"Modnum"
Boîte à outils Scilab
pour les systèmes de communication
Guide de référence

Version provisoire

IRCOM Group
Alan Layec

7 février 2006

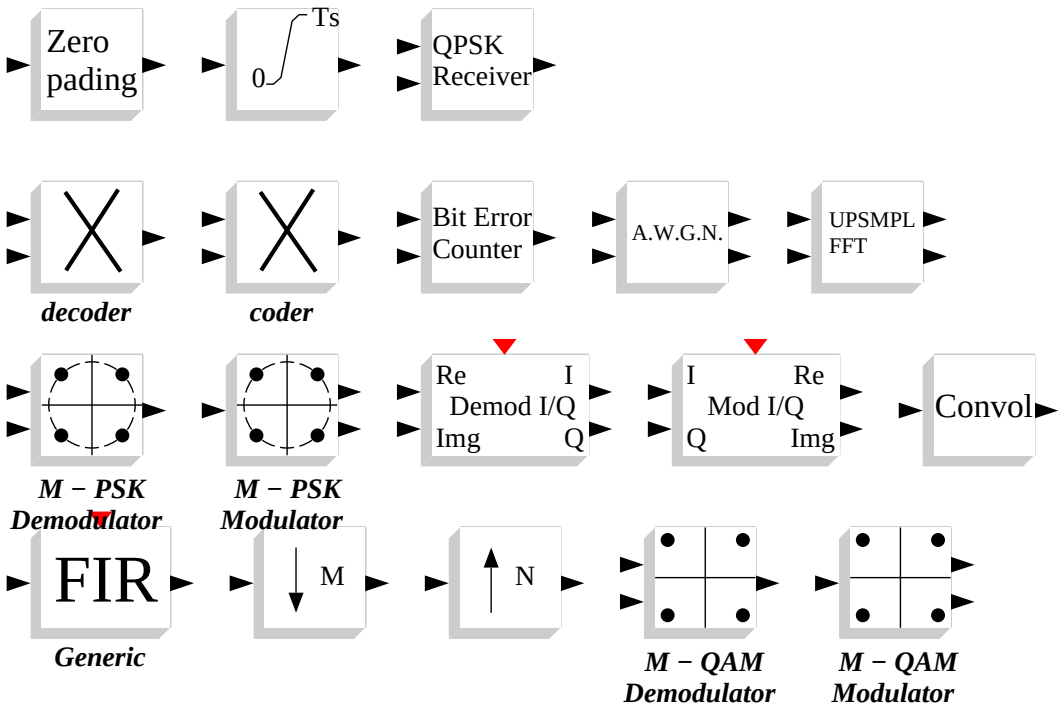
Table des matières

Première partie

Blocs Scicos

Chapitre 1

Palette Communication



1.0.1 Description

La palette Communication est développée pour les communications bande de base utilisant des modulations numériques type PSK/QAM. Elle concerne seulement les fonctions de base rencontrées dans les sous-systèmes numériques.

Toutes les fonctions de cette palette sont réalisées avec des blocs discret de base (DBB) et couvrent de simples opérations comme de changement de cadence, de modulation symbole, d'insertion/décimation de séquences, de filtrage à réponse impulsionnelle finie et aussi des canaux de bases.

1.1 AWGN_f - Bloc canal à bruit blanc gaussien additif

- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `AWGN_f.sci`

1.1.1 Description

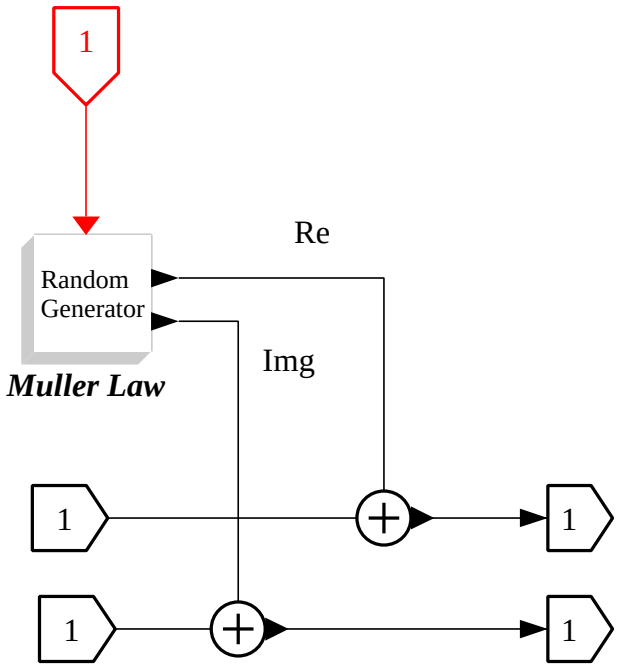
Ce bloc réalise un canal additif à bruit blanc gaussien. Il additionne des échantillons bruités au vecteur des entrées régulières et la sortie régulière peut s'exprimer par la formule suivante :

$$y_k = u_k + \tilde{n}_k \tag{1.1}$$

où u_k est l'élément k du vecteur d'entrée (le premier port est la partie réelle et le deuxième la partie imaginaire), y_k est le vecteur complexe de sortie et $\tilde{n}_k$ l'échantillon bruité calculé par la loi "box Muller".

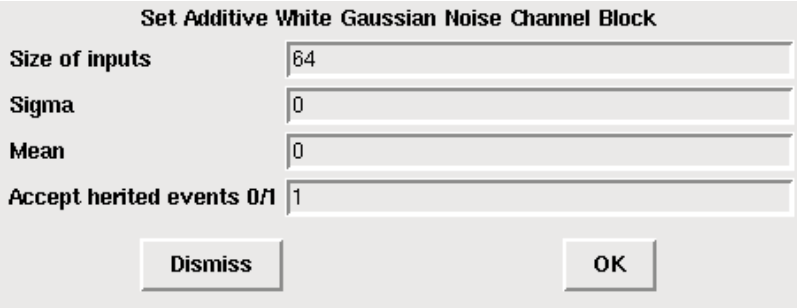
Ce bloc calcule toujours des échantillons de bruit complexes, mais comme montré par le modèle équivalent, il peut-être utilisé pour des vecteurs de signaux réels en utilisant seulement les premiers ports réguliers d'entrée/sortie.

1.1.2 Modèle équivalent en Super Bloc



1.1.3 Boîte de dialogue

- **Size of inputs :**
  Type 'vec' de taille 1. La taille des ports réguliers d'entrée.
- **Sigma :**
  Type 'vec' de taille 1. La variance du générateur de bruit.
- **Mean :**
  Type 'vec' de taille 1. La moyenne du générateur de bruit.



– **Accepted herited (0/1) ? :**
Type ’vec’ de taille 1. Active la propriété d’héritage par évènement. (supprime le port d’entrée évènementiel)

1.1.4 Fonction de calcul (type 4)

```
/*
 * awgn Scicos Additive White Gaussian Noise channel block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 23 décembre 2004 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"

/* Cette fonction de simulation calcule un vecteur aléatoire gaussien
 * par la méthode "Muler Box" et l’ajoute au vecteur complexe d’entrée
 *
 * y1[i]=u1[i]+mean+sigma*sqrt(-log(rand1))*cos(2*%pi*rand2)
 * y2[i]=u2[i]+mean+sigma*sqrt(-log(rand1))*sin(2*%pi*rand2)
 *
 * où mean et sigma sont des paramètres et rand1 et rand2 sont
 * des variables aléatoires uniformément réparties
 *
 * entrée régulière : u1[0..nu-1] vecteur des parties rélles des échantillons d’entrée
 *                    u2[0..nu-1] vecteur des parties imaginaires des échantillons d’entrée
 * sortie régulière : y1[0..ny-1] vecteur des parties rélles des échantillons de sortie
 *                    y2[0..ny-1] vecteur de parties imaginaires des échantillons de sortie
 * entrée évènementielle : Dates de déclenchement
 * sortie évènementeille : néant
 *
 * paramètre entier : néant
 * paramètre réel : rpar[0] : valeur du sigma
 *                  rpar[1] : valeur de la moyenne
 */

/*prototype*/
void awgn(scicos_block *block,int flag)
{
  /*déclaration des variables*/
  double *u1,*u2;
  double *y1,*y2;
  int k;
  int ny;

  /*récupération de l’adresses des ports réguliers*/
  u1=(double *)block->inptr[0];
  u2=(double *)block->inptr[1];
  y1=(double *)block->outptr[0];
  y2=(double *)block->outptr[1];

  /*récupère taille de sortie*/
  ny=block->outsz[0];

  /*Le flag 1 calcule le registre de sortie y*/
  if(flag==1)
  {
   /*Appel noiseiq_c*/
   noiseiq_c(&ny,&block->rpar[0],&block->rpar[1],&y1[0],&y2[0]);

   /*Appel complxa_c*/
   cmplxa_c(&ny,(k=1,&k),&u1[0],&u2[0],&y1[0],&y2[0],&y1[0],&y2[0]);
  }
}
```

1.1.5 Voir aussi

– NOISEBLK_f - Bloc générateur de bruit blanc gaussien (Bloc Scicos)
– noiseblk_c - routine de calcul générateur de bruit blanc gaussien (Routine de calcul bas-niveau)

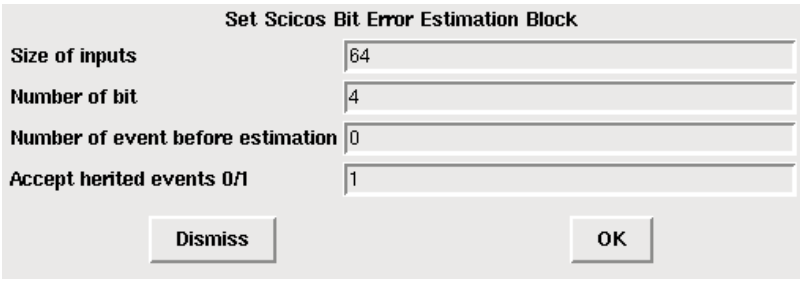
1.2 BITERROR_f - Bloc estimation du taux d'erreur binaire

- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `BITERROR_f.sci`

1.2.1 Description

Ce bloc calcul le taux d'erreur binaire. Il prend en entrée des vecteurs de nombres entiers non signés.

1.2.2 Boîte de dialogue



- **Size of inputs :**
  Type 'vec' de taille 1. La taille des ports réguliers d'entrée.
- **Number of bit :**
  Type 'vec' de taille 1. Le nombre de bit par entier.
- **Number of event before counting :**
  Type 'vec' de taille 1. Le nombre d'activation avant estimation.
- **Accept herited events 0/1 :**
  Type 'vec' de taille 1. Active la propriété d'héritage par évènement. (supprime le port d'entrée évènementiel)

1.2.3 Fonction de calcul (type 4)

```
/* teb2 Scicos block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 9 Novembre 2004 - IRCOM GROUP - Author : A.Layec
 * 15 Fev 2005 : rajout option sur nbre d'évènement
 */

/* REVISION HISTORY :
 * $Log$
 */

#include <stdio.h>
#include "scicos_block.h"

/* Cette fonction de simulation réalise la comparaison
 * bit à bit de deux vecteurs d'entiers non signé.
 * Lorsque qu'un élément des vecteurs diffère, la fonction
 * cherche quel bit est faux et incrémente son état discret pour
 * chaque bit faux(flag2), puis elle recopie l'état discret
 * sur la sortie(flag1)
 *
 * Entrée régulière : inptr[0][0..Nu-1] : Vecteur d'entier 1
 *                    inptr[1][0..Nu-1] : Vecteur d'entier 2
 * Sortie régulière : outptr[0][0] : Valeur du compteur d'erreur
 * Entrée évènementielle : à la rigueur(hérit)
 * Sortie évènementielle : néant
 *
 * Paramètres entier : insz[0] : taille des vecteurs d'entrée
```

```
 *                     ipar[0] : Nombre de bits
 *                     ipar[1] : Nombre d'èvenement avant comptage
 * état discret      : z[0] : compteur nb d'erreur
 *                     z[1] : compteur evènement
 */

/*prototype*/
void teb2(scicos_block *block,int flag)
{
  /*déclaration des variables*/
  int a,b,c;
  int i,j;

  /*Le flag 1 recopie la valeur de z[0] dans outptr[0][0]*/
  if(flag == 1)
  {
    block->outptr[0][0]=block->z[0];
    block->z[1]=block->z[1]+1;
  }

  /*Le flag 2 réalise la comparaison bit à bit*/
  else if(flag == 2)
  {
   /*fprintf(stderr,"z[1]=%d\n",(int)block->z[1]);*/
   if((int)block->z[1]>block->ipar[1])
   {
    for(i=0;i<block->insz[0];i++)
    {
     a=(int)block->inptr[0][i];
     b=(int)block->inptr[1][i];
     /*fprintf(stderr,"a=%d, b=%d\n",a,b);*/
     c=a^b;
     if(c!=0) /*si différence entre les éléments i*/
     {
      for(j=0;j<block->ipar[0];j++)
      {
       if((1<<j & c)==1<<j)  /*comparaison bit à bit*/
       {
        /*fprintf(stderr,"bit %d faux\n",j);*/
        block->z[0]++; /*incrémente compteur*/
       }
      }
     }
    }
   }
  }
}
```

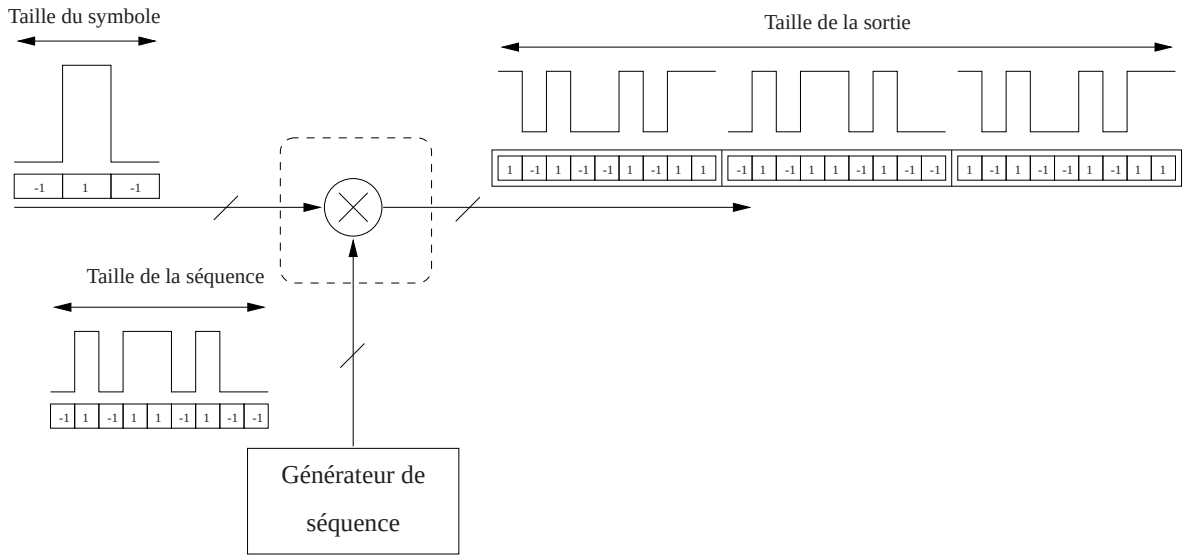
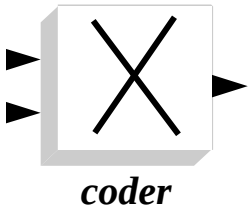


FIG. 1.1 – Modulation d'un vecteur symbole par une séquence

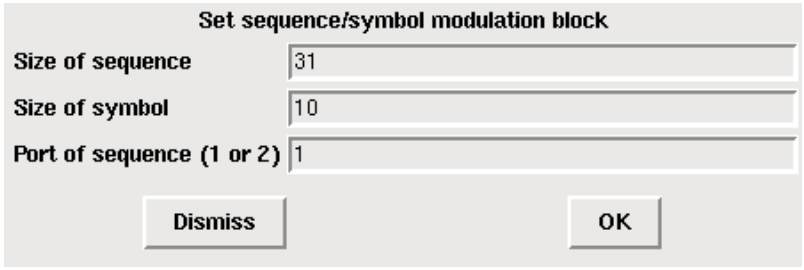
1.3 CODINSERVEC_f - Bloc modulation séquence/symbole



- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `CODINSERVEC_f.sci`

1.3.1 Description

1.3.2 Boîte de dialogue



- **Size of sequence :**
  Type 'vec' de taille -1. La longeur de la séquence. (en nombre d'échantillons)
- **Size of symbol :**
  Type 'vec' de taille -1. La taille du vecteur symbole.
- **Port of sequence :**
  Type 'vec' de taille -1. Le numéro du port (1 ou 2) où est appliqué la séquence.

1.3.3 Fonction de calcul (type 4)

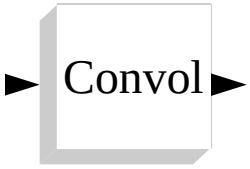
```
/* codinservec Vectorial code insertor block
 * Type 4 simulation function ver 1.0 - scilab-3.0
```

```
 * 16 dec 2004 - IRCOM Lab - Author : Alan
 */

 /* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include "machine.h"
#include <stdio.h>

/* entrées régulières : u1[0..nu1-1] : vecteur d'entrée de code où symbole 1
 *                      u2[0..nu2-1] : vecteur d'entrée de code où symbole 2
 *
 * sorties régulières : y[0..ny-1] : vecteur de sorties de taille ny=nu1*nu2
 *
 * paramètres entiers : ipar[0] : numéro du port d'entrée contenant le code
*/

/*prototype*/
void codinservec(scicos_block *block,int flag)
{
  /*Déclaration des variables*/
  double *uc,*us;
  double *y;
  int nuc,nus,ny;
  int n_c,n_s; /* numéro des ports code(n_c) et symbole(n_s)*/

  /*determination des numéros de ports*/
  n_c=block->ipar[0];
  if (n_c==0) n_s=1;
  else n_s=0;

  /*Récupération des adresses des ports réguliers*/
  uc=(double *)block->inptr[n_c];
  us=(double *)block->inptr[n_s];
  y=(double *)block->outptr[0];

  /*Récupération de la taille des ports d'entrées*/
  nuc=block->insz[n_c];
  nus=block->insz[n_s];

  /*Appel routine codeinser_c*/
  codinser_c(&nuc,&nus,&uc[0],&us[0],&y[0]);
}
```

1.3.4 Voir aussi

– DECODVEC_f - Bloc démodulation séquence/symbole (Bloc Scicos)

1.4 CONVOLGEN_f - Bloc convolution

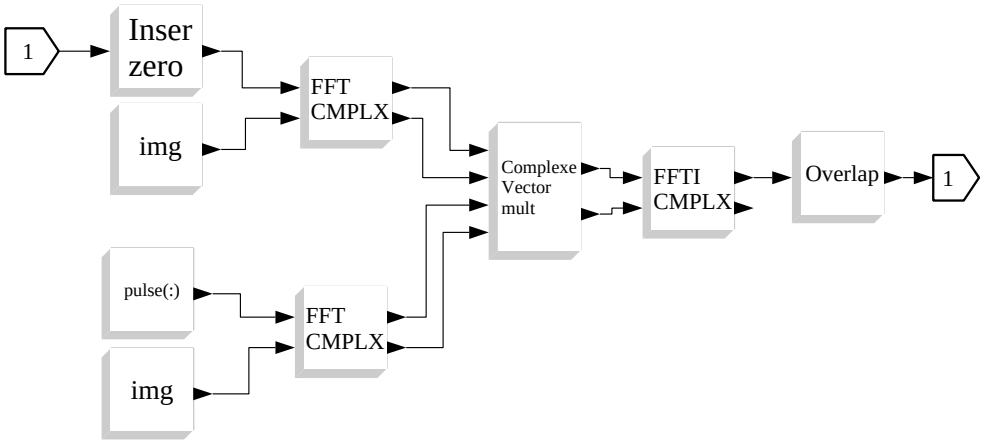


– **Palette :** Communication.cosf - Palette Communication
– **Fonction d'interface :** `CONVOLGEN_f.sci`

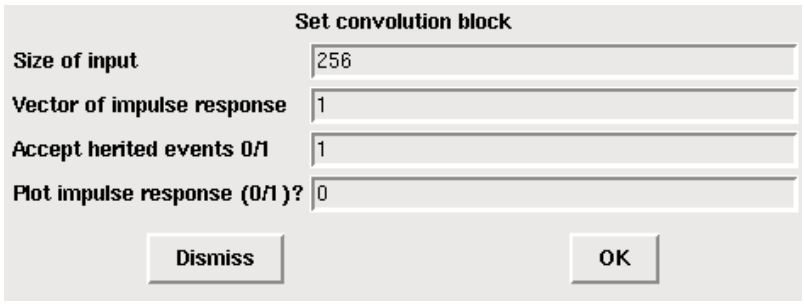
1.4.1 Description

Ce bloc réalise une convolution discrète avec des Transformées de Fourier et une méthode d'empiétement additive. Le vecteur d'entrée u, de taille nu est convolué avec un vecteur réponse impulsionnelle de taille nb_coef définit par le paramètre 'Vector of impulse response'. Le résultat est délivré dans le vecteur de sortie y de taille ny=nu.

1.4.2 Modèle équivalent en Super Bloc



1.4.3 Boîte de dialogue



– **Size of input :**
Type 'vec' de taille 1. La taille du port régulier d'entrée.
– **Vector of impulse response :**
Type 'vec' de taille -1. Vecteur de la réponse impulsionnelle.
– **Accept herited events 0/1 :**
Type 'vec' de taille 1. Active la propriété d'héritage par évènement. (supprime le port d'entrée évènementiel)

– **Plot impulse response (0/1) ? :**
Type ’vec’ de taille 1. Trace la réponse impulsionnelle dans une fenêtre graphique. Le nombre entier donne l’id de la fenêtre.

1.4.4 Fonction de calcul (type 4)

```
/* convol Scicos convolution block by fft method
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 20 Décembre 2004  Author : - IRCOM GROUP - A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include "machine.h"
#include <stdio.h>

/* Entrées régulières : u[0..nu-1] : vecteur d’entrée de taille nu.
 *                                   Echantillons temporels du signal à  filtrer
 *
 * sorties régulières : y[0..ny-1] : vecteur de sortie de taille ny=nu.
 *                                   Echantillons temporels du signal filtré
 *
 * paramètres entiers : nz : taille du vecteur Z = nb_coef. nombre de coefficient du filtre
 *                      ipar[0] = m1 taille vecteur tampon
 *
 * paramètres réels   : rpar[0..nz-1] : rep imulse. Réponse impulsionnelle du filtre
 *
 */


/*Prototype*/
void convol(scicos_block *block,int flag)
{
 /*Déclaration des pointeurs*/
 double *y;
 double *u;
 double *z;
 double *z__;
 double *z1_r, *z1_i, *z2_r, *z2_i, *y_r, *y_i;
 /*Déclaration de variables taille et compteur*/
 int nu,ny;
 int i,k,l;
 int m1,nb_coef;

 /*Récupération de l’adresse des registres*/
 y=(double *)block->outptr[0];
 u=(double *)block->inptr[0];
 z=block->z;

 /*Récupération des tailles des vecteurs*/
 nu=block->insz[0];
 ny=block->outsz[0];
 nb_coef=block->nz;

 /*Récupération de la taille des vecteurs complexes*/
 m1=block->ipar[0];

 /*La flag 1 calcule y*/
 if(flag==1)
  {
   /*Allocation de 3*2 vecteurs de taille m1*/
   if ((*block->work=scicos_malloc(sizeof(double)*(m1*2*3)))== NULL)
   {
    set_block_error(-16);
    return;
   }

   /*Récupération de l’adresse de départ du vecteur alloué*/
   z__=*block->work;

   /*Déclaration de pointeurs auxiliaires*/
   z1_r = &(z__[0])   ; z1_i = &(z__[m1]);   /*vecteur du complexe 1*/
   z2_r = &(z__[2*m1]); z2_i = &(z__[3*m1]); /*vecteur du complexe 2*/
   y_r  = &(z__[4*m1]); y_i  = &(z__[5*m1]); /*vecteur du complexe résultat*/

   /*Recopie u[] dans z1_r[]*/
   F2C(dcopy)(&nu,&u[0],(k=1,&k),&z1_r[0],(k=1,&k));

   /*Recopie rpar[] dans z2_r[]*/
   F2C(dcopy)(&nb_coef,&(block->rpar[0]),(k=1,&k),&z2_r[0],(k=1,&k));

   /*Appel convolr_c*/
   convolr_c(&nu,&nb_coef,&m1,&z1_r[0],&z1_i[0],&z2_r[0],&z2_i[0],&y_r[0],&y_i[0],&z[0]);

   /*Recopie y_r[] dans y[]*/
   F2C(dcopy)(&ny,&y_r[0],(k=1,&k),&y[0],(k=1,&k));

   /*Libère mémoire allouée*/
   scicos_free(*block->work);
  }
}
```

1.4.5 Voir aussi

- RIFGEN_f - Bloc filtre à réponse impulsionnelle finie générique (Bloc Scicos)
- FFTCMPLX_f - Bloc de Transformation de Fourier Rapide (Bloc Scicos)
- VECTMULTCMPLX_f - Bloc multiplicateur vectoriel de signaux complexes (Bloc Scicos)
- INSERZEROVEC_f - Bloc insertion de zéro (Bloc Scicos)
- OVERLAP_f - Bloc empiétement (Bloc Scicos)

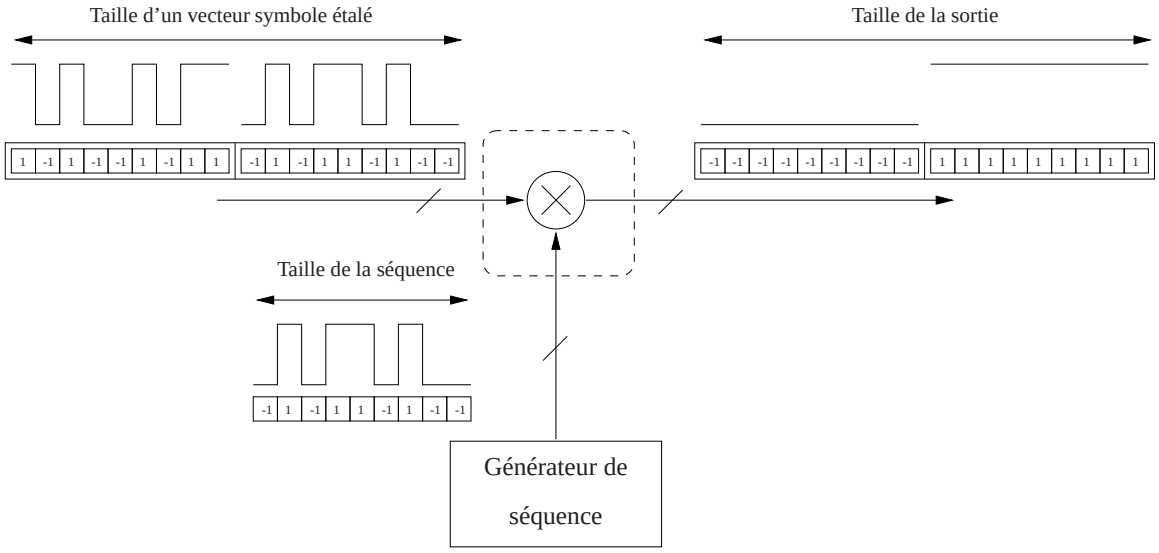
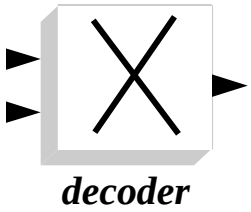


FIG. 1.2 – Démodulation d'un vecteur de symboles étalés par une séquence

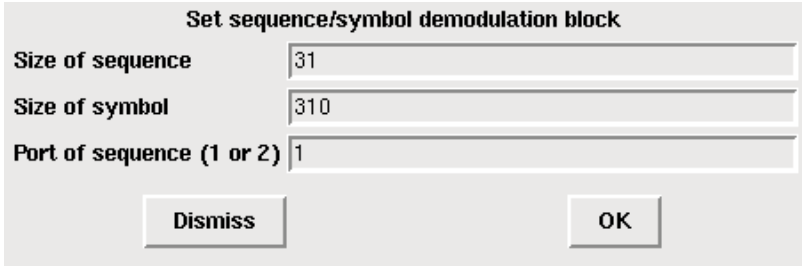
1.5 DECODVEC_f - Bloc démodulation séquence/symbole



- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `DECODVEC_f.sci`

1.5.1 Description

1.5.2 Boîte de dialogue



- **Size of sequence :**
  Type 'vec' de taille -1. La longueur de la séquence. (en nombre d'échantillons)
- **Size of symbol :**
  Type 'vec' de taille -1. La taille du vecteur symbole.
- **Port of sequence :**
  Type 'vec' de taille -1. Le numéro du port (1 or 2) où est appliqué la séquence.

1.5.3 Fonction de calcul (type 4)

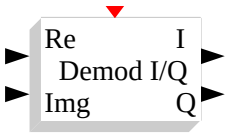
```
/* decodvec Vectorial decoder block
 * Type 4 simulation function ver 1.0 - scilab-3.0
```

```
 * 14 dec 2004 - IRCOM Lab - Author : Alan
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include "machine.h"
#include <stdio.h>

/*
 * entrées régulières : u1[0..nu1-1] : vecteur d'entrée de code où symbole 1
 *                      u2[0..nu2-1] : vecteur d'entrée de code où symbole 2
 *
 * sorties régulières : y[0..ny-1] : vecteur de sorties de taille ny=nu1 ou ny=nu2
 *
 * paramètres entiers : ipar[0] : numéro du port d'entrée contenant le code
 */

/*prototype*/
void decodvec(scicos_block *block,int flag)
{
  /*Déclaration des variables*/
  double *uc,*us;
  double *y;
  int i,j,k;
  int nuc,nus,ny;
  int n_c,n_s; /* numéro des ports code(n_c) et symbole(n_s)*/

  /*determination des numéros de ports*/
  n_c=block->ipar[0];
  if (n_c==0)
   n_s=1;
  else
   n_s=0;

  /*Récupération des adresses des ports réguliers*/
  uc=(double *)block->inptr[n_c];
  us=(double *)block->inptr[n_s];
  y=(double *)block->outptr[0];

  /*fprintf(stderr,"flag=%d,t=%f\n",flag,get_scicos_time()); */
  /*Récupération de la taille des ports d'entrées*/
  nuc=block->insz[n_c];
  nus=block->insz[n_s];

  /*attention*/
  k=nus/nuc; /*doit tjs etre entier*/

  /*Appel routine decod_c*/
  decod_c(&nuc,&k,&uc[0],&us[0],&y[0]);
}
```

1.5.4 Voir aussi

– CODINSERVEC_f - Bloc modulation séquence/symbole (Bloc Scicos)

1.6 DEMODIQ_f - Bloc démodulateur I/Q



- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `DEMODIQ_f.sci`

1.6.1 Description

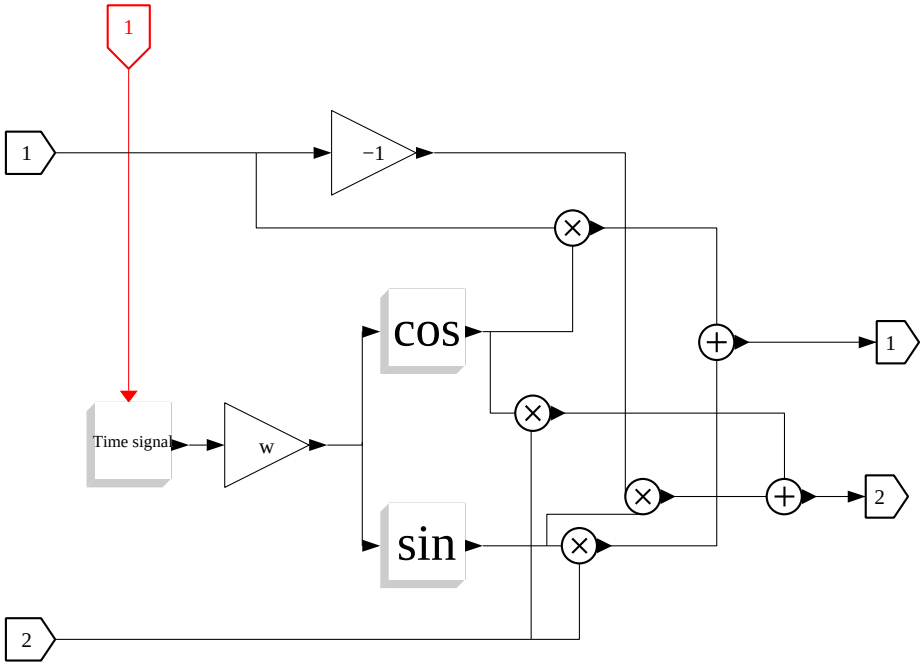
Ce bloc realise une démodulation complexe dans le domaine temporel. Il calcule :

$$y_1(t) = u_1 \cos(\omega_o t) + u_2 \sin(\omega_o t) \tag{1.2}$$

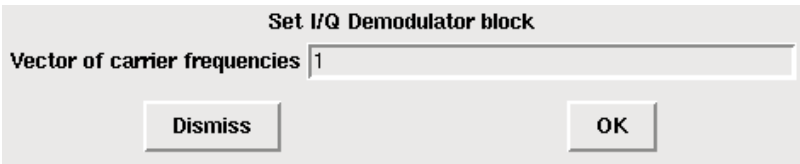
$$y_2(t) = u_2 \cos(\omega_o t) - u_1 \sin(\omega_o t) \tag{1.3}$$

avec u1,u2 les entrées régulières et y1,y2 les sorties. wo est renseigné par le paramètre "vector of frequencies" de la boîte de dialogue. Ce bloc fonctionne de paire avec le bloc MODIQ_f.

1.6.2 Modèle équivalent en Super Bloc



1.6.3 Boîte de dialogue



- **Vector of carrier frequency :**
  Type 'vec' de taille -1. Le vecteur des pulsations libres des oscillateurs locaux.

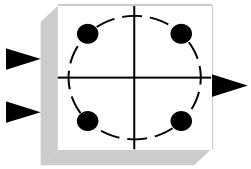
1.6.4 Fonction de calcul (type 2)

```
/* demodiq Scicos demodulator I/Q block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 8 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <math.h>

/* Cette fonction calcule la partie réelle et imaginaire de l'enveloppe
 * complexe en bande de base à partir de la partie réelle et imaginaire
 * de l'enveloppe modulée autour d'une fréquence wo :
 * y1=u1*cos(wot)+u2*sin(wot)
 * y2=u1*sin(wot)-u2*cos(wot)
 * où y1, y2 sont les sorties, u1 et u2 sont la partie réelle et imaginaire
 * du signal modulé et wo un paramètre réel en rad/s. Les porteuses sont en fait
 * sur-échantillonnées par la variable *t. Le rythme des déclenchements défini
 * donc la période de sur-échantillonnage.
 *
 * entrées régulières: u1[0..insz[0]-1] vecteur des parties réelles du signal modulé
 *                     u2[0..insz[0]-1] vecteur des parties imaginaires du signal modulé
 * sorties régulières : y1[0..insz[0]-1] vecteur des parties réelles du signal en bande de base
 *                     y2[0..insz[0]-1] vecteur des parties imaginaires du signal en bande de base
 * entrée évènementielles : instants de suréchantillonnage
 * sorties évènementielles : néant
 * paramatres réels : rpar[0..insz[0]-1] : vecteur des pulsations porteuses
 */

/*prototype*/
void demodiq(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
             ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
   /*déclaration des variables*/
   double *y1,*y2;
   double *u1,*u2;
   int nu,i;

   /*récupération des adresses des ports réguliers*/
   y1=(double *)outptr[0];
   y2=(double *)outptr[1];
   u1=(double *)inptr[0];
   u2=(double *)inptr[1];

   /*récupération de la taille des ports d'entrées*/
   nu=insz[0];

   /*Le flag 1 calcule la partie réelle et imaginaire de
    * l'enveloppe complexe en bande de base.
    */
   if(*flag==1)
   {
    for(i=0;i<nu;i++)
     {
      /*Calcul du registre y*/
      y1[i]=cos(*t*rpar[i])*u1[i]+sin(*t*rpar[i])*u2[i];
      y2[i]=sin(*t*rpar[i])*u1[i]-cos(*t*rpar[i])*u2[i];
     }
   }
}
```

1.6.5 Voir aussi

– MODIQ_f - Bloc modulateur I/Q (Bloc Scicos)

1.7 DEMODPSK_f - Bloc de démodulation par états de phase M-aires



- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `DEMODPSK_f.sci`

1.7.1 Description

Ce démodulateur M-PSK prend en entrée deux composantes [I ;Q] (respectivement [u1 ;u2]) en valeurs réelles et calcule un numéro symbole en valeur entière en accord avec le bloc modulateur M-PSK. La phase est évaluée avec l'équation :

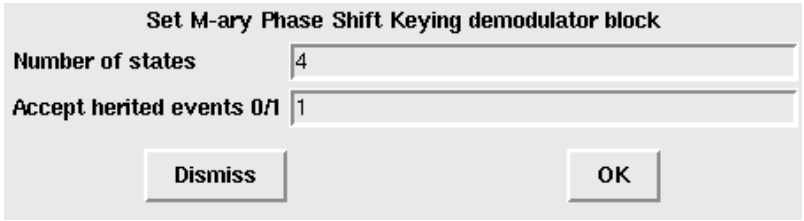
$$\varphi_{demod} = \arctan\left(\frac{-u_2}{u_1}\right) \tag{1.4}$$

et le bloc calcule la valeur entière de sortie avec la formule :

$$y = (int)\frac{\varphi_{demod} M}{2\pi} \tag{1.5}$$

où M est le nombre d'états.

1.7.2 Boîte de dialogue



- **Number of states :**
  Type 'vec' de taille -1. Nombre d'états de la modulation.
- **Accept herited events 0/1 :**
  Type 'vec' de taille 1. Active la propriété d'héritage par évènement. (supprime le port d'entrée évènementiel).

1.7.3 Fonction de calcul (type 4)

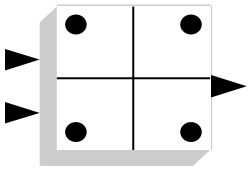
```
/* demodpsk Scicos demodulator MPSK block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 22 décembre 2004 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include "machine.h"
#include <math.h>
#define M_PI 3.14159265358979323846

/* Cette fonction de simulation réalise un démodulateur MPSK.
 * A chaque instant de déclenchement les valeurs de I et Q sont
 * lues en entrée et la fonction calcule dans un premier temps
 * arctang[-Q/I] pour récupérer la valeur de la phase. Ensuite
```

```
 * elle calcule la valeur du numéro symbole par la formule
 * y=(int)((phi*n)/(2*%pi)) avec n le nombre d'états et phi la valeur
 * de arctang[-Q/I].
 *
 * entrées régulières: u1[0..insz[0]-1] vecteur des signaux I
 *                     u2[0..insz[0]-1] vecteur des signaux Q
 * sorties régulières : y[0..insz[0]-1] vecteur des numéros symboles
 * entrée évènementielles : instants de déclenchement
 * sorties évènementielles : néant
 * paramatres entiers : ipar[0..insz[0]-1] : vecteur du nombre d'état
 */

/* demodpskv_c routine de calcul de démodulation PSK
 *
 * Entrées :
 * n      : taille des vecteur originaux
 * m      : nombre d'états (vecteur)
 * i_c    : vecteur de la composante I
 * q_c    : vecteur de la composante Q
 * Sorties :
 * y      : vecteur du numéro symbole
 *
 * dépendance :
 * math.h
 */

void demodpskv_c(int *n,int *m,double *i_c,double *i_q,double*y)
{
 /*Déclaration des variables*/
 int i;
 double phi;

 for(i=0;i<(*n);i++)
 {
  /*Calcul de la phase*/
  phi=atan2(-i_q[i],i_c[i]);
  if(phi<0) phi = phi + 2*M_PI;
  /*Calcul du numéro symbole*/
  y[i]=(int)(phi*m[i]/(2*M_PI));
 }
 return;
}

/*Prototype*/
void demodpsk(scicos_block *block,int flag)
{
 /*Déclaration des variables*/
 double *y;
 double *u1,*u2;

 /*Récupération des adresses des ports réguliers*/
 y=(double *)block->outptr[0];
 u1=(double *)block->inptr[0];
 u2=(double *)block->inptr[1];

 /*Le flag 1 calcule le numéro symbole*/
 if(flag==1)
 {
  /*Appel routine demodpsk_c*/
  demodpskv_c(&block->insz[0],&block->ipar[0],&u1[0],&u2[0],&y[0]);
 }
}
```

1.7.4 Voir aussi

- MODPSK_f - Bloc de modulation par états de phase M-aires (Bloc Scicos)
- MODQAM_f - Bloc modulateur de Modulation d'Amplitude en Quadrature (Bloc Scicos)
- DEMODQAM_f - Bloc démodulateur de Modulation d'Amplitude en Quadrature (Bloc Scicos)

1.8 DEMODQAM_f - Bloc démodulateur de Modulation d'Amplitude en Quadrature



– **Palette :** Communication.cosf - Palette Communication
– **Fonction d'interface :** `DEMODQAM_f.sci`

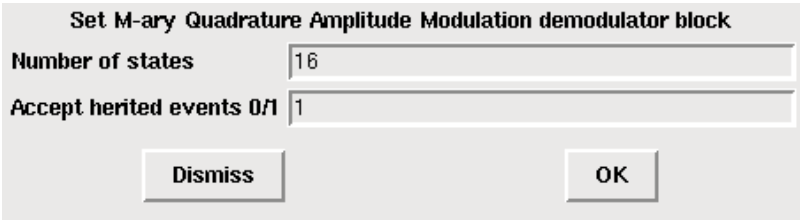
1.8.1 Description

Le bloc démodulateur M-QAM fonctionne de paire avec le bloc modulateur M-QAM. Pour fonctionner correctement, les valeurs en entrée de ce bloc doivent être entières, comme définies à la sortie du bloc modulateur. Le paramètre 'Number of states' de la boîte de dialogue, qui définit le nombre d'états de la modulation, doit être calculé avec la formule :

$$M = 2^k \tag{1.6}$$

avec k un nombre entier pair.

1.8.2 Boîte de dialogue



– **Number of states :**
Type 'vec' de taille -1. Le nombre d'états de la modulation.
– **Accept herited events 0/1 :**
Type 'vec' de taille 1. Active la propriété d'héritage par évènement. (supprime le port d'entrée évènementiel)

1.8.3 Fonction de calcul (type 4)

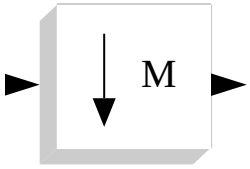
```
/* demodqam Scicos demodulator QAM block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 5 janvier 2005 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>
#include "scicos_block.h"

/* Cette fonction de simulation réalise un decodage QAM défini par la fonction
 * modqam.c. Les signaux I et Q reçu en entrée sont converti aux valeurs des
 * bits de poids fort et des bits de poids faibles du numéro symbole.
 * Les valeurs converties sont ensuite additionnées et la somme est placée
 * dans le registre de sortie y[].
 *
 * Entrées régulières : u1[0..insz[0]-1]: vecteur des signaux I
 *                      u2[0..insz[0]-1]: vecteur des signaux Q
 * Sortie régulière :   u[0..insz[0]-1] : vecteur des numéros symboles
 * Entrée évènementielle : instants de décision
 * Sortie évènementielle : néant
```

```
 * Paramètres entiers : ipar[0..insz[0]-1] : vecteur des nombres de bits.
 *
 * nb : ne fonctionne que pour des modulations avec un nombre de valeurs
 *     identiques pour chaque axes I et Q(16QAM, 64QAM, 256QAM, ...)
 */

/* demodqamv_c routine de calcul d'un démodulataur mQAM
 *
 * Entrées :
 * n :longueur des vecteurs d'entrées
 * m :longueurs des mots binaires en nombre de bits (vecteur)
 * i_c :adresse de départ du vecteur de la composante I
 * q_c :adresse de départ du vecteur de la composante Q
 *
 * Sortie :
 * y : adresse de départ du vecteur du symbole
 *
 * Dépendances:
 */
void demodqamv_c(int *n,int *m,double *i_c,double *q_c,double *y)
{
 /*déclaration des variables*/
 int i,j;
 int ng,nd;
 int d,g;

 for(i=0;i<(*n);i++)
 {
  /*Récupération du nombre d'états*/
  /*n=block->ipar[i];*/

  /*Calcul des sélecteurs binaires (c'est maladroit!!)*/
  nd=(1<<(m[i])/2)-1;
  ng=(1<<(m[i]))-1-nd;
  /*fprintf(stderr,"nd=%d, ng=%d\n", nd,ng);*/

  /*récupération des valeurs d'entrée*/
  d=(int)i_c[i];
  g=(int)q_c[i];

  /*Calcul les nombres binaires droit et gauche*/
  d=(d+nd)/2;
  g=((g+nd)/2)<<((m[i])/2);

  /*Calcul du registre de sortie y[]*/
  y[i]=d+g;
 }
 return;
}

/*prototype*/
void demodqam(scicos_block *block,int flag)
{
 /*déclaration des variables*/
 double *y;
 double *u1,*u2;
 int i,n;
 int nu;
 int nd,ng;
 int d,g;

 /*Récupération des adresses des ports réguliers*/
 y=(double *)block->outptr[0];
 u1=(double *)block->inptr[0];
 u2=(double *)block->inptr[1];

 /*récupération de la taille des ports d'entrées*/
 nu=block->insz[0];

 /*Le flag 1 calcule le numéro symbole*/
 if(flag==1)
 {
  /*Appel demoqamv_c*/
  demodqamv_c(&nu,&block->ipar[0],&u1[0],&u2[0],&y[0]);
 }
}
```

1.8.4 Voir aussi

- MODQAM_f - Bloc modulateur de Modulation d'Amplitude en Quadrature (Bloc Scicos)
- MODPSK_f - Bloc de modulation par états de phase M-aires (Bloc Scicos)
- DEMODPSK_f - Bloc de démodulation par états de phase M-aires (Bloc Scicos)

1.9 DOWNSMPL_f - Bloc réducteur de cadence

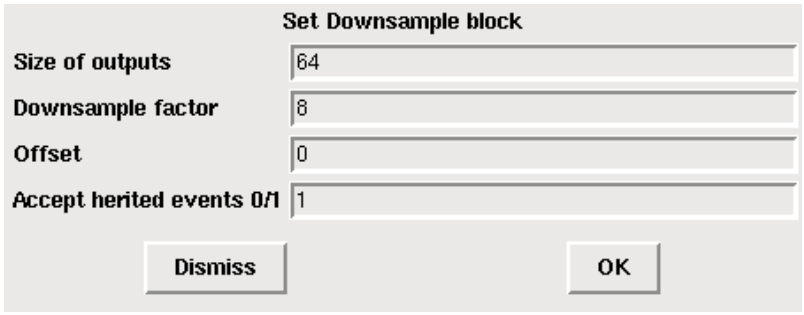


– **Palette :** Communication.cosf - Palette Communication
– **Fonction d'interface :** `DOWNSMPL_f.sci`

1.9.1 Description

C'est le bloc qui fonctionne de paire avec le bloc UPSMPL_f. Il réalise une diminution du taux d'échantillonage de l'entrée en prenant un élément parmi M, où M et le facteur de sous-échantillonnage.

1.9.2 Boîte de dialogue



– **Size of outputs :**
  Type 'vec' de taille 1. La taille du port régulier de sortie.
– **Downsample factor :**
  Type 'vec' de taille 1. Le facteur de sous-échantillonnage.
– **Offset :**
  Type 'vec' de taille 1. La valeur initiale du compteur discret, donne la valeur du premier échantillon qui est pris en entrée.
– **Accept herited events 0/1 :**
  Type 'vec' de taille 1. Active la propriété d'héritage par évènement. (supprime le port d'entrée évènementiel)

1.9.3 Fonction de calcul (type 4)

```
/* sousecht Scicos temporal undersamplig
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 22 Décembre 2004 Author : - IRCOM GROUP - A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include "machine.h"
#include <stdio.h>

/* Cette fonction de simulation réalise un sous échantillonnage d'un
 * vecteur d'entrée de taille nu vers un vecteur de sortie de taille ny.
 * Pour chaque vecteur, la fonction recopie les valeurs de u dans y
 * lorsque l'échantillon testé est celui qui correspond au numéro count
 * exemple count = c.i + nech
 *
 * Entrée régulière : signal surechantilonné
 * Sortie régulière : signal sous échantillonnés
 * Entrée évènementielle : néant
 * Sortie évènementielle : néant
 *
```

```
 * Paramètres entier :     insz[0] : taille du vecteur en entrée
 *                         ipar[0] : nombre d'échantillons
 *
 * état discret      :     z[0] : valeur initiale du numéro échantillons
 */

/*prototype*/
void sousecht(scicos_block *block,int flag)
{
 /*Déclaration des variables*/
 double *y;
 double *u;
 int i,j;
 int nu,nech,counter;

 /*Récupération des adresses des ports réguliers*/
 y=(double *)block->outptr[0];
 u=(double *)block->inptr[0];

 /*Récupération du nombre d'échantillons*/
 nu=block->insz[0];
 nech=block->ipar[0];

 /*Le flag1*/
 if(flag==1)
 {
  counter=(int)block->z[0];
  /*Appel routine sousecht_c*/
  sousecht_c(&nu,&nech,&counter,&u[0],&y[0]);
 }
}
```

1.9.4 Voir aussi

– UPSMPL_f - Bloc élévateur de cadence (Bloc Scicos)

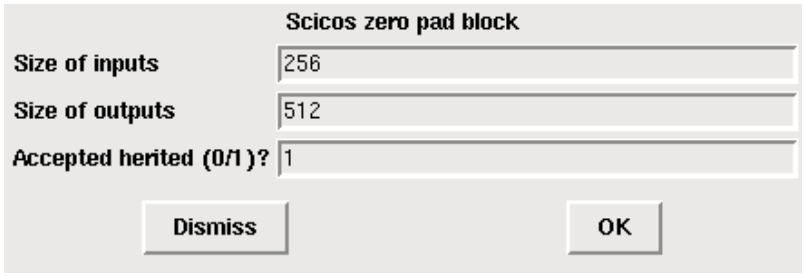
1.10 INSERZEROVEC_f - Bloc insertion de zéro

- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `INSERZEROVEC_f.sci`

1.10.1 Description

Add here a paragraph of the function description.

1.10.2 Boîte de dialogue



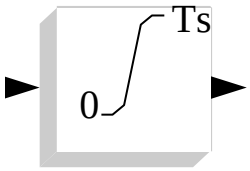
- **Size of inputs :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Size of outputs :**
  Type 'vec' de taille 1. La description du paramètre 2.
- **Accepted herited (0/1) ? :**
  Type 'vec' de taille 1. La description du paramètre 3.

1.10.3 Fonction de calcul (type 4)

```
/* inserzerovec Scicos vectorial zero instertion
 * Type 4 simulation function ver 1.0 - scilab-2.6&2.7
 * 4 janvier 2005 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include "scicos_block.h"

/* Cette fonction transforme un vecteur de taille Nu en un vecteur
 * de taile Ny, avec Ny>Nu, et en ajoutant des zéros dans l'espace > Nu
 *
 * Entrée régulière  : u[0..nu-1] : vecteur d'entrée de taille nu
 * sorties régulière : y[0..ny-1] : vecteur de sortie de taille ny
 */

/*prototype*/
void inserzerovec(scicos_block *block,int flag)
{
  /*Déclaration des variables*/
  double *y;
  double *u;
  int i,nu,ny,k;

  /*Récupération des adresses des ports réguliers*/
  y=(double *)block->outptr[0];
  u=(double *)block->inptr[0];

  /*Récupération des tailles des ports réguliers*/
  nu=block->insz[0];
  ny=block->outsz[0];
```

```
  /*recopie u[] dans y[]*/
  F2C(dcopy)(&nu,&u[0],(k=1,&k),&y[0],(k=1,&k));

  /*ajoute les zéros*/
  for(i=nu;i<ny;i++) y[i]=0;
}
```

1.11 INTSYMB_f - Bloc intégrateur discret symbole

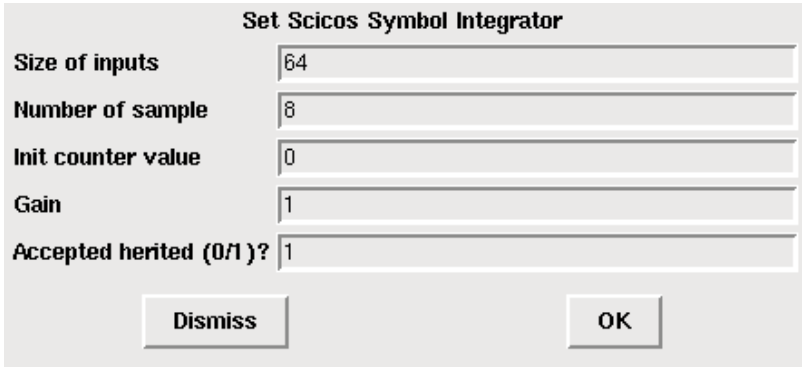


- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `INTSYMB_f.sci`

1.11.1 Description

Add here a paragraph of the function description.

1.11.2 Boîte de dialogue



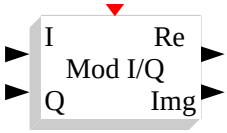
- **Size of inputs :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Number of sample :**
  Type 'vec' de taille 1. La description du paramètre 2.
- **Init counter value :**
  Type 'vec' de taille 1. La description du paramètre 3.
- **Gain :**
  Type 'vec' de taille 1. La description du paramètre 4.
- **Accepted herited (0/1) ? :**
  Type 'vec' de taille 1. La description du paramètre 5.

1.11.3 Fonction de calcul (type 4)

```
/* int_symb scicos Symbol Integrator
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 22 Décembre 2004 Author : - IRCOM GROUP - A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>
#include "scicos_block.h"
/* Entrée régulière : u[0..nu] : vecteur à intégrer
 * Sortie régulière : y[0..nu]=integral(u[0..nu]) : vecteur intégré
 * Entrée évènementielle : (à la rigueur)
 * Sortie évènementielle : néant
 *
 * Paramètres entier : ipar[0] : Longueur en échantillons du symbole
 *                     ipar[1] : N° Echantillon initial d'integration
 *                     insz[0] : taille du vecteur d'entrée
 *
```

```
 * Paramètres réels :  rpar[0] : gain en sortie
 *
 * Etat discret : z[0] : mémoire valeur intégrée précedente (pour le bout du vecteur)
 *
 */

/*prototype*/
void int_symb(scicos_block *block,int flag)
{
  /*Déclaration des variables*/
  double *y;
  double *u;
  double step;
  int i,j;
  int count,init_c,nu,nech;

  /*Récupération des adresses des ports réguliers*/
  y=(double *)block->outptr[0];
  u=(double *)block->inptr[0];

  /*Récupération du nombre d'échantillons*/
  nu=block->insz[0];
  nech=block->ipar[0];
  init_c=block->ipar[1];
  step=block->rpar[0];

  /*Le flag1*/
  if(flag==1)
  {
   /*Appel intsymb_c*/
   intsym_c(&nu,&nech,&init_c,&step,&u[0],&y[0],&block->z[0]);
  }
}
```

1.12 MODIQ_f - Bloc modulateur I/Q



- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `MODIQ_f.sci`

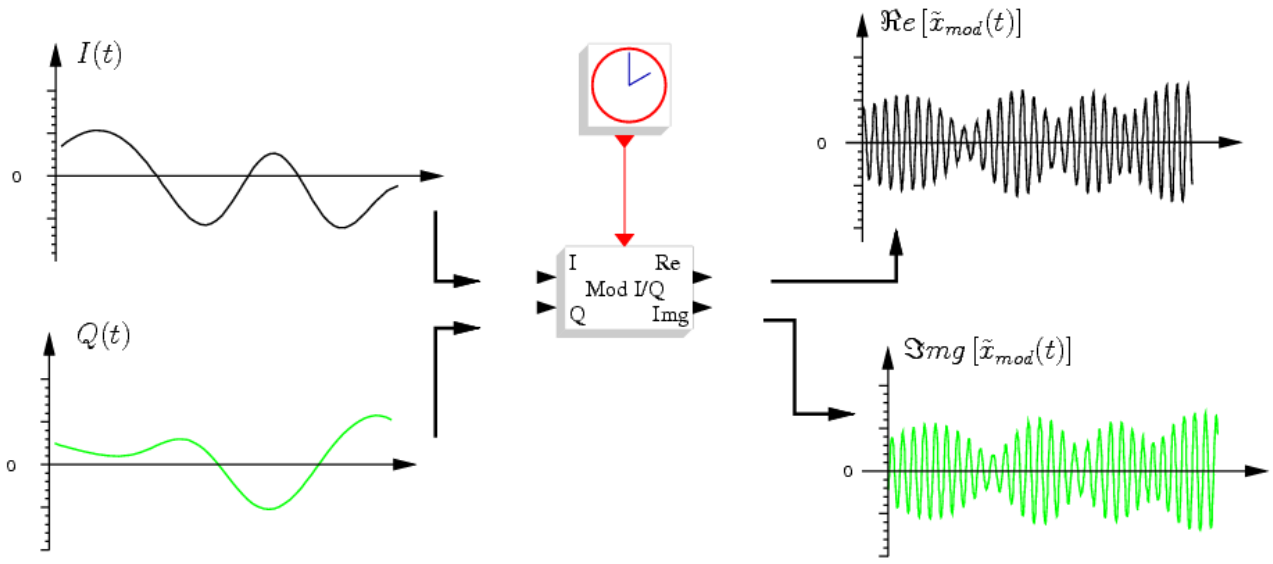
1.12.1 Description

Ce bloc realise un oscillateur local complexe. Il calcule :

$$y_1(t) = u_1(t)\cos(\omega_o t) - u_2(t)\sin(\omega_o t) \tag{1.7}$$

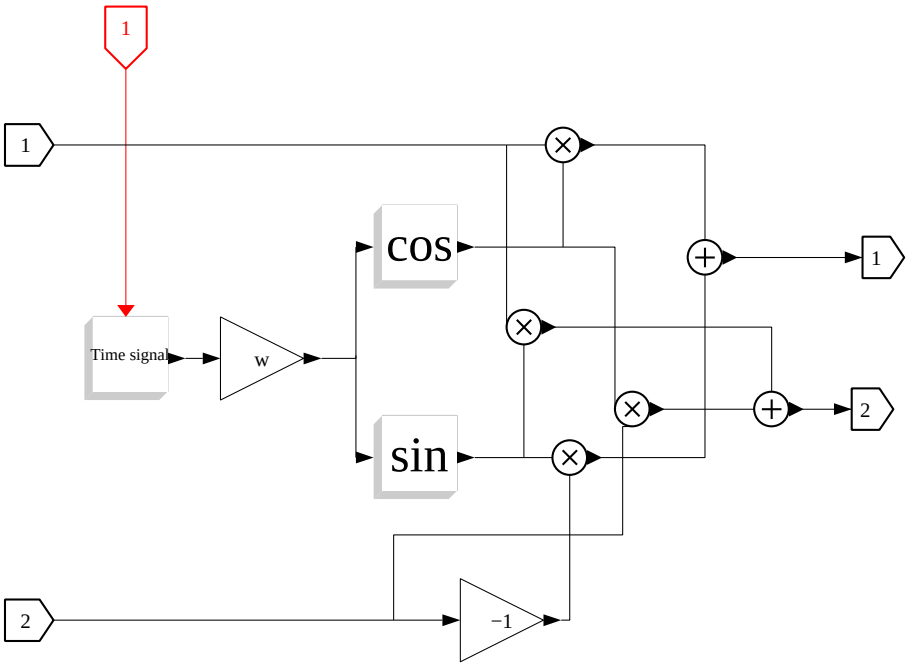
$$y_2(t) = u_1(t)\sin(\omega_o t) + u_2(t)\cos(\omega_o t) \tag{1.8}$$

où u1,u2 sont les deux entrées régulières et y1, y2 les sorties régulières. w0 est renseigné par le paramètre 'Vector of carrier frequency' de la boîte de dialogue. Ce bloc est typiquement utilisé pour réaliser une transposition de composantes symboles en bande de base autour d'une fréquence porteuse comme le montre la figure suivante :

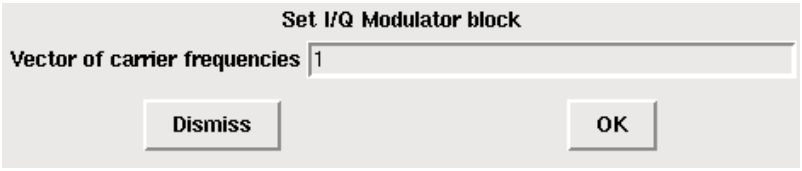


Les dates d'activation de ce bloc déterminent le pas d'échantillonnage des grandeurs de sorties.

1.12.2 Modèle équivalent en Super Bloc



1.12.3 Boîte de dialogue



– **Vector of carrier frequency :**
Type ’vec’ de taille -1. Le vecteur des pulsations libres des oscillateurs locaux.

1.12.4 Fonction de calcul (type 2)

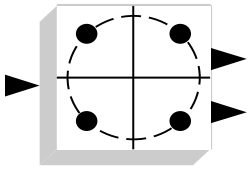
```
/* modiq Scicos modulator I/Q block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 8 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <math.h>

/* Cette fonction calcule la partie réelle et imaginaire de l’enveloppe
 * complexe modulée autour d’une fréquence wo à partir de la partie réelle et imaginaire
 * de l’enveloppe complexe en bande de base :
 * y1=u1*cos(wot)+u2*sin(wot)
 * y2=u1*sin(wot)-u2*cos(wot)
 * où y1, y2 sont les sorties, u1 et u2 sont la partie réelle et imaginaire
 * du signal en bande de base et wo un paramètre réel en rad/s. Les porteuses
 * sont en fait sur-échantillonnées par la variable *t. Le rythme des
 * déclenchements défini donc la période de sur-échantillonnage.
 *
 * entrées régulières: u1[0..insz[0]-1] vecteur des parties réelles du signal en bande de base
 *                     u2[0..insz[0]-1] vecteur des parties imaginaires du signal en bande de base
 * sorties régulières : y1[0..insz[0]-1] vecteur des parties réelles du signal modulé
 *                      y2[0..insz[0]-1] vecteur des parties imaginaires du signal modulé
 * entrée évènementielles : instants de suréchantillonnage
 * sorties évènementielles : néant
 * paramatres réels : rpar[0..insz[0]-1] : vecteur des pulsations porteuses
 */

/*prototype*/
void modiq(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
           ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
```

```
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
 /*déclaration des variables*/
 double *y1,*y2;
 double *u1,*u2;
 int nu,i;

 /*récupération des adresses des ports réguliers*/
 y1=(double *)outptr[0];
 y2=(double *)outptr[1];
 u1=(double *)inptr[0];
 u2=(double *)inptr[1];

 /*récupération de la taille des ports d'entrées*/
 nu=insz[0];

 /* Le flag1 calcule la partie réelle et imaginaire
  * du signal modulé
  */
 if(*flag==1)
 {
  for(i=0;i<nu;i++)
  {
   /*Calcul du registre y*/
   y1[i]=cos(*t*rpar[i])*u1[i]+sin(*t*rpar[i])*u2[i];
   y2[i]=sin(*t*rpar[i])*u1[i]-cos(*t*rpar[i])*u2[i];
  }
 }
}
```

1.12.5 Voir aussi

– DEMODIQ_f - Bloc démodulateur I/Q (Bloc Scicos)

1.13 MODPSK_f - Bloc de modulation par états de phase M-aires



- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `MODPSK_f.sci`

1.13.1 Description

Ce modulateur M-PSK calcule les valeurs des composantes en phase/quadrature (I/Q) à partir des valeurs entières données en entrée. La phase est calculée grâce à l'équation :

$$\varphi = \pi\left(\frac{2u+1}{M}\right) \tag{1.9}$$

où u est la valeur entière d'entrée et M est le nombre d'états définit par le paramètre 'Number of states' de la boîte de dialogue. Les valeurs de sorties sont données par :

$$y_1 = cos\left(\varphi\right) \tag{1.10}$$
$$y_2 = -\sin\left(\varphi\right). \tag{1.11}$$

Pour des modulations 4 et 8-PSK, ce bloc réalise les constellations suivantes :

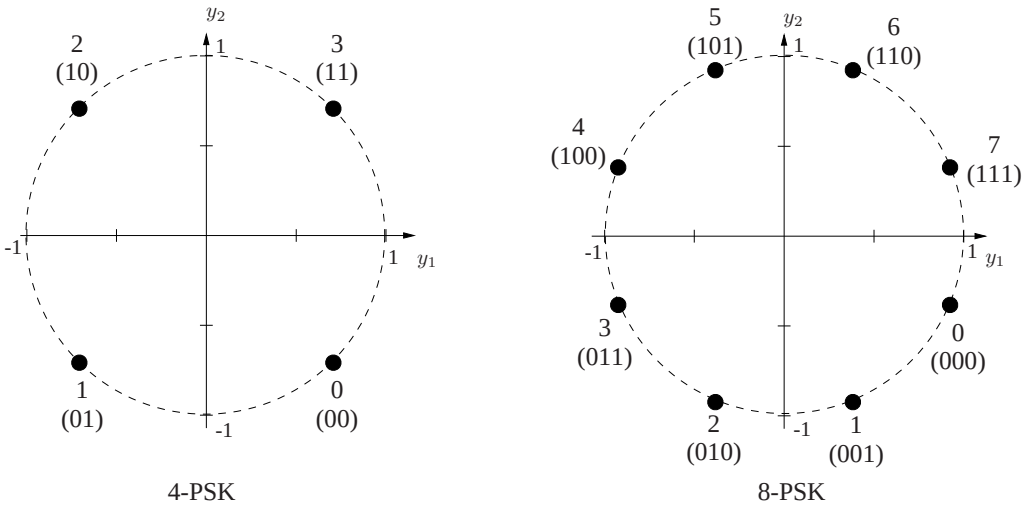
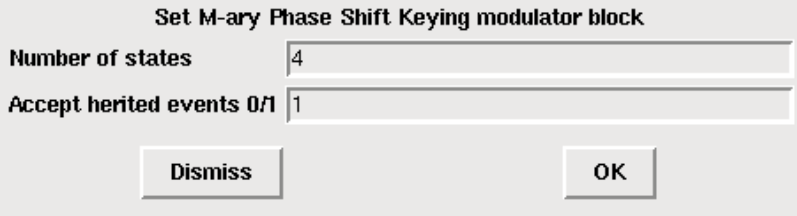


FIG. 1.3 – Constellations 4 et 8-PSK

1.13.2 Boîte de dialogue



– **Number of states :**
  Type 'vec' de taille -1. Nombre d'états de la modulation.
– **Accept herited events 0/1 :**
  Type 'vec' de taille 1. Active la propriété d'héritage par évènement. (supprime le port d'entrée évènementiel)

1.13.3 Fonction de calcul (type 4)

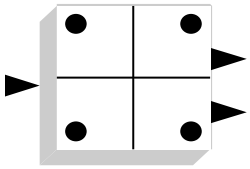
```
/* modpsk Scicos Mary Phase Shift Keying modulator block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 21 décembre 2004 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include "machine.h"
#include <math.h>
#include <stdio.h>
#define M_PI 3.14159265358979323846

/* Cette fonction de simulation réalise un codeur MPSK.
 * Les sorties y1 et y2 correspondent aux valeurs de I et de Q
 * codées en fonction du nombre d'états et du numéro symbole
 * présent à l'entrée u[] aux dates de déclanchements.
 * La fonction calcule :
 * I=cos(%pi*(2*u+1)/m)
 * Q=-sin(%pi*(2*u+1)/m)
 * où u est l'entrée et m le nombre d'états.
 *
 * entrées régulières      : u[0..insz[0]-1]  : vecteur des numéros symboles
 * sorties régulières      : y1[0..insz[0]-1] : vecteur des composantes I
 *                           y2[0..insz[0]-1] : vecteur des composantes Q
 * entrée évènementielles  : instants de déclenchement.
 * sortie évènementielle   : néant
 * paramètres entiers      : ipar[0..insz[0]-1]:m vecteur des nombre d'états
 */

/* modpskv_c routine de calcul des composantes I et Q en fonction
 * d'un numéro symbole codé par états de phase (psk)
 *
 * Entrées :
 * n   : longueur des vecteurs (scalaire)
 * m   : nombre d'état Attention - ici vecteur
 * u   : numéro symbole (vecteur d'entrée)
 * Sorties :
 * i_c : valeur de la composante i (vecteur de sortie 1)
 * q_c : valeur de la composante q (vecteur de sorite 2)
 *
 * Dépendances :
 * math.h
 */
void modpskv_c(int *n,int *m,double *u,double *i_c, double *q_c)
{
 /*Déclaration des variables compteurs*/
 int i;

 for(i=0;i<(*n);i++)
 {
  i_c[i]=cos(M_PI*(2*u[i]+1)/m[i]);
  q_c[i]=-sin(M_PI*(2*u[i]+1)/m[i]);
 }
 return;
}

/*prototype*/
void modpsk(scicos_block *block,int flag)
{
 /*Déclaration des variables*/
 double *y1,*y2;
 double *u;
 int i;

 /*Récupération des adresses des ports réguliers*/
 y1=(double *)block->outptr[0];
 y2=(double *)block->outptr[1];
```

```
 u=(double *)block->inptr[0];

 /*
 Le flag 1 calcule la valeur de I et de Q en fonction
 du numéro symbole u[] et du nombre du nombre d'état m
 */
 if(flag==1||flag==6)
 {
  /*Appel routine modpsk*/
  modpskv_c(&block->insz[0],&block->ipar[0],&u[0],&y1[0],&y2[0]);
 }
}
```

1.13.4 Voir aussi

- DEMODPSK_f - Bloc de démodulation par états de phase M-aires (Bloc Scicos)
- MODQAM_f - Bloc modulateur de Modulation d'Amplitude en Quadrature (Bloc Scicos)
- DEMODQAM_f - Bloc démodulateur de Modulation d'Amplitude en Quadrature (Bloc Scicos)

1.14 MODQAM_f - Bloc modulateur de Modulation d'Amplitude en Quadrature



M − QAM Modulator

- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `MODQAM_f.sci`

1.14.1 Description

Ce bloc modulateur M-QAM calcule une constellation rectangulaire de composantes symbole I/Q à partir de valeurs entières non-signées données par l'entrée régulière et définies dans l'intervalle]0 ;M-1]. M est le paramètre de la boîte de dialogue 'Number of states'. Ce paramètre doit être un entier calculé par :

$$M = 2^k \tag{1.12}$$

où M est le nombre désiré d'états de la modulation et k un nombre entier pair. Les valeurs produites en sortie ne sont pas normalisées. Par exemple, pour une modulation 16-QAM, ce bloc donne la constellation suivante :

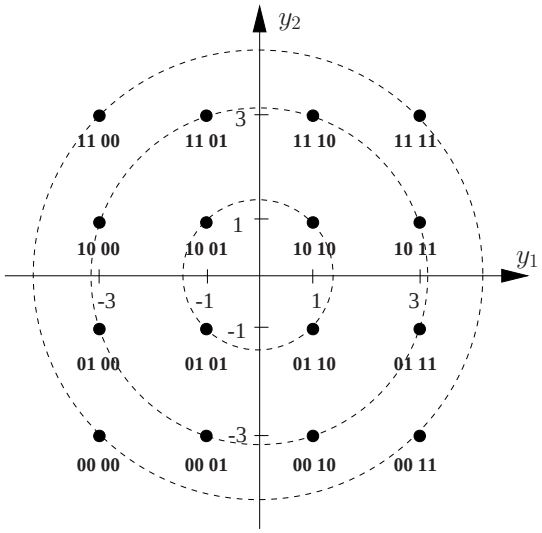
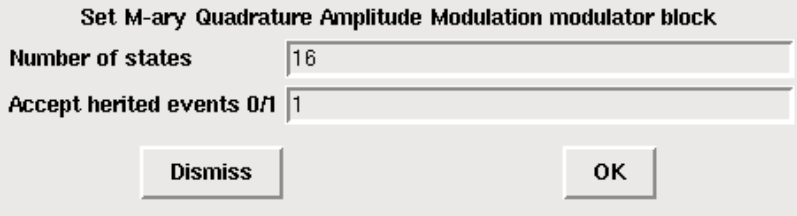


FIG. 1.4 – Constellation 16-QAM

1.14.2 Boîte de dialogue



- **Number of states :**
  Type 'vec' de taille -1. Le nombre d'états de la modulation.
- **Accept herited events 0/1 :**
  Type 'vec' de taille 1. Active la propriété d'héritage par évènement. (supprime le port d'entrée évènementiel)

1.14.3 Fonction de calcul (type 4)

```
/* modqam Scicos QAM modulator block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 5 janvier 2004 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>
#include "scicos_block.h"

/* Cette fonction de simulation réalise un codage type MQAM.
 * Le numéro symbole doit etre définis par un nombre entier non signé dont
 * la valeur est codée par un nombre de bits pair (4 bits, 6 bits, 8 bits,...).
 * Les sélecteurs ng (sélecteur bits de poid faibles) et nd (sélecteurs bits de poid fort)
 * viennent tronquer la valeur du port d'entrée.
 * On associe les valeurs entières des bits de poid faibles tronquées aux valeurs de l'axe
 * I, et on associe les valeurs entières des bits de poids forts tronquées et décalées
 * aux valeurs de l'axe Q.
 *
 * Entrée régulière :   u[0..insz[0]-1] : vecteur des numéros symboles
 * Sorties régulières : y1[0..insz[0]-1]: vecteur des signaux I
 *                      y2[0..insz[0]-1]: vecteur des signaux Q
 * Entrée évènementielle : instants de déclenchement
 * Sortie évènementielle : néant
 * Paramètres entiers : ipar[0..insz[0]-1] : vecteur des nombres de bits.
 *
 * nb : ne fonctionne que pour des modulations avec un nombre de valeurs
 *      identiques pour chaque axes I et Q(16QAM, 64QAM, 256QAM, ...)
 */

/* modqamv_c routine de calcul d'un modulataur mQAM
 *
 * Entrées :
 * n :longueur du vecteur d'entrée
 * m :longueurs des mots binaires en nombre de bits (vecteur)
 * u :adresse de départ du vecteur du symbole en entrée
 *
 * Sortie :
 * i_c : adresse de départ du vecteur de la composante I
 * q_c : adresse de départ du vecteur de la composante Q
 *
 * Dépendances:
 */
void modqamv_c(int *n,int *m,double *u,double *i_c,double *q_c)
{
 /*déclaration des variables compteurs*/
 int i,j;
 int ng,nd;

 for(j=0;j<(*n);j++)
 {
  /*récupération du nombre de bits associé à chaque axes*/
  /*n = block->ipar[j];*/

  /*Calcul des sélecteurs binaires (c'est maladroit!!)*/
  nd=(1<<(m[j])/2)-1;
  ng=(1<<(m[j]))-1-nd;

  /*récupération de la valeur du port d'entrée*/
  i=(int)u[j];

  /*Calcul de la valeur de I et de Q*/
  i_c[j]=((i&nd)*2)-nd;
  q_c[j]=(((i&ng)>>((m[j])/2))*2)-nd;
 }
 return;
}
```

```
/*prototype*/
void modqam(scicos_block *block,int flag)
{
  /*déclaration des variables*/
  double *y1,*y2;
  double *u;
  int i,j,n,nu;
  int nd,ng;

  /*Récupération des adresses des ports réguliers*/
  y1=(double *)block->outptr[0];
  y2=(double *)block->outptr[1];
  u=(double *)block->inptr[0];

  /*récupération du nombre de symbole en entrée*/
  nu=block->insz[0];

  /* Le flag 1 calcule les valeurs de I et de Q
   * suivant le numéro symbole u et suivant le
   * nombre de bits associé à chaque axes
   */

  if(flag==1||flag==6)
  {
   /*Appel modqamv_c*/
   modqamv_c(&nu,&block->ipar[0],&u[0],&y1[0],&y2[0]);
  }
}
```

1.14.4 Voir aussi

- DEMODQAM_f - Bloc démodulateur de Modulation d'Amplitude en Quadrature (Bloc Scicos)
- MODPSK_f - Bloc de modulation par états de phase M-aires (Bloc Scicos)
- DEMODPSK_f - Bloc de démodulation par états de phase M-aires (Bloc Scicos)

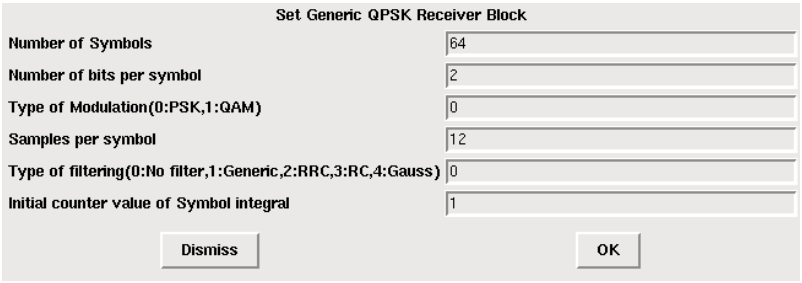
1.15 QPSKREC_f - Bloc récepteur QPSK

- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `QPSKREC_f.sci`

1.15.1 Description

Add here a paragraph of the function description.

1.15.2 Boîte de dialogue



- **Number of Symbols :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Number of bits per symbol :**
  Type 'vec' de taille 1. La description du paramètre 2.
- **Type of Modulation(0 :PSK,1 :QAM) :**
  Type 'vec' de taille 1. La description du paramètre 3.
- **Samples per symbol :**
  Type 'vec' de taille 1. La description du paramètre 4.
- **Type of filtering(0 :No filter,1 :Generic,2 :RRC,3 :RC,4 :Gauss) :**
  Type 'vec' de taille 1. La description du paramètre 5.
- **Initial counter value of Symbol integral :**
  Type 'vec' de taille 1. La description du paramètre 6.

1.15.3 Fonction de calcul (type 4)

```
/* mpskemit Scicos Mary Phase Shift Keying Emmiter
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 22 décembre 2004 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include "machine.h"
#include <math.h>
#include <stdio.h>

/* Cette fonction de simulation réalise un récepteur MPSK
 * La sortie y1 correspond aux valeurs du symbole démodulé.
 * suilant les valeurs I et Q présentes aux l'entrées u1[] et u2[]
 * aux dates de déclanchements.
 *
 * entrées régulières      : u1[0..insz[0]-1]  : vecteur des composantes I
 *                           u2[0..insz[0]-1]  : vecteur des composantes Q
```

```
 * sorties régulières      : y[0..insz[0]-1/nech] : vecteur du symbole démodulé
 *
 * entrée évènementielles  : instants de déclenchement.
 * sortie évènementielle   : néant
 * paramètres entiers      : ipar[0] :nbr de coef du filtre
 *                           ipar[1] :taille du vecteur en puissance de 2 (pour filtre avec fft842)
 *                           ipar[2] :nech nombre d'échantillons
 *                           ipar[3] :inits_c valeur initiale du compteur integrateur
 *                           ipar[4] :initso_c valeur initiale du compteur sous-échantillonneur
 *                           ipar[5] :m nombre d'états
 *
 * paramètres réels        : rpar[0..nb_coef-1] : vecteur de la réponse impulsionnelle
 *                           rpar[nb_coef]      : pas d'intégration de l'intégrateur symbole
 *                           rpar[nb_coef+1]    : amplitude de sortie de l'intégrateur
 *
 * état discret            : z[0..nb_coef-1]         : vecteur du mot mémoire voie I
 *                           z[nb_coef..2*nb_coef-1] : vecteur du mot mémoire voie Q
 *                           z[2*nb_coef]   : mot mémoire inégrateur symbole voie I
 *                           z[2*nb_coef+1] : mot mémoire inégrateur symbole voie Q
 */

/*prototype*/
void mpskrec(scicos_block *block,int flag)
{
 /*Déclaration des variables*/
 double *y;
 double *u1,*u2;
 double *z;
 double step,amplc;
 int nu,ny;
 int i,k;

 int m1,nb_coef,nech,inits_c,initso_c,m;

 double *z__;
 double *zi1_r, *zi1_i, *zi2_r, *zi2_i, *yi_r, *yi_i;
 double *zq1_r, *zq1_i, *zq2_r, *zq2_i, *yq_r, *yq_i;

 /*Récupération des adresses des ports réguliers*/
 y=(double *)block->outptr[0];
 u1=(double *)block->inptr[0];
 u2=(double *)block->inptr[1];
 z=block->z;

 /*Récupération des valeurs des variables*/
 nu=block->insz[0];
 ny=block->outsz[0];
 nb_coef=block->ipar[0];
 m1=block->ipar[1];
 nech=block->ipar[2];
 inits_c=block->ipar[3]; /*Valeur initiale du compteur intégrateur symbole*/
 initso_c=block->ipar[4]; /*Valeur initiale du compteur sous-échantillonneur*/
 m=block->ipar[5];
 step=block->rpar[nb_coef];
 amplc=block->rpar[nb_coef+1];
 /*
 Le flag 1 calcule la valeur de I et de Q en fonction
 du numéro symbole u[] et du nombre du nombre d'état m
 */
 if(flag==1||flag==6)
 {
  /*Allocation de 2*(3*2) vecteurs de taille m1*/
  if ((*block->work=scicos_malloc(sizeof(double)*(m1*2*(2*3))))== NULL)
  {
   set_block_error(-16);
   return;
  }

  /*Récupération de l'adresse de départ du vecteur alloué*/
  z__=*block->work;

  /*Déclaration de pointeurs auxiliaires*/
  zi1_r = &(z__[0])   ;  zi1_i = &(z__[m1]);   /*vecteur voie i du complexe 1*/
  zi2_r = &(z__[2*m1]);  zi2_i = &(z__[3*m1]); /*vecteur voie i du complexe 2*/
  yi_r  = &(z__[4*m1]);  yi_i  = &(z__[5*m1]); /*vecteur voie i du complexe résultat*/
  zq1_r = &(z__[6*m1]);  zq1_i = &(z__[7*m1]); /*vecteur voie q du complexe 1*/
  zq2_r = &(z__[8*m1]);  zq2_i = &(z__[9*m1]); /*vecteur voie q du complexe 2*/
  yq_r  = &(z__[10*m1]); yq_i  = &(z__[11*m1]);/*vecteur voie q du complexe résultat*/

  /*Recope u1[] et u2[] dans zi1_r et zq1_r*/
  F2C(dcopy)(&nu,&u1[0],(k=1,&k),&zi1_r[0],(k=1,&k));
  F2C(dcopy)(&nu,&u2[0],(k=1,&k),&zq1_r[0],(k=1,&k));

  /*Recopie rpar[] dans zi2_r[] et zq2_r*/
  F2C(dcopy)(&nb_coef,&(block->rpar[0]),(k=1,&k),&zi2_r[0],(k=1,&k));
  F2C(dcopy)(&nb_coef,&(block->rpar[0]),(k=1,&k),&zq2_r[0],(k=1,&k));

  /*Appel convolr_c*/
  convolr_c(&nu,&nb_coef,&m1,&zi1_r[0],&zi1_i[0],&zi2_r[0],&zi2_i[0],&yi_r[0],&yi_i[0],&z[0]);
  convolr_c(&nu,&nb_coef,&m1,&zq1_r[0],&zq1_i[0],&zq2_r[0],&zq2_i[0],&yq_r[0],&yq_i[0],&z[nb_coef]);

  /*Appel intsymb_c*/
  intsym_c(&nu,&nech,&inits_c,&step,&yi_r[0],&zi1_r[0],&z[2*nb_coef]);
  intsym_c(&nu,&nech,&inits_c,&step,&yq_r[0],&zi2_r[0],&z[2*nb_coef+1]);

  /*Appel comp_c*/
  comp_c(&nu,&amplc,&zi1_r[0],&yi_r[0]);
  comp_c(&nu,&amplc,&zi2_r[0],&yq_r[0]);

  /*Appel sousecht_c*/
  sousecht_c(&nu,&nech,&initso_c,&yi_r[0],&zi1_r[0]);
```

```
  sousecht_c(&nu,&nech,&initso_c,&yq_r[0],&zi2_r[0]);

  /*Appel demodpsk_c*/
  demodpsk_c(&ny,&m,&zi1_r[0],&zi2_r[0],&y[0]);

  /*Libère mémoire allouée*/
  scicos_free(*block->work);
 }
}
```

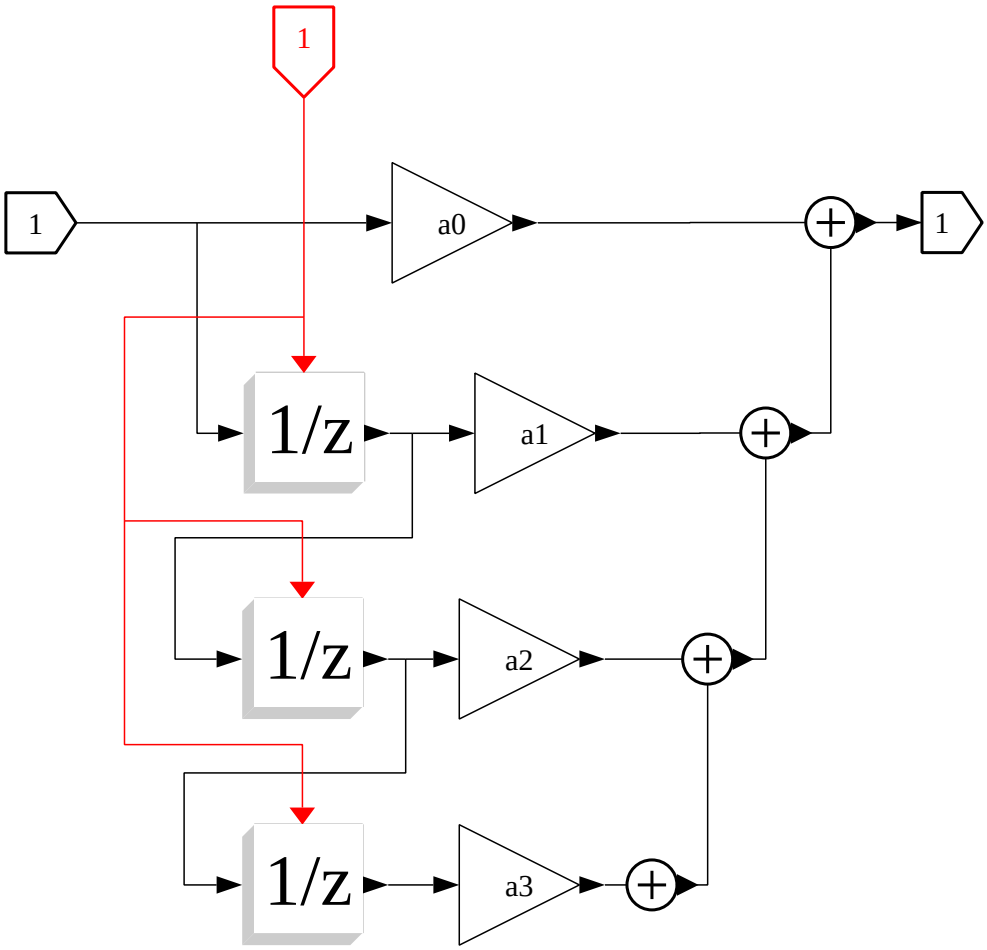
1.16 RIFGEN_f - Bloc filtre à réponse impulsionnelle finie générique

- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `RIFGEN_f.sci`

1.16.1 Description

Add here a paragraph of the function description.

1.16.2 Modèle équivalent en Super Bloc



1.16.3 Boîte de dialogue

- **Type of RIF :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Accepted herited (0/1) ? :**
  Type 'vec' de taille 1. La description du paramètre 2.

1.16.4 Fonction de calcul (type 4)

```
/* nfilter Scicos Temporal RIF block
 * Type 4 simulation function ver 1.0 - scilab-2.6&2.7
 * 13 octobre 2003 - IRCOM GROUP - Author : A.Layec
 * 17 janvier 2005 : passage type 4
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include "scicos_block.h"
#include <stdio.h>

/* Cette fonction de simulation réalise le filtrage temporel
 * par réponse impulsionnelle finie.
 * Elle est capable de réaliser le produit de convolution
 * d'un vecteur d'entrée par un vecteur des réponses impulsionnelles.
 * Pour l'instant, le vecteur des réponses impulsionnelles est de taille fixe (nb_coef)
 * pour chaque filtre mais chaque réponse impulsionnelle peut avoir une forme différente.
 *
 * L'opération réalisée est
 * y[][k]=rpar[][k]*u[][k] où * représente ici le produit de convolution discret.
 * a chaque rythme d'échantillonnage la valeur de u[][k] est stocké dans z[][k]
 * et la valeur de y[][k] est calculée
 *
 * entrées régulières : vecteur des échantillons des signaux d'entrées u[0..Nu-1].
 * sortie régulières : vecteur des échantillons des signaux filtrés y[0..Nu-1].
 * entrée évènementielles : fréquence d'échantillonnage.
 * sorties évènementielles : néant.
 * paramètres réels :
 * rpar[0..nbcoef-1,nbcoef..2*nbcoef-1,...,(Nu-1)*nbcoef-1..Nu*nbcoef-1] : vecteur des réponses impulsionnelles
 * paramètres entiers :
 * ipar[0] : nbcoef : nombre de coefficients de la réponse impulsionnelle
 * insz[0] : Nu : nombre de signaux à filtrer
 * Vecteur Z :
 * z[0..nbcoef-1,nbcoef..2*nbcoef-1,...,(Nu-1)*nbcoef-1..Nu*nbcoef-1] =
 * u[0][k-nbcoef-1],u[0][k-nbcoef-2],..,u[0][k],...,u[Nu-1][k-nbcoef-1],u[Nu-1][k-nbcoef-2],..,u[Nu-1][k]
 */

/*prototype*/
void nfilter(scicos_block *block,int flag)
{
  /*déclaration des variables*/
  double *y;
  double *u;
  int *sz,*nbcoef;
  int i,nu;
  int ptrui,ptrz;
  /*récupération des adresses des ports*/
  y=(double *)block->outptr[0];
  u=(double *)block->inptr[0];

  /*récupération des paramètres*/
  nu=block->ipar[0];
  nbcoef=&block->ipar[1];
  sz=&block->ipar[1+nu];

  /*Le flag 1 réalise le produit de convolution discret*/
  if(flag==1||flag==6)
  {
   ptrui=0;
   ptrz=0;

   for (i=0;i<nu;i++)
   {
    /*fprintf(stderr,"nb_coef[%d]=%d,sz[%d]=%d \n",i,nbcoef[i],i,sz[i]);*/
    /*Appel nfilter*/
    nfilter_c(&sz[i],&nbcoef[i],&u[ptrui],&block->rpar[ptrz],&y[ptrui],&block->z[ptrz]);
    if(nu>1)
    {
     ptrui=sz[i];
     ptrz=nbcoef[i];
    }
   }
  }
}
```

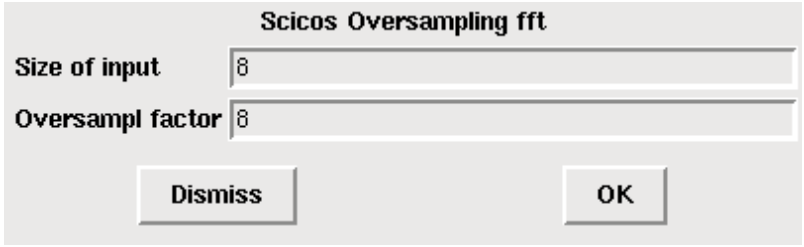
1.17 UPSMPLFFT_f - Bloc élévateur de cadence (OBSOLETE)

- **Palette :** Communication.cosf - Palette Communication
- **Fonction d'interface :** `UPSMPLFFT_f.sci`

1.17.1 Description

Add here a paragraph of the function description.

1.17.2 Boîte de dialogue



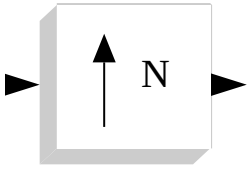
- **Size of input :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Oversampl factor :**
  Type 'vec' de taille 1. La description du paramètre 2.

1.17.3 Fonction de calcul (type 2)

```
/* surech Scicos oversampling by fft
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7&3.0
 * 23 décembre 2003 - IRCOM GROUP - Author : A.Layec
 */

 /* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>

/* Cette fonction réalise le suréchantillonnage par fft. Elle recopie N fois le vecteur
 * d'entrée[u1;u2] dans le vecteur de sortie [y1;y2]
 *
 * entrées réguliéres : u1[0..nu-1] : parties réelles du signal fréquentiel
 *                      u2[0..nu-1] : parties imaginaires du signal fréquenteil
 * sorties régulières : y1[0..ny-1] : parties réelles du signal fréquentiel surechantillonné
 *                      y2[0..ny-1] : parties imaginaires du signal fréquentiel surechantillonné
 * paramètres entiers : ipar[0] : N facteur de surechantillonnage
 *
 * nb : ny=N*nu
 */

/*prototype*/
void surechfft(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
               ipar,nipar,inptr,insz,nin,outptr,outsz,nout)

int *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
  /*Déclaration des variables*/
  double *y1,*y2;
  double *u1,*u2;
  int i,j,N,nu,ny,k;

  /*Récupération des adresses des ports réguliers*/
  y1=(double *)outptr[0];
```

```
  y2=(double *)outptr[1];
  u1=(double *)inptr[0];
  u2=(double *)inptr[1];

  /*Récupération de la taille du port d'entrée*/
  nu=insz[0];
  ny=outsz[0];

  /*Récupération du facteur de suréchantillonnage*/
  N=ipar[0];

  /*Récopie u dans y*/
  for(i=0;i<N;i++)
  {
   for(j=0;j<nu;j++)
   {
    y1[i*nu+j]=u1[j];
    y2[i*nu+j]=u2[j];
   }
  }
}
```

1.18 UPSMPL_f - Bloc élévateur de cadence



– **Palette :** Communication.cosf - Palette Communication
– **Fonction d'interface :** `UPSMPL_f.sci`

1.18.1 Description

Le bloc élévateur de cadence augmente le taux d'échantillonnage du signal d'entrée en répétant les valeurs ou en insérant des zéros. L'augmentation de la cadence du pas d'échantillonnage est déterminée par le paramètre 'Upsample factor' de la boîté de dialogue.

Ce bloc est capable de traiter des signaux d'entrée de type 'sample-based' et 'frame-based' avec le choix des valeurs du paramètre 'Type of upsample' de la boîte de dialogue. Les choix 0 et 1 sont les signaux 'sample-based' et les chois 1 et 2 pour les signaux 'frame-based'.

L'exemple suivant montre les deux types de sortie que peut réaliser ce bloc avec un facteur de sur-échantillonnage de 8.

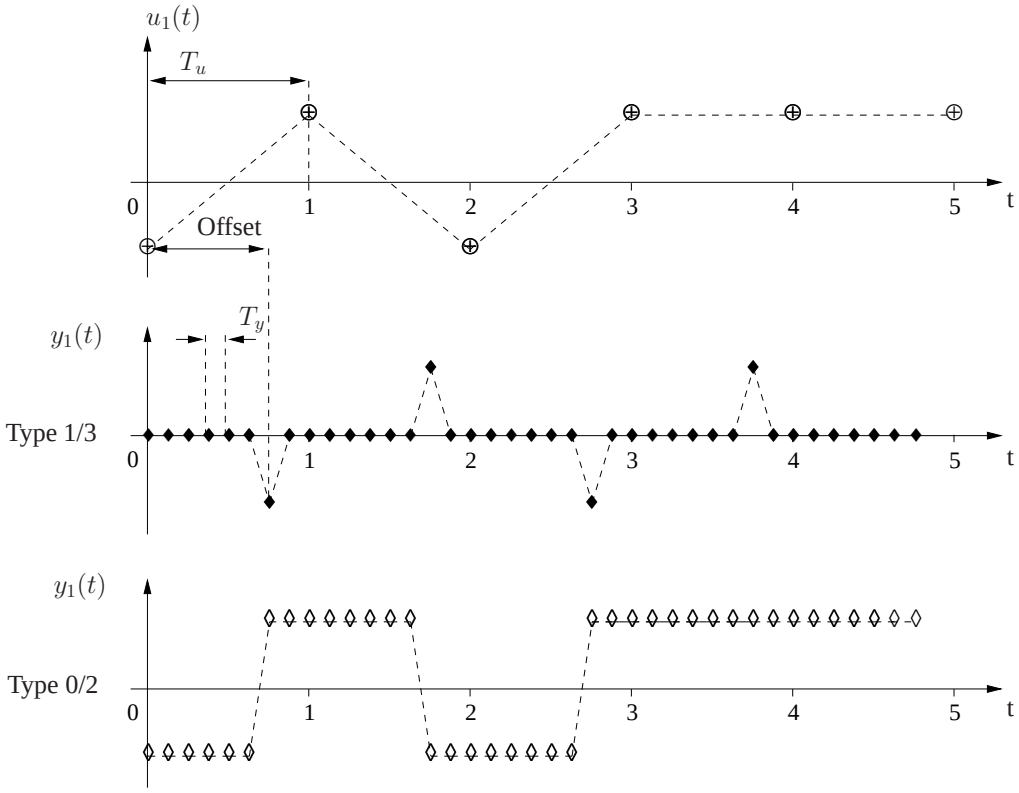
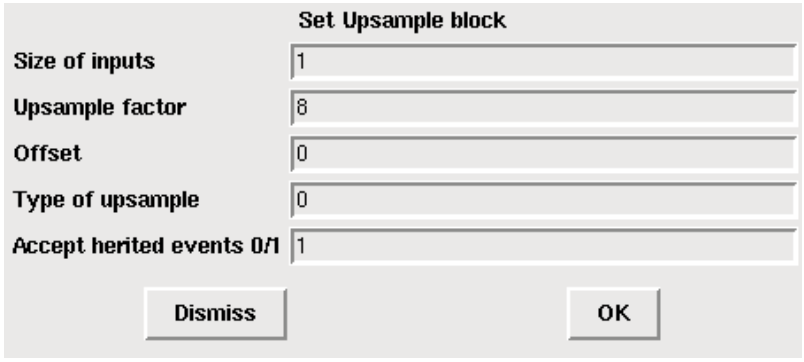


FIG. 1.5 – Entrée et sortie régulière du bloc UPSMPL_f

1.18.2 Boîte de dialogue

– **Size of inputs :**
Type 'vec' de taille 1. La taille du port régulier d'entrée.



- **Upsample factor :**
  Type 'vec' de taille 1. Le facteur de sur-échantillonnage.
- **Offset :**
  Type 'vec' de taille 1. La valeur initiale du compteur discret, donne le décalage du premier échantillon de sortie.
- **Type of upsample :**
  Type 'vec' de taille 1. Le type d'élévation de cadence.
  - **0 :**
    sur-échantillonnage 'sample-based' par répétition des valeurs.
  - **1 :**
    sur-échantillonnage 'sample-based' par insertion de zéros.
  - **2 :**
    sur-échantillonnage 'frame-based' par répétition des valeurs.
  - **3 :**
    sur-échantillonnage 'frame-based' par insertion de zéros.
- **Accept herited events 0/1 :**
  Type 'vec' de taille 1. Active la propriété d'héritage par évènement. (supprime le port d'entrée évènementiel)

1.18.3 Fonction de calcul (type 4)

```
/* surecht Scicos temporal oversamplig and zero insert block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 21 décembre 2004 Author : - IRCOM GROUP - A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include "machine.h"
#include <stdio.h>
/* Entrée régulière : signal à surechantilonner
 * Sortie régulière : signal suréchantilonné
 * Entrée évènementielle : néant
 * Sortie évènementielle : néant
 *
 * Paramètres entier : insz[0] : taille du vecteur en entrée
 *                     ipar[0] : nombre d'échantillons
 *                     ipar[1] : option
 *                               0/1 : traitement séquentiel
 *                               2/3 : traitement vectoriel
 *                               0/2 : pas d'insertion de zéro
 *                               1/3 : insertion zéro
 *
 * état discret     : z[0] : valeur initiale du numéro échantillons
 */

/*prototype*/
void surecht(scicos_block *block,int flag)
{
  /*déclaration des variables*/
  double *y;
  double *u;
  int i,opt,nu,nech,ny;
  int counter;

  /*Récupération des adresses des ports réguliers*/
  y=(double *)block->outptr[0];
  u=(double *)block->inptr[0];
```

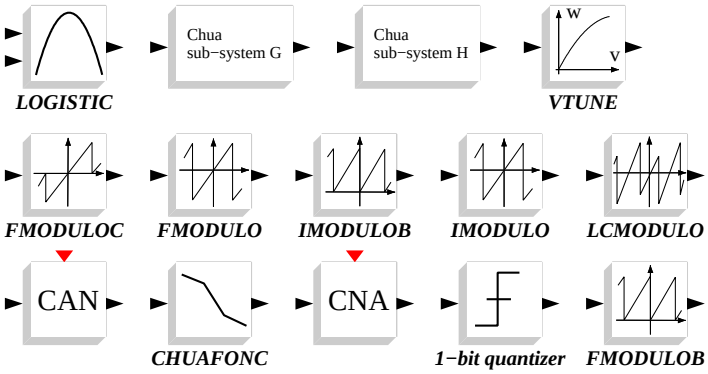
```
  /*récupération du nombre d'échantillons*/
  nu=block->insz[0];
  nech=block->ipar[0];
  ny=block->outsz[0];
  opt=block->ipar[1];

  /* Le flag1 test la valeur du compteur échantillon
   * et délivre u[] dans y[] quand le compteur
   * arrive à la valeur ipar[0]
   */
  if(flag==1)
    {/*Vector*/
     if (opt==2||opt==3)
     {
      opt=opt-2;
      counter=(int)block->z[0];
      /*Appel routine surecht_c*/
      surecht_c(&opt,&nu,&nech,&counter,&u[0],&y[0]);
     }
     /*scalar*/
     else if(opt==0)
     {
      /*si z[]=ipar[0] y[]=u[]*/
      if((int)block->z[0]==nech)
      {
       block->z[0]=1; /*RAZ compteur*/
       for(i=0;i<block->insz[0];i++) y[i]=u[i];
      }
      /*si z[]!=ipar[0] y[]=0*/
      else
      {
       for(i=0;i<block->insz[0];i++) y[i]=0;
       block->z[0]++; /*incrémente compteur*/
      }
     }
     else if(opt==1)
     {
      for(i=0;i<block->insz[0];i++) y[i]=u[i];
     }
    }
}
```

1.18.4 Voir aussi

– DOWNSMPL_f - Bloc réducteur de cadence (Bloc Scicos)

Chapitre 2

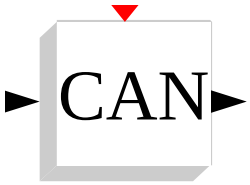
Palette Non-linéaire



2.0.1 Description

Add here a paragraph of the function description.

2.1 CAN_f - Bloc Convertisseur Analogique Numérique

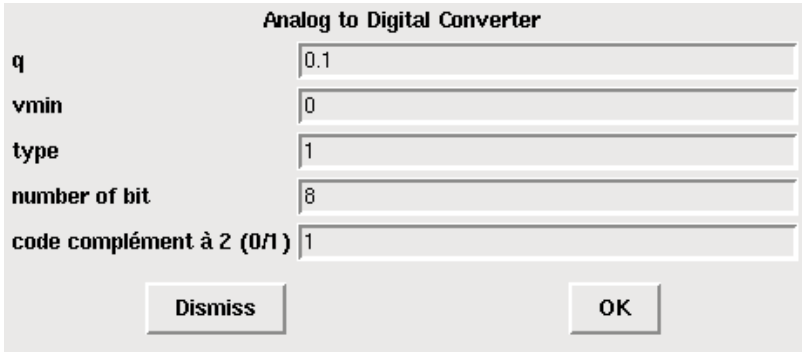


– **Palette :** NonLinear.cosf - Palette Non-linéaire
– **Fonction d'interface :** `CAN_f.sci`

2.1.1 Description

Add here a paragraph of the function description.

2.1.2 Boîte de dialogue



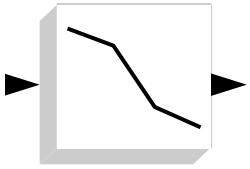
– **q :**
Type 'vec' de taille -1. La description du paramètre 1.
– **vmin :**
Type 'vec' de taille -1. La description du paramètre 2.
– **type :**
Type 'vec' de taille -1. La description du paramètre 3.
– **number of bit :**
Type 'vec' de taille -1. La description du paramètre 4.
– **code complément à 2 (0/1) :**
Type 'vec' de taille -1. La description du paramètre 5.

2.1.3 Fonction de calcul (type 1)

```
c     can.f Analog to Digital Convertor
c     IRCOM Group - A. Layec

c     REVISION HISTORY :
c     $Log$
c
      subroutine can(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,
     &     rpar,nrpar,ipar,nipar,u,nu,y,ny)

      double precision t,xd(*),x(*),z(*),tvec(*),rpar(*),u(*),y(*)
      integer flag,nevprt,nx,nz,ntvec,nrpar,ipar(*)
      integer nipar,nu,ny

      integer i,j
      double precision fsr

      do 15 i=1,nu

         if(ipar(i).eq.1)then

           if (u(i).lt.0.0d0)then
```

```
            y(i)=rpar(i)*(ANINT(u(i)/rpar(i)+0.5d0))
          else
            y(i)=rpar(i)*(ANINT(u(i)/rpar(i)-0.5d0))
          endif

        elseif(ipar(i).eq.2)then
c      qzrnd.f
          if (u(i).lt.0.0d0)then
            y(i)=rpar(i)*(ANINT(u(i)/rpar(i)+0.5d0)-0.5d0)
          else
            y(i)=rpar(i)*(ANINT(u(i)/rpar(i)-0.5d0)+0.5d0)
          endif

        elseif(ipar(i).eq.3) then
c      qzflr.f
          y(i)=rpar(i)*ANINT(u(i)/rpar(i)+0.5d0)

        elseif(ipar(i).eq.4) then
c      qzcel.f
          y(i)=rpar(i)*ANINT(u(i)/rpar(i)-0.5d0)
        endif

 15   continue

      do 20 i=1,nu
       fsr=(2**ipar(i+nu))*rpar(i)
       if(y(i).lt.rpar(nu+i)) then
         y(i)=rpar(nu+i)
       elseif(y(i).gt.(rpar(nu+i)+fsr))then
         y(i)=rpar(nu+i)+fsr
       endif
c      write(6,'(''fsr='',e10.3,'' ipar(i+nu)='',i1)') fsr,ipar(i+nu)
c      write(6,'(''rpar(nu+i)+fsr='',e10.3)') rpar(nu+i)+fsr
 20   continue

      do 30 i=1,nu
       y(i)=aint( (y(i)-rpar(nu+i))/rpar(i))
c      write(6,'(''y(i)='',e10.3,''s= '',i1)') y(i),2**(ipar(i+nu)-1)
       if(ipar(2*nu+i).eq.1)then
         y(i)=y(i)-(2**(ipar(i+nu)-1))
       endif
 30   continue

      end
```

2.2 CHUAFONC_f - Bloc fonction non-linéaire de Chua



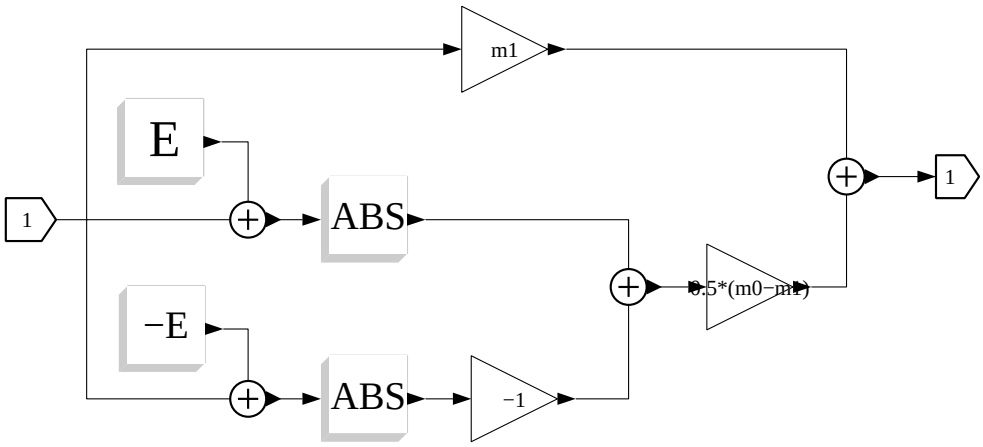
CHUAFONC

– **Palette :** NonLinear.cosf - Palette Non-linéaire
– **Fonction d'interface :** `CHUAFONC_f.sci`

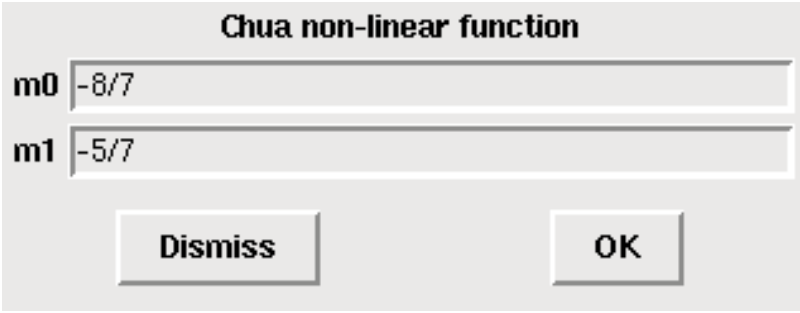
2.2.1 Description

Add here a paragraph of the function description.

2.2.2 Modèle équivalent en Super Bloc



2.2.3 Boîte de dialogue



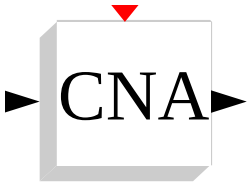
– **m0 :**
Type 'vec' de taille -1. La description du paramètre 1.
– **m1 :**
Type 'vec' de taille -1. La description du paramètre 2.

2.2.4 Fonction de calcul (type 2)

```
/* chuafonc Scicos Chua non-linear function block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 8 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */
```

```
/* REVISION HISTORY :
 * $Log$
 */

#define Abs(x) ( ( (x) >= 0) ? (x) : -( x) )
#include "machine.h"

/* Cette fonction de simulation réalise la fonction non linéaire de la diode de Chua :
 *
 * y=m1*u+1/2*(m0-m1)*(abs(u+E)-abs(u-E))
 *
 * où m0 et m1 sont des vecteurs de paramètres réels,
 * E=1 et u est le vecteurs des entrées, y le vecteur des sorties
 *
 * Entrée régulière : u[0..nu-1] : vecteur des entrées
 * Sortie régulière : y[0..nu-1] : vecteur des sorties
 * Entrée évènementielle : néant (héritage)
 * sortie évènementielle : néant
 * paramètres : rpar[0..nu-1] vecteur m0
 *              rpar[nu..2*nu-1] vecteur m1
 */

/*prototype*/
void
chuafonc(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
            ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
    /*déclaration des variables*/
    double *y;
    double *u;
    int nu,i;

    /*récupération des adresses des ports réguliers*/
    y=(double *)outptr[0];
    u=(double *)inptr[0];

    /*récupération de la taille du port d'entrée u*/
    nu = insz[0];

    /*Calcule du registre de sortie*/
    for(i=0;i<nu;i++)
        y[i]=rpar[nu+i]*u[i]+0.5*(rpar[i]-rpar[nu+i])*(Abs(u[i]+1)-Abs(u[i]-1));
}
```

2.3 CNA_f - Bloc Convertisseur Numérique Analogique

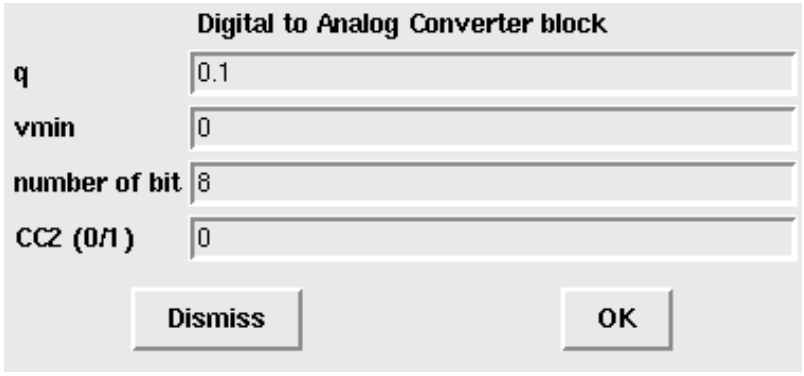


– **Palette :** NonLinear.cosf - Palette Non-linéaire
– **Fonction d'interface :** `CNA_f.sci`

2.3.1 Description

Add here a paragraph of the function description.

2.3.2 Boîte de dialogue

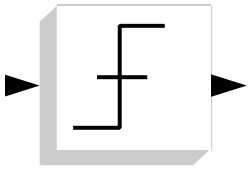


– **q :**
Type 'vec' de taille -1. La description du paramètre 1.
– **vmin :**
Type 'vec' de taille -1. La description du paramètre 2.
– **number of bit :**
Type 'vec' de taille -1. La description du paramètre 3.
– **CC2 (0/1) :**
Type 'vec' de taille -1. La description du paramètre 4.

2.3.3 Fonction de calcul (type 1)

```
c     cna.f Digital to Analog Convertor
c     IRCOM Group - A. Layec

c     REVISION HISTORY :
c     $Log$
c
      subroutine cna(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,
     &     rpar,nrpar,ipar,nipar,u,nu,y,ny)

      double precision t,xd(*),x(*),z(*),tvec(*),rpar(*),u(*),y(*)
      integer flag,nevprt,nx,nz,ntvec,nrpar,ipar(*)
      integer nipar,nu,ny

      integer i,j
      double precision fsr

      do 15 i=1,nu
         if(ipar(i+nu).eq.1) then
            y(i)=rpar(i)*(u(i)+(2**(ipar(i)-1)))+rpar(i+nu)
         else
            y(i)=rpar(i)*u(i)+rpar(i+nu)
         endif
 15   continue

      end
```

2.4 COMP_f - Bloc quantificateur 1 bit

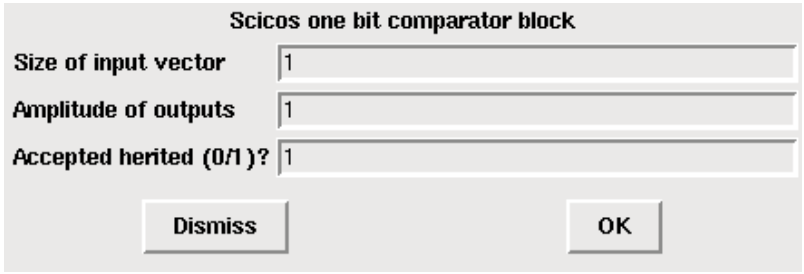


- **Palette :** NonLinear.cosf - Palette Non-linéaire
- **Fonction d'interface :** `COMP_f.sci`

2.4.1 Description

Add here a paragraph of the function description.

2.4.2 Boîte de dialogue



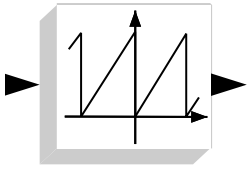
- **Size of input vector :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Amplitude of outputs :**
  Type 'vec' de taille -1. La description du paramètre 2.
- **Accepted herited (0/1) ? :**
  Type 'vec' de taille 1. La description du paramètre 3.

2.4.3 Fonction de calcul (type 4)

```
/* comp Scicos sign comparison block
 * Type 4 simulation function - scilab-3.0
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"

/* Cette fonction de simulation realise la comparaison de signe
 * du vecteur d'entrée. Le vecteur de sortie prend les valeurs
 * +- ampl suivant le signe du vecteur d'entrée.
 * Si le paramètre rpar est un vecteur on appelle compv_c
 * si c'est un scalaire on appelle comp_c.
 *
 * entrée régulière : u[0..nu-1] vecteur d'entrée
 * sortie régulière : y[0..ny-1] vecteur de sortie
 * entrée évènementielle : Heritage ou explicite
 * sortie évènementeille : néant
 *
 * paramètre entier : néant
 * paramètre réel : rpar[0] ou rpar[0..nu-1] amplitude de sortie
 */

/*
 * compv_c routine de calcul de comparaison de signe
 *
 * Entrées :
 * n    : taille des vecteurs
 * amp  : amplitude des sorties (vecteur)
 * u    : vecteur d'entrée
```

```
 * Sorties :
 * y   : vecteur de sortie
 */
void compv_c(int *n,double *ampl,double *u,double *y)
{
 int i;

 for(i=0;i<(*n);i++)
 {
  if(u[i]>=0) y[i]=ampl[i];
  else y[i]=-ampl[i];
 }
 return;
}

/*Prototype*/
void comp(scicos_block *block,int flag)
{
 /*déclaration*/
  double *u,*y;
  int i;
  /*récupération des adresses des ports réguliers*/
  u=(double *)block->inptr[0];
  y=(double *)block->outptr[0];

  if(flag==1)
  {
   if (block->nrpar==1)
    /*Appel comp_c*/
    comp_c(&block->insz[0],&block->rpar[0],&u[0],&y[0]);
   else
    /*Appel compv_c*/
    compv_c(&block->insz[0],&block->rpar[0],&u[0],&y[0]);
  }
}
```

2.5 FMODULOB_f - Bloc fonction modulob réel

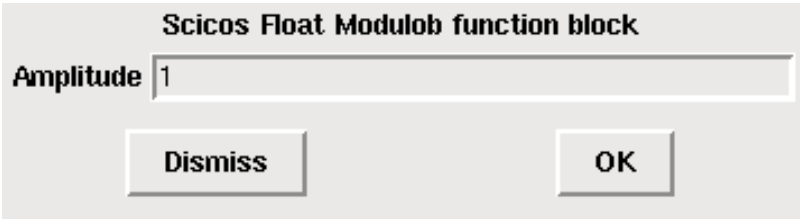


- **Palette :** NonLinear.cosf - Palette Non-linéaire
- **Fonction d'interface :** `FMODULOB_f.sci`

2.5.1 Description

Add here a paragraph of the function description.

2.5.2 Boîte de dialogue



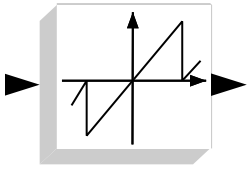
- **Amplitude :**
  Type 'vec' de taille -1. La description du paramètre 1.

2.5.3 Fonction de calcul (type 2)

```
/* fmodulob Scicos float modulob function
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7&3.0
 * 19 octobre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include<math.h>

/* Cette fonction de simulation réalise la fonction non-lineaire
 * modulo sur des nombres définis en virgule flottante.
 * y=u-m*floor(u/m)
 * entrées régulières : vecteur des entrées u[0..nu-1]
 * sorties régulières : vecteur des sorties y[nu..2nu-1]
 * paramètres : rpar[0..nu-1] : m : vecteur des amplitudes
 * entrées d'évenement : néant
 * sorties d'évenement : néant
 *
 * rmq : la sortie est tjs positive
 */

/*prototype*/
void fmodulob(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
              ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
 /*déclaration des variables*/
 int i,nu;
 double *y;
 double *u;

 /*Récupération des adresses des ports réguliers*/
 y=(double *)outptr[0];
 u=(double *)inptr[0];

 /*Récupération de la taille du port d'entrée*/
 nu=insz[0];

 /*Calcul de la valeur modulo et place dans le registre de sortie*/
```

```
 for(i=0;i<nu;i++) y[i] = u[i]-rpar[i]*floor(u[i]/rpar[i]);
}
```

2.6 FMODULOC_f - Bloc fonction moduloc réel



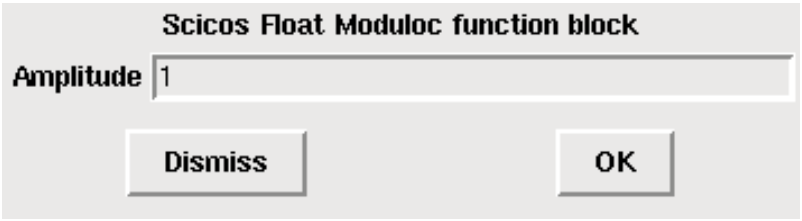
FMODULOC

- **Palette :** NonLinear.cosf - Palette Non-linéaire
- **Fonction d'interface :** `FMODULOC_f.sci`

2.6.1 Description

Add here a paragraph of the function description.

2.6.2 Boîte de dialogue



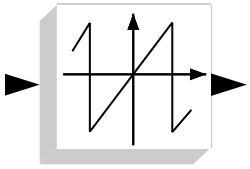
- **Amplitude :**
  Type 'vec' de taille -1. La description du paramètre 1.

2.6.3 Fonction de calcul (type 2)

```
/* fmoduloc Scicos float moduloc function
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7&3.0
 * 19 octobre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"

/* Cette fonction de simulation réalise la fonction non-lineaire
 *  modulo sur des nombres définis en virgule flottante.
 * y=u-m*(int)(u/m)
 * entrées régulières : vecteur des entrées u[0..nu-1]
 * sorties régulières : vecteur des sorties y[nu..2nu-1]
 * paramètres : rpar[0..nu-1] : m : vecteur des amplitudes
 * entrées d'évenement : néant
 * sorties d'évenement : néant
 *
 * la sortie est tjs négative lorsque l'entrée est négative
 * la sortie est tjs positive lorsque l'entrée est positive
 */

/*prototype*/
void fmoduloc(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
              ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
 /*déclaration des variables*/
 int nu,i;
 double *y;
 double *u;

 /*Récupération des adresses des ports réguliers*/
 y=(double *)outptr[0];
 u=(double *)inptr[0];

 /*récupération de la taille du port d'entrée régulier*/
 nu=insz[0];

 /*Calcul de la valeur modulo et place la dans le registre y[]*/
```

```
 for(i=0;i<nu;i++) y[i] = u[i]-rpar[0]*(int)(u[i]/rpar[0]);
}
```

2.7 FMODULO_f - Bloc fonction modulo réel



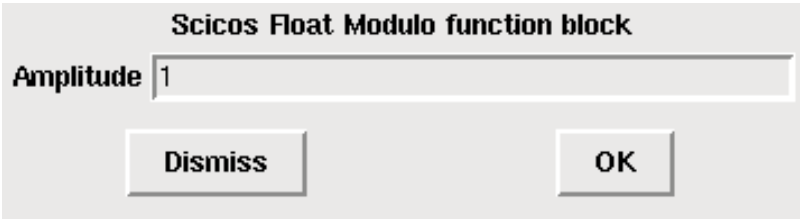
FMODULO

- **Palette :** NonLinear.cosf - Palette Non-linéaire
- **Fonction d'interface :** `FMODULO_f.sci`

2.7.1 Description

Add here a paragraph of the function description.

2.7.2 Boîte de dialogue



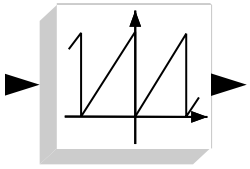
- **Amplitude :**
  Type 'vec' de taille -1. La description du paramètre 1.

2.7.3 Fonction de calcul (type 2)

```
/* fmodulo Scicos float modulo function
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 19 octobre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include<math.h>

/* Cette fonction de simulation réalise la fonction non-lineaire
 * modulo sur des nombres définis en virgule flottante.
 * y=u-0.5*m-m*floor( (u-0.5m) / m)-0.5*m
 * entrées régulières : vecteur des entrées u[0..nu-1]
 *
 * sorties régulières : vecteur des sorties y[nu..2nu-1]
 * paramètres : rpar[0..nu-1] : m : vecteur des amplitudes
 * entrées d'évenement : néant
 * sorties d'évenement : néant
 *
 * rmq : la sortie est soit positive soit négative
 */

/*prototype*/
void fmodulo(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
             ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
 /*déclaration des variables*/
 int i,nu;
 double *y;
 double *u;

 /*Récupération des adresses des ports réguliers*/
 y=(double *)outptr[0];
 u=(double *)inptr[0];

 /*Récupération de la taille du port d'entrée*/
 nu=insz[0];
```

```
 /*Calcul de la valeur modulo et place dans le registre de sortie*/
 for(i=0;i<nu;i++)
      y[i] = (u[i]-0.5*rpar[i])-rpar[i]*floor((u[i]-0.5*rpar[i])/rpar[i])-0.5*rpar[i];
}
```

2.8 IMODULOB_f - Bloc fonction modulob entier



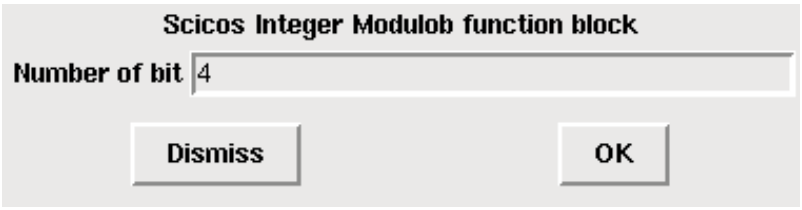
IMODULOB

- **Palette :** NonLinear.cosf - Palette Non-linéaire
- **Fonction d'interface :** `IMODULOB_f.sci`

2.8.1 Description

Add here a paragraph of the function description.

2.8.2 Boîte de dialogue



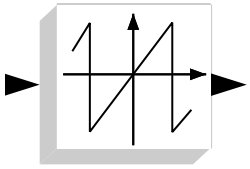
- **Number of bit :**
  Type 'vec' de taille -1. La description du paramètre 1.

2.8.3 Fonction de calcul (type 2)

```
/* imodulob Scicos integer modulob function block
 * Type 2 simulation function - scilab-2.6&2.7&3.0
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>

/* Cette fonction de simulation propose de réaliser la fonction
 * modulo rencontrée dans les systèmes numériques traitant les opérations
 * sur entier en code complément à 2.
 * entrées régulières : u[0..nu-1]
 * sorties régulières : y[0..nu-1] = mod[u[0..nu-1]]
 * paramètres entiers : ipar[0..nu-1] : nombre de bits des mots entiers.
 *
 * Rmq : l'entrée de type double est tronquée à sa valeur entière.
 *           la sortie est tjs positive
 */

/*Prototype*/
void imodulob(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
              ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
    /*Déclaration des variables*/
    int i,nu;
    double *y;
    double *u;
    long ent; /*Déclaration d'un entier de type long long int*/

    /*Récupération des adresses des ports réguliers*/
    y=(double *)outptr[0];
    u=(double *)inptr[0];

    /*Récupération de la taille du port d'entrée*/
    nu=insz[0];

    for(i=0;i<nu;i++)
    {
```

```
     /*Récupération de la valeur d'entrée*/
     ent = (long) u[i];

     /*Conversion en nombre entier non signé*/
     ent &= (2<<(ipar[i]-1)) - 1;

     /*Place nombre converti dans le registre de sortie*/
     y[i] = ent;
    }
}
```

2.9 IMODULO_f - Bloc fonction modulo entier



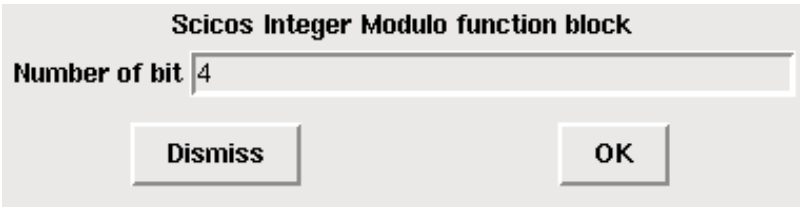
IMODULO

- **Palette :** NonLinear.cosf - Palette Non-linéaire
- **Fonction d'interface :** `IMODULO_f.sci`

2.9.1 Description

Add here a paragraph of the function description.

2.9.2 Boîte de dialogue



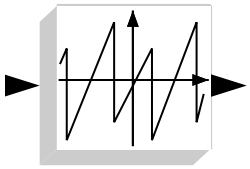
- **Number of bit :**
  Type 'vec' de taille -1. La description du paramètre 1.

2.9.3 Fonction de calcul (type 2)

```
/* imodulo Scicos integer modulo function block
 * Type 2 simulation function - scilab-2.6&2.7&3.0
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>

/* Cette fonction de simulation propose de réaliser la fonction
 * modulo rencontrée dans les systèmes numériques traitant les opérations
 * sur entier en code complément à 2.
 * entrées régulières : u[0..nu-1]
 * sorties régulières : y[0..nu-1] = mod[u[0..nu-1]]
 * paramètres entiers : ipar[0..nu-1] : nombre de bits des mots entiers.
 *
 *Rmq : l'entrée de type double est tronquée à sa valeur entière.
 *          la sortie est soit négative soit positive
 */

/*Prototype*/
void
imodulo(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
        ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
    /*Déclaration des variables*/
    int i,nu;
    double *y;
    double *u;
    long ent; /*Déclaration d'un entier de type long*/

    /*Récupération des adresses des ports réguliers*/
    y=(double *)outptr[0];
    u=(double *)inptr[0];

    /*Récupération de la taille du port d'entrée*/
    nu=insz[0];

    for (i=0;i<nu;i++)
```

```
    {
     /*Récupération de la valeur d'entrée*/
     ent = (long) u[i];

     /*Réalisation du tronquage code complément à 2*/
     ent -= 2<<(ipar[i]-2);
     ent &= (2<<(ipar[i]-1)) - 1;
     ent -= 2<<(ipar[i]-2);

     /*Place valeur tronquée dans le registre de sortie*/
     y[i] = ent;
   }
}
```

2.10 LCMODULO_f - Bloc fonction décalage à gauche modulo entier



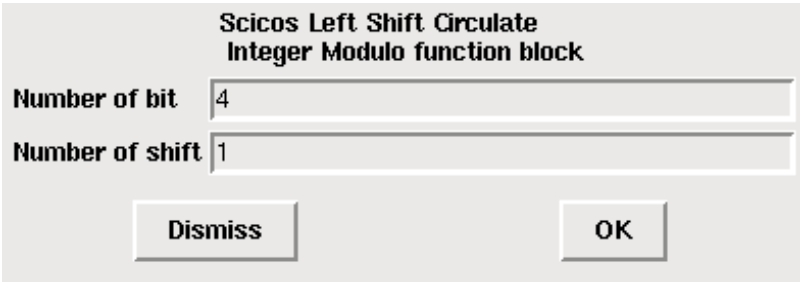
LCMODULO

- **Palette :** NonLinear.cosf - Palette Non-linéaire
- **Fonction d'interface :** `LCMODULO_f.sci`

2.10.1 Description

Add here a paragraph of the function description.

2.10.2 Boîte de dialogue



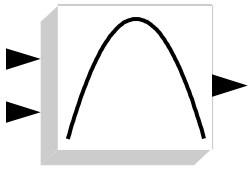
- **Number of bit :**
  Type 'vec' de taille -1. La description du paramètre 1.
- **Number of shift :**
  Type 'vec' de taille -1. La description du paramètre 2.

2.10.3 Fonction de calcul (type 2)

```
/* lcmodulo Scicos left shift circulate integer modulo function block
 * Type 2 simulation function - scilab-2.6&2.7&3.0
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>

/* Cette fonction de simulation propose de réaliser la fonction
 * décalage circulaire à gauche modulo rencontrée dans
 * les systèmes numériques traitant les opérations sur entier en code complément à 2.
 * entrées régulières : u[0..nu-1]
 * sorties régulières : y[0..nu-1] = lcmod[u[0..nu-1]]
 * paramètres entiers : ipar[0..nu-1] : nombre de bits des mots entiers.
 *                                ipar[nu..2*nu-1] : nombre décalage
 *
 * Rmq : l'entrée de type double est tronquée à sa valeur entière.
 *           la sortie est soit négative soit positive
 */

/*Prototype*/
void lcmodulo(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
              ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
    /*Déclaration des variables*/
    int i,j,nu;
    double *y;
    double *u;
    long ent; /*Déclaration d'un entier de type long long int*/
```

```
    /*Récupération des adresses des ports réguliers*/
    y=(double *)outptr[0];
    u=(double *)inptr[0];

    /*Récupération de la taille du port d'entrée*/
    nu=insz[0];

    for (i=0;i<nu;i++)
    {
     /*Récupération de la valeur d'entrée*/
     ent = (long) u[i];

     /*Conversion en nombre non signé*/
     ent &= (2<<(ipar[i]-1)) - 1;
     /*fprintf(stderr,"lcmodulo u=%f, ent=%d\n", u[i],ent);*/

     /*Réalisation de l'opération décalage circulaire à gauche*/
     for(j=0;j<ipar[nu+i];j++)
     {
      if((((1<<(ipar[i]-1))&ent)>>(ipar[i]-1))==1)
         {
          ent <<= 1;
          ent += 1;
         }
      else ent <<=1;
      ent &= (2<<(ipar[i]-1)) - 1;
      /*fprintf(stderr,"lcmodulo j=%d, ent_decal=%d\n",j,ent);*/
     }

     /*Conversion en nombre signé (cc2)*/
     ent -= 2<<(ipar[i]-2);
     ent &= (2<<(ipar[i]-1)) - 1;
     ent -= 2<<(ipar[i]-2);

     /*Place valeur de ent dans le registre de sortie*/
     y[i] = ent;
    }
}
```

2.11 LOGISTIQUE_f - Bloc fonction logistique

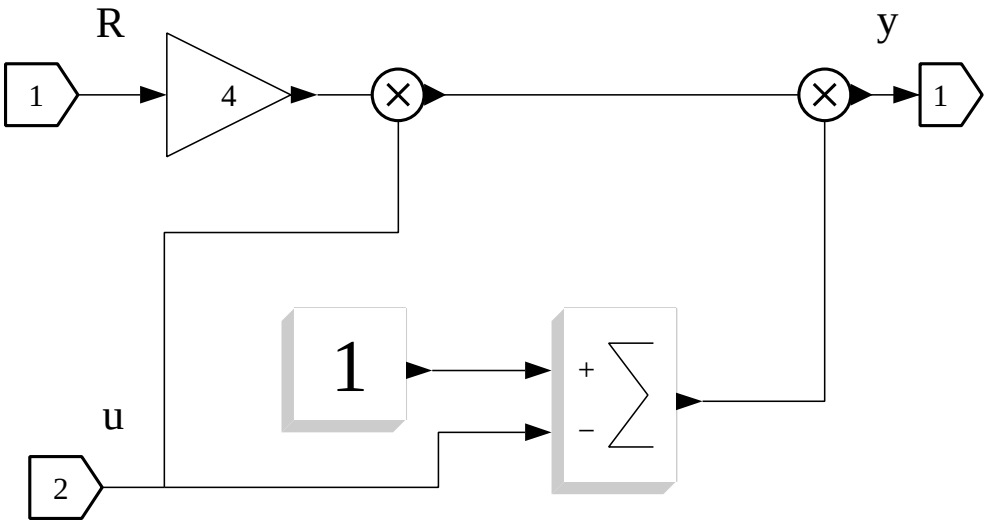


- **Palette :** NonLinear.cosf - Palette Non-linéaire
- **Fonction d'interface :** `LOGISTIQUE_f.sci`

2.11.1 Description

Add here a paragraph of the function description.

2.11.2 Modèle équivalent en Super Bloc

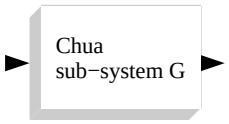


2.11.3 Fonction de calcul (type 2)

```
/* logistique Scicos logistique function block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 18 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"

/* Cette fonction de simulation réalise la fonction non linéaire logistique :
 * y = 4*R*u*(1-u)
 * où R est un paramètre, y la sortie et u l'entrée.
 *
 * Entrées régulières : u1[0..nu-1] : vecteur du paramètre R
 *                      u2[0..nu-1] : vecteur d'entrée
 * Sortie régulière : y1[0..nu-1] : vecteur des sorties
 * Entrée évènementielle : néant (héritage)
 * sortie évènementielle : néant
 * paramètres : néant.
 */

/*prototype*/
void logistique(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
                ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
 /*déclaration des variables*/
 double *y1;
 double *u1;
 double *u2;
 int nu,i;
```

```
 /*récupération des adresses des ports réguliers*/
 y1=(double *)outptr[0];
 u1=(double *)inptr[0];
 u2=(double *)inptr[1];

 /*récupération de la taille du port d'entrée u1*/
 nu = insz[0];

 /*Calcule du registre de sortie*/
 for(i=0;i<nu;i++) y1[i]=4*u1[i]*u2[i]*(1-u2[i]);
}
```

2.12 SYSTEMG_f - Bloc du sous-système G du circuit Chua

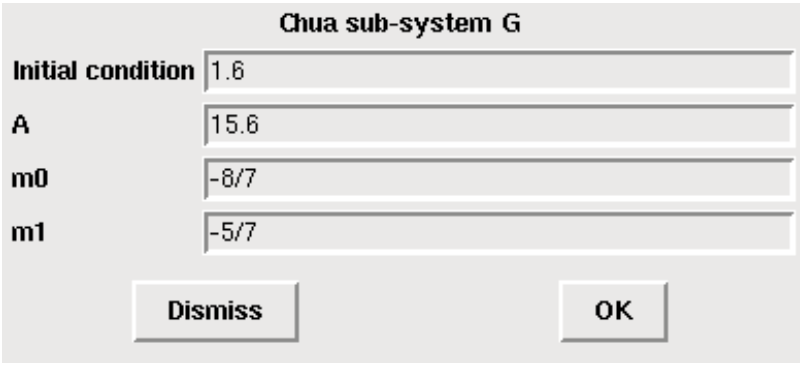


- **Palette :** NonLinear.cosf - Palette Non-linéaire
- **Fonction d'interface :** `SYSTEMG_f.sci`

2.12.1 Description

Add here a paragraph of the function description.

2.12.2 Boîte de dialogue



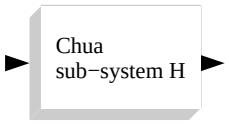
- **Initial condition :**
  Type 'vec' de taille -1. La description du paramètre 1.
- **A :**
  Type 'vec' de taille -1. La description du paramètre 2.
- **m0 :**
  Type 'vec' de taille -1. La description du paramètre 3.
- **m1 :**
  Type 'vec' de taille -1. La description du paramètre 4.

2.12.3 Fonction de calcul (type 2)

```
/* systemg Scicos Chua sub-systeme G block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7&3.0
 * 19 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#define Abs(x) ( ( (x) >= 0) ? (x) : -( x) )
#include "machine.h"

/* Cette fonction de simulation réalise le sous-système G
 * du circuit de Chua:
 *
 * y=x
 * dx/dt=A*((u-x)-Gb*x+1/2*(Ga-Gb)*(abs(u+E)-abs(u-E)))
 *
 * où A, Ga et Gb sont des vecteurs de paramètres réels,
 * E=1 et u est le vecteurs des entrées, y le vecteur des sorties
 * x le vecteur de la variable d'état
 *
 * Entrée régulière : u[0..nu-1] : vecteur des entrées
 * Sortie régulière : y[0..nu-1] : vecteur des sorties
 * Entrée évènementielle : néant (héritage)
 * sortie évènementielle : néant
 * paramètres : rpar[0..nu-1] vecteur A
 *              rpar[nu..2*nu-1] vecteur Ga
```

```
 *              rpar[2*nu..3*nu-1] vecteur Gb
 * etat continu : x[0..nu-1] vecteur de la variable d'état x
 */

/*prototype*/
void systemg(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
             ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
 /*déclaration des variables*/
 double *y;
 double *u;
 int nu,i;

 /*récupération des adresses des ports réguliers*/
 y=(double *)outptr[0];
 u=(double *)inptr[0];

 /*récupération de la taille du port d'entrée u*/
 nu = insz[0];

 /*Le flag 1 place l'état continu x dans le reg y*/
 if(*flag==1||*flag==6)
 {
  for(i=0;i<nu;i++) y[i]=x[i];
 }

 /*Le flag 0 calcule la dérivée de l'état continu*/
 else if(*flag==0)
 {
  for(i=0;i<nu;i++)
    xd[i]=rpar[i]*((u[i]-x[i])-((rpar[2*nu+i]*x[i])+0.5*(rpar[nu+i]-rpar[2*nu+i])*(Abs(x[i]+1)-Abs(x[i]-1))));
 }
}
```

2.13 SYSTEMH_f - Bloc du sous-système H du circuit Chua

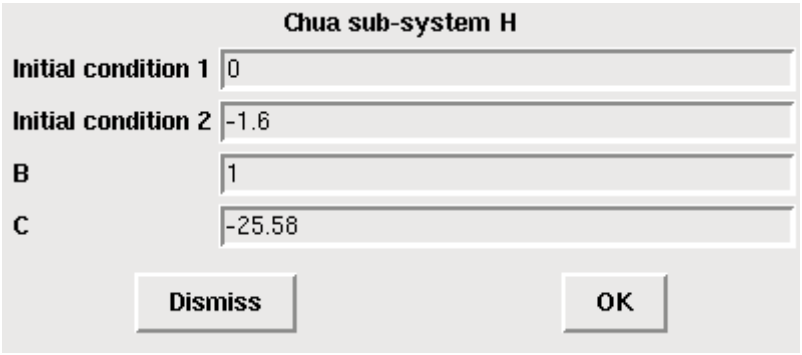


- **Palette :** NonLinear.cosf - Palette Non-linéaire
- **Fonction d'interface :** `SYSTEMH_f.sci`

2.13.1 Description

Add here a paragraph of the function description.

2.13.2 Boîte de dialogue



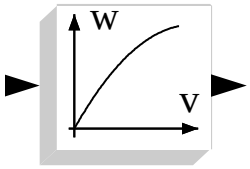
- **Initial condition 1 :**
  Type 'vec' de taille -1. La description du paramètre 1.
- **Initial condition 2 :**
  Type 'vec' de taille -1. La description du paramètre 2.
- **B :**
  Type 'vec' de taille -1. La description du paramètre 3.
- **C :**
  Type 'vec' de taille -1. La description du paramètre 4.

2.13.3 Fonction de calcul (type 2)

```
/* systemh Scicos Chua sub-systeme H block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7&3.0
 * 19 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"

/*Cette fonction de simulation réalise le sous-systeme H
 * du circuit de Chua :
 *
 * y=x1
 *
 * dx1/dt=B*(u-x1+x2)
 * dx2/dt=C*x1
 *
 * où B et C sont des vecteurs de paramètres réels,
 * E=1, u est le vecteurs des entrées, y le vecteur des sorties
 * x1 et x2 les variables d'états
 *
 * Entrée régulière : u[0..nu-1] : vecteur des entrées
 * Sortie régulière : y[0..nu-1] : vecteur des sorties
 * Entrée évènementielle : néant (héritage)
 * sortie évènementielle : néant
 * paramètres : rpar[0..nu-1] vecteur B
```

```
 *              rpar[nu..2*nu-1] vecteur C
 * etat continu : x[0..nu-1] vecteur de la variable d'état x1
 *                x[nu..2*nu-1] vecteur de la variable d'état x2
 */

/*prototype*/
void systemh(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
             ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
 /*déclaration des variables*/
 double *y;
 double *u;
 int nu,i;

 /*récupération des adresses des ports réguliers*/
 y=(double *)outptr[0];
 u=(double *)inptr[0];

 /*récupération de la taille du port d'entrée u*/
 nu = insz[0];

 /*Le flag 1 place l'état continu x dans le reg y*/
 if(*flag==1||*flag==6)
 {
  for(i=0;i<nu;i++) y[i]=x[i];
 }

 /*Le flag 0 calcule la dérivée de l'état continu*/
 else if(*flag==0)
 {
  for(i=0;i<nu;i++)
  {
   xd[i]=rpar[i]*(u[i]-x[i]+x[nu+i]);
   xd[nu+i]=rpar[nu+i]*x[i];
  }
 }
}
```

2.14 VTUNE_f - Bloc caractéristique non-linéaire continue d'OCT

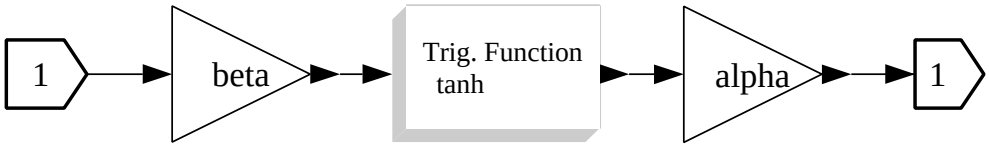


- **Palette :** NonLinear.cosf - Palette Non-linéaire
- **Fonction d'interface :** `VTUNE_f.sci`

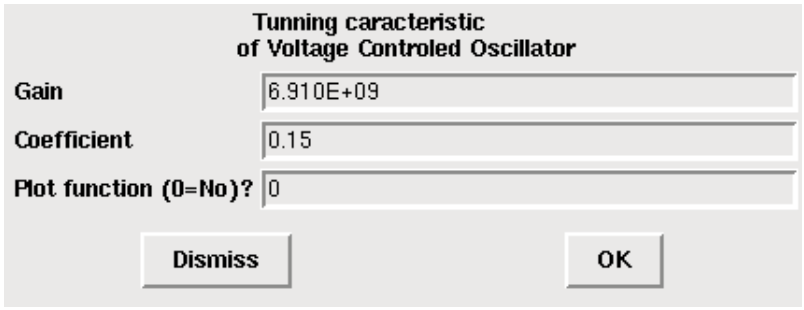
2.14.1 Description

Add here a paragraph of the function description.

2.14.2 Modèle équivalent en Super Bloc



2.14.3 Boîte de dialogue



- **Gain :**
  Type 'vec' de taille -1. La description du paramètre 1.
- **Coefficient :**
  Type 'vec' de taille -1. La description du paramètre 2.
- **Plot function (0=No) ? :**
  Type 'vec' de taille 1. La description du paramètre 3.

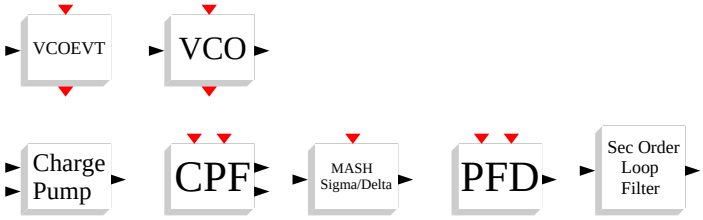
2.14.4 Fonction de calcul (type 2)

```
/* tanhblk Scicos hyperbolic tangent block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 13 octobre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include<math.h>

/* Ce bloc réalise l'opération y[0]=rpar[0]*tanh(rpar[1]*u[0])
 * entrées régulières : vecteur des entrées u[0..nu-1]
 * sorties régulières : vecteurs des sorties y[0..nu-1]
 * entrées et sorties d'évenement : néant
 * paramètres rélls rpar[0..nu-1] : vecteur du gain
 *                 rpar[nu..2*nu-1] : vecteur du coefficient
```

```
 */

/*prototype*/
void tanhblk(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
             ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
 /*déclaration des variables*/
 double *y;
 double *u;
 double t1,t2;
 int i,nu;

 /*récupération des adresses des ports réguliers*/
 y = (double *)outptr[0];
 u = (double *)inptr[0];

 /*récupération de la taille du port d'entrée*/
 nu=insz[0];

 for(i=0;i<nu;i++)
 {
  t1=exp(rpar[nu+i]*u[i]);
  t2=exp(-rpar[nu+i]*u[i]);
  y[i]=rpar[i]*((t1-t2)/(t1+t2));
 }
}
```

Chapitre 3

Palette boucle à verrouillage de phase



3.0.1 Description

Add here a paragraph of the function description.

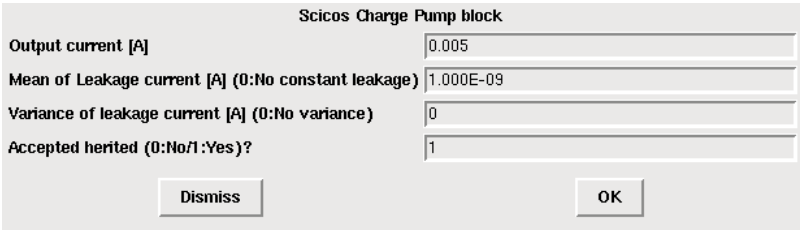
3.1 CHARGEPUMP_f - Bloc Pompe de Charge

- **Palette :** Pll.cosf - Palette boucle à verrouillage de phase
- **Fonction d'interface :** `CHARGEPUMP_f.sci`

3.1.1 Description

Add here a paragraph of the function description.

3.1.2 Boîte de dialogue



- **Output current [A] :**
  Type 'vec' de taille -1. La description du paramètre 1.
- **Mean of Leakage current [A] (0 :No constant leakage) :**
  Type 'vec' de taille -1. La description du paramètre 2.
- **Variance of leakage current [A] (0 :No variance) :**
  Type 'vec' de taille -1. La description du paramètre 3.
- **Accepted herited (0 :No/1 :Yes) ? :**
  Type 'vec' de taille 1. La description du paramètre 4.

3.1.3 Fonction de calcul (type 4)

```
/* pompecharge Scicos Charge Pump block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 9 octobre 2003 - IRCOM GROUP - Author : A.Layec
 * 10 janvier 2005 : passage type 4
 *                   rajout du bruit sur le courant de fuite
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include "scicos_block.h"
#include <math.h>
#include <stdio.h>
#include <stdlib.h>  /*pour RAND_MAX*/

/* Cette fonction de simulation réalise l'opération d'une pompe de charge
 *
 * entrées régulières : u[0..nu-1] : Vecteur des Signaux discrets Up(t)
 *                      u[nu..2nu-1] : Vecteur des signaux discrets Down(t)
 * sortie régulière : y[0..nu] : Vecteur des signaux discrets Icp(t)
 *
 * entrées d'événements : nev=0 à la rigueur (Ref(t) union Div(t))
 *                        sinon néant (héritage)
 *
 * sorties d'évènement : néant
 * paramètres  entiers : outsz[0] : taille des vecteurs (nu)
 *                       ipar[0]  : 0=pas de bruit sur le courant de fuite
 *                                  1=fuite constante
 *                                  2=bruit normal sur le courant de fuite
 * paramètres réels   : rpar[0..nu-1]      : courant max de la pompe de charge
 *                      rpar[nu..2nu-1]    : courant de fuite moyen de la pompe de charge
```

```
 *                     rpar[2*nu..3*nu-1] : deviation du courant de fuite (si ipar[0]=1)
 */

/*prototype*/
void pompecharge(scicos_block *block,int flag)
{
 /*déclaration*/
 double *y;
 double *up,*down;
 int nu;
 int typ_leak;
 double *Io,*Ileak_d,*Ileak_m;
 /*fprintf(stderr,"flag=%d\n",flag);*/
 /*Récupération de l'adresse des ports d'entrée et de sortie*/
 y=(double *)block->outptr[0];
 up=(double *)block->inptr[0];
 down=(double *)block->inptr[1];

 /*Stock taille de la sortie dans nu*/
 nu=block->outsz[0];

 /*Récupération des paramètres*/
 typ_leak=block->ipar[0];
 Io=&block->rpar[0];
 Ileak_m=&block->rpar[nu];
 Ileak_d=&block->rpar[2*nu];

 /*Appel chargepump_c*/
 chargepump_c(&nu,&up[0],&down[0],&Io[0],&typ_leak,&Ileak_m[0],&Ileak_d[0],&y[0]);
}
```

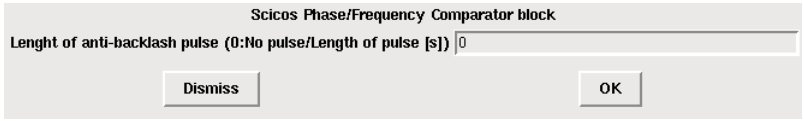
3.2 CPF_f - Bloc comparateur pahse/fréquence trois états

- **Palette :** Pll.cosf - Palette boucle à verrouillage de phase
- **Fonction d'interface :** `CPF_f.sci`

3.2.1 Description

Add here a paragraph of the function description.

3.2.2 Boîte de dialogue



- **Lenght of anti-backlash pulse (0 :No pulse/Length of pulse [s]) :**
  Type 'vec' de taille 1. La description du paramètre 1.

3.2.3 Fonction de calcul (type 4)

```
/* cpf Scicos Phase/frequency Comparator block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 6 janvier 2005 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include <stdio.h>

/* Cette fonction de simulation réalise un comparateur phase fréquence
 * trois états. Elle calcule les signaux discrets Up(t) et Down(t) qui
 * sont les sorties logiques de deux bascules D. Les entrées de ces bascules
 * sont placées à 1 et les entrées d'horloges de celles-ci sont les entrées
 * évènementielles du bloc. Lorsque les sorties des deux bascules sont toutes
 * les deux placées à 1, la fonction place les sorties du bloc y[0] et y[1] à 0.
 *
 * entrées d'évenements : dates d'événements Ref(t), Div(t)
 * sortie d'évenement  : néant
 * entrée régulières : néant
 * Sortie régulières : Up(t) et Down(t)
 * paramètres : néant
 * Etats discrets : z[0] et z[1] mémorisent les états précédents des sorties des bascules
 */

/*prototype*/
void cpf(scicos_block *block,int flag)
{
 /*déclaration*/
 int k;
 int nev;
 double *y1, *y2;

 /*récupération des adresses*/
 y1=(double *)block->outptr[0];
 y2=(double *)block->outptr[1];
 nev=block->nevprt;

 /* Le flag 1 met à jour les sorties un test est réalisé sur les états précédents*/
 if(flag==1)
 {
  /*Appel cpf_c*/
  cpf_c((k=1,&k),&nev,&block->z[0],&block->z[1],&y1[0],&y2[0]);
 }
}
```

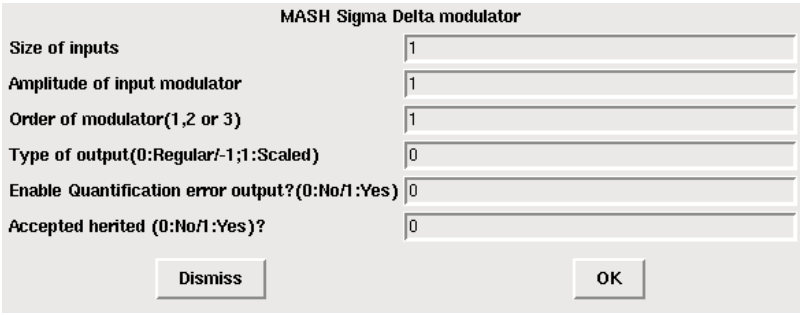
3.3 MASHBLK_f - Bloc modulateur Sigma-Delta

– **Palette :** Pll.cosf - Palette boucle à verrouillage de phase
– **Fonction d'interface :** `MASHBLK_f.sci`

3.3.1 Description

Add here a paragraph of the function description.

3.3.2 Boîte de dialogue



– **Size of inputs :**
  Type 'vec' de taille 1. La description du paramètre 1.
– **Amplitude of input modulator :**
  Type 'vec' de taille 1. La description du paramètre 2.
– **Order of modulator(1,2 or 3) :**
  Type 'vec' de taille 1. La description du paramètre 3.
– **Type of output(0 :Regular/-1 ;1 :Scaled) :**
  Type 'vec' de taille 1. La description du paramètre 4.
– **Enable Quantification error output ?(0 :No/1 :Yes) :**
  Type 'vec' de taille 1. La description du paramètre 5.
– **Accepted herited (0 :No/1 :Yes) ? :**
  Type 'vec' de taille 1. La description du paramètre 6.

3.3.3 Fonction de calcul (type 4)

```
/* mash Scicos discret first order sigma delta modulator 1 bit block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 26 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include "scicos_block.h"
#include<stdio.h>

/* Ce bloc est un modulateur sigma delta (du premier au troisième  ordre)
 * discret avec un comparateur de signe (quantificateur 1 bit)
 * A chaque coup d'horloge le signal d'erreur est évaluée et la sortie est
 * calculée. L'intégrateur discret est réalisé par la méthode des dérivées
 * à gauche du premier ordre. Un éventuel gain est à dispostion pour effectuer
 * une comparaison avec un signal d'entrée de niveau supérieur à 1. Une deuxième
 * sortie est disponbile (bruit de quantification) pour réaliser la
 * mise en cascade des modulateurs.
```

```
 *
 * entrée régulières : u[0..nu-1] : vecteur du signal à moduler
 * sortie régulières : y1[0..nu-1] : vecteur du signal de sortie (+1 ou -1)
 *                     y2[0..nu-1] : bruit de quantification (si ipar[0]==1)
 *
 * paramètre réel rpar[0..nu-1] : vecteur du gain éventuel
 * paramètres entiers : ipar[0] : typ de sortie (0: sans bruit, 1:avec bruit)
 *                      ipar[1] : odre du modulateur (1,2 ou 3)
 *                      ipar[2] : typ de sortie (0:régulière, 1: diminuée)
 *
 * etats discret:
 * 1er ordre
 *              z[0..nu-1] : intégrale du signal d'erreur du modulateur 1
 *              z[nu..2*nu-1] : signal de sortie du modulateur 1
 * 2eme ordre
 *              z[0..nu-1]      : integral du sgnal d'erreur du modulateur 1
 *              z[nu..2*nu-1]   : integral du signal d'erreur du modulateur 2
 *              z[2*nu..3*nu-1] : signal de sortie du modulateur 1
 *              z[3*nu..4*nu-1] : signal de sortie du modulateur 2
 *              z[4*nu..5*nu-1] : etat mémoire du modulateur du 2eme ordre
 * 3eme ordre
 *              z[0..nu-1]      : integral du sgnal d'erreur du modulateur 1
 *              z[nu..2*nu-1]  : integral du signal d'erreur du modulateur 2
 *              z[2*nu..3*nu-1]  : integral du signal d'erreur du modulateur 3
 *              z[3*nu..4*nu-1] : signal de sortie du modulateur 1
 *              z[4*nu..5*nu-1] : signal de sortie du modulateur 2
 *              z[5*nu..6*nu-1] : etat mémoire du modulateur du 2eme ordre
 *              z[6*nu..7*nu-1] : signal de sortie du modulateur 3
 *              z[7*nu..8*nu-1] : etat mémoire 1 du modulateur du 3eme ordre
 *              z[8*nu..9*nu-1]: etat mémoire 2 du modulateur du 3eme ordre
 * vecteur de travail dyna. :
 * si order==1 et si ipar[0]=0
 * z__[0..nu-1] : bruit de quantification
 *
 * si order>1 et ipar[0]==0
 * z__[0..nu-1] : bruit de quantification
 * z__[nu..2*nu-1] : vecteur de travail pour mash2_c et mash3_c
 *
 * si order>1 et ipar[0]==0
 * z__[0..nu-1] : vecteur de travail pour mash2_c et mash3_c
 *
 */

/*prototype*/
void mash(scicos_block *block,int flag)
{
  /*déclaration des variables*/
  int nu,i,typ_out,typ,order;
  double *y1;
  double *y2;
  double *w;
  double *u;
  double m;
  double *z__;

  /*Récupération des adresses des ports réguliers*/
  y1=(double *)block->outptr[0];
  u=(double *)block->inptr[0];

  /*Récupération des paramètres*/
  nu=block->insz[0];
  m=block->rpar[0];
  typ=block->ipar[0];
  order=block->ipar[1];
  typ_out=block->ipar[2];

  if(flag==1)
  {
   if (typ==1) y2=(double *)block->outptr[1]; /*avec sortie de bruit*/
   /* alloue le workspace si nécessaire*/
   if((order==1)&&(typ==0))
   {
    if ((*block->work=scicos_malloc(sizeof(double)*nu))== NULL)
     {
      set_block_error(-16);
      return;
     }
     z__=*block->work;
     y2 = &(z__[0]);
   }

   if((order>1)&&(typ==0))
   {
     if ((*block->work=scicos_malloc(sizeof(double)*2*nu))== NULL)
     {
      set_block_error(-16);
      return;
     }
     z__=*block->work;
     y2 = &(z__[0]);
     w = &(z__[nu]);
   }

   if((order>1)&&(typ==1))
   {
    if ((*block->work=scicos_malloc(sizeof(double)*nu))== NULL)
     {
      set_block_error(-16);
      return;
     }
     z__=*block->work;
```

```
    w = &(z__[0]);
   }

   /*Appel Mash1*/
   if(order==1) mash1_c(&nu,&m,&u[0],&block->z[0],&block->z[1],&y1[0],&y2[0]);

   /*Appel Mash2*/
   else if(order==2) mash2_c(&nu,&m,&u[0],&block->z[0],&block->z[2],&y1[0],&y2[0],&w[0]);

   /*Appel Mash3*/
   else if(order==3) mash3_c(&nu,&m,&u[0],&block->z[0],&block->z[3],&y1[0],&y2[0],&w[0]);

   /*remet en forme le vecteur de sortie si nécéssaire*/
   if(typ_out!=0) for(i=0;i<nu;i++) y1[i]=(y1[i]+(typ_out))/2;

   /*libére le workspace*/
   if((order!=1)||((order==1)&&(typ==0))) scicos_free(*block->work);
  }
}
```

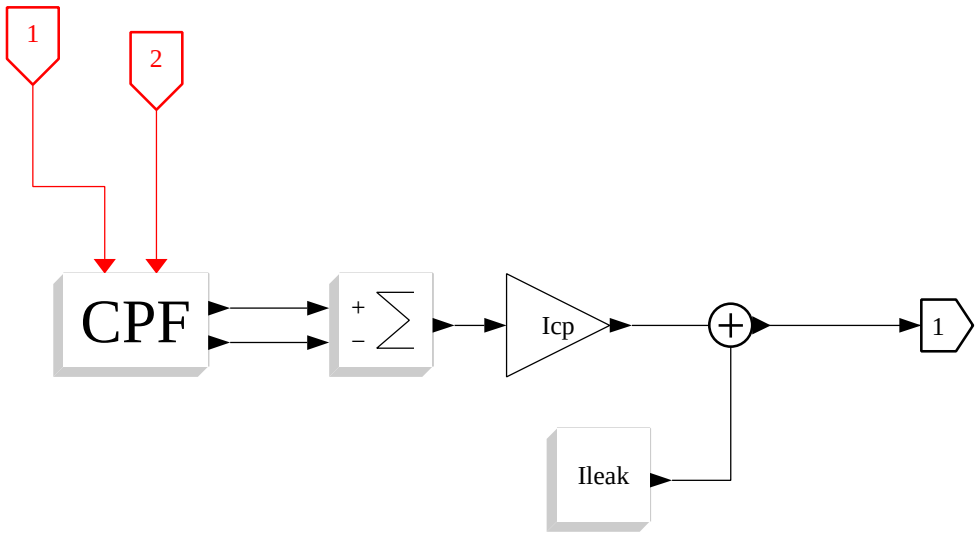
3.4 PFD_f - Bloc Détecteur phase/fréquence trois états

– **Palette :** Pll.cosf - Palette boucle à verrouillage de phase
– **Fonction d'interface :** `PFD_f.sci`

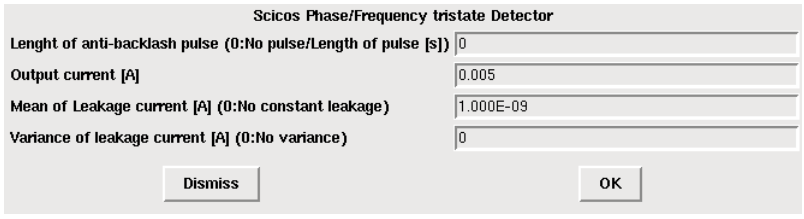
3.4.1 Description

Add here a paragraph of the function description.

3.4.2 Modèle équivalent en Super Bloc



3.4.3 Boîte de dialogue

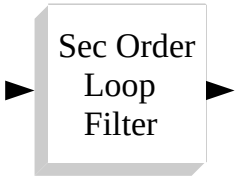


– **Lenght of anti-backlash pulse (0 :No pulse/Length of pulse [s]) :**
Type 'vec' de taille -1. La description du paramètre 1.
– **Output current [A] :**
Type 'vec' de taille -1. La description du paramètre 2.
– **Mean of Leakage current [A] (0 :No constant leakage) :**
Type 'vec' de taille -1. La description du paramètre 3.
– **Variance of leakage current [A] (0 :No variance) :**
Type 'vec' de taille -1. La description du paramètre 4.

3.4.4 Fonction de calcul (type 4)

```
/* pfd Scicos Phase Frequency Detector block including
 * D Flip/Flop Tristate Phase/Frequency comparator and charge Pump
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 13 janvier 2004 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include <stdio.h>

/* entrées régulières      : néant
 * sorties : y[0..ny-1]    : vecteur de sortie du courant de la pompe de charge
 * entrées évènementielles : nev = 1 : entrée d'horloge de mise à jour du cpf
 *                           nev = 2 : entrée d'horloge du diviseur de retour
 * sortie évènementielles  :
 * paramètres entiers : outsz[0] : taille du vecteur (ny)
 *                      ipar[0..ny] : type de bruit de la pompe de charge (typ_leak)
 *                              0: pas de fuite
 *                              1: fuite constante
 *                              2: fuite bruit normal
 * paramètres réels   : rpar[0..ny-1]      : courant max de la pompe de charge
 *                      rpar[ny..2ny-1]    : courant de fuite moyen de la pompe de charge
 *                      rpar[2*ny..3*ny-1] : deviation du courant de fuite
 * Etats discrets : z[0],z[ny] : etats mémoires des bascules
 * Etats discrets dynamiques : up et down valeur de sortie du CPF
 */

/*prototype*/
void pfd(scicos_block *block,int flag)
{
 /*déclaration*/
 int ny;
 int nev;
 double *y;
 double *z__;
 double *up,*down;
 int typ_leak;
 double *Io,*Ileak_d,*Ileak_m;

 /*récupération des adresses*/
 y=(double *)block->outptr[0];
 nev=block->nevprt;

 /*Récupération des paramètres*/
 ny=block->outsz[0];
 typ_leak=block->ipar[0];
 Io=&block->rpar[0];
 Ileak_m=&block->rpar[ny];
 Ileak_d=&block->rpar[2*ny];

 if(flag==1)
 {
 /*Allocation de 2 vecteurs de taille ny*/
 if ((*block->work=scicos_malloc(sizeof(double)*(2*ny)))== NULL)
 {
  set_block_error(-16);
  return;
 }
 /*Récupération de l'adresse de départ du vecteur alloué*/
 z__=*block->work;

 /*Déclaration de pointeurs auxiliaires*/
 up=&(z__[0]);down=&(z__[ny]);   /*vecteur up et down*/

 /*Appel cpf_c*/
 cpf_c(&ny,&nev,&block->z[0],&block->z[ny],&up[0],&down[0]);

 /*Appel chargepump_c*/
 chargepump_c(&ny,&up[0],&down[0],&Io[0],&typ_leak,&Ileak_m[0],&Ileak_d[0],&y[0]);

 /*Libère mémoire allouée*/
 scicos_free(*block->work);
 }
}
```

3.5 SOLOOPFILTER_f - Bloc filtre de boucle pour PLL de type 2

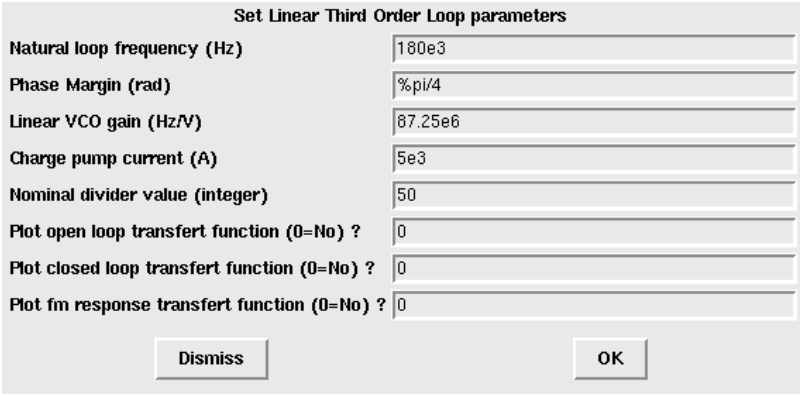


– **Palette :** Pll.cosf - Palette boucle à verrouillage de phase
– **Fonction d'interface :** `SOLOOPFILTER_f.sci`

3.5.1 Description

Add here a paragraph of the function description.

3.5.2 Boîte de dialogue



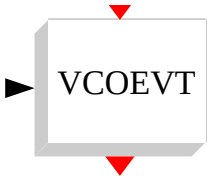
– **Natural loop frequency (Hz) :**
Type 'vec' de taille 1. La description du paramètre 1.
– **Phase Margin (rad) :**
Type 'vec' de taille 1. La description du paramètre 2.
– **Linear VCO gain (Hz/V) :**
Type 'vec' de taille 1. La description du paramètre 3.
– **Charge pump current (A) :**
Type 'vec' de taille 1. La description du paramètre 4.
– **Nominal divider value (integer) :**
Type 'vec' de taille 1. La description du paramètre 5.
– **Plot open loop transfert function (0=No) ? :**
Type 'vec' de taille 1. La description du paramètre 6.
– **Plot closed loop transfert function (0=No) ? :**
Type 'vec' de taille 1. La description du paramètre 7.
– **Plot fm response transfert function (0=No) ? :**
Type 'vec' de taille 1. La description du paramètre 8.

3.5.3 Fonction de calcul (type 1)

```
      subroutine csslti(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,
     &     rpar,nrpar,ipar,nipar,u,nu,y,ny)
c     Copyright INRIA
c     Scicos block simulator
c     continuous state space linear system simulator
c     rpar(1:nx*nx)=A
c     rpar(nx*nx+1:nx*nx+nx*nu)=B
```

```
c     rpar(nx*nx+nx*nu+1:nx*nx+nx*nu+nx*ny)=C
c     rpar(nx*nx+nx*nu+nx*ny+1:nx*nx+nx*nu+nx*ny+ny*nu)=D
c
      double precision t,xd(*),x(*),z(*),tvec(*),rpar(*),u(*),y(*)
      integer flag,nevprt,nx,nz,ntvec,nrpar,ipar(*)
      integer nipar,nu,ny

c
      la=1
      lb=nx*nx+la
      lc=lb+nx*nu

      if(flag.eq.1.or.flag.eq.6) then
c     y=c*x+d*u
         ld=lc+nx*ny
         call dmmul(rpar(lc),ny,x,nx,y,ny,ny,nx,1)
         call dmmul1(rpar(ld),ny,u,nu,y,ny,ny,nu,1)
c         if(t.gt.64.0) write(6,'(e15.8,10(e10.3,x))') t,x(1),x(2),
c     $        u(1),y(1)
      elseif(flag.eq.0) then
c     xd=a*x+b*u
         call dmmul(rpar(la),nx,x,nx,xd,nx,nx,nx,1)
         call dmmul1(rpar(lb),nx,u,nu,xd,nx,nx,nu,1)
c         if(t.gt.64.0) write(6,'(e15.8,10(e10.3,x))') t,x(1),x(2),
c     $        xd(1),xd(2),u(1)
      endif
      return
      end
```

3.6 VCOEVT_f - Bloc passage à zéro discret

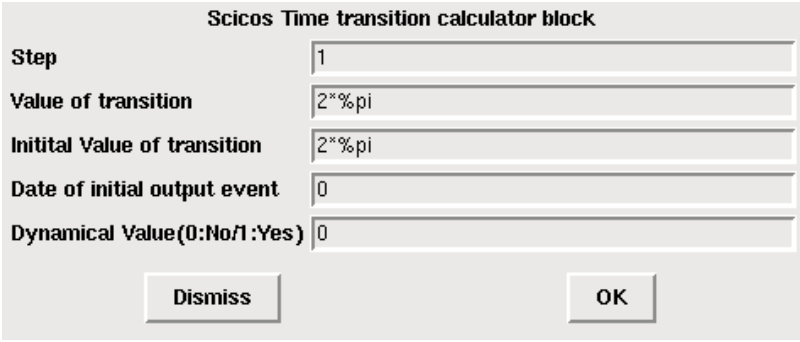


- **Palette :** Pll.cosf - Palette boucle à verrouillage de phase
- **Fonction d'interface :** `VCOEVT_f.sci`

3.6.1 Description

Add here a paragraph of the function description.

3.6.2 Boîte de dialogue



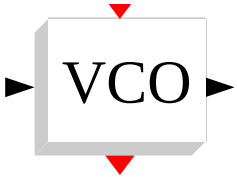
- **Step :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Value of transition :**
  Type 'vec' de taille 1. La description du paramètre 2.
- **Initital Value of transition :**
  Type 'vec' de taille 1. La description du paramètre 3.
- **Date of initial output event :**
  Type 'vec' de taille 1. La description du paramètre 4.
- **Dynamical Value(0 :No/1 :Yes) :**
  Type 'vec' de taille 1. La description du paramètre 5.

3.6.3 Fonction de calcul (type 4)

```
/* vcoevt Scicos Time transition calculator block
 * Type 4 simulation function ver 1.1 - scilab-3
 * 7 Avril 2004 - IRCOM GROUP - Author : A.Layec
 * 21 Novembre 2004 : rajout option ipar[0]
 * 10 janvier 2005 : passage au type 4
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include "scicos_block.h"

/* Cette fonction réalise le calcul approximé de la date d'évenement
 * correspondant au passage du signal d'entrée aux valeurs rpar[1]
 * (rpar[1]*u2[0] si ipar[0]=1).
 * Rmq : L'entrée est considérée strictement croissante et monotone.
 *
 * entrées régulières : u1[0] : wot+phi(t)
 *                      u2[0] : entrée dynamique(rapport de division)
 * sortie régulière : néant
 * entrée d'évenement : Pas de calcul : Tsampl
```

```
 * sortie d'évenement : date du passage à rpar[1] (ou rpar[1]*u2[0])
 * paramètre entier : ipar[0] : 0 : passage aux valeurs rpar[1]
 *                              1 : passage aux valeurs rpar[1]*u2[0]
 * paramètres réels : rpar[0] : Tsampl
 *                    rpar[1] : valeur de passage en radian
 * état discret : z[0] : u(k-1)
 *                z[1] : valeur de u à la date de passage
 */

/*prototype*/
void vcoevt(scicos_block *block,int flag)
{
 /*déclaration des variables*/
 double *u1;
 double v; /*valeur de passage*/

 /*Récupération des adresses des ports d'entrées*/
 u1=(double *)block->inptr[0];

 /*Test si entrée 2*/
 if (block->ipar[0]==1)
 {
  /*déclaration*/
  double *u2;
  /*récupération adresse*/
  u2=(double *)block->inptr[1];
  v=block->rpar[1]*u2[0];
 }
 else v=block->rpar[1];

 /*le flag 3 calcul la date de transition*/
 if(flag==3)
  {
   /*Test si la valeur est >=rpar[1]*/
   if((u1[0]-block->z[1])>=v)
    {
     /*Calcul la date par extrapolation*/
     /*tvec[0] = *t + (((z[1]+v)-z[0])*rpar[0])/(u1[0]-z[0]);*/
     block->evout[0]=(((block->z[1]+v)-block->z[0])*block->rpar[0])/(u1[0]-block->z[0]);
     /*mémorise valeur à la date*/
     block->z[1] = block->z[1] + v;
    }
    /*mémorise u*/
    block->z[0]=u1[0];
  }
}
```

3.7 VCO_f - Bloc Oscillateur Contrôlé en Tension

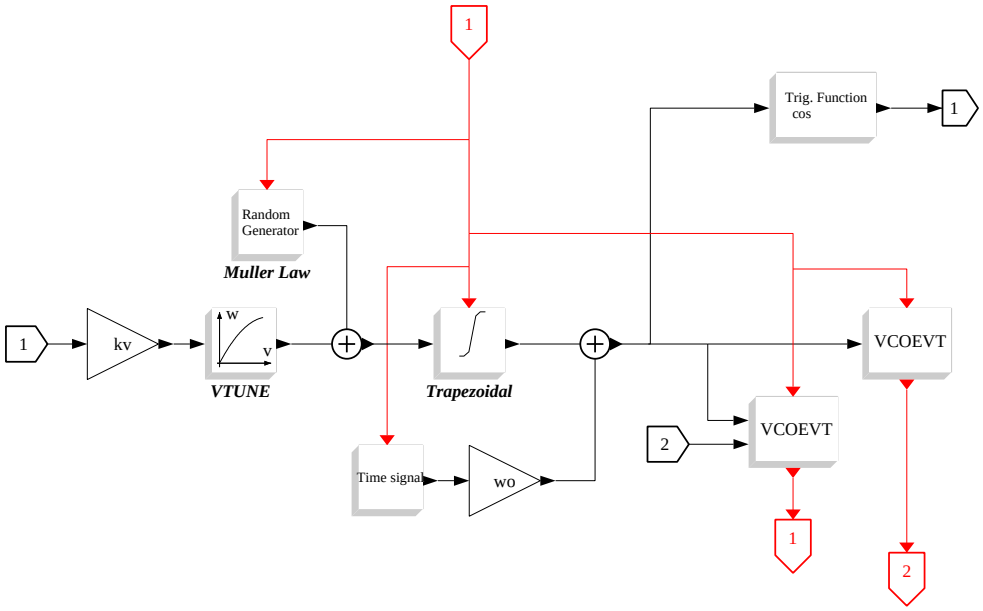


– **Palette :** Pll.cosf - Palette boucle à verrouillage de phase
– **Fonction d'interface :** `VCO_f.sci`

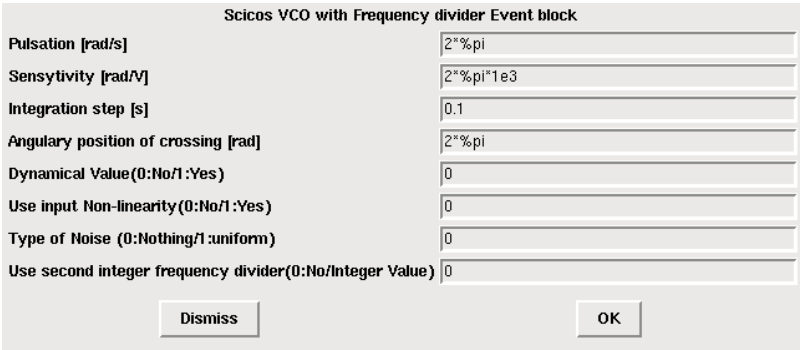
3.7.1 Description

Add here a paragraph of the function description.

3.7.2 Modèle équivalent en Super Bloc



3.7.3 Boîte de dialogue



– **Pulsation [rad/s] :**
Type 'vec' de taille -1. La description du paramètre 1.
– **Sensytivity [rad/V] :**
Type 'vec' de taille -1. La description du paramètre 2.

- **Integration step [s] :**
  Type 'vec' de taille 1. La description du paramètre 3.
- **Angulary position of crossing [rad] :**
  Type 'vec' de taille 1. La description du paramètre 4.
- **Dynamical Value(0 :No/1 :Yes) :**
  Type 'vec' de taille 1. La description du paramètre 5.
- **Use input Non-linearity(0 :No/1 :Yes) :**
  Type 'vec' de taille 1. La description du paramètre 6.
- **Type of Noise (0 :Nothing/1 :uniform) :**
  Type 'vec' de taille 1. La description du paramètre 7.
- **Use second integer frequency divider(0 :No/Integer Value) :**
  Type 'vec' de taille 1. La description du paramètre 8.

3.7.4 Fonction de calcul (type 4)

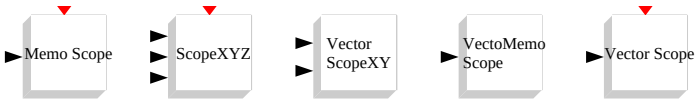
```
/* vco Scicos VCO with frequency divider Event block
 * Type 4 simulation function ver 1.1 - scilab-3.0
 * 1 decembre 2003 - IRCOM GROUP - Author : A.Layec
 * 10 janvier 2005 : rajout option du rapport de division dynamique
 *                   passage type 4
 *                   remplace otutes les anciennes routines vco
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <math.h>
#include <stdio.h>
#include "scicos_block.h"

/* Calcul la forme temporelle de sortie d'un VCO et ses instants de transitions
 * produit un évènement en sortie toutes les N fois (N:rapport de division)
 *
 * entrées régulières : u1[0] : u3(t) tension de commande
 *                      u2[0] : valeur du rapport de division
 *
 * sorties régulières : y[0] : ys(t)=cos(wot+phi(t))
 *                 avec phi(t)=integral(Kv*u3(t))dt pour le cas linéaire
 *                phi(t)=intégral(gain*tanh(coef*u3(t)))dt pour le cas non-linéaire
 *
 * entrées évènementielles : Tstep pas de résolution
 *
 * sorties évènementielles : evout[0] : Instant de passage aux valeurs rpar[3]*(ipar[0]+u1[0])
 *
 * paramètres réels :   rpar[0] : pas d'intégration
 *                      rpar[1] : wo pulsation libre de l'oscillateur
 *                      rpar[2] : kv sensibilité
 *                      rpar[3] : instant de transition
 *                    si ipar[1]==1 :
 *                      rpar[4] : gain de la fonction non-linéaire
 *                      rpar[5] : coeffficient de la fonction non-linéaire
 *                      rpar[6] : moyenne du bruit sur la commande
 *                      rpar[7] : variance du bruit sur la commande
 *
 * paramètres entiers : ipar[0] : réalise une détection dynamique
 *                                0 : passage aux valeurs rpar[1]
 *                                1 : passage aux valeurs rpar[1]*u2[0]
 *                      ipar[1] : réalise la fonction non-linéaire en entrée du VCO
 *                                0 : linéaire
 *                                1 : alpha*tanh(beta*u1[0])
 *                      ipar[2] : réalise un deuxième diviseur de fréquence à rapport entier
 *                                0 : pas de sencond diviseur
 *                               sinon valeur du 2eme rapport
 *                      ipar[3] : type de bruit
 *                                0 : pas de bruit
 *                                1 : bruit normal sur la commande d'entrée
 *
 * état discret : z[0] : valeur de la phase instantannée
 *               z[1] : valeur de l'entrée à l'état précédent
 *                z[2] : valeur de l'angle instantannée
 *               z[3] : valeur de 'angle à l'état précédent
 *                z[4] : compteur incremental de l'angle (ex: ..+2*%pi*N[k-1]+2*%pi*N[k]+..)
 *                z[5] : valeur de la distance à parcourir jusqu'à la prochaine transition (ex : 2*%pi*N)
 *                z[6] : compteur incremental de l'angle du 2eme diviseur (ex: ..+2*%pi*N[k-1]+2*%pi*N[k]+..)
 */

/*prototype*/
void vco(scicos_block *block,int flag)
{
 /*déclaration des variables*/
 double *y;
 double *u1;
 double *u2;

 double step; /*pas d'intégration*/
 double wo; /*pulsation libre*/
 double kv; /*sensibilité*/
```

```
double v; /*valeur de passage*/
double t; /*valeur du temps de simulation*/
double f_u; /*valeur de l'entrée(avec sa fonction)*/
double alpha; /*gain de la fonc non-linéaire*/
double beta; /*coef de la func non-linéaire*/

/*Récupération des adresses des ports réguliers*/
y=(double *)block->outptr[0];
u1=(double *)block->inptr[0];
u2=(double *)block->inptr[1];
step=block->rpar[0];
wo=block->rpar[1];
kv=block->rpar[2];
v=block->rpar[3];

/* Le flag 1 calcule l'intégrale discrète (trapèze) de la tension d'entrée
 * calcul de l'angle instantannée theta(t)=(wot+phi(t))
 * Place le cos(theta(t)) dans y[]
 */
if(flag==1)
{
 /*Calcul de la fonction d'entrée*/
 if (block->ipar[1]!=0)
 {
  alpha=block->rpar[4];
  beta=block->rpar[5];
  f_u=alpha*tanh(beta*u1[0]);
 }
 else f_u=u1[0];
 /*multiplication par sensibilité linéaire*/
 f_u=kv*f_u;

 if (block->ipar[3]==1) /*Test si bruit*/
 {
  /*déclaration*/
  double phi_n;
  int i,k;

  /*Appel noiseblk_c*/
  noiseblk_c((i=1,&i),(k=1,&k),&block->rpar[7],&block->rpar[6],&phi_n);

  /*rajout bruit*/
  f_u=f_u+(phi_n);
 }
 /*Calcul de la phase instantannée : intégrale de la valeur d'entrée par la méthode des trapèzes*/
 block->z[0]=(step/2)*(f_u+block->z[1])+block->z[0];
 /*Mise en mémoire de f_u*/
 block->z[1]=f_u;
 /*Calcul de l'angle wo*t+phi(t)*/
 t=get_scicos_time(); /*récupération de la valeur du temps*/
 block->z[2]=t*wo+block->z[0];

 /*Calcul du cos(wo*t+phi(t))*/
 y[0]=cos(block->z[2]);
}

/* Le flag 3 détermine l'instant de passage de theta(t) à rpar[3]*ipar[0]->z[5]
 * par une méthode d'extrapolation et place le résultat dans tvec[0]
 */

else if(flag==3)
{
 if (block->ipar[0]==1)
 {
  /*déclaration*/
  double *u2;
  /*récupération adresse*/
  u2=(double *)block->inptr[1];
  /*fprintf(stderr,"U2[0]=%f\n",u2[0]);*/
  block->z[5]=v*u2[0];
 }
 else block->z[5]=v;
 /*block->z[5]=v*block->ipar[0];*/

 /*Teste si la valeur de transistion est dépassée*/
 if((block->z[2]-block->z[4])>=block->z[5])
 {
  /*Calcul de l'instant de transition*/
  block->evout[0]=(((block->z[4]+block->z[5])-block->z[3])*step)/(block->z[2]-block->z[3]);

  /*Incrément valeur z[4]*/
  block->z[4]=block->z[4]+block->z[5];
 }

 /*Test si il existe un deuxième port de sortie evènenemtiel*/
 /*fprintf(stderr,"nevout=%d\n",block->nevout);*/
 if(block->ipar[2]!=0)
 {
  if((block->z[2]-block->z[6])>=v*block->ipar[2])
  {
   /*Calcul de l'instant de transition*/
   block->evout[1]=(((block->z[6]+v*block->ipar[2])-block->z[3])*step)/(block->z[2]-block->z[3]);
   /*Incrément valeur z[6]*/
   block->z[6]=block->z[6]+v*block->ipar[2];
  }
 }
}

/*Le flag 2 mémorise la valeur de thetha_k-1 dans z[3]*/
else if(flag==2)
{
```

```
  /*Mise en mémoire de*/
  block->z[3]=block->z[2];
 }
}
```

Chapitre 4

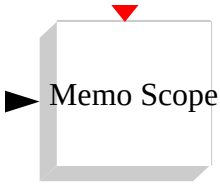
Palette Sinks



4.0.1 Description

Add here a paragraph of the function description.

4.1 MEMOSCOPE_f - Bloc oscilloscope à mémoire

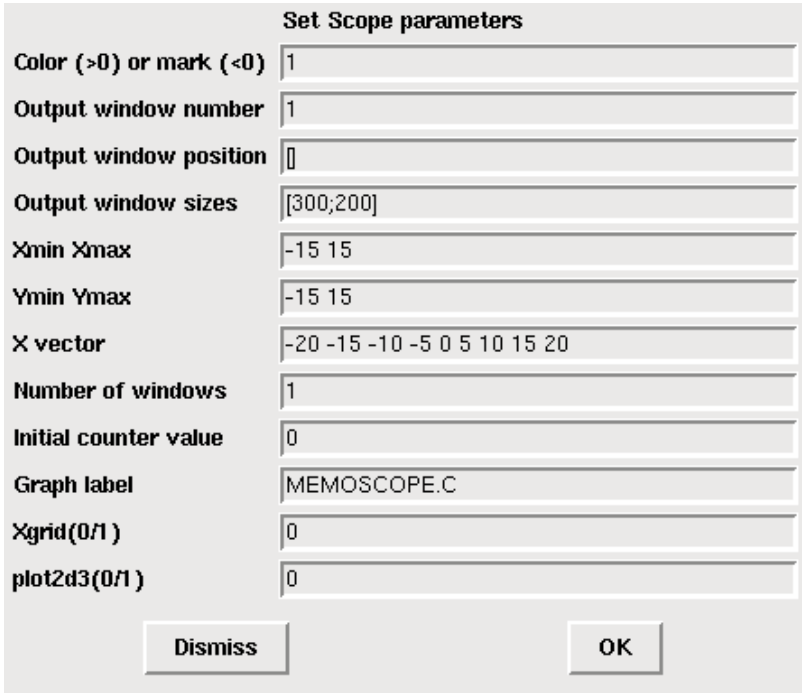


- **Palette :** Sinks.cosf - Palette Sinks
- **Fonction d'interface :** `MEMOSCOPE_f.sci`

4.1.1 Description

Add here a paragraph of the function description.

4.1.2 Boîte de dialogue



- **Color (>0) or mark (<0) :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Output window number :**
  Type 'vec' de taille 1. La description du paramètre 2.
- **Output window position :**
  Type 'vec' de taille -1. La description du paramètre 3.
- **Output window sizes :**
  Type 'vec' de taille -1. La description du paramètre 4.
- **Xmin Xmax :**
  Type 'vec' de taille 2. La description du paramètre 5.
- **Ymin Ymax :**
  Type 'vec' de taille 2. La description du paramètre 6.
- **X vector :**
  Type 'vec' de taille -1. La description du paramètre 7.

- **Number of windows :**
  Type ’vec’ de taille 1. La description du paramètre 8.
- **Initial counter value :**
  Type ’vec’ de taille 1. La description du paramètre 9.
- **Graph label :**
  Type ’str’ de taille 1. La description du paramètre 10.
- **Xgrid(0/1) :**
  Type ’vec’ de taille 1. La description du paramètre 11.
- **plot2d3(0/1) :**
  Type ’vec’ de taille 1. La description du paramètre 12.

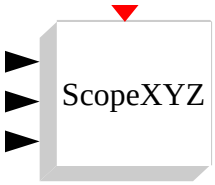
4.1.3 Fonction de calcul (type 2)

```
/* memoscope Scicos vector memory visualization block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 23 octobre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#define PI0 (integer *) 0
#define PD0 (double *) 0
#include "machine.h"
#include <stdio.h>

/*
 * Cette fonction de simulation est un oscilloscope à mémoire.
 * La largeur de la fenetre est fournie par le paramètre ipar[9]
 * (nombre d’échantillons:in_size) et la longueur de la mémoire est fournie
 * par le paramètre ipar[2] (nombre de fenetre à mémoriser).
 * A chaque coup d’horloge la valeur d’entrée u[] est stockée
 * dans le buffeur z. La taille totale du buffer est ipar[9]*ipar[2] + 1
 * Un compteur interne est incrémenté jusqu’a sa valeur finale ipar[9]*ipar[2]
 * et lorsque cette valeur est atteinte, le contenu du buffer est envoyé
 * a la routine graphique plot2d.
 * entrée régulières : u[0] valeur d’entrée
 * sorties régulières : néant
 * entrée évènementielle : date de déclenchement
 * sortie évènementielle : néant.
 * état discret : z[0..in_size-1]
 *                z[in_size..(2*in_size)-1]
 *                    ..
 *                    ..
 *                z[(ipar[2]-1)*in_size..(ipar[2]*in_size)-1] : buffer échantillons
 *                z[in_size*ipar[2]] : compteur échantillons
 *
 *  Integer Parameter
 *  ipar[0]    : window number
 *  ipar[1]    : color flag
 *  ipar[2]    : number of window (buffer_size=ipar[2]*ipar[9])
 *  ipar[3]    : dash,color or mark choice
 *  ipar[4]    : line or mark size
 *  ipar[5..6] : position of window
 *  ipar[7..8] : size of window
 *  ipar[9]    : in_size (length of window)
 *  ipar[10]   : additionnal option
 *                 (xgrid (0/1)   : bit 1)
 *  ipar[11]   : GRAPH_label_size (length of title)
 *  ipar[12..12+ipar[11]] : Char code of title
 *
 *  Real Parameter
 *  rpar[0] : xmin
 *  rpar[1] : xmax
 *  rpar[2] : ymin
 *  rpar[3] : ymax
 *  rpar[4..4+ipar[9]] : xvector
 */

/*prototype*/
void memoscope(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
               ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
   /* Déclaration des variables
    * Variables compteur
    */
   int i,j,k;
    /*Pour paramètres réels*/
   double xmin,xmax,ymin,ymax;
    /*Pour paramatères entiers*/
   int wid,graphcolor,wpos[2],wdim[2],in_size,option;
   int GRAPH_label_size,GRAPH_label_CODE[40];
    /* Autres*/
   char name[4];  /*Chaine de caractères pour le nom du driver graphique*/
   char strf[4];  /*Chaine de caractère pour controle affichage*/
   char buf[41];  /*Chaine pour recevoir la légende*/
   int nax[4];    /*Tableau entier pour controle des graduations*/
```

```
double rect[4];  /*Tableau double pour les coordonnées de la fenetre*/
double frect[4]; /*Cela sert à setscale2d*/
int ww,verbose=0,narg; /*Parametres entiers pour passage paramètre*/
double *u;  /*Pour scicos*/
/*int style[ipar[2]];*/
/*Tableau d'entier pour définition des styles de chaque courbes*/
int *style;
double *pos;
style=(int *) malloc(sizeof(int)*ipar[2]);
/*double pos[ipar[9]*ipar[2]];*/
/*Tableau de réels pour la définition de la position de chaque point*/
pos = (double *) malloc(sizeof(double)*ipar[9]*ipar[2]);


/*Récupération de l'adresse du port d'entrée régulier*/
u=(double *)inptr[0];

/*Récupération des paramètres entiers*/
wid=ipar[0];
graphcolor=ipar[1];
wpos[0]=ipar[5];
wpos[1]=ipar[6];
wdim[0]=ipar[7];
wdim[1]=ipar[8];
in_size=ipar[9];
option=ipar[10];
GRAPH_label_size=ipar[11];

/*Récupération des paramètres réels*/
xmin=rpar[0];xmax=rpar[1]; ymin=rpar[2];ymax=rpar[3];

/*Place xmin,ymin,xmax et ymax dans rect[]*/
rect[0]=xmin;rect[1]=ymin;rect[2]=xmax;rect[3]=ymax;

/*Initialise strf, nax et frec*/
strf[0]='0';strf[1]='1';strf[2]='1';strf[3]='\0';
nax[0]=2;nax[1]=10;nax[2]=2;nax[3]=10;
frect[0]=0;frect[1]=0;frect[2]=1;frect[3]=1;

/*Le flag2 place u[] dans z[] et affiche z[] si nécéssaire*/
if(*flag==2)
{
 /*Récupère dans k valeur compteur échantillons*/
 k=(int)z[in_size*ipar[2]];

 /*Affecte z[k] à la valeur de u[0]*/
 z[k]=u[0];

 /*Incrémente compteur échantillons*/
 z[in_size*ipar[2]]++;

 if(z[in_size*ipar[2]]==in_size*ipar[2])
 {
  /*RAZ compteur échantillons*/
  z[in_size*ipar[2]]=0;

  /*Retourne le numéro de fenetre active dans ww*/
  C2F(dr1)("xget","window",&verbose,&ww,&narg,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Assigne le numéro de fenetre wid si nécéssaire*/
  if(ww!=wid)
   C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Assigne la couleur*/
  C2F(dr1)("xset","use color",&graphcolor,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Place le flag wresize pour redimensionner automatiquement la fenetre*/
  C2F(dr1)("xset","wresize",(j=1,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Efface la fenetre courante*/
  C2F(dr1)("xclear","v",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Que fait xstart?*/
  C2F(dr)("xstart","v",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Récupère la légende*/
  for(i=0;i<ipar[11];i++) GRAPH_label_CODE[i]=ipar[12+i];
  buf[ipar[11]]='\0';

  /*Converti les codes de la légende en caractère*/
  C2F(cvstr)(&GRAPH_label_size,&GRAPH_label_CODE[0],&buf[0],(j=1,&j));

  /*Initialise le type de tracé*/
  C2F(dr1)("xset","dashes",(j=0,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Affectation de style[] et de pos[]*/
  for (j=0;j<ipar[2];j++)
  {
   style[j]=graphcolor;
   for (i=0;i<ipar[9];i++) pos[j*ipar[9]+i]=rpar[4+i];
  }

  /*Execute plot2d*/
  C2F(plot2d)(&pos[0],&z[0],(j=ipar[2],&j),&in_size,&style[0],&strf,&buf,&rect,&nax,0L,0L);

  /*Affiche la legende*/
  Legends((i=1,&i),(j=1,&j),&buf);

  /*Test le bit 1 de option*/
  if((option&1)==1) C2F(xgrid)((j=1,&j));
 }
```

```
  }
  /*le flag4 initialise la fenetre graphique*/
  else if(*flag==4)
  {
   /* Vérification du driver graphique
    * Le nom du driver est retourné dans la variable name
    */
   C2F(dr1)("xgetdr",name,PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);
   /*fprintf(stderr,"name=%s\n", name);*/

   /*Change le driver en position Rec si nécéssaire*/
   if(name!="Rec")
     C2F(dr1)("xsetdr","Rec",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Je sais pas ce que cela fait sciwin()?!? (j'ai pas trouvé le code)*/
   C2F(sciwin)();

   /*Retourne le numéro de fenetre active dans ww*/
   C2F(dr1)("xget","window",&verbose,&ww,&narg,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /* Assigne le numéro de fenetre wid si nécéssaire*/
   if(ww!=wid)
    C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Teste la position de la fenetre et replace la si necessaire*/
   if((wpos[0]>=0)&&(wpos[1]>0))
    C2F(dr1)("xset","wpos",&wpos[0],&wpos[1],PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Teste la dimension de la fenetre et redimensionne la si nécessaire*/
   if((wdim[0]>=0)&&(wdim[1]>0))
    C2F(dr1)("xset","wdim",&wdim[0],&wdim[1],PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Reassigne la fenetre pour forcer les changements*/
   C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Ajuste l'echelle et la forme des axes (? correspond aux options de plot2d)*/
   C2F(setscale2d)(&frect,&rect,"nn",0L);

   /*Assigne la couleur*/
   C2F(dr1)("xset","use color",&graphcolor,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Place le flag wresize pour redimensionner automatiquement la fenetre*/
   C2F(dr1)("xset","wresize",(j=1,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Initialise la fonction logique pour dessiner (?)*/
   C2F(dr1)("xset","alufunction",(j=3,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Efface la fenetre courante*/
   C2F(dr1)("xclear","v",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Que fait xstart?*/
   C2F(dr)("xstart","v",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Récupère la légende*/
   for(i=0;i<ipar[11];i++) GRAPH_label_CODE[i]=ipar[12+i];
   buf[ipar[11]]='\0';

   /*Converti les codes de la légende en caractère*/
   C2F(cvstr)(&GRAPH_label_size,&GRAPH_label_CODE[0],&buf[0],(j=1,&j));

   /*Initialise le type de tracé*/
   C2F(dr1)("xset","dashes",(j=0,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Execute plot2d*/
   C2F(plot2d)(&rect[0],&rect[1],(j=1,&j),(k=1,&k),&graphcolor,&strf,&buf,&rect,&nax,0L,0L);

   /*Applique la légende au titre de la fenetre*/
   C2F(dr)("xname",&buf,PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);
  }
  free(style);
  free(pos);
}
```

4.2 SCOPXYZ_f - Bloc oscilloscope de trajectoire 3D

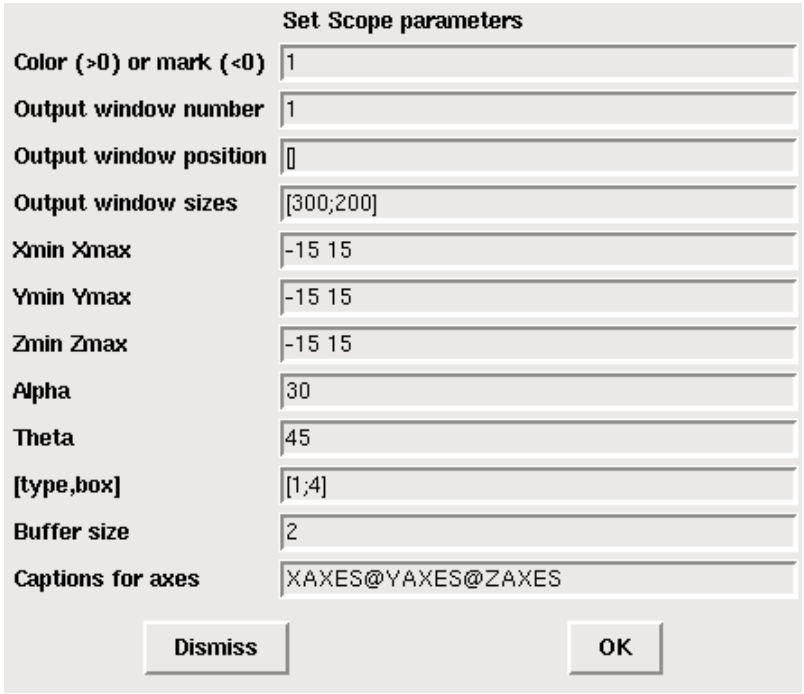


– **Palette :** Sinks.cosf - Palette Sinks
– **Fonction d'interface :** `SCOPXYZ_f.sci`

4.2.1 Description

Add here a paragraph of the function description.

4.2.2 Boîte de dialogue



– **Color (>0) or mark (<0) :**
Type 'vec' de taille 1. La description du paramètre 1.
– **Output window number :**
Type 'vec' de taille 1. La description du paramètre 2.
– **Output window position :**
Type 'vec' de taille -1. La description du paramètre 3.
– **Output window sizes :**
Type 'vec' de taille -1. La description du paramètre 4.
– **Xmin Xmax :**
Type 'vec' de taille 2. La description du paramètre 5.
– **Ymin Ymax :**
Type 'vec' de taille 2. La description du paramètre 6.
– **Zmin Zmax :**
Type 'vec' de taille 2. La description du paramètre 7.

- **Alpha :**
  Type ’vec’ de taille 1. La description du paramètre 8.
- **Theta :**
  Type ’vec’ de taille 1. La description du paramètre 9.
- **[type,box] :**
  Type ’vec’ de taille 2. La description du paramètre 10.
- **Buffer size :**
  Type ’vec’ de taille 1. La description du paramètre 11.
- **Captions for axes :**
  Type ’str’ de taille 1. La description du paramètre 12.

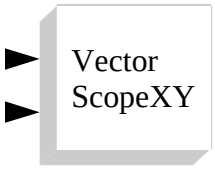
4.2.3 Fonction de calcul (type 2)

```
/* scopxyz Scicos 3d visualization block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 24 octobre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#define PI0 (integer *) 0
#define PD0 (double *) 0
#include "machine.h"
#include <stdio.h>
#include <stdlib.h>

/* Cette fonction de simulation est un oscilloscope à 3 dimensions.
 * Elle permet l’observation de l’évolution d’une trajectoire dont
 * les valeurs des coordonnées x,y,z, en chaque instant de déclenchement
 * de la fonction, sont présentes sur les entrées u1,u2 et u3.
 * Cette fonction utilise la routines graphique param3d1.
 * A chaque coup d’horloge, les valeurs des coordonnées sont stockée dans
 * le buffer z[]. La taille totale de z est 3*buffer_size + 1. Un compteur
 * est incrémenté jusqu’a sa valeur finale buffer_size et lorsque cette valeur
 * est atteinte, le contenu du buffer z est envoyé à la routine param3d1.
 *
 * Entrées régulières : u1[0], u2[0] et u3[0] sont les coordonnées du point.
 * Soties régulières : néant.
 * Entrée évènementielle : Instant de déclenchement.
 * Sortie évènementielle : néant.
 * Registre interne (taille (3*buffer_size)+1)
 * z[0..buffer_size-1] : x
 * z[buffer_size..2*buffer_size-1] : y
 * z[2*buffer_size..3*buffer_size-1] : z
 * z[3*buffer_size] : compteur échantillons
 *
 * Integer Parameter
 * ipar[0]    : window number
 * ipar[1]    : color flag
 * ipar[2]    : buffer_size
 * ipar[3]    : dash,color or mark choice
 * ipar[4]    : line or mark size
 * ipar[5..6] : position of window
 * ipar[7..8] : size of window
 * ipar[9]    : type
 * ipar[10]   : box
 * ipar[11]   : GRAPH_label_size (length of axes captions)
 * ipar[12..12+ipar[11]] : Char code of axes captions
 *
 * Real Parameter
 * rpar[0] : xmin
 * rpar[1] : xmax
 * rpar[2] : ymin
 * rpar[3] : ymax
 * rpar[4] : zmin
 * rpar[5] : zmax
 * rpar[6] : alpha
 * rpar[7] : theta
 */

/*prototype*/
void scopxyz(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
             ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;

{
   /* Déclaration des variables
    * Variables compteur
    */

   int i,j,k;
   /*Pour paramètres réels*/
   double xmin,xmax,ymin,ymax,zmin,zmax,alpha,theta;
   /*Pour paramatères entiers*/
   int wid,graphcolor,wpos[2],wdim[2],buffer_size,option;
   int GRAPH_label_size,GRAPH_label_CODE[40];
   /*Autres*/
```

```
char name[4]; /*Chaine de caractères pour le nom du driver graphique*/
char buf[41]; /*Chaine pour recevoir la légende*/
int eflag[3]; /*Flag pour param3d1*/
double rect[6]; /*Tableau double pour les coordonnées de la fenetre*/
int ww,verbose=0,narg; /*Parametres entiers pour passage paramètre*/
double *u1,*u2,*u3;
int *style;
style = (int *) malloc(sizeof(int)*ipar[2]);

/*Récupération de l'adresse du port d'entrée régulier*/
u1=(double *)inptr[0];
u2=(double *)inptr[1];
u3=(double *)inptr[2];

/*Récupération des paramètres entiers*/
wid=ipar[0];
graphcolor=ipar[1];
buffer_size=ipar[2];
wpos[0]=ipar[5];
wpos[1]=ipar[6];
wdim[0]=ipar[7];
wdim[1]=ipar[8];

GRAPH_label_size=ipar[11];

/*Récupération des paramètres réels*/
xmin=rpar[0];xmax=rpar[1]; ymin=rpar[2];ymax=rpar[3];
zmin=rpar[4];zmax=rpar[5]; alpha=rpar[6]; theta=rpar[7];

/*Place xmin,xmax,ymin,ymax,zmin,zmax dans rect[]*/
rect[0]=xmin;rect[1]=xmax;rect[2]=ymin;rect[3]=ymax;
rect[4]=zmin;rect[5]=zmax;

/*Initialise eflag (à quoi sert eflag[0]?,eflag[2]->type,eflag[3]->box)*/
eflag[0]=1;eflag[1]=ipar[9];eflag[2]=ipar[10];

/*Le flag2 place u[] dans z[] et affiche z[] si nécéssaire*/
if(*flag==2)
{
  /*récupère valeur compteur échantillons dans k*/
  k=(int)z[3*buffer_size];

  /*Place u[] dans z[]*/
  z[k]=u1[0];
  z[buffer_size+k]=u2[0];
  z[2*buffer_size+k]=u3[0];

  /*Incrémente compteur échantillons*/
  z[3*buffer_size]++;

  /* Test la valeur compteur échantillons
   * pour savoir si celle ci a atteint la valeur buffer_size
   */
  if(z[3*buffer_size]==buffer_size)
  {
   /*Retourne le numéro de fenetre active dans ww*/
   C2F(dr1)("xget","window",&verbose,&ww,&narg,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Assigne le numéro de fenetre wid si nécéssaire*/
   if(ww!=wid)
    C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   /*Récupère la légende des axes*/
   for(i=0;i<ipar[11];i++) GRAPH_label_CODE[i]=ipar[12+i];
   buf[ipar[11]]='\0';

   /*Converti les codes de la légende en caractère*/
   C2F(cvstr)(&GRAPH_label_size,&GRAPH_label_CODE[0],&buf[0],(j=1,&j));
   /*buf[0]='@';buf[1]='@';buf[2]='Y';buf[3]='@';buf[4]='Z';buf[5]='a';buf[6]='\0';*/
   /*Affectation de style[]*/
   for (j=0;j<ipar[2];j++) style[j]=graphcolor;

   /*Execute param3d1*/
   C2F(param3d1)(&z[0],&z[buffer_size],&z[2*buffer_size],&buffer_size,(j=1,&j),(k=1,&k),&style[0],&theta,&alpha,&buf,&eflag,&rect,0L);

   /* Teste la valeur de graphcolor pour eviter un tracé
    * discontinu si la valeur est >0
    */
   if(graphcolor<=0)
     /*RAZ compteur échantillons*/
     z[3*buffer_size]=0;
   else
   {
     /*Place valeur compteur échantillons à 1*/
     z[3*buffer_size]=1;

     /*Recopie derniers echantillons dans z[0],z[buffer_size] et z[2*buffer_size]*/
     z[0]=u1[0];
     z[buffer_size]=u2[0];
     z[2*buffer_size]=u3[0];
   }
  }
 }
/*le flag4 initialise la fenetre graphique*/
else if(*flag==4)
{
 /* Vérification du driver graphique
  * Le nom du driver est retourné dans la variable name
  */
 C2F(dr1)("xgetdr",name,PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);
 /*fprintf(stderr,"name=%s\n", name);*/
```

```
    /*Change le driver en position Rec si nécéssaire*/
    if(name!="Rec")
      C2F(dr1)("xsetdr","Rec",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Je sais pas ce que cela fait sciwin()?!? (j'ai pas trouvé le code)*/
    C2F(sciwin)();

    /*Retourne le numéro de fenetre active dans ww*/
    C2F(dr1)("xget","window",&verbose,&ww,&narg,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Assigne le numéro de fenetre wid si nécéssaire*/
    if(ww!=wid)
     C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Teste la position de la fenetre et replace la si necessaire*/
    if((wpos[0]>=0)&&(wpos[1]>0))
     C2F(dr1)("xset","wpos",&wpos[0],&wpos[1],PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Teste la dimension de la fenetre et redimensionne la si nécessaire*/
    if((wdim[0]>=0)&&(wdim[1]>0))
     C2F(dr1)("xset","wdim",&wdim[0],&wdim[1],PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Reassigne la fenetre pour forcer les changements*/
    C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Assigne la couleur*/
    C2F(dr1)("xset","use color",&graphcolor,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Place le flag wresize pour redimensionner automatiquement la fenetre*/
    C2F(dr1)("xset","wresize",(j=1,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Initialise la fonction logique pour dessiner (?)*/
    C2F(dr1)("xset","alufunction",(j=3,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Efface la fenetre courante*/
    C2F(dr1)("xclear","v",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Que fait xstart?*/
    C2F(dr)("xstart","v",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Récupère la légende*/
    for(i=0;i<ipar[11];i++) GRAPH_label_CODE[i]=ipar[12+i];
    buf[ipar[11]]='\0';

    /*Converti les codes de la légende en caractère*/
    C2F(cvstr)(&GRAPH_label_size,&GRAPH_label_CODE[0],&buf[0],(j=1,&j));

    /*Initialise le type de tracé*/
    C2F(dr1)("xset","dashes",(j=0,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Affectation de style[]*/
    for (j=0;j<ipar[2];j++) style[j]=graphcolor;

    /*Execute param3d1*/
    C2F(param3d1)(&z[0],&z[buffer_size],&z[2*buffer_size],&buffer_size,(j=1,&j),(k=0,&k),&style,&theta,&alpha,&buf,&eflag,&rect,0L);

    /*Applique la légende au titre de la fenetre*/
    /*C2F(dr)("xname",&buf,PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);*/
   }
   free(style);
}
```

4.3 SCOXYVEC_f - Bloc oscilloscope vectoriel

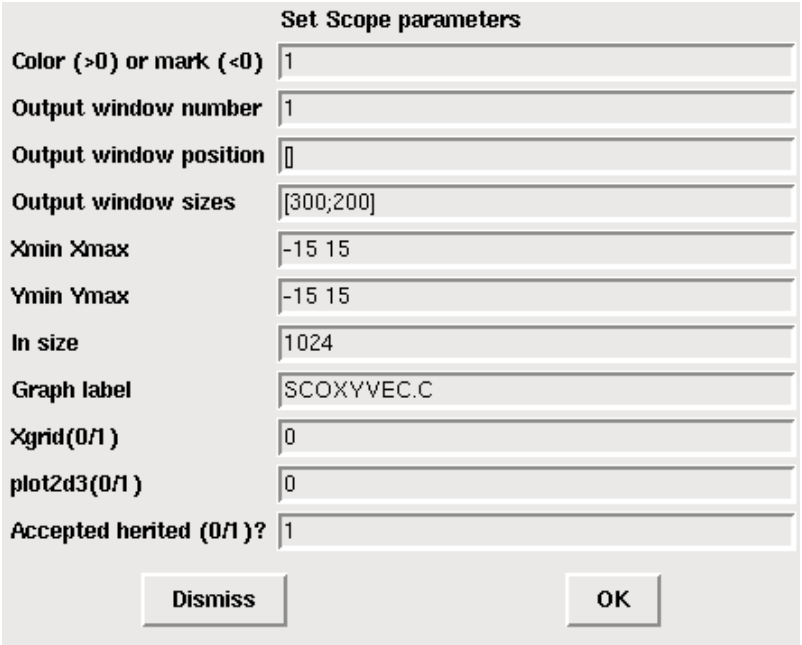


- **Palette :** Sinks.cosf - Palette Sinks
- **Fonction d'interface :** `SCOXYVEC_f.sci`

4.3.1 Description

Add here a paragraph of the function description.

4.3.2 Boîte de dialogue



- **Color (>0) or mark (<0) :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Output window number :**
  Type 'vec' de taille 1. La description du paramètre 2.
- **Output window position :**
  Type 'vec' de taille -1. La description du paramètre 3.
- **Output window sizes :**
  Type 'vec' de taille -1. La description du paramètre 4.
- **Xmin Xmax :**
  Type 'vec' de taille 2. La description du paramètre 5.
- **Ymin Ymax :**
  Type 'vec' de taille 2. La description du paramètre 6.
- **In size :**
  Type 'vec' de taille 1. La description du paramètre 7.
- **Graph label :**
  Type 'str' de taille 1. La description du paramètre 8.

– **Xgrid(0/1) :**
  Type ’vec’ de taille 1. La description du paramètre 9.
– **plot2d3(0/1) :**
  Type ’vec’ de taille 1. La description du paramètre 10.
– **Accepted herited (0/1) ? :**
  Type ’vec’ de taille 1. La description du paramètre 11.

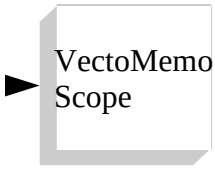
4.3.3 Fonction de calcul (type 2)

```
/* memoscope Scicos vector memory visualization block
 *Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 *23 octobre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#define PI0 (integer *) 0
#define PD0 (double *) 0
#include "machine.h"
#include <stdio.h>

/*Cette fonction de simulation est un oscilloscope à mémoire.
 * La largeur de la fenetre est fournie par le paramètre ipar[9]
 * (nombre d’échantillons:in_size) et la longueur de la mémoire est fournie
 * par le paramètre ipar[2] (nombre de fenetre à mémoriser).
 * A chaque coup d’horloge la valeur d’entrée u[] est stockée
 * dans le buffeur z. La taille totale du buffer est ipar[9]*ipar[2] + 1
 * Un compteur interne est incrémenté jusqu’a sa valeur finale ipar[9]*ipar[2]
 * et lorsque cette valeur est atteinte, le contenu du buffer est envoyé
 * a la routine graphique plot2d.
 * entrée régulières : u[0..nu-1] valeur d’entrée
 * sorties régulières : néant
 * entrée évènementielle : date de déclenchement
 * sortie évènementielle : néant.
 * état discret : z[0] : ne sert à rien
 *
 * Integer Parameter
 * ipar[0]    : window number
 * ipar[1]    : color flag
 * ipar[2]    : number of window (buffer_size=ipar[2]*ipar[9])
 * ipar[3]    : dash,color or mark choice
 * ipar[4]    : line or mark size
 * ipar[5..6] : position of window
 * ipar[7..8] : size of window
 * ipar[9]    : in_size (length of window)
 * ipar[10]   : additionnal option
 *                 (xgrid (0/1)   : bit 1)
 * ipar[11]   : GRAPH_label_size (length of title)
 * ipar[12..12+ipar[11]] : Char code of title
 *
 * Real Parameter
 * rpar[0] : xmin
 * rpar[1] : xmax
 * rpar[2] : ymin
 * rpar[3] : ymax
 * rpar[4..4+ipar[9]] : xvector
 */
/*prototype*/
void
scoxyvec(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
        ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;

{
   /* Déclaration des variables
    * Variables compteur
    */
   int i,j,k;
   /*Pour paramètres réels*/
   double xmin,xmax,ymin,ymax;
   /*Pour paramatères entiers*/
   int wid,graphcolor,wpos[2],wdim[2],in_size,option;
   int GRAPH_label_size,GRAPH_label_CODE[40];
   /*Autres*/
   char name[4]; /*Chaine de caractères pour le nom du driver graphique*/
   char strf[4]; /*Chaine de caractère pour controle affichage*/
   char buf[41]; /*Chaine pour recevoir la légende*/
   int nax[4]; /*Tableau entier pour controle des graduations*/
   double rect[4]; /*Tableau double pour les coordonnées de la fenetre*/
   double frect[4]; /*Cela sert à setscale2d*/
   int ww,verbose=0,narg; /*Parametres entiers pour passage paramètre*/
   /*Pour scicos*/
   double *u1,*u2;

   /*Récupération de l’adresse du port d’entrée régulier*/
   u1=(double *)inptr[0];
   u2=(double *)inptr[1];

   /*Récupération des paramètres entiers*/
   wid=ipar[0];
```

```
graphcolor=ipar[1];
wpos[0]=ipar[5];
wpos[1]=ipar[6];
wdim[0]=ipar[7];
wdim[1]=ipar[8];
in_size=ipar[9];
option=ipar[10];
GRAPH_label_size=ipar[11];

/*Récupération des paramètres réels*/
xmin=rpar[0];xmax=rpar[1]; ymin=rpar[2];ymax=rpar[3];

/*Place xmin,ymin,xmax et ymax dans rect[]*/
rect[0]=xmin;rect[1]=ymin;rect[2]=xmax;rect[3]=ymax;

/*Initialise strf, nax et frec*/
strf[0]='0';strf[1]='1';strf[2]='1';strf[3]='\0';
nax[0]=2;nax[1]=10;nax[2]=2;nax[3]=10;
frect[0]=0;frect[1]=0;frect[2]=1;frect[3]=1;

/*Le flag2 place u[] dans la fentêtre graphique*/
if(*flag==2)
{
  /*Retourne le numéro de fenetre active dans ww*/
  C2F(dr1)("xget","window",&verbose,&ww,&narg,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Assigne le numéro de fenetre wid si nécéssaire*/
  if(ww!=wid)
    C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Assigne la couleur*/
  C2F(dr1)("xset","use color",&graphcolor,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Place le flag wresize pour redimensionner automatiquement la fenetre*/
  C2F(dr1)("xset","wresize",(j=1,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Efface la fenetre courante*/
  C2F(dr1)("xclear","v",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Que fait xstart?*/
  C2F(dr)("xstart","v",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Récupère la légende*/
  for(i=0;i<ipar[11];i++) GRAPH_label_CODE[i]=ipar[12+i];
  buf[ipar[11]]='\0';

  /*Converti les codes de la légende en caractère*/
  C2F(cvstr)(&GRAPH_label_size,&GRAPH_label_CODE[0],&buf[0],(j=1,&j));

  /*Initialise le type de tracé*/
  C2F(dr1)("xset","dashes",(j=1,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Execute plot2d*/
  C2F(plot2d)(&u1[0],&u2[0],(i=1,&i),(j=ipar[9],&j),&graphcolor,&strf,&buf,&rect,&nax,0L,0L);

  /*Affiche la legende*/
  Legends((i=1,&i),(j=1,&j),&buf);

  /*Test le bit 1 de option*/
  if((option&1)==1) C2F(xgrid)((j=1,&j));
}
/*le flag4 initialise la fenetre graphique*/
else if(*flag==4)
{
 /* Vérification du driver graphique
  * Le nom du driver est retourné dans la variable name
  */
 C2F(dr1)("xgetdr",name,PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);
 /*fprintf(stderr,"name=%s\n", name);*/

 /*Change le driver en position Rec si nécéssaire*/
 if(name!="Rec")
   C2F(dr1)("xsetdr","Rec",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Je sais pas ce que cela fait sciwin()?!? (j'ai pas trouvé le code)*/
 C2F(sciwin)();

 /*Retourne le numéro de fenetre active dans ww*/
 C2F(dr1)("xget","window",&verbose,&ww,&narg,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Assigne le numéro de fenetre wid si nécéssaire*/
 if(ww!=wid)
  C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Teste la position de la fenetre et replace la si necessaire*/
 if((wpos[0]>=0)&&(wpos[1]>0))
  C2F(dr1)("xset","wpos",&wpos[0],&wpos[1],PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Teste la dimension de la fenetre et redimensionne la si nécessaire*/
 if((wdim[0]>=0)&&(wdim[1]>0))
  C2F(dr1)("xset","wdim",&wdim[0],&wdim[1],PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Reassigne la fenetre pour forcer les changements*/
 C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Ajuste l'echelle et la forme des axes (? correspond aux options de plot2d)*/
 C2F(setscale2d)(&frect,&rect,"nn",0L);

 /*Assigne la couleur*/
 C2F(dr1)("xset","use color",&graphcolor,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);
```

```
    /*Place le flag wresize pour redimensionner automatiquement la fenetre*/
    C2F(dr1)("xset","wresize",(j=1,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Initialise la fonction logique pour dessiner (?)*/
    C2F(dr1)("xset","alufunction",(j=3,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Efface la fenetre courante*/
    C2F(dr1)("xclear","v",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Que fait xstart?*/
    C2F(dr)("xstart","v",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Récupère la légende*/
    for(i=0;i<ipar[11];i++) GRAPH_label_CODE[i]=ipar[12+i];
    buf[ipar[11]]='\0';

    /*Converti les codes de la légende en caractère*/
    C2F(cvstr)(&GRAPH_label_size,&GRAPH_label_CODE[0],&buf[0],(j=1,&j));

    /*Initialise le type de tracé*/
    C2F(dr1)("xset","dashes",(j=0,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Execute plot2d*/
    C2F(plot2d)(&rect[0],&rect[1],(j=1,&j),(k=1,&k),&graphcolor,&strf,&buf,&rect,&nax,0L,0L);

    /*Applique la légende au titre de la fenetre*/
    C2F(dr)("xname",&buf,PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);
   }
}
```

4.4 VECTOMEMOSCOPE_f - Bloc oscilloscope vectoriel à mémoire

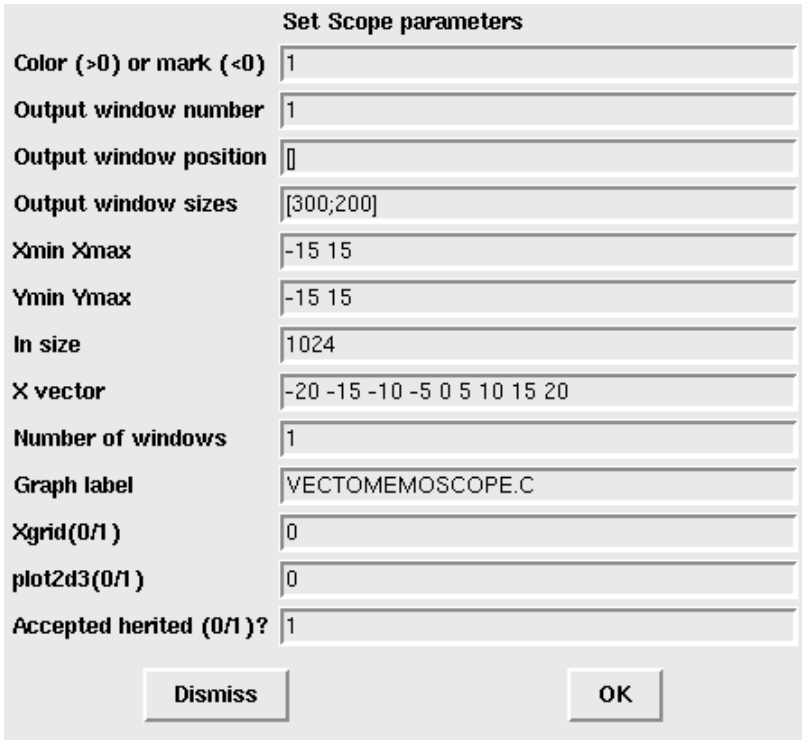


- **Palette :** Sinks.cosf - Palette Sinks
- **Fonction d'interface :** `VECTOMEMOSCOPE_f.sci`

4.4.1 Description

Add here a paragraph of the function description.

4.4.2 Boîte de dialogue



- **Color (>0) or mark (<0) :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Output window number :**
  Type 'vec' de taille 1. La description du paramètre 2.
- **Output window position :**
  Type 'vec' de taille -1. La description du paramètre 3.
- **Output window sizes :**
  Type 'vec' de taille -1. La description du paramètre 4.
- **Xmin Xmax :**
  Type 'vec' de taille 2. La description du paramètre 5.
- **Ymin Ymax :**
  Type 'vec' de taille 2. La description du paramètre 6.

- **In size :**
  Type ’vec’ de taille -1. La description du paramètre 7.
- **X vector :**
  Type ’vec’ de taille -1. La description du paramètre 8.
- **Number of windows :**
  Type ’vec’ de taille 1. La description du paramètre 9.
- **Graph label :**
  Type ’str’ de taille 1. La description du paramètre 10.
- **Xgrid(0/1) :**
  Type ’vec’ de taille 1. La description du paramètre 11.
- **plot2d3(0/1) :**
  Type ’vec’ de taille 1. La description du paramètre 12.
- **Accepted herited (0/1) ? :**
  Type ’vec’ de taille 1. La description du paramètre 13.

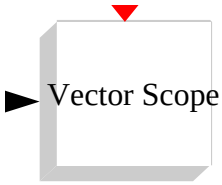
4.4.3 Fonction de calcul (type 2)

```
/* memoscope Scicos vector memory visualization block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 23 octobre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#define PI0 (integer *) 0
#define PD0 (double *) 0
#include "machine.h"
#include <stdio.h>

/* Cette fonction de simulation est un oscilloscope à mémoire.
 * La largeur de la fenetre est fournie par le paramètre ipar[9]
 * (nombre d’échantillons:in_size) et la longueur de la mémoire est fournie
 * par le paramètre ipar[2] (nombre de fenetre à mémoriser).
 * A chaque coup d’horloge la valeur d’entrée u[] est stockée
 * dans le buffeur z. La taille totale du buffer est ipar[9]*ipar[2] + 1
 * Un compteur interne est incrémenté jusqu’a sa valeur finale ipar[9]*ipar[2]
 * et lorsque cette valeur est atteinte, le contenu du buffer est envoyé
 * a la routine graphique plot2d.
 * entrée régulières : u[0..nu-1] valeur d’entrée
 * sorties régulières : néant
 * entrée évènementielle : date de déclenchement
 * sortie évènementielle : néant.
 * état discret : z[0] : ne sert à rien
 *
 * Integer Parameter
 * ipar[0]    : window number
 * ipar[1]    : color flag
 * ipar[2]    : number of window (buffer_size=ipar[2]*ipar[9])
 * ipar[3]    : dash,color or mark choice
 * ipar[4]    : line or mark size
 * ipar[5..6] : position of window
 * ipar[7..8] : size of window
 * ipar[9]    : in_size (length of window)
 * ipar[10]   : additionnal option
 *                (xgrid (0/1)   : bit 1)
 * ipar[11]   : GRAPH_label_size (length of title)
 * ipar[12..12+ipar[11]] : Char code of title
 *
 * Real Parameter
 * rpar[0] : xmin
 * rpar[1] : xmax
 * rpar[2] : ymin
 * rpar[3] : ymax
 * rpar[4..4+ipar[9]] : xvector
 */
/*prototype*/
void vectomemoscope(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
                    ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
   /*Déclaration des variables
    * Variables compteur
    */
   int i,j,k;
   /*Pour paramètres réels*/
   double xmin,xmax,ymin,ymax;
   /*Pour paramatères entiers*/
   int wid,graphcolor,wpos[2],wdim[2],in_size,option;
   int GRAPH_label_size,GRAPH_label_CODE[40];
   /*Autres*/
   char name[4]; /*Chaine de caractères pour le nom du driver graphique*/
   char strf[4]; /*Chaine de caractère pour controle affichage*/
   char buf[41]; /*Chaine pour recevoir la légende*/
```

```
int nax[4];   /*Tableau entier pour controle des graduations*/
double rect[4]; /*Tableau double pour les coordonnées de la fenetre*/
double frect[4]; /*Cela sert à setscale2d*/
int ww,verbose=0,narg; /*Parametres entiers pour passage paramètre*/
double *u; /*Pour scicos*/
int *style;
double *pos;
/*int style[ipar[2]];*/
/*Tableau d'entier pour définition des styles de chaque courbes*/
style=(int *) malloc(sizeof(int)*ipar[2]);
/*Tableau de réels pour la définition de la position de chaque point*/
pos=(double *) malloc(sizeof(double)*ipar[9]*ipar[2]);
/*double pos[ipar[9]*ipar[2]];*/

/*Récupération de l'adresse du port d'entrée régulier*/
u=(double *)inptr[0];

/*Récupération des paramètres entiers*/
wid=ipar[0];
graphcolor=ipar[1];
wpos[0]=ipar[5];
wpos[1]=ipar[6];
wdim[0]=ipar[7];
wdim[1]=ipar[8];
in_size=ipar[9];
option=ipar[10];
GRAPH_label_size=ipar[11];

/*Récupération des paramètres réels*/
xmin=rpar[0];xmax=rpar[1]; ymin=rpar[2];ymax=rpar[3];

/*Place xmin,ymin,xmax et ymax dans rect[]*/
rect[0]=xmin;rect[1]=ymin;rect[2]=xmax;rect[3]=ymax;

/*Initialise strf, nax et frec*/
strf[0]='0';strf[1]='1';strf[2]='1';strf[3]='\0';
nax[0]=2;nax[1]=10;nax[2]=2;nax[3]=10;
frect[0]=0;frect[1]=0;frect[2]=1;frect[3]=1;

/*Le flag2 place u[] dans la fentêtre graphique*/
if(*flag==2)
{
  /*Retourne le numéro de fenetre active dans ww*/
  C2F(dr1)("xget","window",&verbose,&ww,&narg,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Assigne le numéro de fenetre wid si nécéssaire*/
  if(ww!=wid)
    C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Assigne la couleur*/
  C2F(dr1)("xset","use color",&graphcolor,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Place le flag wresize pour redimensionner automatiquement la fenetre*/
  C2F(dr1)("xset","wresize",(j=1,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Efface la fenetre courante*/
  C2F(dr1)("xclear","v",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Que fait xstart?*/
  C2F(dr)("xstart","v",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Récupère la légende*/
  for(i=0;i<ipar[11];i++) GRAPH_label_CODE[i]=ipar[12+i];
  buf[ipar[11]]='\0';

  /*Converti les codes de la légende en caractère*/
  C2F(cvstr)(&GRAPH_label_size,&GRAPH_label_CODE[0],&buf[0],(j=1,&j));

  /*Initialise le type de tracé*/
  C2F(dr1)("xset","dashes",(j=0,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

  /*Affectation de style[] et de pos[]*/
  for (j=0;j<ipar[2];j++)
  {
   style[j]=graphcolor;
   for (i=0;i<ipar[9];i++) pos[j*ipar[9]+i]=rpar[4+i];
  }

  /*Execute plot2d*/
  C2F(plot2d)(&pos[0],&u[0],(j=ipar[2],&j),&in_size,&style[0],&strf,&buf,&rect,&nax,0L,0L);

  /*Affiche la legende*/
  Legends((i=1,&i),(j=1,&j),&buf);

  /*Test le bit 1 de option*/
  if((option&1)==1) C2F(xgrid)((j=1,&j));
}
/*le flag4 initialise la fenetre graphique*/
else if(*flag==4)
{
 /* Vérification du driver graphique
  * Le nom du driver est retourné dans la variable name
  */
 C2F(dr1)("xgetdr",name,PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);
 /*fprintf(stderr,"name=%s\n", name);*/

 /*Change le driver en position Rec si nécéssaire*/
 if(name!="Rec")
   C2F(dr1)("xsetdr","Rec",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Je sais pas ce que cela fait sciwin()?!? (j'ai pas trouvé le code)*/
```

```
    C2F(sciwin)();

    /*Retourne le numéro de fenetre active dans ww*/
    C2F(dr1)("xget","window",&verbose,&ww,&narg,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Assigne le numéro de fenetre wid si nécéssaire*/
    if(ww!=wid)
     C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Teste la position de la fenetre et replace la si necessaire*/
    if((wpos[0]>=0)&&(wpos[1]>0))
     C2F(dr1)("xset","wpos",&wpos[0],&wpos[1],PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Teste la dimension de la fenetre et redimensionne la si nécessaire*/
    if((wdim[0]>=0)&&(wdim[1]>0))
     C2F(dr1)("xset","wdim",&wdim[0],&wdim[1],PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Reassigne la fenetre pour forcer les changements*/
    C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Ajuste l'echelle et la forme des axes (? correspond aux options de plot2d)*/
    C2F(setscale2d)(&frect,&rect,"nn",0L);

    /*Assigne la couleur*/
    C2F(dr1)("xset","use color",&graphcolor,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Place le flag wresize pour redimensionner automatiquement la fenetre*/
    C2F(dr1)("xset","wresize",(j=1,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Initialise la fonction logique pour dessiner (?)*/
    C2F(dr1)("xset","alufunction",(j=3,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Efface la fenetre courante*/
    C2F(dr1)("xclear","v",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Que fait xstart?*/
    C2F(dr)("xstart","v",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Récupère la légende*/
    for(i=0;i<ipar[11];i++) GRAPH_label_CODE[i]=ipar[12+i];
    buf[ipar[11]]='\0';

    /*Converti les codes de la légende en caractère*/
    C2F(cvstr)(&GRAPH_label_size,&GRAPH_label_CODE[0],&buf[0],(j=1,&j));

    /*Initialise le type de tracé*/
    C2F(dr1)("xset","dashes",(j=0,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Execute plot2d*/
    C2F(plot2d)(&rect[0],&rect[1],(j=1,&j),(k=1,&k),&graphcolor,&strf,&buf,&rect,&nax,0L,0L);

    /*Applique la légende au titre de la fenetre*/
    C2F(dr)("xname",&buf,PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);
   }
   free(style);
   free(pos);
}
```

4.5 VECTORSCOPE_f - Bloc oscilloscope vectoriel

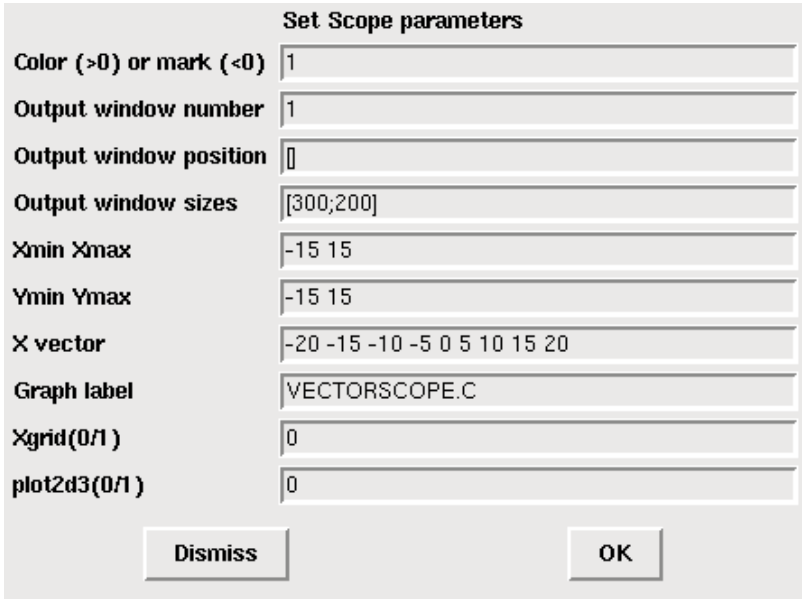


– **Palette :** Sinks.cosf - Palette Sinks
– **Fonction d'interface :** `VECTORSCOPE_f.sci`

4.5.1 Description

Add here a paragraph of the function description.

4.5.2 Boîte de dialogue



– **Color (>0) or mark (<0) :**
Type 'vec' de taille 1. La description du paramètre 1.
– **Output window number :**
Type 'vec' de taille 1. La description du paramètre 2.
– **Output window position :**
Type 'vec' de taille -1. La description du paramètre 3.
– **Output window sizes :**
Type 'vec' de taille -1. La description du paramètre 4.
– **Xmin Xmax :**
Type 'vec' de taille 2. La description du paramètre 5.
– **Ymin Ymax :**
Type 'vec' de taille 2. La description du paramètre 6.
– **X vector :**
Type 'vec' de taille -1. La description du paramètre 7.
– **Graph label :**
Type 'str' de taille 1. La description du paramètre 8.

– **Xgrid(0/1) :**
  Type ’vec’ de taille 1. La description du paramètre 9.
– **plot2d3(0/1) :**
  Type ’vec’ de taille 1. La description du paramètre 10.

4.5.3 Fonction de calcul (type 2)

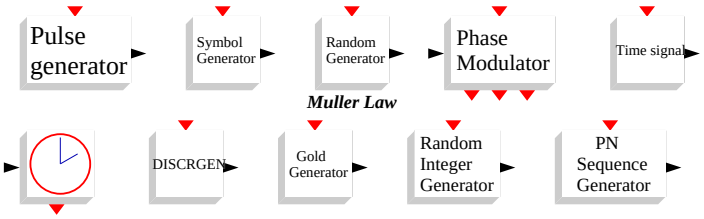
```
/* vectorscope Scicos vector visualization block
 * Originaly write in fortran in the LARY_CR package
 * Rewrite in C with new options
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 23 octobre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#define PI0 (integer *) 0
#define PD0 (double *) 0
#include "machine.h"
#include <stdio.h>

/* Cette fonction de simulation permet d’ouvrir une fenetre graphique
 * et de visualiser la valeur d’un vecteur passer via le port
 * d’entrée u[]. A chaque coup d’horloge, le contenu de la fenetre
 * est effacé et réactualisé avec la valeur du vecteur d’entrée.
 * entrée régulières : u[0..in_size-1] vecteur à visualiser
 * sorties régulières : néant
 * entrée évènementielle : date de déclenchement
 * sortie évènementielle : néant.
 * état discret : z[0..in_size-1] : buffer
 *
 * Integer Parameter
 * ipar[0]    : window number
 * ipar[1]    : color flag
 * ipar[2]    : number of window (buffer_size=ipar[2]*ipar[9])
 * ipar[3]    : dash,color or mark choice
 * ipar[4]    : line or mark size
 * ipar[5..6] : position of window
 * ipar[7..8] : size of window
 * ipar[9]    : in_size (length of window)
 * ipar[10]   : additionnal option
 *                (xgrid (0/1)   : bit 1)
 *                (plot2d3 (0/1) : bit 2)
 * ipar[11]   : GRAPH_label_size (length of title)
 * ipar[12..12+ipar[11]] : Char code of title
 *
 * Real Parameter
 * rpar[0] : xmin
 * rpar[1] : xmax
 * rpar[2] : ymin
 * rpar[3] : ymax
 * rpar[4..4+ipar[6]] : xvector
 */

/*prototype*/
void
vectorscope(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
        ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;

{
   /* Déclaration des variables
    * Variables compteur
    */
   int i,j,k;
   /*Pour paramètres réels*/
   double xmin,xmax,ymin,ymax;
   /*Pour paramatères entiers*/
   int wid,graphcolor,wpos[2],wdim[2],in_size,option;
   int GRAPH_label_size,GRAPH_label_CODE[40];
   /*Autres*/
   char name[4]; /*Chaine de caractères pour le nom du driver graphique*/
   char strf[4]; /*Chaine de caractère pour controle affichage*/
   char buf[41]; /*Chaine pour recevoir la légende*/
   int nax[4]; /*Tableau entier pour controle des graduations*/
   double rect[4]; /*Tableau double pour les coordonnées de la fenetre*/
   double frect[4]; /*Cela sert à setscale2d*/
   int ww,verbose=0,narg; /*Parametres réels pour passage paramètre*/
   /*Pour scicos*/
   int nev;
   double *u;

   /*Récupération de l’adresse du port d’entrée régulier*/
   u=(double *)inptr[0];
   /*Récupération du numéro du port d’activation*/
   nev=*nevprt;

   /*Récupération des paramètres entiers*/
   wid=ipar[0];
   graphcolor=ipar[1];
   wpos[0]=ipar[5];
   wpos[1]=ipar[6];
```

```
wdim[0]=ipar[7];
wdim[1]=ipar[8];
in_size=ipar[9];
option=ipar[10];
GRAPH_label_size=ipar[11];
for(i=0;i<ipar[11];i++) GRAPH_label_CODE[i]=ipar[12+i];
buf[ipar[11]]='\0';

/*Récupération des paramètres réels*/
xmin=rpar[0];xmax=rpar[1]; ymin=rpar[2];ymax=rpar[3];

/*Place xmin,ymin,xmax et ymax dans rect[]*/
rect[0]=xmin;rect[1]=ymin;rect[2]=xmax;rect[3]=ymax;

/*Initialise strf, nax et frec*/
strf[0]='1';strf[1]='1';strf[2]='1';strf[3]='\0';
nax[0]=2;nax[1]=10;nax[2]=2;nax[3]=10;
frect[0]=0;frect[1]=0;frect[2]=1;frect[3]=1;

/* Vérification du driver graphique
 * Le nom du driver est retourné dans la variable name
 */
C2F(dr1)("xgetdr",name,PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);
/*fprintf(stderr,"name=%s\n", name);*/

/*Change le driver en position Rec si nécéssaire*/
if(name!="Rec")
  C2F(dr1)("xsetdr","Rec",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

/*Le flag2 place u[] dans z[] et affiche z[]*/
if(*flag==2)
{
 /*Place u[] dans z[]*/
 for(i=0;i<in_size;i++) z[i]=u[i];

 /*Retourne le numéro de fenetre active dans ww*/
 C2F(dr1)("xget","window",&verbose,&ww,&narg,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Assigne le numéro de fenetre wid si nécéssaire*/
 if(ww!=wid)
  C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Assigne la couleur?*/
 C2F(dr1)("xset","use color",&graphcolor,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Place le flag wresize pour redimensionner automatiquement la fenetre*/
 C2F(dr1)("xset","wresize",(j=1,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Efface la fenetre courante*/
 C2F(dr1)("xclear","v",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Que fait xstart?*/
 C2F(dr)("xstart","v",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Converti les codes de la légende en caractère*/
 C2F(cvstr)(&GRAPH_label_size,&GRAPH_label_CODE[0],&buf[0],(j=1,&j));

 /*Initialise le type de tracé*/
 C2F(dr1)("xset","dashes",(j=0,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Test le bit 2 de option*/
 if(((option&2)>>1)==1)
     /*Lance plot2d3*/
     C2F(plot2d3)("gnn",&rpar[4],&z[1],(j=1,&j),&in_size,&graphcolor,&strf,&buf,&rect,&nax,0L,0L,0L);
 else
     /*Lance plot2d*/
     C2F(plot2d)(&rpar[4],&z[1],(j=1,&j),&in_size,&graphcolor,&strf,&buf,&rect,&nax,0L,0L);

 /*Test le bit 1 de option*/
 if((option&1)==1) C2F(xgrid)((j=1,&j));
}
/*le flag4 initialise la fenetre graphique*/
else if(*flag==4)
{
 /*Je sais pas ce que cela fait sciwin()?!? (j'ai pas trouvé le code)*/
 C2F(sciwin)();

 /*Retourne le numéro de fenetre active dans ww*/
 C2F(dr1)("xget","window",&verbose,&ww,&narg,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Assigne le numéro de fenetre wid si nécéssaire*/
 if(ww!=wid)
  C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Teste la position de la fenetre et replace la si necessaire*/
 if((wpos[0]>=0)&&(wpos[1]>0))
  C2F(dr1)("xset","wpos",&wpos[0],&wpos[1],PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Teste la dimension de la fenetre et redimensionne la si nécessaire*/
 if((wdim[0]>=0)&&(wdim[1]>0))
  C2F(dr1)("xset","wdim",&wdim[0],&wdim[1],PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Reassigne la fenetre pour forcer les changements*/
 C2F(dr1)("xset","window",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

 /*Ajuste l'echelle et la forme des axes (? correspond aux options de plot2d)*/
 C2F(setscale2d)(&frect,&rect,"nn",0L);

 /*Assigne la couleur?*/
 C2F(dr1)("xset","use color",&graphcolor,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);
```

```
    /*Place le flag wresize pour redimensionner automatiquement la fenetre*/
    C2F(dr1)("xset","wresize",(j=1,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Initialise la function logique pour dessiner (?)*/
    C2F(dr1)("xset","alufunction",(j=3,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Efface la fenetre courante*/
    C2F(dr1)("xclear","v",PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Que fait xstart?*/
    C2F(dr)("xstart","v",&wid,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Converti les codes de la légende en caractère*/
    C2F(cvstr)(&GRAPH_label_size,&GRAPH_label_CODE[0],&buf[0],(j=1,&j));

    /*Initialise le type de tracé*/
    C2F(dr1)("xset","dashes",(j=0,&j),PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

    /*Lance plot2d*/
    C2F(plot2d)(&rect[0],&rect[1],(j=1,&j),(k=1,&k),&graphcolor,&strf,&buf,&rect,&nax,0L,0L);

    /*Applique la légende au titre de la fenetre (?)*/
    C2F(dr)("xname",&buf,PI0,PI0,PI0,PI0,PI0,PI0,PD0,PD0,PD0,PD0,0L,0L);

   }

}
```

Chapitre 5

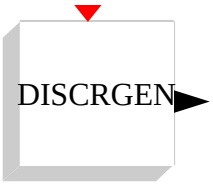
Palette Source



5.0.1 Description

Add here a paragraph of the function description.

5.1 DISCRGEN_f - Bloc générateur discret (PAS ENCORE DISPONIBLE)

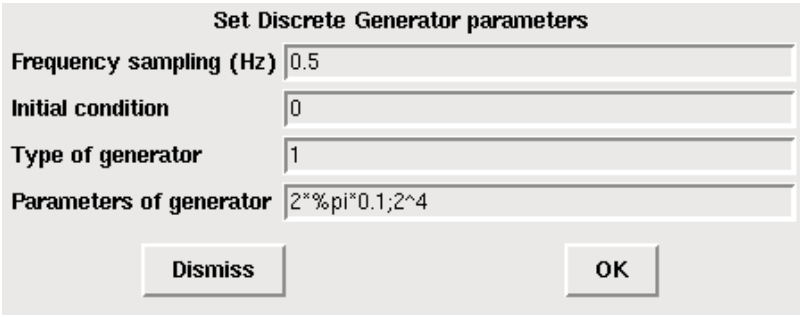


- **Palette :** Sources.cosf - Palette Source
- **Fonction d'interface :** `DISCRGEN_f.sci`

5.1.1 Description

Add here a paragraph of the function description.

5.1.2 Boîte de dialogue



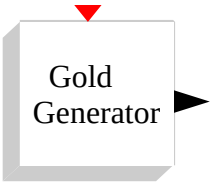
- **Frequency sampling (Hz) :**
  Type 'vec' de taille -1. La description du paramètre 1.
- **Initial condition :**
  Type 'vec' de taille -1. La description du paramètre 2.
- **Type of generator :**
  Type 'vec' de taille -1. La description du paramètre 3.
- **Parameters of generator :**
  Type 'vec' de taille -1. La description du paramètre 4.

5.1.3 Fonction de calcul (type 4)

```
/* discrgen Scicos discrete generator block
 * Type 4 simulation function - scilab-3.0
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include <stdio.h>
#include <math.h>
#define M_PI 3.14159265358979323846

/* rpar[0] : amplitude
 * rpar[1] : longueur de la période (en échantillons)
 * ipar[0] : option (1:cos,2:sin)
 */

/* cwr_c : routine de calcul d'une onde à fréquence constante (1 composante)
 *
 * Entrées :
 * n       : longueur du vecteur
 * ampl    : amplitude de l'onde
 * opt     : option de la routine (1:cos, 2:sin)
 * theta_i : pas de l'angle
 *
```

```
 * Sorties :
 * y : adresse de départ du vecteur de sortie
 *
 * Entrées/Sorties :
 * theta   : angle instantanée
 *
 * Dépendances :
 * math.h
 *
*/
void cwr_c(int *n,double *ampl,int *opt,double *theta_i,double *y,double *theta)
{
 /*déclaration*/
 int i;

 for (i=0;i<(*n);i++)
 {
  switch (*opt)
  {
   case 1 :
   {
    y[i]=(*ampl)*cos((*theta));
    break;
   }
   case 2 :
   {
    y[i]=(*ampl)*sin((*theta));
    break;
   }
  }
  *theta=(*theta)+(*theta_i);
 }
 return;
}

void discrgen(scicos_block *block,int flag)
{
 /*déclaration des variables*/
 int i,k;
 int ny;
 int opt;
 double inc; /*incrément de l'angle*/
 double ampl;
 double *y;

 /*récupération de la taille de sortie*/
 ny=block->outsz[0];

 /*récupération de l'adresse de sortie*/
 y=(double *)block->outptr[0];

 /*récupération des paramètres*/
 ampl=block->rpar[0];
 inc=(2*M_PI)/block->rpar[1];
 opt=block->ipar[0];

 if(flag==1)
 {
  /*Appel cw_r*/
  cwr_c(&ny,&ampl,&opt,&inc,&y[0],&block->z[0]);
 }
}
```

5.2 GENGOLD_f - Bloc générateur de séquence de Gold

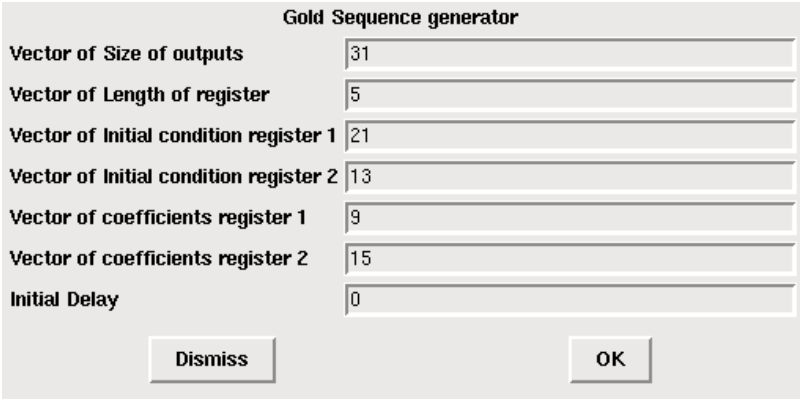


– **Palette :** Sources.cosf - Palette Source
– **Fonction d'interface :** `GENGOLD_f.sci`

5.2.1 Description

Add here a paragraph of the function description.

5.2.2 Boîte de dialogue



– **Vector of Size of outputs :**
Type 'vec' de taille -1. La description du paramètre 1.
– **Vector of Length of register :**
Type 'vec' de taille -1. La description du paramètre 2.
– **Vector of Initial condition register 1 :**
Type 'vec' de taille -1. La description du paramètre 3.
– **Vector of Initial condition register 2 :**
Type 'vec' de taille -1. La description du paramètre 4.
– **Vector of coefficients register 1 :**
Type 'vec' de taille -1. La description du paramètre 5.
– **Vector of coefficients register 2 :**
Type 'vec' de taille -1. La description du paramètre 6.
– **Initial Delay :**
Type 'vec' de taille 1. La description du paramètre 7.

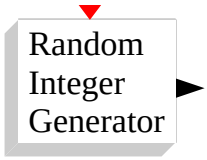
5.2.3 Fonction de calcul (type 4)

```
/* Gengold Scicos Gold Sequences Generator block
 * Type 4 simulation function ver 1.1 - scilab-3.0
 * 15 décembre 2004 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>
```

```
#include "scicos_block.h"

/* Cette fonction de simulation est un générateur de sequences de gold.
 * Les valeurs de sortie sont soit -1 soit 1 et dependent d'une combinaison linéaire
 * des valeurs des registres internes Z. Les opérations sont réalisées grâce à des
 * opérateurs de manipulations de bit, et les mots binaires sont stockés par leurs valeurs entières.
 * La fonction est capable de délivrer des vecteurs de nb_gen générateurs.
 */

/* Entrée régulière : néant
 * Sorties régulières : y[0..ny[0]-1]                              : sortie du générateur 1
 *                      y[ny[0]..ny[0]+ny[1]-1]                   : sortie du générateur 2
 *                      ...
 *                      y[ny[nb_gen-2]..ny[nb_gen-2]+ny[nb_gen-1]] : sortie du générateur nb_gen
 *
 * Entrée évenementielle : période de déclenchement
 * Sortie évènementielle : néant.
 *
 * Etats discrets : z[0..nb_gen-1]        : valeur des registres à décalage 1
 *                  z[nb_gen..2*nb_gen-1] : valeur des registres à décalage 2
 *
 * paramètres entiers : ipar[0]                    : nb_gen nombre de générateurs
 *                      ipar[1..nb_gen]            : ny[0..nb_gen-1] longeur des vecteurs de sortie des géné
 *                      ipar[nb_gen+1..2*nb_gen]   : longueur des registres (nbre de bascules)
 *                      ipar[2*nb_gen+1..3*nb_gen] : valeurs des coefficients des registres 1
 *                      ipar[3*nb_gen+1..4*nb_gen] : valeurs des coeff des registres 2
 */

void gengold(scicos_block *block,int flag)
{
  /*déclaration des variables*/
  int ny;
  int nb_gen;
  int reg1,reg2,N;
  int coef1,coef2;
  int k;
  int n;
  int y_ptr;
  double *y;

  /*Récupération de l'adresse du port de sortie*/
  y=(double *)block->outptr[0];

  /*Récupération du nombre de générateurs*/
  nb_gen=block->ipar[0];

  /*Fait décalage initial*/
  if(flag==4)
  {
   /*Déclaration*/
   int m,decal;
   /*double y_trash[0];*/
   m=1;

   /*récupère le nombre de décalage initial*/
   decal=(int) block->rpar[0];

   for (k=0;k<nb_gen;k++)
   {
    /*Récupération de la longueur des registres*/
    N=block->ipar[nb_gen+1+k];

    /*Récupération de la valeur du registre 1 et 2*/
    reg1=(int) block->z[k];
    reg2=(int) block->z[nb_gen+k];

    /*Récupération de la valeur des coef*/
    coef1=block->ipar[2*nb_gen+1+k];
    coef2=block->ipar[3*nb_gen+1+k];

    for(n=0;n<decal;n++)
    {
     /*Appel routine gold_c*/
     gold_c(&N,&m,&y[0],&reg1,&reg2,&coef1,&coef2);
    }
    /*Sauvegarde la valeur du registre1 dans z[0] et la valeur du reg 2 dans z[1]*/
    block->z[k]=reg1;
    block->z[nb_gen+k]=reg2;
   }

  }
  /*Seulement sur flag 1*/
  else if(flag==1)
  {
   for (k=0;k<nb_gen;k++)
   {
    /*Définition de l'adresse de départ de la sortie du géné k*/
    if(k!=0) y_ptr=y_ptr+block->ipar[k];
    else y_ptr=0;

    /*Récupération de la longueur du vecteur de sortie*/
    ny=block->ipar[1+k];

    /*Récupération de la longueur des registres*/
    N=block->ipar[nb_gen+1+k];

    /*Récupération de la valeur du registre 1 et 2*/
    reg1=(int) block->z[k];
    reg2=(int) block->z[nb_gen+k];

    /*Récupération de la valeur des coef*/
```

```
    coef1=block->ipar[2*nb_gen+1+k];
    coef2=block->ipar[3*nb_gen+1+k];

    /*Appel routine gold_c*/
    gold_c(&N,&ny,&y[y_ptr],&reg1,&reg2,&coef1,&coef2);

    /*Sauvegarde la valeur du registre1 dans z[0] et la valeur du reg 2 dans z[1]*/
    block->z[k]=reg1;
    block->z[nb_gen+k]=reg2;
   }
  }
}
```

5.3 GENINT_f - Bloc générateur de nombre entier

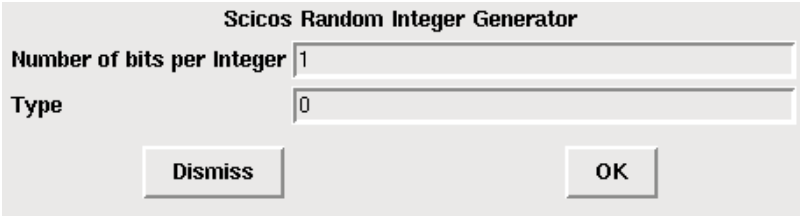


- **Palette :** Sources.cosf - Palette Source
- **Fonction d'interface :** `GENINT_f.sci`

5.3.1 Description

Add here a paragraph of the function description.

5.3.2 Boîte de dialogue



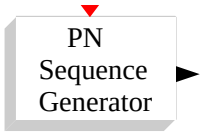
- **Number of bits per Integer :**
  Type 'vec' de taille -1. La description du paramètre 1.
- **Type :**
  Type 'vec' de taille -1. La description du paramètre 2.

5.3.3 Fonction de calcul (type 4)

```
/* genint Scicos Random Integer Generator block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 3 janvier 2005 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <math.h>
#include "scicos_block.h"

/* Ce bloc est générateur de nombre entier aléatoire. Il peut générer des trains binaires
 * type RZ ou NRZ ou bien des nombres entiers de longueurs déterminées, signés ou non signés
 *
 * entrées régulières : néant
 * sorties régulières : sortie du générateur
 * entrée évènementielle : détermine la cadence de génération
 *
 * paramètres entiers : ipar[0..outsz[0]-1] : vecteur du nombre de bit du mot de sortie
 *                      ipar[outsz[0]..2*outsz[0]-1]: vecteur du type de générateur
 *
 * nb : Type 0: génére 1 seul bit codé NRZ
 *           1: génère 1 seul bit codé RZ, ou un mot non signé
 *           2: génère un mot codé code complément à 2
 */

/* genintv_c routine de calcul de mots entiers aléatoires
 *
 * Entrées :
 * n   : taille du  vecteur de sortie (scalaire)
 * m   : longueur des mots binaires des éléments du vecteur de sorties (vecteur)
 * typ : type des générateurs binaires (vecteur)
 * (typ=0: génére 1 seul bit codé NRZ
 *    =1: génère 1 seul bit codé RZ, ou un mot non signé
 *    =2: génère un mot codé code complément à 2)
 * Sorties :
 * y : vecteur de sortie
 *
 * dépendances
 * math.h
```

```
 */
void genintv_c(int *n,int *m,int *typ,double *y)
{
 /*déclaration des variables compteurs*/
 int k,i,j;

 for(i=0;i<(*n);i++)
 {
  switch (typ[i])
  {
   /*Type 0 -> dédié signal NRZ*/
   case 0:
   {
    y[i]=(int)((rand()&1)*2-1);
    break;
   }
   /*Type 1 -> entier non signé*/
   case 1 :
   {
    k=1;
    k=k<<m[i];
    j=rand();
    y[i]=(j&(k-1));
    break;
   }
   /*Type 2 -> entier code complémént à 2*/
   case 2 :
   {
    j=rand();
    j -= 2<<(m[i]-2);
    j &= (2<<(m[i]-1)) - 1;
    j -= 2<<(m[i]-2);
    y[i]=j;
    break;
   }
   break;
  }
 }
 return;
}

/*prototype*/
void genint(scicos_block *block,int flag)
{
    /*Déclaration des variables*/
    double *y;
    int i,j,k;
    int ny;

    /*Récupération de l'adresse du port de sortie régulier*/
    y=(double *)block->outptr[0];

    /*Récupération de la taille du vecteur de sortie*/
    ny=block->outsz[0];

    /*Le flag 1 recopie le vecteur Z dans le vecteur Y*/
    if (flag==1||flag==6)
    {
        k=1;
        F2C(dcopy)(&ny,&block->z[0],&k,&y[0],&k);
    }
    /*Le flag 2 calcul les valeurs ou mots binaires dans Z*/
    else if(flag==2||flag==4)
    {
     /*Appel binv_c*/
     genintv_c(&ny,&block->ipar[0],&block->ipar[ny],&block->z[0]);
    }
}
```

5.4 GENMLLSRS_f - Bloc générateur de séquence PN

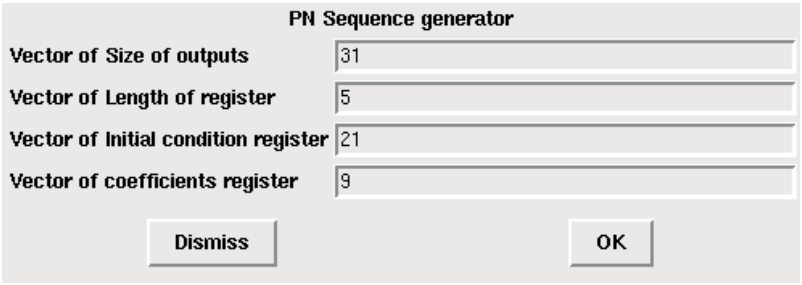


- **Palette :** Sources.cosf - Palette Source
- **Fonction d'interface :** `GENMLLSRS_f.sci`

5.4.1 Description

Add here a paragraph of the function description.

5.4.2 Boîte de dialogue



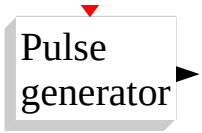
- **Vector of Size of outputs :**
  Type 'vec' de taille -1. La description du paramètre 1.
- **Vector of Length of register :**
  Type 'vec' de taille -1. La description du paramètre 2.
- **Vector of Initial condition register :**
  Type 'vec' de taille -1. La description du paramètre 3.
- **Vector of coefficients register :**
  Type 'vec' de taille -1. La description du paramètre 4.

5.4.3 Fonction de calcul (type 4)

```
/* Genmllsrs Scicos Maximal Length Linear feddback Shift Regiter Sequences Generator block
 * Type 4 simulation function ver 1.1 - scilab-3.0
 * 15 décembre 2004 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>
#include "scicos_block.h"

/* Cette fonction de simulation est un générateur de sequences pseudoaléatoires
 * à longueur maximale.
 * Les valeurs de sortie sont soit -1 soit 1 et dependent d'une combinaison linéaire
 * des valeurs des registres internes Z. Les opérations sont réalisées grâce à des
 * opérateurs de manipulations de bit, et les mots binaires sont stockés par leurs valeurs entières.
 * La fonction est capable de délivrer des vecteurs de nb_gen générateurs.
 *
 * Entrée régulière : néant
 * Sorties régulières : y[0..ny[0]-1]                              : sortie du générateur 1
 *                      y[ny[0]..ny[0]+ny[1]-1]                    : sortie du générateur 2
 *                      ...
 *                      y[ny[nb_gen-2]..ny[nb_gen-2]+ny[nb_gen-1]] : sortie du générateur nb_gen
 *
 * Entrée évenementielle : période de déclenchement
 * Sortie évènementielle : néant.
 *
 * Etats discrets : z[0..nb_gen-1]        : valeur des registres à décalage
 *
 * paramètres entiers : ipar[0]                   : nb_gen nombre de générateurs
```

```
 *                      ipar[1..nb_gen]            : ny[0..nb_gen-1] longeur des vecteurs de sortie des géné
 *                      ipar[nb_gen+1..2*nb_gen]   : longueur des registres (nbre de bascules)
 *                     ipar[2*nb_gen+1..3*nb_gen] : valeurs des coefficients des registres
 */

void genmllsrs(scicos_block *block,int flag)
{
  /*déclaration des variables*/
  int ny;
  int nb_gen;
  int coef;
  int reg,N;
  int i,j,k,l;
  int y_ptr;
  double *y;

  /*Récupération des adresses des ports réguliers*/
  y=(double *)block->outptr[0];

  /*Récupération du nombre de générateurs*/
  nb_gen=block->ipar[0];

  /*Seulement sur flag 1*/
  if(flag==1)
  {
   for(k=0;k<nb_gen;k++)
   {
    /*Définition de l'adresse de départ de la sortie du géné k*/
    if(k!=0) y_ptr=y_ptr+block->ipar[k];
    else y_ptr=0;

    /*Récupération de la longueur du vecteur de sortie*/
    ny=block->ipar[1+k];

    /*Récupération de la longueur du registre*/
    N=block->ipar[nb_gen+1+k];

    /*Récupération de la valeur du registre*/
    reg=(int) block->z[k];

    /*Récupération de la valeur des coef*/
    coef=block->ipar[2*nb_gen+1+k];

     /*Appel routine mllsrs_c*/
    mllsrs_c(&N,&ny,&y[y_ptr],&reg,&coef);

    /*Sauvegarde la valeur du registre dans z[0]*/
    block->z[k] = reg;
   }
  }
}
```

5.5 GENPULSE_f - Bloc générateur d'impulsion

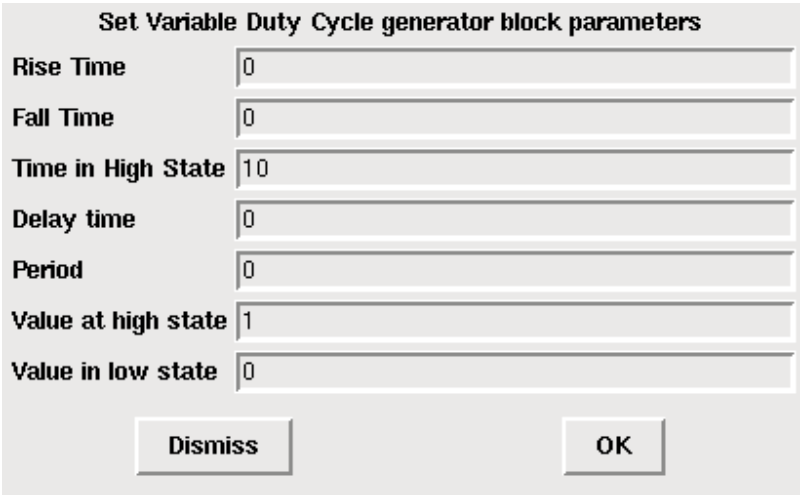


- **Palette :** Sources.cosf - Palette Source
- **Fonction d'interface :** `GENPULSE_f.sci`

5.5.1 Description

Add here a paragraph of the function description.

5.5.2 Boîte de dialogue



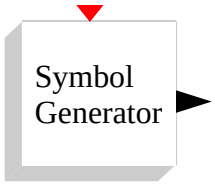
- **Rise Time :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Fall Time :**
  Type 'vec' de taille 1. La description du paramètre 2.
- **Time in High State :**
  Type 'vec' de taille 1. La description du paramètre 3.
- **Delay time :**
  Type 'vec' de taille 1. La description du paramètre 4.
- **Period :**
  Type 'vec' de taille 1. La description du paramètre 5.
- **Value at high state :**
  Type 'vec' de taille 1. La description du paramètre 6.
- **Value in low state :**
  Type 'vec' de taille 1. La description du paramètre 7.

5.5.3 Fonction de calcul (type 2)

```
/* Variable duty cycle pulse generator
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * february 12th 2004 IRCOM Group R. Quere
 */

 /* REVISION HISTORY :
 * $Log$
 */
```

```
#include "machine.h"
#include <math.h>

/* This function returns a variable duty cycle pulse waveform
 *  The parameters of the pulse are in the rpar vector in the following order
 *  rpar[0]=Trise   Rise time of the pulse (default=0)
 *  rpar[1]=Tfall   Fall time of the pulse (default=0)
 *  rpar[2]=Thigh   Time duration in the high state (default=1)
 *  rpar[3]=Td      Time delay of the pulse (default=0)
 *  rpar[4]=Tp      Period of the pulse, if Tp=0 there is only one pulse (default=0)
 *  rpar[5]=Ahigh   Value of the pulse in the high state (default=0)
 *  rpar[6]=Alow    Value of the pulse in the low state (default=0)
 */


void genpulse(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
       ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
  /*déclaration des variables*/
  double *y;
  double t1, t2, t3, t4, tc, tp, Ahigh, Alow;
  int nu,i;

  y=(double *)outptr[0];

  t1=rpar[3];
  t2=t1+rpar[0];
  t3=t2+rpar[2];
  t4=t3+rpar[3];
  tp=rpar[4];
  Ahigh=rpar[5];
  Alow=rpar[6];

  if(*flag==4)
     *y=z[0];
  else if (*flag==1 || *flag==6)
     *y=z[0];
  else if (*flag==2)
   { if (tp == 0)
     {if (*t < t1)
         z[0]=Alow;
      else if (*t<t2)
         z[0]=(*t-t1)*(Ahigh-Alow)/(t2-t1)+Alow;
      else if (*t<t3)
         z[0]=Ahigh;
      else if (*t<t4)
         z[0]=(*t-t3)*(Alow-Ahigh)/(t4-t3)+Ahigh;
      else
        z[0]=0;
     }
     else
     { tc=*t-floor(*t/tp)*tp;
       if (tc < t1)
         z[0]=Alow;
       else if (tc<t2)
         z[0]=(tc-t1)*(Ahigh-Alow)/(t2-t1)+Alow;
       else if (tc<t3)
         z[0]=Ahigh;
       else if (tc<t4)
         z[0]=(tc-t3)*(Alow-Ahigh)/(t4-t3)+Ahigh;
       else
        z[0]=Alow;
     }
   }
}
```

5.6 GENSYMB_f - Bloc générateur de symbole

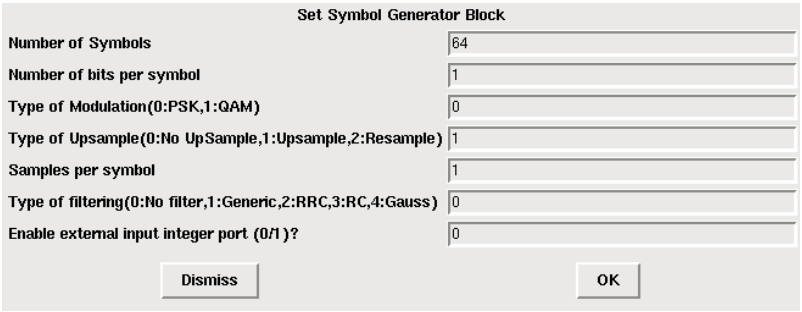


– **Palette :** Sources.cosf - Palette Source
– **Fonction d'interface :** `GENSYMB_f.sci`

5.6.1 Description

Add here a paragraph of the function description.

5.6.2 Boîte de dialogue



– **Number of Symbols :**
  Type 'vec' de taille 1. La description du paramètre 1.
– **Number of bits per symbol :**
  Type 'vec' de taille 1. La description du paramètre 2.
– **Type of Modulation(0 :PSK,1 :QAM) :**
  Type 'vec' de taille 1. La description du paramètre 3.
– **Type of Upsample(0 :No UpSample,1 :Upsample,2 :Resample) :**
  Type 'vec' de taille 1. La description du paramètre 4.
– **Samples per symbol :**
  Type 'vec' de taille 1. La description du paramètre 5.
– **Type of filtering(0 :No filter,1 :Generic,2 :RRC,3 :RC,4 :Gauss) :**
  Type 'vec' de taille 1. La description du paramètre 6.
– **Enable external input integer port (0/1) ? :**
  Type 'vec' de taille 1. La description du paramètre 7.

5.6.3 Fonction de calcul (type 4)

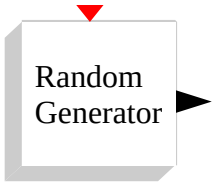
```
/* genmsymb Scicos Generic Symbol Generator
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 5 janvier 2005 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include "machine.h"
#include <math.h>
#include <stdio.h>

/* entrées régulières      : u[0..insz[0]-1]  : vecteur des numéros symboles
 * sorties régulières      : y1[0..nech*insz[0]-1] : vecteur des composantes I
```

```
 *                          y2[0..nech*insz[0]-1] : vecteur des composantes Q
 * entrée évènementielles  : instants de déclenchement.
 * sortie évènementielle   : néant
 * paramètres entiers      : ipar[0] :nombre de symbole à générer
 *                           ipar[1] :longueur en bit des symboles
 *                           ipar[2] :m nombre d'états
 *                           ipar[3] :nech nombre d'échantillons
 *                           ipar[4] :compteur initial du suréchantillonneur
 *                           ipar[5] :nbr de coef du filtre
 *                           ipar[6] :taille du vecteur en puissance de 2 (pour filtre avec fft842)
 *                           ipar[7] :flag générateur séquentiel(0:scalaire/1:Vectoriel)
 *
 * paramètres réels        : rpar[0..nb_coef-1] : vecteur de la réponse impulsionnelle
 * état discret            : flag_seq=0
 *                             z[0..nb_coef-1]         : vecteur du mot mémoire voie I
 *                             z[nb_coef..2*nb_coef-1] : vecteur du mot mémoire voie Q
 *                           flag_seq=1
 *                            z[0] : compteur sur-échantillonneur
 *                            z[1..nb_coef]         : vecteur du mot mémoire voie I
 *                            z[nb_coef+1..2*nb_coef] : vecteur du mot mémoire voie Q
 */

/*prototype*/
void gensymb(scicos_block *block,int flag)
{
 /*Déclaration des variables*/
 double *y1,*y2;
 /*double *u;*/
 double *z;
 int ny;
 int i,k;
 int nu,n,m,m1,nech,init_c,nb_coef;
 int opt;
 int flag_seq;
 double *z__;
 double *zi1_r, *zi1_i, *zi2_r, *zi2_i, *yi_r, *yi_i;
 double *zq1_r, *zq1_i, *zq2_r, *zq2_i, *yq_r, *yq_i;

 /*Récupération des valeurs des variables*/
 ny=block->outsz[0];
 nu=block->ipar[0];
 n=block->ipar[1];
 m=block->ipar[2];
 nech=block->ipar[3];
 init_c=block->ipar[4];
 nb_coef=block->ipar[5];
 m1=block->ipar[6];
 flag_seq=block->ipar[7];

 /*Récupération des adresses des ports réguliers*/
 y1=(double *)block->outptr[0];
 if (n>1) y2=(double *)block->outptr[1];
 z=block->z;
 /* Le flag 1 calcule la valeur de I et de Q en fonction
  * du numéro symbole u[] et du nombre du nombre d'état m
 */
 if(flag==1)
 {
  /********/
  /*Scalar*/
  /********/
  if(flag_seq)
  {
   /*fprintf(stderr,"flag_seq=%d,n=%d,nu=%d,nb_coef=%d,nz=%d\n",flag_seq,n,nu,nb_coef,block->nz);*/
   if((int)block->z[0]==nech)
   {
    block->z[0]=1; /*RAZ compteur*/
    if (nu>0)
    {
     if (n>1)
     {
      /*Appel genint*/
      genint_c(&nu,&n,(k=1,&k),&y1[0]);

      /*Appel routine modpsk*/
      modpsk_c(&nu,&m,&y1[0],&y1[0],&y2[0]);
     }
     else genint_c(&nu,&n,(k=0,&k),&y1[0]);
    }
    else
    {
     /*déclaration*/
     double *u;
     /*Récupération de l'adresse du port d'entrée*/
     u=(double *)block->inptr[0];
     /*récupération de la taille du port d'entrée*/
     nu=block->insz[0];
     /*Recopie u[] dans y1[]*/
     y1[0]=u[0];
     if (n>1)
      /*Appel routine modpsk*/
      modpsk_c(&nu,&m,&y1[0],&y1[0],&y2[0]);
    }
   }
   else
   {
    y1[0]=0;
    if(n>1) y2[0]=0;

    block->z[0]++; /*incrémente compteur*/
   }
```

```
 if (nu<=0)
  /*récupération de la taille du port d'entrée*/
  nu=block->insz[0];

 if (nb_coef!=0)
 {
  /*Appel nfilter*/
  nfilter_c(&nu,&nb_coef,&y1[0],&block->rpar[0],&y1[0],&block->z[1]);
  if (n>1) nfilter_c(&nu,&nb_coef,&y2[0],&block->rpar[0],&y2[0],&block->z[nb_coef+1]);
 }
}
/********/
/*Vector*/
/********/
else
{
 /*fprintf(stderr,"n=%d\n",n);*/
 if(m1!=0) /*fait un test sur filtre*/
 {
  if(n>1) /*fait un test sur nombre d'état*/
  {
   /*Allocation de 2*(3*2) vecteurs de taille m1*/
   if ((*block->work=scicos_malloc(sizeof(double)*(m1*2*(2*3))))== NULL)
   {
    set_block_error(-16);
    return;
   }
  }
  else
  {
   /*Allocation de (3*2) vecteurs de taille m1*/
   if ((*block->work=scicos_malloc(sizeof(double)*(m1*(2*3))))== NULL)
   {
    set_block_error(-16);
    return;
   }
  }
  /*Récupération de l'adresse de départ du vecteur alloué*/
  z__=*block->work;

  /*Déclaration de pointeurs auxiliaires*/
  zi1_r = &(z__[0])   ; zi1_i = &(z__[m1]);   /*vecteur voie i du complexe 1*/
  zi2_r = &(z__[2*m1]); zi2_i = &(z__[3*m1]); /*vecteur voie i du complexe 2*/
  yi_r  = &(z__[4*m1]); yi_i  = &(z__[5*m1]); /*vecteur voie i du complexe résultat*/
  if(n>1)
  {
   zq1_r = &(z__[6*m1]); zq1_i = &(z__[7*m1]);   /*vecteur voie q du complexe 1*/
   zq2_r = &(z__[8*m1]); zq2_i = &(z__[9*m1]); /*vecteur voie q du complexe 2*/
   yq_r  = &(z__[10*m1]); yq_i  = &(z__[11*m1]); /*vecteur voie q du complexe résultat*/
  }
 }
 else
 {
  zi1_r = &(y1[0]); yi_r  = &(z[0]); /*vecteur voie i*/
  if(n>1)
  {
   zq1_r = &(y2[0]); yq_r  = &(z[nb_coef]); /*vecteur voie q*/
  }
 }
 if (nu>0) /*fait un test sur présence port entrée*/
 {
  /*Appel genint_c*/
  if(n>1) genint_c(&nu,&n,(k=1,&k),&zi1_r[0]);
  else genint_c(&nu,&n,(k=0,&k),&zi1_r[0]);
 }
 else if(nu==0)
 {
  /*déclaration*/
  double *u;
  /*Récupération de l'adresse du port d'entrée*/
  u=(double *)block->inptr[0];
  /*récupération de la taille du port d'entrée*/
  nu=block->insz[0];
  /*Recopie u[] dans zi1_r[]*/
  F2C(dcopy)(&nu,&u[0],(k=1,&k),&zi1_r[0],(k=1,&k));
 }

 if(n>1)
 /*Appel routine modpsk*/
  modpsk_c(&nu,&m,&zi1_r[0],&yi_r[0],&yq_r[0]);
 else
  /*Recopie zi1_r[] dans yi_r[]*/
  F2C(dcopy)(&nu,&zi1_r[0],(k=1,&k),&yi_r[0],(k=1,&k));

 /*Appel routine surecht_c*/
 surecht_c((opt=1,&opt),&nu,&nech,&init_c,&yi_r[0],&zi1_r[0]);
 if(n>1) surecht_c((opt=1,&opt),&nu,&nech,&init_c,&yq_r[0],&zq1_r[0]);

 if(m1!=0)
 {
  /*Recopie rpar[] dans zi2_r[] et zq2_r*/
  F2C(dcopy)(&nb_coef,&(block->rpar[0]),(k=1,&k),&zi2_r[0],(k=1,&k));
  if(n>1) F2C(dcopy)(&nb_coef,&(block->rpar[0]),(k=1,&k),&zq2_r[0],(k=1,&k));

  /*Appel convolr_c*/
  convolr_c((k=nu*nech,&k),&nb_coef,&m1,&zi1_r[0],&zi1_i[0],&zi2_r[0],&zi2_i[0],&yi_r[0],&yi_i[0],&z[0]);
  if(n>1) convolr_c((k=nu*nech,&k),&nb_coef,&m1,&zq1_r[0],&zq1_i[0],&zq2_r[0],&zq2_i[0],&yq_r[0],&yq_i[0],&z[nb_coef]);

  /*Recopie y_r[] dans y[]*/
  F2C(dcopy)(&ny,&yi_r[0],(k=1,&k),&y1[0],(k=1,&k));
  if(n>1) F2C(dcopy)(&ny,&yq_r[0],(k=1,&k),&y2[0],(k=1,&k));
```

```
    /*Libère mémoire allouée*/
    scicos_free(*block->work);
   }
  }
 }
}
```

5.7 NOISEBLK_f - Bloc générateur de bruit blanc gaussien



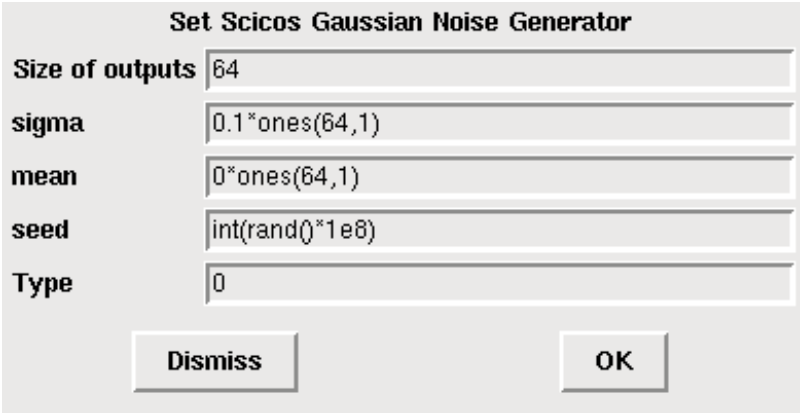
Muller Law

- **Palette :** Sources.cosf - Palette Source
- **Fonction d'interface :** `NOISEBLK_f.sci`

5.7.1 Description

Add here a paragraph of the function description.

5.7.2 Boîte de dialogue



- **Size of outputs :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **sigma :**
  Type 'vec' de taille -1. La description du paramètre 2.
- **mean :**
  Type 'vec' de taille -1. La description du paramètre 3.
- **seed :**
  Type 'vec' de taille 1. La description du paramètre 4.
- **Type :**
  Type 'vec' de taille 1. La description du paramètre 5.

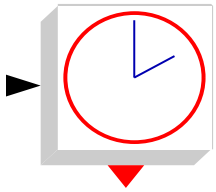
5.7.3 Fonction de calcul (type 4)

```
/* noiseblk Scicos Gaussian random generator block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 22 décembre 2004 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include <math.h>
#include <stdlib.h>  /*pour RAND_MAX*/
#define M_PI 3.14159265358979323846

/* noiseblkv_c routine de calcul d'échantillons bruités par la méthode "Box Muller Law"
 *
 * Entrées :
 * n    : taille des vecteurs
```

```
 * typ  : type de sortie (0:cos/1:sin)
 * sig  : vecteurs des variances
 * mean : vecteurs des moyennes
 * Sorties :
 * y    : vecteur des sorties
 *
 * dépendances
 * math.h
 */
void noiseblkv_c(int *n,int *typ,double *sig,double *mean,double *y_i,double *y_q)
{
 /*déclaration des variables*/
 int i;
 double rand1, rand2;
 double rand_m;
 double ampl, phase;

 /*récupération de la valeur de RAND_MAX*/
 rand_m=RAND_MAX;

 for(i=0;i<(*n);i++)
 {
  /*calcul rand1*/
  rand1=rand()/rand_m;
  /*test rand1*/
  while((rand1<=0)||(rand1>=1)) rand1=rand()/rand_m;

  /*calcul rand2*/
  rand2=rand()/rand_m;
  /*test rand2*/
  while((rand2<=0)||(rand2>=1)) rand2=rand()/rand_m;

  /*Calcul amplitude et phase*/
  ampl=sig[i]*sqrt(-log(rand1));
  phase=2*M_PI*rand2;

  /*Calcul y*/
  switch(*typ)
  {
   case 0 :
   {
    y_i[i]=mean[i]+ampl*cos(phase);
    break;
   }

   case 1 :
   {
    y_q[i]=mean[i]+ampl*sin(phase);
    break;
   }

   case 2 :
   {
    y_i[i]=mean[i]+ampl*cos(phase);
    y_q[i]=mean[i]+ampl*sin(phase);
    break;
   }
  }
 }
 return;
}

void noiseblk(scicos_block *block,int flag)
{
  /*Déclaration des variables*/
  double *y1,*y2;
  int ny,typ;

  /*récupération de l'adresses des ports réguliers*/
  y1=(double *)block->outptr[0];

  /*récupère taille de sortie*/
  ny=block->outsz[0];

  /*récupère le type de générateur*/
  typ=block->ipar[1];

  if(flag==6)
  {
   srand(block->ipar[0]);
  }

  else if(flag==1)
  {
   switch(typ)
   {
    case 0 :
    {
     /*Appel noiseblk_c*/
     noiseblkv_c(&ny,&typ,&block->rpar[0],&block->rpar[ny],&y1[0],y2);
     break;
    }

    case 1 :
    {
     /*Appel noiseblk_c*/
     noiseblkv_c(&ny,&typ,&block->rpar[0],&block->rpar[ny],y2,&y1[0]);
     break;
    }

    case 2 :
```

```
    {
     /*récupération de l'adresse de sortie*/
     y2=(double *)block->outptr[1];
     /*Appel noiseblk_c*/
     noiseblkv_c(&ny,&typ,&block->rpar[0],&block->rpar[ny],&y1[0],&y2[0]);
     break;
    }
   }
  }
}
```

5.8 PCLOCK_f - Bloc générateur d'évènement modulé en phase intégré

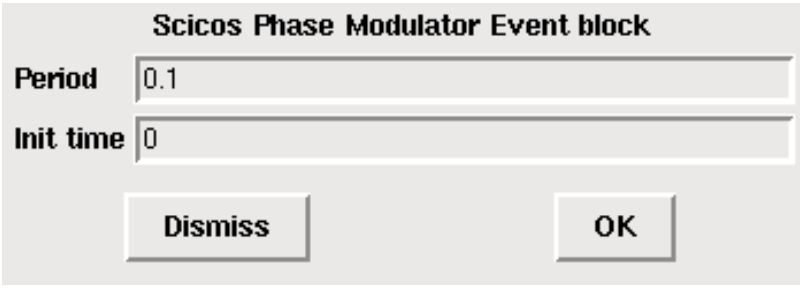


- **Palette :** Sources.cosf - Palette Source
- **Fonction d'interface :** `PCLOCK_f.sci`

5.8.1 Description

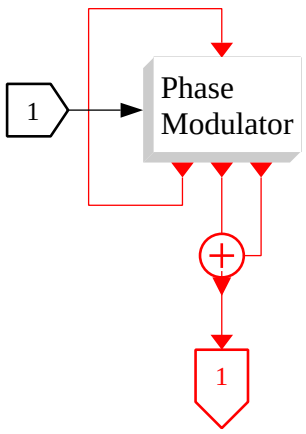
Add here a paragraph of the function description.

5.8.2 Boîte de dialogue

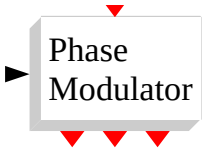


- **Period :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Init time :**
  Type 'vec' de taille 1. La description du paramètre 2.

5.8.3 Contenu du super-bloc compilé



5.9 PEVTDLY_f - Bloc générateur d'évènement modulé en phase

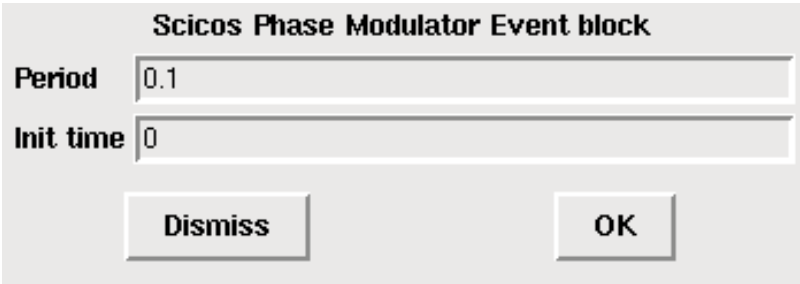


- **Palette :** Sources.cosf - Palette Source
- **Fonction d'interface :** `PEVTDLY_f.sci`

5.9.1 Description

Add here a paragraph of the function description.

5.9.2 Boîte de dialogue



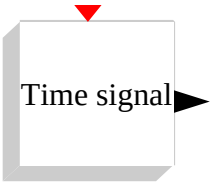
- **Period :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Init time :**
  Type 'vec' de taille 1. La description du paramètre 2.

5.9.3 Fonction de calcul (type 4)

```
/* pevtdly Scicos Phase Modulator Event block
 * Type 2 simulation function ver 1.0 - scilab-3.0
 * 7 octobre 2003 - IRCOM GROUP - Author : A.Layec
 * 14 janvier 2005 : passage type4/rajout option
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include "scicos_block.h"
#include <math.h>
#define M_PI 3.14159265358979323846

/* Cette fonction de simulation calcul des instants de transistion
 * modulée en phase.
 * Entrées régulières : delta_{phi}
 * Sorties régulières : néant
 * Entrées évènementielles : instant de déclenchement
 * Sortie évènementielles : evout[0] : horloge fixe
 *                          evout[1] u tvec[2] : sortie modulée
 * paramères : rpar[0] : période de déclenchement
 *
 * nb : la mémoire z[0] est utilisée pour savoir sur quel port tvec[1]
 *      ou tvec[2] l'événement modulé doit être dirigé.
 */

/*prototype*/
void pevtdly(scicos_block *block,int flag)
{
 /*déclaration*/
 double T; /*période fixe*/
 double *u;
 /*récupération de l'adresse du port d'entrée*/
 u=(double *)block->inptr[0];

 /*récupération des paramètres*/
 T=block->rpar[0];
```

```
 /*le flag 3 calcul les dates de sorties*/
 if (flag==3)
 {
  /*calcul de l'horloge fixe.*/
  block->evout[0]=T;
  /*Test sur valeur d'entrée*/
  if (u[0]<(2*M_PI) && u[0]>-(2*M_PI))
  {
   /*test sur z[0]*/
   if(block->z[0]==0) /*si z[0]=1 alors calcul evout[1]*/
   {
    block->z[0]=1;
    block->evout[1]=T*(1+(u[0]/(2*M_PI)));
   }
   else /*sinon calcul tvec[2]*/
   {
    block->z[0]=0;
    block->evout[2]=T*(1+(u[0]/(2*M_PI)));
   }
  }
 }
}
```

5.10 Time_f - Bloc estimateur temporel

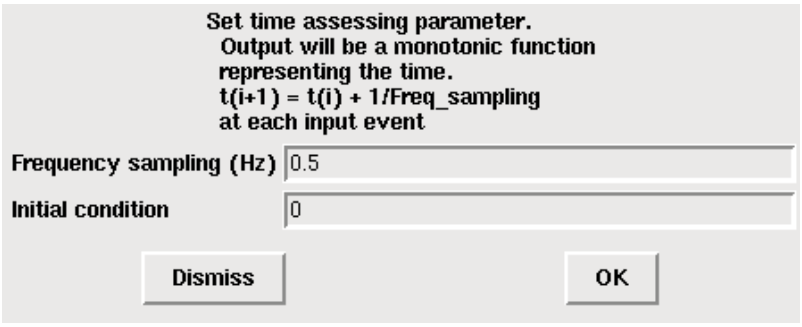


- **Palette :** Sources.cosf - Palette Source
- **Fonction d'interface :** `Time_f.sci`

5.10.1 Description

Add here a paragraph of the function description.

5.10.2 Boîte de dialogue



- **Frequency sampling (Hz) :**
  Type 'vec' de taille -1. La description du paramètre 1.
- **Initial condition :**
  Type 'vec' de taille -1. La description du paramètre 2.

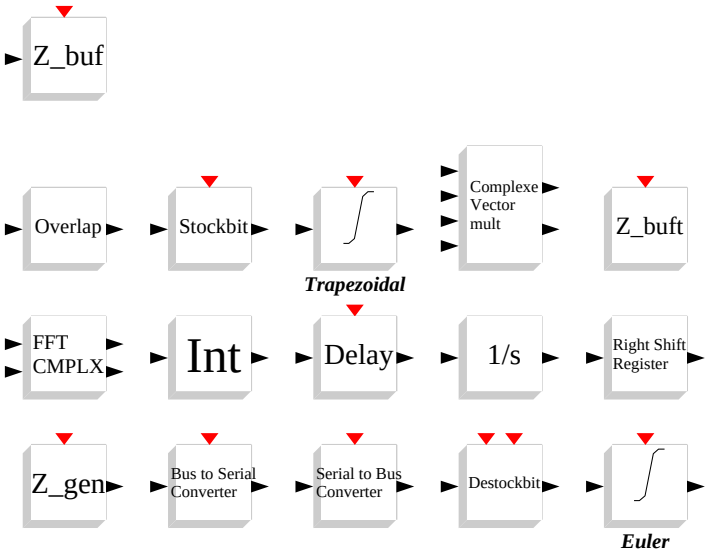
5.10.3 Fonction de calcul (type 4)

```
/* time Scicos time signal block
 * Originaly write in fortran in the LARY_CR package
 * Rewrite in C with capabylity to compute output vector
 * Type 4 simulation function ver 1.0 - scilab-2.6&2.7
 * 19 novembre 2003 - IRCOM GROUP - Author : A.Layec
 * 10 janvier 2004 : passage type 4
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include "scicos_block.h"

/* Cette fonction de simulation délivre la valeur de la variable t
 * à chaque instant de déclenchement :
 * y[k]=y[k-1]+T[k]
 * avec T[k] le vecteur des paramètres et y[k] le vecteur des sortie
 *
 * Entrée régulière : néant.
 * Sorties régulières : y[0..nu-1] : sorties des nu signaux.
 * Entrée évenementielle : période de déclenchement
 * Sortie évènementielle : néant.
 *
 * Etats discrets : z[0..nu-1]    : valeur des y[k-1]
 * paramètres réels : rpar[0..nu-1] : vecteurs des paramètres (T[k]=1/rpar)
 */

/*prototype*/
void time(scicos_block *block,int flag)
{
 /*déclaration des variables*/
 double *y;
 int i,nu;
```

```
 /*Récupération de l'adresse du port de sortie régulier*/
 y=(double *)block->outptr[0];

 /*récupération de la taille du port de sortie*/
 nu=block->outsz[0];

 /*Le flag 1 place le registre z dans le registre y*/
 if(flag==1||flag==6)
 {
  for(i=0;i<nu;i++) y[i]=block->z[i];
 }

 /*la flag 2 calcul l'incrément temporel*/
 else if(flag==2)
 {
  for(i=0;i<nu;i++) block->z[i]=block->z[i]+1/block->rpar[i];
 }
}
```

Chapitre 6

Palette Outils



6.0.1 Description

Add here a paragraph of the function description.

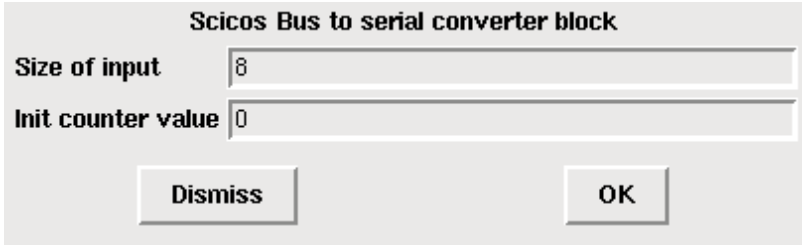
6.1 CONVPS_f - Bloc convertisseur parallèle-série

– **Palette :** Tools.cosf - Palette Outils
– **Fonction d'interface :** `CONVPS_f.sci`

6.1.1 Description

Add here a paragraph of the function description.

6.1.2 Boîte de dialogue



– **Size of input :**
Type 'vec' de taille 1. La description du paramètre 1.
– **Init counter value :**
Type 'vec' de taille 1. La description du paramètre 2.

6.1.3 Fonction de calcul (type 2)

```
/* convsp Scicos Convertisseur parralèle série
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 21 décembre 2003 - IRCOM GROUP - Author : A.Layec
 */

 /* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>

/* Cette fonction réalise la conversion "parallèle-série". Le vecteur d'entrée de taille
 * nu est lu périodiquement par pas de 1 block.
 *
 * entrées régulières : u[0..nu-1] : vecteur d'entrée de taille nu
 * sorties régulières : y[0] : vecteur de sortie de taille 1
 * état discret : z[0] : compteurs échantillons
 */

/*prototype*/
void convps(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
            ipar,nipar,inptr,insz,nin,outptr,outsz,nout)

int *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
  /*Déclaration*/
  double *y;
  double *u;

  /*Récupération des adresses des ports réguliers*/
  y=(double *)outptr[0];
  u=(double *)inptr[0];

  /*Uniquement sur flag 1*/
  if(*flag==1)
  {
```

```
   /*Test valeur compteur échantillons*/
   if((int)z[0]==insz[0])
   {
    /*RAZ compteur*/
    z[0]=0;
    /*lit position 0*/
    y[0]=u[0];
    /*Incremente compteur*/
    z[0]++;
   }
   else
   {
    /*Lit position z[0] de u[]*/
    y[0]=u[(int)z[0]];
    /*Incrémente compteur*/
    z[0]++;
   }
  /*fprintf(stderr,"insz[0]=%d, a=%d, u[a]=%f, y[0]=%f\n", insz[0],a, u[a],y[0]);*/
 }
}
```

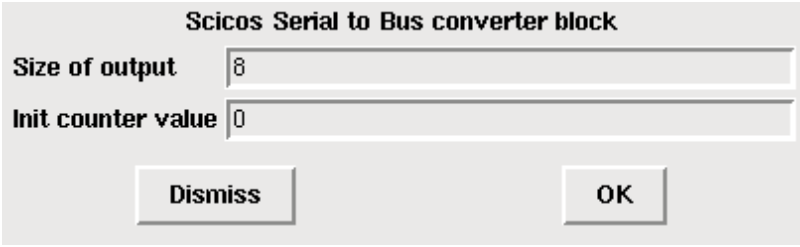
6.2 CONVSP_f - Bloc convertisseur série-parallèle

- **Palette :** Tools.cosf - Palette Outils
- **Fonction d'interface :** `CONVSP_f.sci`

6.2.1 Description

Add here a paragraph of the function description.

6.2.2 Boîte de dialogue



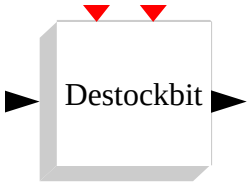
- **Size of output :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Init counter value :**
  Type 'vec' de taille 1. La description du paramètre 2.

6.2.3 Fonction de calcul (type 2)

```
/* convsp Scicos Convertisseur série parralèle
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7&3.0
 * 21 décembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"

/* Cette fonction réalise la conversion "série-parallèle". Le valeur du vecteur d'entrée
 * U[] est stockée dans Z[] et est délivrée lorsque le nombre d'échantillons stocké est N,
 * avec N la taille du port de sortie.
 *
 * entrées régulières : u[0] : valeur d'entrée
 * sorties régulières : y[0..N-1] : vecteur de sortie
 * états discrets : z[0..N-1] : valeur u[k-N,..,k-2,k-1,k]
 *                  z[N] : compteur échantillons
 */

/*prototype*/
void convsp(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
            ipar,nipar,inptr,insz,nin,outptr,outsz,nout)

int *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
  /*Déclaration des variables*/
  double *y;
  double *u;
  int ny,k;

  /*Récupération des adresses de ports réguliers*/
  y=(double *)outptr[0];
  u=(double *)inptr[0];

  /*récupération de la taille du port de sortie*/
```

```
  ny=outsz[0];

  /*Uniquement sur le flag 1*/
  if(*flag==1)
  {
   /*Test valeur compteur échantillons*/
   if((int)z[ny]==outsz[0])
   {
    /*RAZ compteur échantillons*/
    z[ny]=0;
    /*Recopie z dans y*/
    F2C(dcopy)(&ny,&z[0],(k=1,&k),&y[0],(k=1,&k));
    /*Ajout du nouveau point*/
    z[0]=u[0];
    /*Incremente compteur échantillon*/
    z[ny]++;
   }
   else
   {
    /*ajout du nouveau point*/
    z[(int)z[ny]]=u[0];
    /*Incremente compteur échantillon*/
    z[ny]++;
   }
  }
}
```

6.3 DESTOCKBIT_f - Bloc registre à décalage binaire

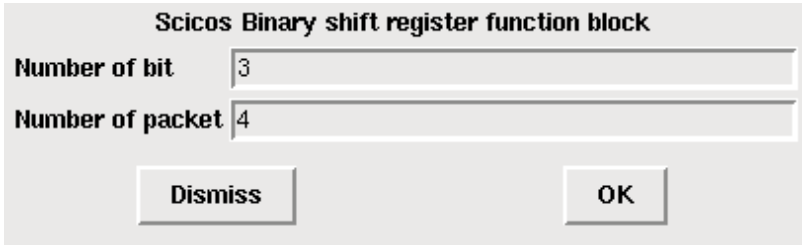


– **Palette :** Tools.cosf - Palette Outils
– **Fonction d'interface :** `DESTOCKBIT_f.sci`

6.3.1 Description

Add here a paragraph of the function description.

6.3.2 Boîte de dialogue



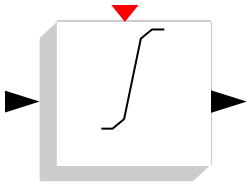
– **Number of bit :**
Type 'vec' de taille -1. La description du paramètre 1.
– **Number of packet :**
Type 'vec' de taille -1. La description du paramètre 2.

6.3.3 Fonction de calcul (type 2)

```
/* destockbit Scicos binary shift register
 * Type 2 simulation function ver 1.1 - scilab-2.6&2.7&3.0
 * 12 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>

/* Cette fonction réalise un registre à décalage binaire.
 * A chaque évènement présent sur port évènementiel numéro 1
 * la fonction réalise une acquistion de la valeur présente
 * sur son port d'entrée régulier supposé etre de taille ipar[0..nu-1]*ipar[nu..2*nu-1]
 * puis délivre un paquet de taille ipar[0..nu-1] sur le port de sortie régulier lorsque
 * un évènement se présente sur son port événementiel 2.
 * A chaque événement 2, un ET logique de ipar[0..nu-1] bits est réalisé
 * et le registre est décalé de ipar[0..nu-1] bits vers la droite.
 *
 * Entrée régulière : u[0..nu-1] vecteur des entiers codés sur Nbit*Nbpaquets
 * Sortie régulière : y[0..nu-1] vecteur des entiers codés sur Nbit
 * Entrée évènementielle : nev=1 : Instant d'acquisition du mot en entrée de taille Nbit*Nbpaquet
 *                         nev=2 : Instant de délivrance du mot en sortie de taille Nbit
 * Sortie évènementielle : néant
 * Paramètres entiers : ipar[0..nu-1] : longueur des paquets du vecteur d'entrée (Nbit)
 *                      ipar[nu..2*nu-1] : Nombre de paquet
 * Mémoires : z[0..nu-1] : valeur paquet
 */

/*prototype*/
void destockbit(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
                ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
  /*déclaration des variables*/
```

```
  double *y;
  double *u;
  int nu,nev,i,j,k;

  /*Récupération des adresses des ports*/
  y=(double *)outptr[0];
  u=(double *)inptr[0];
  nev=*nevprt;

  /*récupération de la taille du port d'entrée régulier*/
  nu=insz[0];

  /* Le flag 1 récupère le paquet à délivrer
   * et délivre sa valeur cc2 dans y[]
   * lorsque le port d'activation est 2 ou 3
   */

  if(*flag==1 && (nev==2||nev==3))
  {
   for(k=0;k<nu;k++)
   {
    /*Récupération paquet entrée*/
    i=(int)z[k];

    /*Tronque valeur de i àux ipar[k] bits de poid faible*/
    j=i&((1<<ipar[k])-1);

    /*Réalise conversion code complément à 2*/
    j-= 2<<(ipar[k]-2);
    j&= (2<<(ipar[k]-1)) - 1;
    j-= 2<<(ipar[k]-2);

    /*place valeur dans y[]*/
    y[k]=j;

    /*Décale paquet de ipar[0] vers la droite*/
    i=i>>ipar[k];

    /*Sauvegarde valeur décalée dans z[k]*/
    z[k]=i;
    /*fprintf(stderr,"z[k]_sauv_bloc2=%f\n", z[k]);*/
   }
  }

  /* Le flag 2 stocke la valeur en entrée dans z[]
   * et la convertie en nombre entier non signé lorsque
   * le numéro du port d'activation est 1 ou 3
   */
  else if(*flag==2 && (nev==1||nev==3))
    {
     for(k=0;k<nu;k++)
     {
      /*aquisition paquet entrée*/
      j=(int)u[k];

      /*Conversion en nombre entier non signé*/
      j &= (2<<((ipar[k]*ipar[nu+k])-1)) - 1;

      /*Sauvegarde paquet entrée dans z[k]*/
      z[k]=j;
     }
    }
}
```

6.4 EULERINTEGRAL_f - Bloc intégrateur à simple pas, méthode Euler

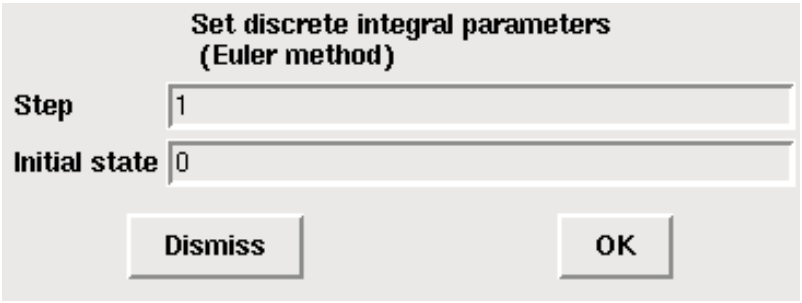


- **Palette :** Tools.cosf - Palette Outils
- **Fonction d'interface :** `EULERINTEGRAL_f.sci`

6.4.1 Description

Add here a paragraph of the function description.

6.4.2 Boîte de dialogue



- **Step :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Initial state :**
  Type 'vec' de taille -1. La description du paramètre 2.

6.4.3 Fonction de calcul (type 2)

```
/* int_euler Scicos discrete integral by euler method block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 17 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"

/* Cette fonction de simulation réalise une intégration discrète
 * du vecteur d'entrée par la méthode d'Euler arrière (dérivée à gauche) :
 * y[k]=h*u[k]+y[k-1]
 * où h est le pas d'intégration, y[k] le vecteur de sortie,
 * et u[k] le vecteur d'entrée
 *
 * entrée régulières : u[0..nu-1] vecteur d'entrée
 * sorties régulières : y[0..nu-1] registre de sortie
 * entrée évènementielle : Instants de déclenchement
 * sortie évènementielle : néant.
 * état discret : z[0..nu-1] vecteur des états de sortie discrets précédents
 *
 * paramètre réel : rpar[0] pas d'intégration
 */

/*prototype*/
void int_euler(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
               ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
  /*déclaration des variables*/
  double *y;
  double *u;
```

```
  int i,nu;

  /*récupération de l'adresses des ports réguliers*/
  y=(double *)outptr[0];
  u=(double *)inptr[0];

  /*Récupération de la taille du port d'entrée*/
  nu=insz[0];

  /*Le flag 1 calcule l'integrale et place le résultat dans le registre y[]*/
  if(*flag==1)
    {
     for(i=0;i<nu;i++) y[i]=rpar[0]*u[i]+z[i];
    }

  /*le flag 2 place l'état de sortie y dans z*/
  else if(*flag==2)
  {
   for(i=0;i<nu;i++) z[i]=y[i];
  }
}
```

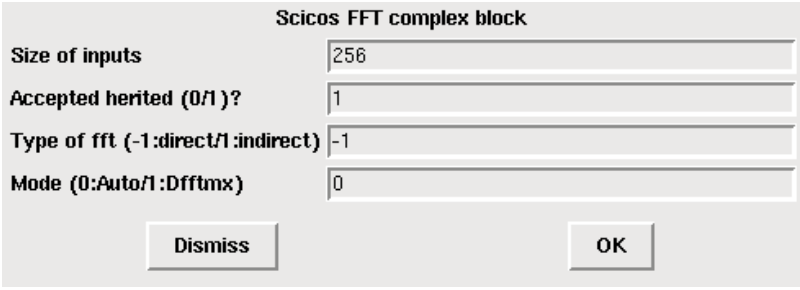
6.5 FFTCMPLX_f - Bloc de Transformation de Fourier Rapide

- **Palette :** Tools.cosf - Palette Outils
- **Fonction d'interface :** `FFTCMPLX_f.sci`

6.5.1 Description

Add here a paragraph of the function description.

6.5.2 Boîte de dialogue



- **Size of inputs :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Accepted herited (0/1) ? :**
  Type 'vec' de taille 1. La description du paramètre 2.
- **Type of fft (-1 :direct/1 :indirect) :**
  Type 'vec' de taille 1. La description du paramètre 3.
- **Mode (0 :Auto/1 :Dfftmx) :**
  Type 'vec' de taille 1. La description du paramètre 4.

6.5.3 Fonction de calcul (type 4)

```
/* fftcmplx Scicos fft complexe
 * Type 4 simulation function ver 1.1b - scilab-3.0
 * 15 décembre 2004 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"
#include "machine.h"
#include <stdio.h>

/* Cette fonction réalise la fft complexe du vecteur {u1[];u2[]}.
 * u1 et u2 sont places dans y1 et y2 respectivement par dcopy puis la fft est réalisée.
 * Suivant la valeur de ipar[0] on réalise une fft directe (0 où -1) ou indirecte (1).
 * Suivant la valeur de ipar[1] on appele fft842(ipar[1]=0) où dfftmx(ipar[1]=1) via dfft2.
 * Si dfftmx est choisie alors on place la taille du mot de travail dans ipar[2].
 *
 * entrées réguliéres : u1[0..nu-1] : vecteur des réels du mot complexe d'entrée
 *                      u2[0..nu-1] : vecteur des imaginaires du mot complexe d'entrée
 *
 * sorties régulières : y1[0..nu-1] : vecteur des réels du mot complexe de sortie
 *                      y2[0..nu-1] : vecteur des imaginaires du mot complexe de sortie
 *
 * paramètres entiers : ipar[0] : signe de l'exponentiel (-1 où 0 : fft directe; 1 : fft indirecte)
 *                      ipar[1] : flag choix fft (0:fft842/1:dfftmx)
 *                      ipar[2] : taille du mot de travail pour dfftmx
 */
```

```
/*prototype*/
void fftcmplx(scicos_block *block,int flag)
{ /*fprintf(stderr,"flag=%d\n",flag);*/
  /*Déclaration des fonctions externes*/
  extern void F2C(fft842)();
  extern void F2C(dfft2)();

  /*Déclaration des variables*/
  double *y1,*y2;
  double *u1,*u2;
  int nu,j,k,ierr;
  double *z__;

  /*Récupération des adresses des ports réguliers*/
  y1=(double *)block->outptr[0];
  y2=(double *)block->outptr[1];
  u1=(double *)block->inptr[0];
  u2=(double *)block->inptr[1];

  /*Récupération de la taille du port d'entrée*/
  nu=block->insz[0];

  /*Recopie u dans y*/
  F2C(dcopy)(&nu,&u1[0],(k=1,&k),&y1[0],(k=1,&k));
  F2C(dcopy)(&nu,&u2[0],(k=1,&k),&y2[0],(k=1,&k));

  /*Appel routine fft*/
  switch (block->ipar[1])
  {
   case 0 : /*Appel fft842*/
    {
     /*fprintf(stderr,"fft842 \n");*/
     if (block->ipar[0]==0 | block->ipar[0]==-1)
      k=0;
     else
      k=1;
     F2C(fft842)(&k,&nu,&y1[0],&y2[0],&ierr);
     break;
    }
   case 1 : /*Appel fftmx*/
    {
     /*fprintf(stderr,"dfftmx \n");*/
     if (block->ipar[0]==0 | block->ipar[0]==-1)
      k=-1;
     else
      k=1;
     /*allocation dynamique*/
     if ((*block->work=scicos_malloc(sizeof(double)*block->ipar[2]))== NULL)
      {
       set_block_error(-16);
       return;
      }
     z__=*block->work;
     F2C(dfft2)(&y1[0],&y2[0],(j=1,&j),&nu,(j=1,&j),&k,&ierr,&z__[0],&(block->ipar[2]));
     scicos_free(*block->work);
     break;
    }
  }
}
```

6.6 INTBLK_f - Bloc parite entière

– **Palette :** Tools.cosf - Palette Outils
– **Fonction d'interface :** `INTBLK_f.sci`

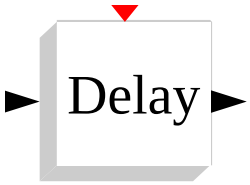
6.6.1 Description

Add here a paragraph of the function description.

6.6.2 Fonction de calcul (type 2)

```
/* intblk Scicos integer block
 * Type 2 simulation function ver 1.1 - scilab-2.6&2.7
 * 18 décembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>

/* Cette fonction de simulation retourne la partie
 * entière du vecteur d'entrée.
 * y=(int)u
 *
 * entrée régulière : u[0..nu-1] vecteur d'entrée
 * sortie régulière : y[0..nu-1] vecteur de sortie
 * entrée et sortie évènementielle : néant
 * paramètre : néant
 */

/*prototype*/
void intblk(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
            ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
  /*déclaration des variables*/
  int i,nu;
  double *y;
  double *u;

  /*Récupération des adresses des ports réguliers*/
  y=(double *)outptr[0];
  u=(double *)inptr[0];

  /*récupération de la taille du port d'entrée*/
  nu=insz[0];

  /*Retourne la valeur entière par (int)u[i] dans le registre de sortie*/
  for(i=0;i<nu;i++) y[i]=(int)u[i];
}
```

6.7 NDELAY_f - Bloc registre à décalage pour signaux multipléxés

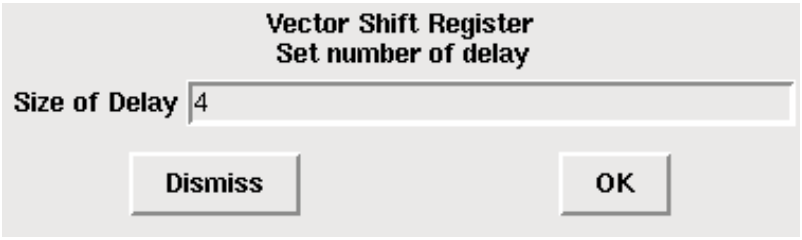


– **Palette :** Tools.cosf - Palette Outils
– **Fonction d'interface :** `NDELAY_f.sci`

6.7.1 Description

Add here a paragraph of the function description.

6.7.2 Boîte de dialogue



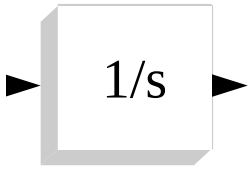
– **Size of Delay :**
Type 'vec' de taille -1. La description du paramètre 1.

6.7.3 Fonction de calcul (type 2)

```
/* ndelay Scicos vector delay block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 18 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"

/* Cette fonction de simulation présente la valeur du vecteur u[]
 * sur le vecteur de sortie y[] après n période de déclenchements.
 * Elle se compose d'un vecteur de registre à décalage dont chaque registre
 * a une taille donnée par le paramètre entier ipar[0..Nu-1] où Nu représente
 * la taille des vecteurs d'entrées/sorties.
 *
 * entrée régulière u[0..Nu-1] : vecteur d'entrée
 * sortie régulièer y[0..Nu-1] : vecteur de sortie
 * entrée évènementielle : période de déclenchement
 * sortie évènementielle : néant
 *
 * paramètre entier ipar[0..Nu-1] :  vecteur des nombres de retard
 * états discrets : z[0..ipar[0]-1] : buffer entrée u[0]
 *                  z[ipar[0]..ipar[1]-1] : buffer entrée u[1]
 *                 ...........
 *                z[ipar[Nu-2]..ipar[Nu-1] : buffer entrée u[Nu-1]
 */

/*prototype*/
void ndelay(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
            ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
 /*Déclaration*/
 double *y;
 double *u;
 int nu,i,j,n;

 /*récupération des adresses des ports réguliers*/
 y = (double *)outptr[0];
```

```
 u = (double *)inptr[0];

 /*récupération de la taille du port d'entrée*/
 nu=insz[0];

 switch (*flag)
 {
  /* Le flag 1 place la première valeur du buffer
   * dans le registre de sortie y[]
   */
  case 1 :
  {
   n = 0;
   for (i=0;i<nu;i++)
   {
    y[i]=z[n];
    n=n+ipar[i];
   }
   break;
  }

  /* Le flag 2 fait un décalage à gauche des buffers
   * et place la valeur de u[] à la sortie des buffers
   */
  case 2 :
  {
   /*Décale buffer*/
   n = 0;
   for (i=0;i<nu;i++)
   {
    for (j=0;j<ipar[i];j++) z[n+j]=z[n+j+1];
    n=n+ipar[i];
   }
   /*Ajoute nouvelle valeur dans buffer*/
   n = 0;
   for(j=0;j<nu;j++)
   {
    n=n+ipar[j];
    z[n-1]=u[j];
   }
   break;
  }
  break;
 }
}
```

6.8 NINTEGRAL_f - Bloc intégrateur continu pour signaux multipléxés

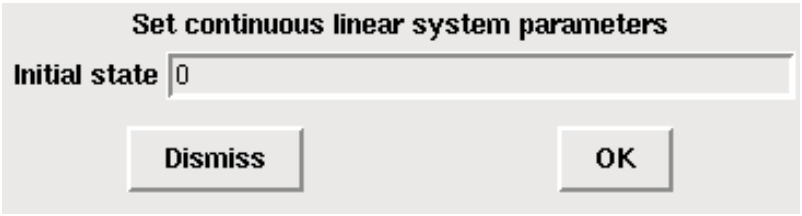


– **Palette :** Tools.cosf - Palette Outils
– **Fonction d'interface :** `NINTEGRAL_f.sci`

6.8.1 Description

Add here a paragraph of the function description.

6.8.2 Boîte de dialogue



– **Initial state :**
Type 'vec' de taille -1. La description du paramètre 1.

6.8.3 Fonction de calcul (type 2)

```
/* nintegr Scicos vector intgerator
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 16 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"

/* Cette fonction de simulation réalise l'intégration simple
 * du vecteur u[] par le solveur ODE intégré et place le résultat dans y[]
 *
 * entrée régulières : u[0..nu-1] vecteur d'entrée
 * sorties régulières : y[0..nu-1] registre de sortie
 * entrée évènementielle : néant (temps dépendant)
 * sortie évènementielle : néant.
 * état continu : u[0..in_size-1] vecteur des états continus
 */

/*Prototype*/
void nintegr(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
             ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
 /*Déclaration*/
 double *y;
 double *u;
 int i,nu;

 /*Récupération des addresses des ports réguliers*/
 y = (double *)outptr[0];
 u = (double *)inptr[0];

 /*Récupération de la taille du port d'entrée régulier*/
 nu=insz[0];

 /*Le flag 1 place le vecteur des états continus x[]*/
 /*dans le registre de sortie y[]*/
 if(*flag == 1)
    for(i=0;i<nu;i++) y[i]=x[i];
```

```
 /*Le flag 2 place les valeurs d'entrée u[]*/
 /*dans le vecteur des dérivées xd[]*/
 else if(*flag ==0)
    for(i=0;i<nu;i++) xd[i]=u[i];
}
```

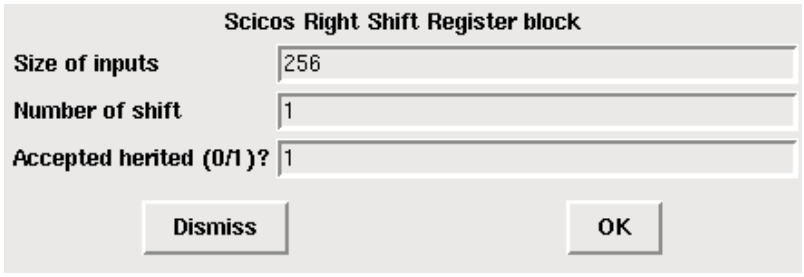
6.9 OVERLAPRSR_f - Bloc registre à décalage discret à mémoire pour signaux multipléxés

– **Palette :** Tools.cosf - Palette Outils
– **Fonction d'interface :** `OVERLAPRSR_f.sci`

6.9.1 Description

Add here a paragraph of the function description.

6.9.2 Boîte de dialogue



– **Size of inputs :**
Type 'vec' de taille 1. La description du paramètre 1.
– **Number of shift :**
Type 'vec' de taille 1. La description du paramètre 2.
– **Accepted herited (0/1) ? :**
Type 'vec' de taille 1. La description du paramètre 3.

6.9.3 Fonction de calcul (type 4)

```
/* overlaprsr Scicos right shift register with memo word block
 * Type 4 simulation function ver 1.0 - scilab-2.6&2.7
 * 24 Décembre 2004 Author : - IRCOM GROUP - A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>
#include "scicos_block.h"

/* Cette fonction de simulation décale le vecteur reçu en entrée de nz
 * cases vers la droite et le recopie en sortie y. L'execendent du mot
 * décalé est stocké dans Z puis placé au début du vecteur suivant.
 *
 * Entrée réguliere : u[0..nu-1] vecteur d'entrée à décaler
 * Sortie reguliere : y[0..nu-1] vecteur de sortie décalé
 *
 * Entrée évènementielle : néant
 * Sortie évènementielle : néant
 *
 * Paramètres entier : insz[0] : taille du vecteur en entrée
 *                     nz : longueur du mot mémoire
 */

/*prototype*/
void overlaprsr(scicos_block *block,int flag)
{
  /*Déclaration des variables*/
```

```
  double *y;
  double *u;
  double *z;
  int nu,i,l;

  /*Récupération des adresses des ports réguliers*/
  y=(double *)block->outptr[0];
  u=(double *)block->inptr[0];
  z=block->z;

  /*Récupération de la taille du port d'entrée*/
  nu=block->insz[0];

  /*Récupération de la longueur du mot mémoire*/
  l=block->nz;

  /* Le flag1 décale à droite le vecteur d'entrée nz fois
   * puis le place dans y et mémorise l'excedent du mot vecteur
   * d'entrée dans z
   */

  if(flag==1)
  {
   /*Appel overlaprsr_c*/
   overlaprsr_c(&nu,&l,&u[0],&y[0],&z[0]);
  }
}
```

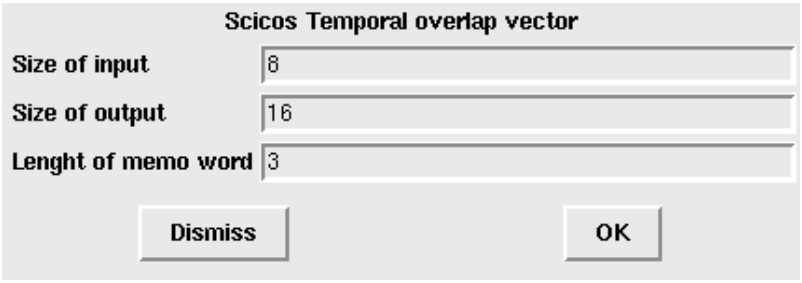
6.10 OVERLAP_f - Bloc empiétement

- **Palette :** Tools.cosf - Palette Outils
- **Fonction d'interface :** `OVERLAP_f.sci`

6.10.1 Description

Add here a paragraph of the function description.

6.10.2 Boîte de dialogue



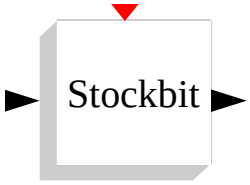
- **Size of input :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Size of output :**
  Type 'vec' de taille 1. La description du paramètre 2.
- **Lenght of memo word :**
  Type 'vec' de taille 1. La description du paramètre 3.

6.10.3 Fonction de calcul (type 4)

```
/* overlap Scicos temporal overlap vector
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 5 décembre 2005 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include "scicos_block.h"

/* Cette fonction transforme un vecteur de taille nu en vecteur de taille ny,
 * avec nu>ny. Elle mémorise l'excés nu-ny et ajoute celui au début du vecteur
 * suivant.
 * Entrées régulières : u[0..nu-1] : vecteur d'entrée de taille nu
 * sorties régulières : y[0..ny-1] : vecteur de sortie de taille ny
 * paramètres entiers : nz : taille du vecteur Z
 */

/*prototype*/
void overlap(scicos_block *block,int flag)
{
  /*déclaration*/
  double *y;
  double *u;
  double *z;
  int i,nu,ny,k,l,ierr;

  /*récupération de l'adresse des ports réguliers*/
  y=(double *)block->outptr[0];
  u=(double *)block->inptr[0];
  z=block->z;
```

```
  /*Récupération de la taille des ports réguliers*/
  nu=block->insz[0];
  ny=block->outsz[0];

  /*La flag 1 calcule y*/
  if(flag==1)
   {
    /*Recopie les ny premiers éléments de u dans y*/
    F2C(dcopy)(&ny,&u[0],(k=1,&k),&y[0],(k=1,&k));

    /*Ajoute les nz éléments précédents au début du vecteur y*/
    for(i=0;i<block->nz;i++) y[i]=u[i]+z[i];
   }
  /*Le flag 2 mais en mémoire l'éxcès nz de u dans Z*/
  else if(flag==2)
  {
   /*Recopie les nz excédentaires de u dans z*/
   F2C(dcopy)((l=block->nz,&l),&u[ny],(k=1,&k),&z[0],(k=1,&k));
  }
}
```

6.11 STOCKBIT_f - Bloc accumulateur de mot binaire

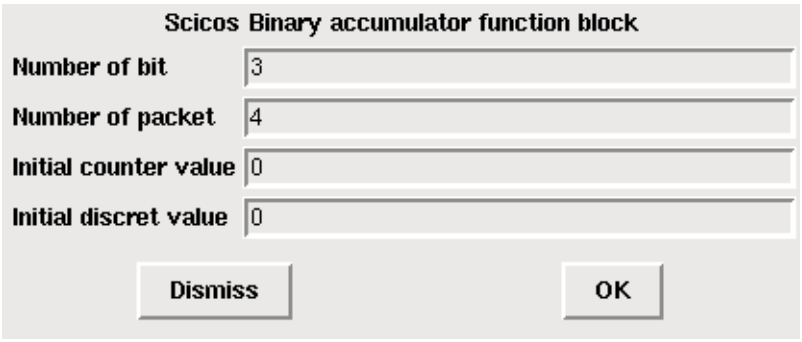


- **Palette :** Tools.cosf - Palette Outils
- **Fonction d'interface :** `STOCKBIT_f.sci`

6.11.1 Description

Add here a paragraph of the function description.

6.11.2 Boîte de dialogue



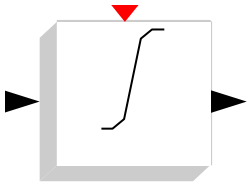
- **Number of bit :**
  Type 'vec' de taille -1. La description du paramètre 1.
- **Number of packet :**
  Type 'vec' de taille -1. La description du paramètre 2.
- **Initial counter value :**
  Type 'vec' de taille -1. La description du paramètre 3.
- **Initial discret value :**
  Type 'vec' de taille -1. La description du paramètre 4.

6.11.3 Fonction de calcul (type 2)

```
/* stockbit Scicos bit accumulator
 * Type 2 simulation function ver 1.1 - scilab-2.6&2.7
 * 12 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include <stdio.h>

/* Cette fonction de simulation réalise un accumulateur de bit
 * Chaque paquet de bit en entrée (de longueur ipar[0..nu-1])
 * est stocké dans un mot de longueur ipar[0..nu-1]*ipar[nu..2*nu-1]
 * où ipar[nu..2*nu-1] représente le nombre de paquets à stocker.
 * Le ième mot est placé à la place i*ipar[0..nu-1] et le mot
 * de sortie de longueur ipar[nu..2*nu-1] est codé en code complément à 2
 *
 * Entrée régulière : u[0..nu-1] vecteur des entiers codés sur Nbit
 * Sortie régulière : y[0..nu-1] vecteur des entiers codés sur Nbit*Nbpaquet
 * Entrée évènementielle : Instant d'acquisition du mot en entrée
 * Sortie évènementielle : néant
 * Paramètres entiers : ipar[0..nu-1] : longueur des paquets du vecteur d'entrée (Nbit)
 *                      ipar[nu..2*nu-1] : Nombre de paquet à stocker
 * Mémoires : z[0..nu-1] : compteur paquet
```

```
 *            z[nu..2*nu-1] : valeur du mot de sortie.
 */

/*prototype*/
void stockbit(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
              ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
  /*déclaration des variables*/
  int i,j,k,nu;
  double *y;
  double *u;

  /*Récupération des adresses des ports réguliers*/
  y=(double *)outptr[0];
  u=(double *)inptr[0];

  /*récupération de la taille du port d'entrée*/
  nu=insz[0];

  if(*flag==1)
  {
   for(k=0;k<nu;k++)
   {
    /*Incrément compteur paquet*/
    z[k]++;

    /*Acquisition de l'entrée dans i*/
    i = (int)u[k];
    /*fprintf(stderr,"stockbit z[k]=%f, u=%f\n", z[k],u[k]);*/

    /*Conversion en nombre entier non signé*/
    i &= (2<<(ipar[k]-1)) - 1;

    /*Décale à gauche le paquet de z[k]*ipar[k] bits*/
    i=i<<(ipar[k]*(int)(z[k]-1));

    /*Incrémente valeur paquet*/
    z[nu+k] = z[nu+k]+i;

    /*Réalise test sur compteur paquet*/
    if (z[k]==ipar[nu+k])
    {
     /*Passe valeur de z[nu+k] dans j*/
     j=(int)z[nu+k];

     /*Réalise conversion code complément à 2*/
     j-= 2<<((ipar[k]*ipar[nu+k])-2);
     j&= (2<<((ipar[k]*ipar[nu+k])-1)) - 1;
     j-= 2<<((ipar[k]*ipar[nu+k])-2);

     /*Place la valeur du paquet dans registre y[]*/
     y[k]=j;
     fprintf(stderr,"stockbit y=%f\n", y[k]);
     /*Met la valeur du compteur paquet à zéro*/
     z[k]=0;

     /*Réinitialise la valeur du paquet de sortie à zéro*/
     z[nu+k]=0;
    }
   }
  }
}
```

6.12 TRAPINTEGRAL_f - Bloc intégrateur à simple pas, méthode Trapèze



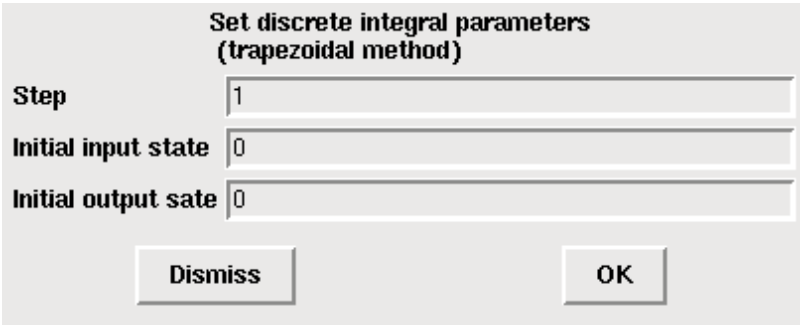
Trapezoidal

- **Palette :** Tools.cosf - Palette Outils
- **Fonction d'interface :** `TRAPINTEGRAL_f.sci`

6.12.1 Description

Add here a paragraph of the function description.

6.12.2 Boîte de dialogue



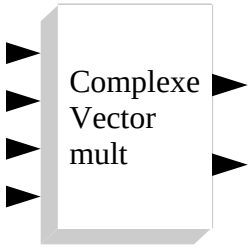
- **Step :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Initial input state :**
  Type 'vec' de taille -1. La description du paramètre 2.
- **Initial output sate :**
  Type 'vec' de taille -1. La description du paramètre 3.

6.12.3 Fonction de calcul (type 2)

```
/* int_trap Scicos discrete integral by trapezoidal method block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 17 novembre 2003 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"

/* Cette fonction de simulation réalise une intégration discrète
 * du vecteur d'entrée par la méthode des trapèzes :
 * y[k]=h/2*(u[k]+u[k-1])+y[k-1]
 * où h est le pas d'intégration, y[k] le vecteur de sortie,
 * et u[k] le vecteur d'entrée
 *
 * entrée régulières : u[0..nu-1] vecteur d'entrée
 * sorties régulières : y[0..nu-1] registre de sortie
 * entrée évènementielle : Instants de déclenchement
 * sortie évènementielle : néant.
 * état discret : z[0..nu-1]    vecteur des états de sortie discrets précédents
 *                z[nu..2*nu-1] vecteur des états d'entrée discrets précédents
 *
 * paramètre réel : rpar[0] pas d'intégration
 *
 */

/*prototype*/
void int_trap(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
              ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
```

```
integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
double x[],xd[],z[],tvec[],rpar[];
double *inptr[],*outptr[],*t;
{
  /*déclaration des variables*/
  double *y;
  double *u;
  int i,nu;

  /*récupération de l'adresses des ports réguliers*/
  y=(double *)outptr[0];
  u=(double *)inptr[0];

  /*Récupération de la taille du port d'entrée*/
  nu=insz[0];

  /*Le flag 1 calcule l'integrale et place le résultat dans le registre y[]*/
  if(*flag==1)
  {
   for(i=0;i<nu;i++) y[i]=(rpar[0]/2)*(u[i]+z[nu+i])+z[i];
  }

  /*le flag 2 place l'état de sortie y et l'état d'entrée dans z*/
  else if(*flag==2)
  {
   for(i=0;i<nu;i++)
   {
    z[i]=y[i];
    z[nu+i]=u[i];
   }
  }
}
```

6.13 VECTMULTCMPLX_f - Bloc multiplicateur vectoriel de signaux complexes

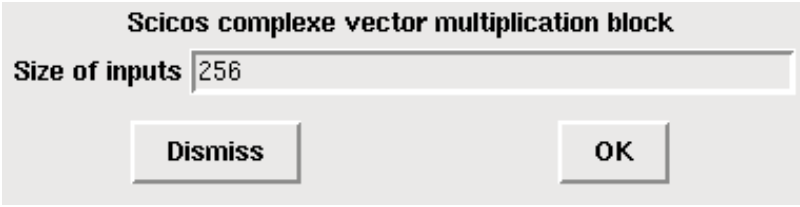


– **Palette :** Tools.cosf - Palette Outils
– **Fonction d'interface :** `VECTMULTCMPLX_f.sci`

6.13.1 Description

Add here a paragraph of the function description.

6.13.2 Boîte de dialogue



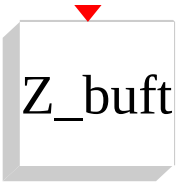
– **Size of inputs :**
Type 'vec' de taille 1. La description du paramètre 1.

6.13.3 Fonction de calcul (type 4)

```
/* vectmultcmplx Scicos complexe vector multiplication
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 3 janvier 2005 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "machine.h"
#include "scicos_block.h"

/* Cette fonction réalise la multiplication de deux vecteurs complexes.
 * y1=u1*u3-u2*u4
 * y2=u1*u4+u2*u3
 * entrées régulières : u1[0..nu-1] : parties réelles du vecteur 1
 *                      u2[0..nu-1] : parties imaginaires du vecteur 1
 *                      u3[0..nu-1] : parties réelles du vecteur 2
 *                      u4[0..nu-1] : parties imaginaires du vecteur 2
 * sorties régulière : y1[0..nu-1] : parties réelles du vecteur de sortie
 *                     y2[0..nu-1] : parties imaginaires du vecteur de sortie
 */

/*prototype*/
void vectmultcmplx(scicos_block *block,int flag)
{
  /*Déclaration*/
  double *y1,*y2;
  double *u1,*u2;
  double *u3,*u4;
  int i,nu;

  /*Récupération des adresses des ports réguliers*/
  y1=(double *)block->outptr[0];
  y2=(double *)block->outptr[1];
  u1=(double *)block->inptr[0];
  u2=(double *)block->inptr[1];
  u3=(double *)block->inptr[2];
  u4=(double *)block->inptr[3];
```

```
  /*Récupération de la taille d'entrée*/
  nu=block->insz[0];

  /*Réalise la multiplication vectorielle complexe*/
  /*Appel cmplxm_c*/
  cmplxm_c(&nu,&u1[0],&u2[0],&u3[0],&u4[0],&y1[0],&y2[0]);
}
```

6.14 ZBUFT_f - Bloc tampon d'enregistrement d'évènement discret

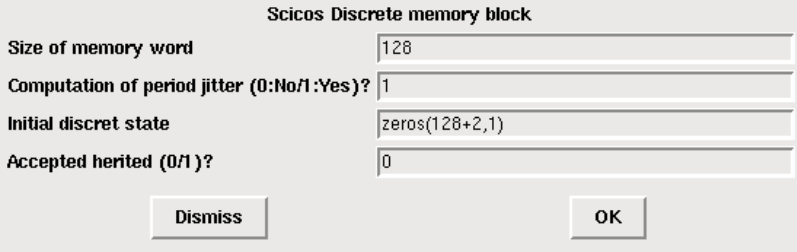


- **Palette :** Tools.cosf - Palette Outils
- **Fonction d'interface :** `ZBUFT_f.sci`

6.14.1 Description

Add here a paragraph of the function description.

6.14.2 Boîte de dialogue



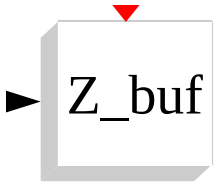
- **Size of memory word :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Computation of period jitter (0 :No/1 :Yes) ? :**
  Type 'vec' de taille 1. La description du paramètre 2.
- **Initial discret state :**
  Type 'vec' de taille -1. La description du paramètre 3.
- **Accepted herited (0/1) ? :**
  Type 'vec' de taille 1. La description du paramètre 4.

6.14.3 Fonction de calcul (type 4)

```
/* z_buft Scicos discret event memory block
 * Type 2 simulation function ver 1.0 - scilab-2.6&2.7
 * 7 Avril 2004 - IRCOM GROUP - Author : A.Layec
 * 4 Mai 2005 : passage type 4
 */

/* REVISION HISTORY :
 * $Log$
 */
 #include "scicos_block.h"

/* Cette fonction de simulation est un buffer qui mémorise
 * les dates de déclenchement dans le vecteur des états discrets z
 * si le paramètre entier ipar[1] est placé à 1 alors la
 * fonction calcul les écarts temporels entre chaques activation
 *
 * entrée régulière : néant
 * sortie régulière : néant
 * entrée évènementielle : Date de déclenchement
 * sortie évènementielle : néant
 *
 * état discret : z[0..n-1] : mot mémoire
 *                z[n]      : date d'activatation précédente
 *                z[n+1]    : compteur échantillon
 * paramètre entier : ipar[0] : longueur du mot mémoire (n)
 *                    ipar[1]  : option : 1 : auto-calcul du jitter period
 */

/*prototype*/
// void
```

```
// z_buft(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,rpar,nrpar,
//          ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
// integer *flag,*nevprt,*nx,*nz,*ntvec,*nrpar,ipar[],*nipar,insz[],*nin,outsz[],*nout;
// double x[],xd[],z[],tvec[],rpar[];
// double *inptr[],*outptr[],*t;
void z_buft(scicos_block *block,int flag)
{
 /*déclaration des variables*/
 int i,j,n;
 double t;

 /*le flag 2 mémorise t dans z[]*/
 if(flag==2)
 {
  /*récupération de la longeur du mot mémoire*/
  n=block->ipar[0];
  /*récupère valeur du compteur échantillons*/
  j=(int)block->z[n+1];
  /*récupère la date d'activation*/
  t=get_scicos_time();
  /*met en mémoire dans z ou calcule la période jitter*/
  if (block->ipar[1]==0)
    block->z[j]=t;
  else if (block->ipar[1]==1)
    block->z[j]=t-block->z[n];
  /*met en mémoire la date d'activation*/
  block->z[n]=t;
  /*incrémente compteur échantillons*/
  j++;
  /*Test valeur compteur échantillons*/
  if(j>=n) j=0;
  /*mémorise compteur dans z[n+1]*/
  block->z[n+1]=j;
 }
}
```

6.15 ZBUF_f - Bloc tampon d'enregistrement discret

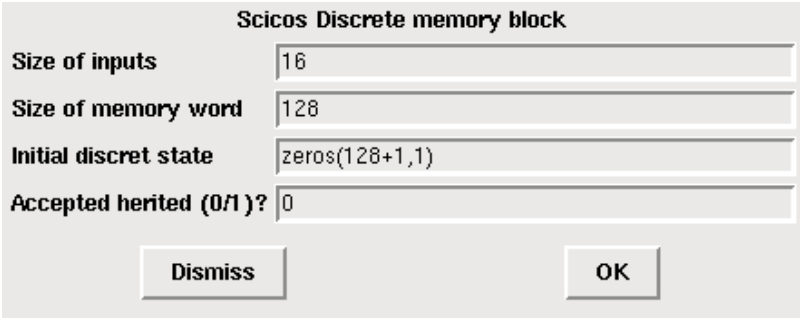


- **Palette :** Tools.cosf - Palette Outils
- **Fonction d'interface :** `ZBUF_f.sci`

6.15.1 Description

Add here a paragraph of the function description.

6.15.2 Boîte de dialogue



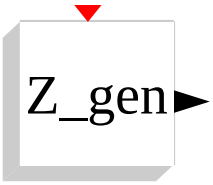
- **Size of inputs :**
  Type 'vec' de taille 1. La description du paramètre 1.
- **Size of memory word :**
  Type 'vec' de taille 1. La description du paramètre 2.
- **Initial discret state :**
  Type 'vec' de taille -1. La description du paramètre 3.
- **Accepted herited (0/1) ? :**
  Type 'vec' de taille 1. La description du paramètre 4.

6.15.3 Fonction de calcul (type 4)

```
/* z_buf Scicos discret memory block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 17 janvier 2005 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"

/* Cette fonction de simulation est un buffer
 * qui mémorise la valeur de son entrée régulière
 * dans le vecteur des états discrets z
 *
 * entrée régulière : u[0..nu-1]
 * sortie régulière : néant
 * entrée évènementielle : Date de déclenchement
 * sortie évènementielle : néant
 *
 * état discret : z[0..ipar[0]-1] : mot mémoire
 *                z[ipar[0]] : compteur
 * paramètre entier : ipar[0] : taille du mot mémoire
 */

/*prototype*/
void z_buf(scicos_block *block,int flag)
```

```
{
 /*déclaration des variables*/
 int i,j;
 /*seulement sur flag 2*/
 if(flag==2)
 {
  /*récupère valeur du compteur échantillons*/
  j=(int)block->z[block->ipar[0]];
  /*met en mémoire la(les) valeur(s) du port réguliers*/
  for(i=0;i<block->insz[0];i++) block->z[j+i]=block->inptr[0][i];
  /*incrémente compteur échantillons*/
  j=j+block->insz[0];
  /*Test valeur compteur échantillons*/
  if(j>=block->ipar[0]) j=0;
  /*mémorise compteur dans z[n+1]*/
  block->z[block->ipar[0]]=j;
 }
}
```

6.16 ZGEN_f - Bloc générateur tampon discret

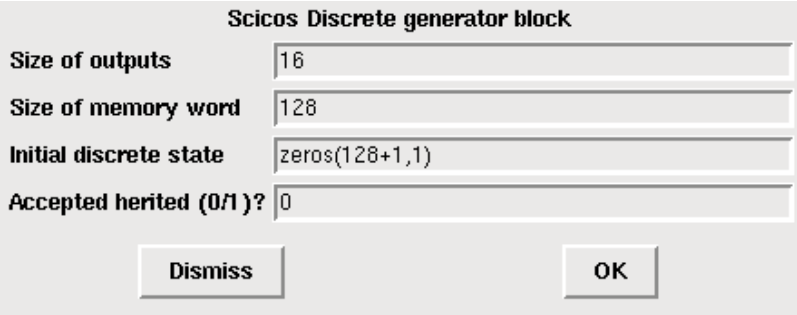


– **Palette :** Tools.cosf - Palette Outils
– **Fonction d'interface :** `ZGEN_f.sci`

6.16.1 Description

Add here a paragraph of the function description.

6.16.2 Boîte de dialogue



– **Size of outputs :**
Type 'vec' de taille 1. La description du paramètre 1.
– **Size of memory word :**
Type 'vec' de taille 1. La description du paramètre 2.
– **Initial discrete state :**
Type 'vec' de taille -1. La description du paramètre 3.
– **Accepted herited (0/1) ? :**
Type 'vec' de taille 1. La description du paramètre 4.

6.16.3 Fonction de calcul (type 4)

```
/* z_buf Scicos discret memory block
 * Type 4 simulation function ver 1.0 - scilab-3.0
 * 17 janvier 2005 - IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "scicos_block.h"

/* ipar[0] : taille du mot mémoire
 * z[0..ipar[0]-1] : mot mémoire
 * z[ipar[0]] : compteur
 */

void z_gen(scicos_block *block,int flag)
{
 int i,j;
 if(flag==1)
  {
   j=(int)block->z[block->ipar[0]];
   for(i=0;i<block->outsz[0];i++) block->outptr[0][i]=block->z[j+i];
   j=j+block->outsz[0];
   if(j>=block->ipar[0]) j=0;
   block->z[block->ipar[0]]=j;
  }
}
```

Deuxième partie

Fonctions Scilab

Chapitre 1

Librairies de fonctions diverses

1.0.1 Description

Add here a paragraph of the function description.

1.1 Dessine un marqueur sur une courbe

- **Nom :** marker
- **Librairie :** misc - Librairies de fonctions diverses

1.1.1 Séquence d'appel

```
marker(h)
```

1.1.2 Paramètres

- **h :** add here the parameter description

1.1.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

1.1.4 Exemple

```
Add here scilab instructions and comments
```

1.1.5 Contenu du fichier

```
//t=0.1:0.1:10-0.1;
//plot(t,sin(t));
//h=gcf();
//
function []=marker(h)
data=h.children(1).children(1).children(1).data;
rep=[];repa=[];p=[];v=[];xx=[];yy=[];
rep(3)=-1;mark_id=1;mark_color=1;sty=5;
while rep(3)<>2 then
  v=[];
  repa=rep;
  rep=xgetmouse();

  //horizontal detection
  v=find(data(:,1)>repa(1) & data(:,1)<rep(1));
  if (size(v,'*'))==0
    v=find(data(:,1)<repa(1) & data(:,1)>rep(1));
  end;
  if (size(v,'*'))>=1
   v=v(1);
   xx=data(v,1);yy=data(v,2);
   if (p<>[]) delete(p); end;
    xpoly(xx,yy,"marks",0);
    p=get("hdl");
    p.mark_style=sty;
```

```
    p.mark_size=5;
    p.foreground=mark_color;
    show_pixmap();
    xinfo("x:"+string(xx)+",y:"+string(yy));
  end;
  if rep(3)==0
     xpoly(xx,yy,"marks",0);
     q=get("hdl");
     q.mark_style=sty;
     q.mark_size=5;
     q.foreground=mark_color;
     show_pixmap();
     disp("x"+string(mark_id)+":"+string(xx)+",y"+string(mark_id)+":"+string(yy))
     mark_color=mark_color+1;mark_id=mark_id+1;
  end;
end;
if (p<>[]) delete(p); end;
endfunction
```

1.1.6 Fonction(s) utilisée(s)

Add here the used function name and references

Chapitre 2

Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.0.1 Description

Add here a paragraph of the function description.

2.1 Ferme la fenêtre graphique n° 0

- **Nom :** del_graphic
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.1.1 Séquence d'appel

```
del_graphic
```

2.1.2 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.1.3 Exemple

```
Add here scilab instructions and comments
```

2.1.4 Contenu du fichier

```
function del_graphic()
while %t
  win=xget("window");
  if win==0 then
    xdel(win);
    win=xget("window");
    if win==0 then
      xdel(win);
      break
    end
  else
  break
  end
end
endfunction
```

2.1.5 Fonction(s) utilisée(s)

Add here the used function name and references

2.2 Exporte les diagrammes scicos (not yet implemented)

- **Nom :** export_to_file
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.2.1 Séquence d'appel

```
export_to_file(diagr,dest)
```

2.2.2 Paramètres

- **diagr :** add here the parameter description
- **dest :** add here the parameter description

2.2.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.2.4 Exemple

```
Add here scilab instructions and comments
```

2.2.5 Contenu du fichier

```
function export_to_file(diagr,dest)
//Load findfile library
//getf(MODNUM+'/macros/util/find_file.sci');

 //find directory path of dname+'.cos' file
 path_dname=return_path_cos_file(diagr);
 if path_dname==[] then
     disp('Unable to find '+dname+'.cos');
     return;
 end

 //load scicos diagram
 load(path_dname+dname+'.cos');

endfunction
```

2.2.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.3 Cherche les numéros des blocs scicos dans une liste Info

- **Nom :** find_num_block
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.3.1 Séquence d'appel

```
n = find_num_block(Info,name)
```

2.3.2 Paramètres

- **Info :** add here the parameter description
- **name :** add here the parameter description
- **n :** add here the parameter description

2.3.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.3.4 Exemple

```
Add here scilab instructions and comments
```

2.3.5 Contenu du fichier

```
//find_num_block
//fonction qui retourne les numéros des blocks
//présents dans une liste Info ayant un nom
//donné.
//
//Entrée : Info liste Info
//         name : nom du block a trouver
//Sortie : n : numéro des blocks dans le diagramme de
//             nom name
function n=find_num_block(Info,name)
%cpr=Info(2)
tt=%cpr.sim("funs");
j=1;
n(j)=0;
for i=1:size(tt)
 if tt(i)==name then
  n(j)=i;
  j=j+1;
  //break;
 else
  //n=0;
 end
end
endfunction
```

2.3.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.4 Crée un repertoire et des scripts de simulation d'un diagramme scicos

- **Nom :** generate_sim_file
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.4.1 Séquence d'appel

```
generate_sim_file(dname)
```

2.4.2 Paramètres

- **dname :** add here the parameter description

2.4.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.4.4 Exemple

```
Add here scilab instructions and comments
```

2.4.5 Contenu du fichier

```
//generate_sim_file
//fonction qui genere un ensemble de fichiers
//pour réaliser des simulations de diagramme scicos
//à partir de la ligne de commande scilab
//Le diagramme scicos est cherché dans les répertoires
//du projet.
//Lorsque le diagramme est trouvé la fonction
//crée un nouveau répertoire dname_sim dans le répertoire
//MODNUM+'/simu'.
//ce répertoire contient un fichier dname_sim.cos, correspondant au
//diagramme à simuler, un fichier dname_sim_ctxt.sce qui correspond
//au context du diagramme et un fichier dname_sim.sce qui correspond
//au script à éxécuter pour simuler le diagramme.
//Entrée : dname le nom du fichier cos à chercher dans MODNUM.
//Sortie : néant
function generate_sim_file(dname)
 //test tt_ml
 if ~exists('tt_ml') then
  tt_ml=return_master_list(MODNUM);
 end

 dname=basename(dname);

 //find directory path of dname+'.cos' file
 path_dname=return_path_cos_file(dname);
 if path_dname==[] then
     disp('Unable to find '+dname+'.cos');
     return;
 end

 //load scicos diagram
 load(path_dname+dname+'.cos');
 context=scs_m.props("context");

 //convert context
 tt_ctxt='scs_m.props.context=[';
 for i=1:size(context,1)
  str=string(context(i));
  if str<>'' then
  str=strsubst(str,'''','''''');
  tt_ctxt=[tt_ctxt;
           '''+str+'''];
  end
 end
 tt_ctxt=[tt_ctxt;'];'];

 //Definition of the target path
 if MSDOS then
  simu_path=MODNUM+'\simu\';
 else
  simu_path=MODNUM+'/simu/';
 end
 if fileinfo(simu_path+dname+'_sim')<>[] then
  ok=%t;j=1;
  while ok
    if fileinfo(simu_path+dname+'_sim'+'_'+string(j))==[] then
     ok=%f;
     path_sim=simu_path+dname+'_sim'+'_'+string(j);
     sim_path=dname+'_sim'+'_'+string(j);
    else
     j=j+1;
    end
  end
 else
    path_sim=simu_path+dname+'_sim';
    sim_path=dname+'_sim';
 end
 //create target path
 if MSDOS then
   mk_cmd='mkdir '+'""'+path_sim+'""';
 else
   mk_cmd='mkdir '+path_sim;
 end
 unix_g(mk_cmd);

 //put cxt in file
 mputl(tt_ctxt,path_sim+'/'+dname+'_sim'+'_ctxt.sce');

 //export diagram file
 scs_m.props.title(1)=dname+'_sim';
 if MSDOS then
  scs_m.props.title(2)=path_sim+'\';
 else
  scs_m.props.title(2)=path_sim+'/';
 end

 //Load scicos library
 exec(load_scicos_func,-1);

 //save diagram
 pal_mode=%f;
 super_block=%f;
 do_save(scs_m);

 //def sript file
 tt_sci=['//Define simulation name';
         'sim_name='''+dname+'_sim'+''';';
         '//Define simulation path';
         'sim_path='''+sim_path+''';';
```

```
        '//Load scicos diagram';
        'load(MODNUM+''/simu/''+sim_path+''/''+sim_name+''.cos'');';
        '//Define context'
        'exec(MODNUM+''/simu/''+sim_path+''/''+sim_name+''_ctxt.sce'');';
        'context=scs_m.props(""context"");execstr(context);';
        '//Define Simulation end time';
        'scs_m.props.tf=Tfin;';
        '//Define other variables';
        '';
        '//Open Log file';
        'u_log=mopen(MODNUM+''/simu/''+sim_path+''/''+sim_name+''.log'',''a'');';
        '//Set flag (for display saved information in scilab window)';
        'flag=1;';
        '//Save inital state of simulation in Log file';
        'wlog_init(u_log,flag);';
        '//Save context in log file';
        'wlog_ctxt(u_log,flag);';
        '//Loop for iterative simulation';
        'j=1;';
        '//Save and display begin simulation date j';
        'wlog_bst(u_log,j,flag);';
        '//substitue context variable to be sweep';
        '//scs_m.props.context=subst_ctxt(scs_m,''varname'',''varname='');';
        '//Initialise Info and %scicos_context variable';
        'if exists(''Info'')==0 then Info=list(); end';
        'if exists(''%scicos_context'')==0 then %scicos_context=struct(); end';
        '//Do simulation with scicos_simulate'
        'Info=scicos_simulate(scs_m,Info,%scicos_context);';
        '//Load result';
        '//myvar=return_state_block(Info,""blockname"")';
        '//Save and display util variable';
        '//wlog_sv(u_log,tlist([''varname''],varvalue),flag);';
        '//Save and Disp final simulation date j';
        'wlog_fst(u_log,j,flag);';
        '//post-processed result';
        '//Save and display post-processed vector(matrix)';
        '//wlog_psv(u_log,tlist([''varname''],varvalue,flag);';
        '//Close final state of simulation in Log file';
        'wlog_final(u_log,flag);';
        '//close log file';
        'mclose(u_log);';
];
//put script text in file
 mputl(tt_sci,path_sim+'/'+dname+'_sim'+'.sce');

endfunction
```

2.4.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.5 Initialisation d'une fenêtre tcl/tk pour les scripts de simulations scilab

- **Nom :** init_ui
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.5.1 Séquence d'appel

```
init_ui
```

2.5.2 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.5.3 Exemple

```
Add here scilab instructions and comments
```

2.5.4 Contenu du fichier

```
function init_ui()
set("figure_style","old");
f = figure("Position",[50 50 500 200],...
     "BackgroundColor",[0.9 0.9 0.9],...
     "Unit", "pixel");
l1 = uicontrol(f,"Position"  , [0 0 500 200],...
 "Style"     , "listbox",...
```

```
 "String"    ,  "",...
 "BackgroundColor",[1 1 1]);
endfunction
```

2.5.5 Fonction(s) utilisée(s)

Add here the used function name and references

2.6 Charge les librairies de scicos

- **Nom :** load_scicos_func
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.6.1 Séquence d'appel

```
load_scicos_func
```

2.6.2 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.6.3 Exemple

```
Add here scilab instructions and comments
```

2.6.4 Contenu du fichier

```
function load_scicos_func()
   set("figure_style","old")
   del_graphic();
   load SCI/macros/scicos/lib
   exec(loadpallibs,-1)
   %tcur=0;%cpr=list();alreadyran=%f;needstart=%t;needcompile=4;%state0=list();
   prot=funcprot();funcprot(0);
   deff('disablemenus()',' ');
   deff('enablemenus()',' ');
   funcprot(prot) ;
endfunction
```

2.6.5 Fonction(s) utilisée(s)

Add here the used function name and references

2.7 Bloc Resistor modifié

- **Nom :** mResistor
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.7.1 Séquence d'appel

```
[x,y,typ] = mResistor(job,arg1,arg2)
```

2.7.2 Paramètres

- **job :** add here the parameter description
- **arg1 :** add here the parameter description
- **arg2 :** add here the parameter description
- **x :** add here the parameter description
- **y :** add here the parameter description
- **typ :** add here the parameter description

2.7.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.7.4 Exemple

```
Add here scilab instructions and comments
```

2.7.5 Contenu du fichier

```
function [x,y,typ]=mResistor(job,arg1,arg2)
// Copyright INRIA
// exemple d'un bloc implicit,
//   -  sans entree ni sortie de conditionnement
//   -  avec une entree et une sortie de type implicit et de dimension 1
//   - avec un dialogue de saisie de parametre
x=[];y=[];typ=[];
select job
case 'plot' then
  R=arg1.graphics.exprs;
  standard_draw(arg1,%f,mResistor_draw_ports)
  //standard_draw(arg1,%f,bidon_draw_ports)
case 'getinputs' then
  [x,y,typ]=mResistor_inputs(arg1)
case 'getoutputs' then
  [x,y,typ]=mResistor_outputs(arg1)
case 'getorigin' then
  [x,y]=standard_origin(arg1)
case 'set' then
  x=arg1;
  graphics=arg1.graphics;exprs=graphics.exprs
  model=arg1.model;
  while %t do
    [ok,R,exprs]=getvalue('Set Resistor block parameter',..
        'R (ohm)',list('vec',1),exprs)
    if ~ok then break,end
    model.rpar=R
    model.equations.parameters(2)=list(R)
    graphics.exprs=exprs
    x.graphics=graphics;x.model=model
    break
  end
case 'define' then
  model=scicos_model()
  R=0.01
  model.rpar=R
  model.sim='resistor'
  model.blocktype='c'
  model.dep_ut=[%t %f]
  mo=modelica()
    mo.model='Resistor'
    mo.inputs='p';
    mo.outputs='n';
    mo.parameters=list('R',list(R))
  model.equations=mo
  model.in=ones(size(mo.inputs,'*'),1)
  model.out=ones(size(mo.outputs,'*'),1)
  exprs=string(R)
  gr_i=[ 'if orient==0|orient==1|orient==%T|orient==%f then';
         'xx=[0,1,1,7,7,8,7,7,1,1]/8;';
         'yy=[1,1,0,0,1,1,1,2,2,1]/2;';
         'rect=xstringl(0,0,''R='+R);'
         'xstring(orig(1)+(sz(1)-rect(3))/2,orig(2)-rect(4)*1.2,''R='+R);';
         'else'
         'yy=[0,1,1,7,7,8,7,7,1,1]/8;';
         'xx=[1,1,0,0,1,1,1,2,2,1]/2;';
         'rect=xstringl(0,0,''R='+R);'
         'xstring(orig(1)+rect(3)*0.5,orig(2)+(sz(2)-rect(4))/2,''R='+R);'
         'end'
         'xpoly(orig(1)+xx*sz(1),orig(2)+yy*sz(2));'
       ]
  x=standard_define([2 0.18],model,exprs,list(gr_i,0))
  x.graphics.in_implicit=['I']
  x.graphics.out_implicit=['I']
end
endfunction
```

```
function mResistor_draw_ports(o)
[orig,sz,orient]=(o.graphics.orig,o.graphics.sz,o.graphics.flip)
xset('pattern',default_color(0))

if orient==0|orient==%F
 out=[0  -1
      1  -1
      1   1
      0   1]*diag([xf/7,yf/14])

 in=[-1  -1
      0  -1
      0   1
     -1   1]*diag([xf/7,yf/14])

 dy=sz(2)/2
 xset('pattern',default_color(1))
 xfpoly(in(:,1)+ones(4,1)*orig(1),in(:,2)+ones(4,1)*(orig(2)+dy),1)
 xpoly(out(:,1)+ones(4,1)*(orig(1)+sz(1)),out(:,2)+ones(4,1)*(orig(2)+dy),"lines",1)
elseif orient==1|orient==%T
 out=[0  -1
     -1  -1
     -1   1
      0   1]*diag([xf/7,yf/14])

 in=[1  -1
     0  -1
     0   1
     1   1]*diag([xf/7,yf/14])
 dy=sz(2)/2
 xset('pattern',default_color(1))
 xfpoly(in(:,1)+ones(4,1)*(orig(1)+sz(1))+1,in(:,2)+ones(4,1)*(orig(2)+sz(2)-dy),1)
 xpoly(out(:,1)+ones(4,1)*orig(1)-1,out(:,2)+ones(4,1)*(orig(2)+dy),"lines",1)
elseif orient==0.5
 in=[-1  -1
      1  -1
      1   1
     -1   1]*diag([xf/14*0.6,yf/7])

 out=[0  -1
      1  -1
      1   1
      0   1]*diag([xf/14,yf/7])
 dx=sz(1)/2
 //xset('pattern',default_color(1))
 xfpoly(in(:,1)+ones(4,1)*(orig(1)+dx),in(:,2)+ones(4,1)*(orig(2)+sz(2)+yf/7),1)
 xpoly(out(:,1)+ones(4,1)*orig(1),out(:,2)+ones(4,1)*(orig(2)-1-yf/7),"lines",1)
elseif orient==1.5
 in=[-1  -1
      1  -1
      1   1
     -1   1]*diag([xf/14*0.6,yf/7])

 out=[1  -1
      0  -1
      0   1
      1   1]*diag([xf/14,yf/7])
 dx=sz(1)/2
 //xset('pattern',default_color(1))
 xfpoly(in(:,1)+ones(4,1)*(orig(1)+dx),in(:,2)+ones(4,1)*(orig(2)-1-yf/7),1)
 xpoly(out(:,1)+ones(4,1)*orig(1),out(:,2)+ones(4,1)*(orig(2)+sz(2)+yf/7),"lines",1)
end
xset('pattern',default_color(0))
endfunction
function [x,y,typ]=mResistor_inputs(arg1)
 xf=60
 yf=40
 orig=arg1.graphics.orig;sz=arg1.graphics.sz;
 orient=arg1.graphics.flip
 dy=sz(2)/2
 dx=sz(1)/2
 if orient==0|orient==%F
  x=orig(1)
  y=orig(2)+dy
 elseif orient==1|orient==%T
  x=orig(1)+sz(1)
  y=orig(2)+sz(2)-dy
 elseif orient==0.5
  x=orig(1)+dx
  y=orig(2)+sz(2)+yf/7
 elseif orient==1.5
  x=orig(1)+dx
  y=orig(2)-1-yf/7
 end
 typ=2
endfunction
function [x,y,typ]=mResistor_outputs(arg1)
 xf=60
 yf=40
 orig=arg1.graphics.orig;sz=arg1.graphics.sz;
 orient=arg1.graphics.flip
 dy=sz(2)/2
 dx=sz(1)/2
 if orient==0|orient==%F
  x=orig(1)+sz(1)
  y=orig(2)+dy
 elseif orient==1|orient==%T
  x=orig(1)-1
  y=orig(2)+dy
 elseif orient==0.5
```

```
  x=orig(1)
  y=orig(2)-1-yf/7
 elseif orient==1.5
  x=orig(1)
  y=orig(2)+sz(2)+yf/7
 end
 typ=2
endfunction
```

2.7.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.8 Fonction do_tild modifiée

- **Nom :** mdo_tild
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.8.1 Séquence d'appel

```
[%pt,scs_m] = mdo_tild(%pt,scs_m)
```

2.8.2 Paramètres

- **%pt :** add here the parameter description
- **scs_m :** add here the parameter description
- **%pt :** add here the parameter description
- **scs_m :** add here the parameter description

2.8.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.8.4 Exemple

```
Add here scilab instructions and comments
```

2.8.5 Contenu du fichier

```
function [%pt,scs_m]=mdo_tild(%pt,scs_m)
// Copyright INRIA
while %t
  if %pt==[] then
    [btn,%pt,win,Cmenu]=cosclick()
    if Cmenu<>[] then
      [%win,Cmenu]=resume(win,Cmenu)
    end
  else
    win=%win;
  end
  xc=%pt(1);yc=%pt(2);%pt=[]
  k=getblock(scs_m,[xc;yc])
  if k<>[] then break,end
end
if get_connected(scs_m,k)<>[] then
  message('Connected block can''t be tilded')
  return
end
o=scs_m.objs(k)
drawobj(o)
if pixmap then xset('wshow'),end
//geom=o(2);geom(3)=~geom(3);o(2)=geom;
pause
mblock=['bidonb';'mResistor';'mResistor2']
geom=o.graphics;
ll=%f;
for i=1:size(mblock,'*')
 if mblock(i)==o.gui then
  geom.flip=geom.flip+0.5;
  if geom.flip==2 then geom.flip=0, end
```

```
  sz_t=geom.sz(1);
  geom.sz(1)=geom.sz(2)
  geom.sz(2)=sz_t
  ll=%t;
  break;
end;
end
if ~ll then geom.flip=~geom.flip, end;
 //geom.flip=~geom.flip;
o.graphics=geom;
drawobj(o)
scs_m_save=scs_m
scs_m.objs(k)=o
[scs_m_save,enable_undo,edited]=resume(scs_m_save,%t,%t)
endfunction
```

2.8.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.9 Fonction getvalue modifiée

- **Nom :** mgetvalue
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.9.1 Séquence d'appel

```
[ok,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18] = mgetvalue(%des
```

2.9.2 Paramètres

- **%desc :** add here the parameter description
- **%lables :** add here the parameter description
- **%typ :** add here the parameter description
- **%ini :** add here the parameter description
- **ok :** add here the parameter description
- **%1 :** add here the parameter description
- **%2 :** add here the parameter description
- **%3 :** add here the parameter description
- **%4 :** add here the parameter description
- **%5 :** add here the parameter description
- **%6 :** add here the parameter description
- **%7 :** add here the parameter description
- **%8 :** add here the parameter description
- **%9 :** add here the parameter description
- **%10 :** add here the parameter description
- **%11 :** add here the parameter description
- **%12 :** add here the parameter description
- **%13 :** add here the parameter description
- **%14 :** add here the parameter description
- **%15 :** add here the parameter description
- **%16 :** add here the parameter description
- **%17 :** add here the parameter description
- **%18 :** add here the parameter description

2.9.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.9.4 Exemple

Add here scilab instructions and comments

2.9.5 Contenu du fichier

```
function [ok,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16, ...
  %17,%18]=mgetvalue(%desc,%lables,%typ,%ini)

 global mydesc
 global mylables
 global mytyp
 global myini

 mydesc=%desc
 mylables=%lables
 mytyp=%typ
 myini=%ini

if %scicos_prob==%t then
ok=%f
        [%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18]=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
return;end
[%lhs,%rhs]=argn(0)

%nn=prod(size(%lables))
if %lhs<>%nn+2&%lhs<>%nn+1 then error(41),end
if size(%typ)<>2*%nn then
  error('typ : list(''type'',[sizes],...)')
end
%1=[];%2=[];%3=[];%4=[];%5=[];%6=[];%7=[];%8=[];%9=[];%10=[];%11=[]; ...
   %12=[];%13=[];%14=[]

if exists('%scicos_context') then
  %mm=getfield(1,%scicos_context)
  for %mi=%mm(3:$)
    if execstr(%mi+'=%scicos_context(%mi)','errcatch')<>0 then
      disp(lasterror())
      ok=%f
      return
    end
  end
end

if %rhs==3 then  %ini=emptystr(%nn,1),end
ok=%t
while %t do
  %str=%ini;
  if %str==[] then ok=%f,break,end
  for %kk=1:%nn
    %cod=ascii(%str(%kk))
    %spe=find(%cod==10)
    if %spe<>[] then
      %semi=ascii(';')
      %cod(%spe)=%semi*ones(%spe')
      %str(%kk)=ascii(%cod)
    end
  end
  %noooo=0
  for %kk=1:%nn
    select part(%typ(2*%kk-1),1:3)
    case 'mat'
      if execstr('%vv=['+%str(%kk)+']','errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)<>1 then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      [%mmmm,%nnnnn]=size(%vv)
      %ssss=string(%sz(1))+' x '+string(%sz(2))
      if %mmmm*%nnnnn==0 then
if  %sz(1)>=0&%sz(2)>=0&%sz(1)*%sz(2)<>0 then %noooo=%kk,break,end
      else
if %sz(1)>=0 then if %mmmm<>%sz(1) then %noooo=%kk,break,end,end
if %sz(2)>=0 then if %nnnnn<>%sz(2) then %noooo=%kk,break,end,end
      end
    case 'vec'
      if execstr('%vv=['+%str(%kk)+']','errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)<>1 then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssss=string(%sz(1))
      %nnnnn=prod(size(%vv))
      if %sz(1)>=0 then if %nnnnn<>%sz(1) then %noooo=%kk,break,end,end
    case 'pol'
      if execstr('%vv=['+%str(%kk)+']','errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)>2 then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssss=string(%sz(1))
      %nnnnn=prod(size(%vv))
      if %sz(1)>=0 then if %nnnnn<>%sz(1) then %noooo=%kk,break,end,end
    case 'row'
      if execstr('%vv=['+%str(%kk)+']','errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)<>1 then %noooo=-%kk,break,end
```

```
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      if %sz(1)<0 then
%ssss='1 x *'
      else
%ssss='1 x '+string(%sz(1))
      end
      [%mmmm,%nnnnn]=size(%vv)
      if %mmmm<>1 then %noooo=%kk,break,end,
      if %sz(1)>=0 then if %nnnnn<>%sz(1) then %noooo=%kk,break,end,end
    case 'col'
      if execstr('%vv=['+%str(%kk)+']','errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)<>1 then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      if %sz(1)<0 then
%ssss='* x 1'
      else
%ssss=string(%sz(1))+' x 1'
      end
      [%mmmm,%nnnnn]=size(%vv)
      if %nnnnn<>1 then %noooo=%kk,break,end,
      if %sz(1)>=0 then if %nnnnn<>%sz(1) then %noooo=%kk,break,end,end
    case 'str'
      %vv=%str(%kk)
      if type(%vv)<>10 then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssss=string(%sz(1))
      %nnnnn=prod(size(%vv))
      if %sz(1)>=0 then if %nnnnn<>1 then %noooo=%kk,break,end,end
    case 'lis'
      if execstr('%vv='+%str(%kk),'errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)<>15& type(%vv)<>16 then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssss=string(%sz(1))
      %nnnnn=size(%vv)
      if %sz(1)>=0 then if %nnnnn<>%sz(1) then %noooo=%kk,break,end,end
    case 'r  '
      if execstr('%vv=['+%str(%kk)+']','errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)<>16 then %noooo=-%kk,break,end
      if typeof(%vv)<>'rational' then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      [%mmmm,%nnnnn]=size(%vv(2))
      %ssss=string(%sz(1))+' x '+string(%sz(2))
      if %mmmm*%nnnnn==0 then
if  %sz(1)>=0&%sz(2)>=0&%sz(1)*%sz(2)<>0 then %noooo=%kk,break,end
      else
if %sz(1)>=0 then if %mmmm<>%sz(1) then %noooo=%kk,break,end,end
if %sz(2)>=0 then if %nnnnn<>%sz(2) then %noooo=%kk,break,end,end
      end
    else
      error('Incorrect type :'+%typ(2*%kk-1))
    end
    execstr('%'+string(%kk)+'=%vv')
  end
  if %noooo>0 then
    message(['answer given for  '+%lables(%noooo);
             'has invalid dimension: ';
             'waiting for dimension  '+%ssss])
    %ini=%str
    ok=%f;break
  elseif %noooo<0 then
    message(['answer given for  '+%lables(-%noooo);
             'has incorrect type :'+ %typ(-2*%noooo-1)])
    %ini=%str
    ok=%f;break
  else
    break
  end
end
if %lhs==%nn+2 then
  execstr('%'+string(%lhs-1)+'=%str')
end
ok=%f
endfunction
```

2.9.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.10 Fonction getvalue modifiée

- **Nom :** mgetvalue2
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.10.1 Séquence d'appel

```
[ok,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18] = mgetvalue2(%de
```

2.10.2 Paramètres

- **%desc :** add here the parameter description
- **%lables :** add here the parameter description
- **%typ :** add here the parameter description
- **%ini :** add here the parameter description
- **ok :** add here the parameter description
- **%1 :** add here the parameter description
- **%2 :** add here the parameter description
- **%3 :** add here the parameter description
- **%4 :** add here the parameter description
- **%5 :** add here the parameter description
- **%6 :** add here the parameter description
- **%7 :** add here the parameter description
- **%8 :** add here the parameter description
- **%9 :** add here the parameter description
- **%10 :** add here the parameter description
- **%11 :** add here the parameter description
- **%12 :** add here the parameter description
- **%13 :** add here the parameter description
- **%14 :** add here the parameter description
- **%15 :** add here the parameter description
- **%16 :** add here the parameter description
- **%17 :** add here the parameter description
- **%18 :** add here the parameter description

2.10.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.10.4 Exemple

```
Add here scilab instructions and comments
```

2.10.5 Contenu du fichier

```
function [ok,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16, ...
  %17,%18]=mgetvalue2(%desc,%lables,%typ,%ini)

 global mydesc
 global mylables
 global mytyp
 global myini

 mydesc=%desc
 mylables=%lables
 mytyp=%typ
 myini=%ini

if %scicos_prob==%t then
ok=%f
        [%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18]=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
return;end
[%lhs,%rhs]=argn(0)

%nn=prod(size(%lables))
if %lhs<>%nn+2&%lhs<>%nn+1 then error(41),end
if size(%typ)<>2*%nn then
  error('typ : list(''type'',[sizes],...)')
end
%1=[];%2=[];%3=[];%4=[];%5=[];%6=[];%7=[];%8=[];%9=[];%10=[];%11=[]; ...
   %12=[];%13=[];%14=[]
```

```
if exists('%scicos_context') then
  %mm=getfield(1,%scicos_context)
  for %mi=%mm(3:$)
    if execstr(%mi+'=%scicos_context(%mi)','errcatch')<>0 then
      disp(lasterror())
      ok=%f
      return
    end
  end
end

if %rhs==3 then  %ini=emptystr(%nn,1),end
ok=%t
while %t do
  %str=%ini;
  if %str==[] then ok=%f,break,end
  for %kk=1:%nn
    %cod=ascii(%str(%kk))
    %spe=find(%cod==10)
    if %spe<>[] then
      %semi=ascii(';')
      %cod(%spe)=%semi*ones(%spe')
      %str(%kk)=ascii(%cod)
    end
  end
  %noooo=0
  for %kk=1:%nn
    select part(%typ(2*%kk-1),1:3)
    case 'mat'
      if execstr('%vv=['+%str(%kk)+']','errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)<>1 then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      [%mmmm,%nnnnn]=size(%vv)
      %ssss=string(%sz(1))+' x '+string(%sz(2))
      if %mmmm*%nnnnn==0 then
if  %sz(1)>=0&%sz(2)>=0&%sz(1)*%sz(2)<>0 then %noooo=%kk,break,end
      else
if %sz(1)>=0 then if %mmmm<>%sz(1) then %noooo=%kk,break,end,end
if %sz(2)>=0 then if %nnnnn<>%sz(2) then %noooo=%kk,break,end,end
      end
    case 'vec'
      if execstr('%vv=['+%str(%kk)+']','errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)<>1 then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssss=string(%sz(1))
      %nnnnn=prod(size(%vv))
      if %sz(1)>=0 then if %nnnnn<>%sz(1) then %noooo=%kk,break,end,end
    case 'pol'
      if execstr('%vv=['+%str(%kk)+']','errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)>2 then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssss=string(%sz(1))
      %nnnnn=prod(size(%vv))
      if %sz(1)>=0 then if %nnnnn<>%sz(1) then %noooo=%kk,break,end,end
    case 'row'
      if execstr('%vv=['+%str(%kk)+']','errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)<>1 then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      if %sz(1)<0 then
%ssss='1 x *'
      else
%ssss='1 x '+string(%sz(1))
      end
      [%mmmm,%nnnnn]=size(%vv)
      if %mmmm<>1 then %noooo=%kk,break,end,
      if %sz(1)>=0 then if %nnnnn<>%sz(1) then %noooo=%kk,break,end,end
    case 'col'
      if execstr('%vv=['+%str(%kk)+']','errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)<>1 then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      if %sz(1)<0 then
%ssss='* x 1'
      else
%ssss=string(%sz(1))+' x 1'
      end
      [%mmmm,%nnnnn]=size(%vv)
      if %nnnnn<>1 then %noooo=%kk,break,end,
      if %sz(1)>=0 then if %nnnnn<>%sz(1) then %noooo=%kk,break,end,end
    case 'str'
      %vv=%str(%kk)
      if type(%vv)<>10 then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssss=string(%sz(1))
      %nnnnn=prod(size(%vv))
      if %sz(1)>=0 then if %nnnnn<>1 then %noooo=%kk,break,end,end
    case 'lis'
      if execstr('%vv='+%str(%kk),'errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)<>15& type(%vv)<>16 then %noooo=-%kk,break,end
```

```
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssss=string(%sz(1))
      %nnnnn=size(%vv)
      if %sz(1)>=0 then if %nnnnn<>%sz(1) then %noooo=%kk,break,end,end
    case 'r  '
      if execstr('%vv=['+%str(%kk)+']','errcatch')<>0  then
%noooo=-%kk,break,
      end
      if type(%vv)<>16 then %noooo=-%kk,break,end
      if typeof(%vv)<>'rational' then %noooo=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      [%mmmm,%nnnnn]=size(%vv(2))
      %ssss=string(%sz(1))+' x '+string(%sz(2))
      if %mmmm*%nnnnn==0 then
if  %sz(1)>=0&%sz(2)>=0&%sz(1)*%sz(2)<>0 then %noooo=%kk,break,end
      else
if %sz(1)>=0 then if %mmmm<>%sz(1) then %noooo=%kk,break,end,end
if %sz(2)>=0 then if %nnnnn<>%sz(2) then %noooo=%kk,break,end,end
      end
    else
      error('Incorrect type :'+%typ(2*%kk-1))
    end
    execstr('%'+string(%kk)+'=%vv')
  end
  if %noooo>0 then
    message(['answer given for  '+%lables(%noooo);
             'has invalid dimension: ';
             'waiting for dimension  '+%ssss])
    %ini=%str
    ok=%f;break
  elseif %noooo<0 then
    message(['answer given for  '+%lables(-%noooo);
             'has incorrect type :'+ %typ(-2*%noooo-1)])
    %ini=%str
    ok=%f;break
  else
    break
  end
end
if %lhs==%nn+2 then
  execstr('%'+string(%lhs-1)+'=%str')
end

endfunction
```

2.10.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.11 Fonction scicos_simulate modifiée

- **Nom :** mscicos_simulate
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.11.1 Séquence d'appel

```
Info = mscicos_simulate(scs_m,Info,%scicos_context,flag,Tbeg,Tfin)
```

2.11.2 Paramètres

- **scs_m :** add here the parameter description
- **Info :** add here the parameter description
- **%scicos_context :** add here the parameter description
- **flag :** add here the parameter description
- **Tbeg :** add here the parameter description
- **Tfin :** add here the parameter description
- **Info :** add here the parameter description

2.11.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.11.4 Exemple

Add here scilab instructions and comments

2.11.5 Contenu du fichier

```
function Info=mscicos_simulate(scs_m,Info,%scicos_context,flag,Tbeg,Tfin)
// Function for running scicos simulation in batch mode
// Info=scicos_simulate(scs_m,Info[,%scicos_context][,flag])
//
// scs_m: scicos diagram (obtained by "load file.cos"). Note that
// the version of file.cos must be the current version. If not, load
// into scicos and save.
//
// %scicos_context: a scilab struct containing values of
// symbolic variables used in the context and Scicos blocks. This
// is often used to change a parameter in the diagram context. In that
// case, make sure that in the diagram context the variable is defined such
// that it can be modified. Say a variable "a" is to be defined in the
// context having value 1, and later in batch mode, we want to change
// the value of "a". In that case, in the context of the diagram place:
//  if ~exists('a') then a=1,end
// If you want then to run the simulation in batch mode using the value
// a=2, set:
// %scicos_context.a=2
//
// Info: a list. It must be list() at the first call, then use output
// Info as input Info for the next calls. Info contains compilation and
// simulation information and is used to avoid recompilation when not
// needed.
//
// flag: string. If it equals 'nw' (no window), then blocks using
// graphical windows are not executed. Note that the list of such
// blocks must be updated as new blocks are added.
//
//
// list of blocks to ignore (blocks using graphics) in nw mode
  Ignoreb=['cscope','cmscope','scope','mscope','scopexy','evscpe','affich']
  //
  load SCI/macros/scicos/lib
  exec(loadpallibs,-1)

  //redefine gui functions
  prot=funcprot();funcprot(0);
  deff('disablemenus()',' ')
  deff('enablemenus()',' ')
  do_terminate=do_terminate1
  funcprot(prot)

  if argn(2)==3 then
    if type(%scicos_context)==10&(stripblanks(%scicos_context)=='nw') then
      Ignore=Ignoreb
      %scicos_context=struct()
    else
      Ignore=[]
    end
  elseif argn(2)==2 then
    Ignore=[]
    %scicos_context=struct()
  elseif stripblanks(flag)=='nw' then
    Ignore=Ignoreb
  else
    Ignore=[]
  end

  //
  if Info<>list() then
    [%tcur,%cpr,alreadyran,needstart,needcompile,%state0]=Info(:)
  else
    %tcur=0;%cpr=list();alreadyran=%f;needstart=%t;needcompile=4;%state0=list();
  end

  //
  tolerances=scs_m.props.tol
  solver=tolerances(6)

  [%scicos_context,ierr]=script2var(scs_m.props.context, ...
    %scicos_context);
  if ierr==0 then
    [scs_m,%cpr,needcompile,ok]=do_eval(scs_m,%cpr)
    if needcompile<>4&size(%cpr)>0 then %state0=%cpr.state,end
    alreadyran=%f
  else
      error(['Incorrect context definition, ';lasterror()] )
  end

  if %cpr==list() then need_suppress=%t, else need_suppress=%f,end

  [%cpr,%state0_n,needcompile,alreadyran,ok]=..
      do_update(%cpr,%state0,needcompile)
  if ~ok then error('Error updating parameters.'),end

  if Tbeg==0 then
   if or(%state0_n<>%state0) then //initial state has been changed
     %state0=%state0_n
     [alreadyran,%cpr]=do_terminate1(scs_m,%cpr)
     choix=[]
   end
```

```
  else

  end
  if %cpr.sim.xptr($)-1<size(%cpr.state.x,'*') & solver<100 then
    warning(['Diagram has been compiled for implicit solver'
     'switching to implicit Solver'])
    solver=100
    tolerances(6)=solver
  elseif (%cpr.sim.xptr($)-1==size(%cpr.state.x,'*')) & ..
( solver==100 & size(%cpr.state.x,'*')<>0) then
    warning(['Diagram has been compiled for explicit solver'
     'switching to explicit Solver'])
    solver=0
    tolerances(6)=solver
  end

  if need_suppress then //this is done only once
    for i=1:length(%cpr.sim.funs)
      if type(%cpr.sim.funs(i))<>13 then
if find(%cpr.sim.funs(i)(1)==Ignore)<>[] then
  %cpr.sim.funs(i)(1)='trash';
end
      end
    end
  end

  switch_to_old_graphics_style()

  if needstart then //scicos initialisation
    if alreadyran then
      [alreadyran,%cpr]=do_terminate1(scs_m,%cpr)
      alreadyran=%f
    end
    %tcur=0
    %cpr.state=%state0
    tf=scs_m.props.tf;
    if tf*tolerances==[] then
      error(['Simulation parameters not set']);
    end

    ierr=execstr('[state,t]=scicosim(%cpr.state,Tbeg,Tfin,%cpr.sim,'+..
 '''start'',tolerances)','errcatch')
    %cpr.state=state
    if ierr<>0 then
      restore_graphics_style()
      error(['Initialisation problem:'])
    end
  end

  ierr=execstr('[state,t]=scicosim(%cpr.state,Tbeg,Tfin,%cpr.sim,'+..
       '''run'',tolerances)','errcatch')

  %cpr.state=state
  if ierr==0 then
    alreadyran=%t
    if tf-t<tolerances(3) then
      needstart=%t
      [alreadyran,%cpr]=do_terminate1(scs_m,%cpr)
    else
      %tcur=t
    end
  else
    restore_graphics_style()
    error(['Simulation problem:';lasterror()])
  end
  restore_graphics_style()
  Info=list(%tcur,%cpr,alreadyran,needstart,needcompile,%state0)
endfunction

function restore_graphics_style()
  global bak

  if Ignore==[] then

    //gg=xget('window')  // for bug in figure_style and winsid
    //xset('window',0) // for bug in figure_style and winsid
    set('old_style',bak) //set('figure_style',bak)
    //xset('window',gg) // for bug in figure_style and winsid
  end
endfunction

function switch_to_old_graphics_style()
  global bak
  if Ignore==[] then
    bak=stripblanks(get("old_style")) //bak=get('figure_style')
    set("old_style","on")
    //set('figure_style','old')
  end
endfunction


function [alreadyran,%cpr]=do_terminate1(scs_m,%cpr)
// Copyright INRIA

  if prod(size(%cpr))<2 then   alreadyran=%f,return,end
  par=scs_m.props;

  if alreadyran then
    alreadyran=%f
    //terminate current simulation
    ierr=execstr('[state,t]=scicosim(%cpr.state,par.tf,par.tf,'+..
```

```
 '%cpr.sim,''finish'',par.tol)',’errcatch’)

    %cpr.state=state
    if ierr<>0 then
      error([’End problem:’;lasterror()])
    end
  end
endfunction
```

2.11.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.12 Fonction tk_getvalue modifiée

- **Nom :** mtk_getvalue
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.12.1 Séquence d’appel

```
[ok,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18] = mtk_getvalue(%
```

2.12.2 Paramètres

- **%desc :** add here the parameter description
- **%labels :** add here the parameter description
- **%typ :** add here the parameter description
- **%ini :** add here the parameter description
- **ok :** add here the parameter description
- **%1 :** add here the parameter description
- **%2 :** add here the parameter description
- **%3 :** add here the parameter description
- **%4 :** add here the parameter description
- **%5 :** add here the parameter description
- **%6 :** add here the parameter description
- **%7 :** add here the parameter description
- **%8 :** add here the parameter description
- **%9 :** add here the parameter description
- **%10 :** add here the parameter description
- **%11 :** add here the parameter description
- **%12 :** add here the parameter description
- **%13 :** add here the parameter description
- **%14 :** add here the parameter description
- **%15 :** add here the parameter description
- **%16 :** add here the parameter description
- **%17 :** add here the parameter description
- **%18 :** add here the parameter description

2.12.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.12.4 Exemple

```
Add here scilab instructions and comments
```

2.12.5 Contenu du fichier

```
function [ok,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18]=mtk_getvalue(%desc,%labels,%typ,%ini)
//  getvalues - %window dialog for data acquisition
//%Synta%
//  [ok,%1,..,%11]=getvalue(desc,labels,typ,ini)
//%Parameters
//  desc    : column vector of strings, dialog general comment
//  labels  : n column vector of strings, labels(i) is the label of
//            the ith required value
//  typ     : list(typ1,dim1,..,typn,dimn)
//            typi : defines the type of the ith required value
//                   if may have the following values:
//                   'mat' : stands for matrix of scalars
//                   'col' : stands for column vector of scalars
//                   'row' : stands for row vector of scalars
//                   'vec' : stands for  vector of scalars
//                   'str' : stands for vector of strings
//                   'lis' : stands for list
//                   'pol' : stands for polynomials
//                   'r'   : stands for rational
//            dimi : defines the size of the ith required value
//                   it must be
//                    - an integer or a 2-vector of integers (-1 stands for
//                      arbitrary dimension)
//                    - an evaluatable character string
//  ini     : n column vector of strings, ini(i) gives the suggested
//            response for the ith required value
//  ok      : boolean ,%t if ok button pressed, %f if cancel button pressed
//  xi      : contains the ith required value if ok==%t
//%Description
// getvalues macro encapsulate x_mdialog function with error checking,
// evaluation of numerical response, ...
//%Remarks
// All correct scilab syntax may be used as responses, for matrices
// and vectors getvalues automatically adds [ ] around the given response
// before numerical evaluation
//%Example
// labels=['magnitude';'frequency';'phase    '];
// [ampl,freq,ph]=getvalue('define sine signal',labels,..
//            list('vec',1,'vec',1,'vec',1),['0.85';'10^2';'%pi/3'])
//
//%See also
// x_mdialog, x_dialog
//!
// Copyright INRIA
[%lhs,%rhs]=argn(0)

ierr=execstr('GetVar_fun=TCL_GetVar','errcatch');
if ierr<>0 then GetVar_fun=TK_GetVar, end;
ierr=execstr('EvalStr_fun=TCL_EvalStr','errcatch');
if ierr<>0 then EvalStr_fun=TK_EvalStr, end;

%nn=prod(size(%labels))
if %lhs<>%nn+2&%lhs<>%nn+1 then error(41),end
if size(%typ)<>2*%nn then
  error('%typ : list(''type'',[sizes],...)')
end
%1=[];%2=[];%3=[];%4=[];%5=[];%6=[];%7=[];%8=[];%9=[];%10=[];%11=[];
%12=[];%13=[];%14=[];
%15=[];%16=[];%17=[];%18=[];

if exists('%scicos_context') then
  %mm=getfield(1,%scicos_context)
  for %mi=%mm(3:$)
    if execstr(%mi+'=%scicos_context(%mi)','errcatch')<>0 then
      disp(lasterror())
      ok=%f
      return
    end
  end
end

if %rhs==3 then  %ini=emptystr(%nn,1),end
ok=%t
while %t do
  %str1=mdialog(%desc,%labels,%ini)
  if %str1==[] then ok=%f,%str=[];break,end
  %str=%str1;
  for %kk=1:%nn
    %cod=ascii(%str(%kk))
    %spe=find(%cod==10)
    if %spe<>[] then
      %semi=ascii(';')
      %cod(%spe)=%semi*ones(%spe')
      %str(%kk)=ascii(%cod)
    end
  end
  %nok=0
  for %kk=1:%nn
    select part(%typ(2*%kk-1),1:3)
    case 'mat'
      %ierr=execstr('%vv=['+%str(%kk)+']','errcatch');
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)<>1 then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      [%mv,%nv]=size(%vv)
      %ssz=string(%sz(1))+' x '+string(%sz(2))
      if %mv*%nv==0 then
if  %sz(1)>=0&%sz(2)>=0&%sz(1)*%sz(2)<>0 then %nok=%kk,break,end
      else
```

```
if %sz(1)>=0 then if %mv<>%sz(1) then %nok=%kk,break,end,end
if %sz(2)>=0 then if %nv<>%sz(2) then %nok=%kk,break,end,end
      end
    case 'vec'
      %ierr=execstr('%vv=['+%str(%kk)+']','errcatch')
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)<>1 then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssz=string(%sz(1))
      %nv=prod(size(%vv))
      if %sz(1)>=0 then if %nv<>%sz(1) then %nok=%kk,break,end,end
    case 'pol'
      %ierr=execstr('%vv=['+%str(%kk)+']','errcatch');
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)>2 then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssz=string(%sz(1))
      %nv=prod(size(%vv))
      if %sz(1)>=0 then if %nv<>%sz(1) then %nok=%kk,break,end,end
    case 'row'
      %ierr=execstr('%vv=['+%str(%kk)+']','errcatch');
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)<>1 then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      if %sz(1)<0 then
%ssz='1 x *'
      else
%ssz='1 x '+string(%sz(1))
      end
      [%mv,%nv]=size(%vv)
      if %mv<>1 then %nok=%kk,break,end,
      if %sz(1)>=0 then if %nv<>%sz(1) then %nok=%kk,break,end,end
    case 'col'
      %ierr=execstr('%vv=['+%str(%kk)+']','errcatch');
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)<>1 then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      if %sz(1)<0 then
%ssz='* x 1'
      else
%ssz=string(%sz(1))+' x 1'
      end
      [%mv,%nv]=size(%vv)
      if %nv<>1 then %nok=%kk,break,end,
      if %sz(1)>=0 then if %mv<>%sz(1) then %nok=%kk,break,end,end
    case 'str'
      %sde=%str1(%kk)
      %spe=find(ascii(%str1(%kk))==10)
      %spe($+1)=length(%sde)+1
      %vv=[];%kk1=1
      for %kkk=1:size(%spe,'*')
%vv(%kkk,1)=part(%sde,%kk1:%spe(%kkk)-1)
%kk1=%spe(%kkk)+1
      end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssz=string(%sz(1))
      %nv=prod(size(%vv))
      if %sz(1)>=0 then if %nv<>%sz(1) then %nok=%kk,break,end,end
    case 'lis'
      %ierr=execstr('%vv='+%str(%kk),'errcatch');
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)<>15& type(%vv)<>16 then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssz=string(%sz(1))
      %nv=size(%vv)
      if %sz(1)>=0 then if %nv<>%sz(1) then %nok=%kk,break,end,end
    case 'r  '
      %ierr=execstr('%vv=['+%str(%kk)+']','errcatch');
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)<>16 then %nok=-%kk,break,end
      if typeof(%vv)<>'rational' then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      [%mv,%nv]=size(%vv(2))
      %ssz=string(%sz(1))+' x '+string(%sz(2))
      if %mv*%nv==0 then
if  %sz(1)>=0&%sz(2)>=0&%sz(1)*%sz(2)<>0 then %nok=%kk,break,end
      else
if %sz(1)>=0 then if %mv<>%sz(1) then %nok=%kk,break,end,end
if %sz(2)>=0 then if %nv<>%sz(2) then %nok=%kk,break,end,end
      end
    else
      error('type non gere :'+%typ(2*%kk-1))
    end
    execstr('%'+string(%kk)+'=%vv')
  end
  if %nok>0 then
    x_message(['answer given for  '+%labels(%nok);
             'has invalid dimension: ';
             'waiting for dimension  '+%ssz])
    %ini=%str
  elseif %nok<0 then
    if %ierr==0 then
      x_message(['answer given for  '+%labels(-%nok);
'has incorrect type :'+ %typ(-2*%nok-1)])
    else
      x_message(['answer given for  '+%labels(-%nok);
'is incorrect see error message in scilab window'])
    end
    %ini=%str
  else
    break
```

```
  end
end
if %lhs==%nn+2 then
  execstr('%'+string(%lhs-1)+'=%str')
end
endfunction

function result=mdialog(title,items,init)
  if argn(2)<1 then
    title=['this is a demo';'sdsdfs sdfsddddddddd w"+...
   " errrereeeeeeeeeeeeeeeeeeeeee']
    items=['asdfaf';'qwedfeqwfwefffffffffffffffffffff']
    end
    if argn(2)<3 then
    init=['asdffasf';'dfsdfsfsfsdfsdfddddddddddddddddddddddddddd']
  end
    title=sci2tcl(title);
    for i=1:size(items,'*')
      items(i)=sci2tcl(items(i))
      init(i)=sci2tcl(init(i))
    end

txt=create_txt(title,items,init);
result=[];

EvalStr_fun(txt)
//done=TK_GetVar('done')
//if done==string(1) then
//  for i=1:size(items,'*')
//    execstr('result(i)=TK_GetVar(''x'+string(i)+''')')
//  end
//end
//TK_EvalStr('destroy $w')

endfunction


function txt=create_txt(title,items,init)
txt=['set w .form'
'catch {destroy $w}'
'toplevel $w'
'set numx [winfo pointerx .]'
'set numy [winfo pointery .]'
'set title '"Set Block properties'"'
'wm geometry $w +$numx+$numy'
'wm title $w $title'
'wm iconname $w '"form'"'
'#positionWindow $w'
'label $w.msg  -wraplength 4i -justify left -text '"'+title+''"'
'frame $w.buttons'
'pack $w.buttons -side bottom -fill x -pady 2m'
'button $w.buttons.dismiss -text Dismiss -command {set done 2}'
'button $w.buttons.code -text OK -command {set done 1}'
'pack $w.buttons.dismiss $w.buttons.code -side left -expand 1'];

for i=1:size(items,'*')
  txt=[txt
       'frame $w.f'+string(i)+' -bd 2'
       'entry $w.f'+string(i)+'.entry -relief sunken -width 40'
       'label $w.f'+string(i)+'.label'
       'pack $w.f'+string(i)+'.entry -side right'
       'pack $w.f'+string(i)+'.label -side left'];
end
for i=1:size(items,'*')
txt=[txt
     '$w.f'+string(i)+'.label config -text '"'+items(i)+''"'];
end

 for i=1:size(items,'*')
   txt=[txt
'$w.f'+string(i)+'.entry insert 0 '"'+init(i)+''"'
];
   end

tt=''
for i=1:size(items,'*')
  tt=tt+'global x'+string(i)+';set x'+string(i)+' [$w.f'+string(i)+'.entry set];'
  //tt=tt+'ScilabEval '"result('+string(i)+')=x'+string(i)+''";'
end
txt=[txt;
     'proc done1 {w} {'+tt+'}'
]
tt=''
for i=1:size(items,'*')
  tt=tt+'$w.f'+string(i)+' '
end
 txt=[txt;
     'pack $w.msg '+tt+'-side top -fill x']
 //     'focus $w.f1.entry']
 //     'set done 0'
 //     'bind $w <Return> {set done 1}'
 //     'bind $w <Destroy> {set done 2}']
      //'tkwait variable done'
      //'if {$done==1} {done1 $w}']
     //'catch {destroy $w}']

endfunction


function tt=sci2tcl(ttx)
  for t=['\',''"','[',']','{','}']
```

```
    ttx=strsubst(ttx,t,'\'+t)
  end
  if size(ttx,'*')<2 then tt=ttx,return,end
  tt=ttx(1)
  ttx=ttx(2:$)';
  for t=ttx
    tt=tt+'\n '+t
  end

endfunction
```

2.12.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.13 Fonction tk_getvalue modifiée

- **Nom :** mtk_getvalue2
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.13.1 Séquence d'appel

```
[ok,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18] = mtk_getvalue2(
```

2.13.2 Paramètres

- **%desc :** add here the parameter description
- **%labels :** add here the parameter description
- **%typ :** add here the parameter description
- **%ini :** add here the parameter description
- **ok :** add here the parameter description
- **%1 :** add here the parameter description
- **%2 :** add here the parameter description
- **%3 :** add here the parameter description
- **%4 :** add here the parameter description
- **%5 :** add here the parameter description
- **%6 :** add here the parameter description
- **%7 :** add here the parameter description
- **%8 :** add here the parameter description
- **%9 :** add here the parameter description
- **%10 :** add here the parameter description
- **%11 :** add here the parameter description
- **%12 :** add here the parameter description
- **%13 :** add here the parameter description
- **%14 :** add here the parameter description
- **%15 :** add here the parameter description
- **%16 :** add here the parameter description
- **%17 :** add here the parameter description
- **%18 :** add here the parameter description

2.13.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.13.4 Exemple

Add here scilab instructions and comments

2.13.5 Contenu du fichier

```
function [ok,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18]=mtk_getvalue2(%desc,%labels,%typ,%ini)
 global mydesc
 global mylables
 global mytyp
 global myini

 mydesc=%desc
 mylables=%labels
 mytyp=%typ
 myini=%ini
//  getvalues - %window dialog for data acquisition
//%Synta%
//  [ok,%1,..,%11]=getvalue(desc,labels,typ,ini)
//%Parameters
//  desc    : column vector of strings, dialog general comment
//  labels  : n column vector of strings, labels(i) is the label of
//            the ith required value
//  typ     : list(typ1,dim1,..,typn,dimn)
//            typi : defines the type of the ith required value
//                   if may have the following values:
//                   'mat' : stands for matrix of scalars
//                   'col' : stands for column vector of scalars
//                   'row' : stands for row vector of scalars
//                   'vec' : stands for  vector of scalars
//                   'str' : stands for vector of strings
//                   'lis' : stands for list
//                   'pol' : stands for polynomials
//                   'r'   : stands for rational
//            dimi : defines the size of the ith required value
//                   it must be
//                    - an integer or a 2-vector of integers (-1 stands for
//                      arbitrary dimension)
//                    - an evaluatable character string
//  ini     : n column vector of strings, ini(i) gives the suggested
//            response for the ith required value
//  ok      : boolean ,%t if ok button pressed, %f if cancel button pressed
//  xi      : contains the ith required value if ok==%t
//%Description
// getvalues macro encapsulate x_mdialog function with error checking,
// evaluation of numerical response, ...
//%Remarks
// All correct scilab syntax may be used as responses, for matrices
// and vectors getvalues automatically adds [ ] around the given response
// before numerical evaluation
//%Example
// labels=['magnitude';'frequency';'phase    '];
// [ampl,freq,ph]=getvalue('define sine signal',labels,..
//            list('vec',1,'vec',1,'vec',1),['0.85';'10^2';'%pi/3'])
//
//%See also
// x_mdialog, x_dialog
//!
// Copyright INRIA
[%lhs,%rhs]=argn(0)

ierr=execstr('GetVar_fun=TCL_GetVar','errcatch');
if ierr<>0 then GetVar_fun=TK_GetVar, end;
ierr=execstr('EvalStr_fun=TCL_EvalStr','errcatch');
if ierr<>0 then EvalStr_fun=TK_EvalStr, end;

%nn=prod(size(%labels))
if %lhs<>%nn+2&%lhs<>%nn+1 then error(41),end
if size(%typ)<>2*%nn then
  error('%typ : list(''type'',[sizes],...)')
end
%1=[];%2=[];%3=[];%4=[];%5=[];%6=[];%7=[];%8=[];%9=[];%10=[];%11=[];
%12=[];%13=[];%14=[];
%15=[];%16=[];%17=[];%18=[];

if exists('%scicos_context') then
  %mm=getfield(1,%scicos_context)
  for %mi=%mm(3:$)
    if execstr(%mi+'=%scicos_context(%mi)','errcatch')<>0 then
      disp(lasterror())
      ok=%f
      return
    end
  end
end

if %rhs==3 then  %ini=emptystr(%nn,1),end
ok=%t
while %t do
  %str1=mdialog(%desc,%labels,%ini)
  if %str1==[] then ok=%f,%str=[];break,end
  %str=%str1;
  for %kk=1:%nn
    %cod=ascii(%str(%kk))
    %spe=find(%cod==10)
    if %spe<>[] then
      %semi=ascii(';')
      %cod(%spe)=%semi*ones(%spe')
```

```
      %str(%kk)=ascii(%cod)
    end
  end
  %nok=0
  for %kk=1:%nn
    select part(%typ(2*%kk-1),1:3)
    case 'mat'
      %ierr=execstr('%vv=['+%str(%kk)+']','errcatch');
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)<>1 then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      [%mv,%nv]=size(%vv)
      %ssz=string(%sz(1))+' x '+string(%sz(2))
      if %mv*%nv==0 then
if  %sz(1)>=0&%sz(2)>=0&%sz(1)*%sz(2)<>0 then %nok=%kk,break,end
      else
if %sz(1)>=0 then if %mv<>%sz(1) then %nok=%kk,break,end,end
if %sz(2)>=0 then if %nv<>%sz(2) then %nok=%kk,break,end,end
      end
    case 'vec'
      %ierr=execstr('%vv=['+%str(%kk)+']','errcatch')
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)<>1 then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssz=string(%sz(1))
      %nv=prod(size(%vv))
      if %sz(1)>=0 then if %nv<>%sz(1) then %nok=%kk,break,end,end
    case 'pol'
      %ierr=execstr('%vv=['+%str(%kk)+']','errcatch');
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)>2 then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssz=string(%sz(1))
      %nv=prod(size(%vv))
      if %sz(1)>=0 then if %nv<>%sz(1) then %nok=%kk,break,end,end
    case 'row'
      %ierr=execstr('%vv=['+%str(%kk)+']','errcatch');
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)<>1 then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      if %sz(1)<0 then
%ssz='1 x *'
      else
%ssz='1 x '+string(%sz(1))
      end
      [%mv,%nv]=size(%vv)
      if %mv<>1 then %nok=%kk,break,end,
      if %sz(1)>=0 then if %nv<>%sz(1) then %nok=%kk,break,end,end
    case 'col'
      %ierr=execstr('%vv=['+%str(%kk)+']','errcatch');
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)<>1 then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      if %sz(1)<0 then
%ssz='* x 1'
      else
%ssz=string(%sz(1))+' x 1'
      end
      [%mv,%nv]=size(%vv)
      if %nv<>1 then %nok=%kk,break,end,
      if %sz(1)>=0 then if %mv<>%sz(1) then %nok=%kk,break,end,end
    case 'str'
      %sde=%str1(%kk)
      %spe=find(ascii(%str1(%kk))==10)
      %spe($+1)=length(%sde)+1
      %vv=[];%kk1=1
      for %kkk=1:size(%spe,'*')
%vv(%kkk,1)=part(%sde,%kk1:%spe(%kkk)-1)
%kk1=%spe(%kkk)+1
      end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssz=string(%sz(1))
      %nv=prod(size(%vv))
      if %sz(1)>=0 then if %nv<>%sz(1) then %nok=%kk,break,end,end
    case 'lis'
      %ierr=execstr('%vv='+%str(%kk),'errcatch');
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)<>15& type(%vv)<>16 then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      %ssz=string(%sz(1))
      %nv=size(%vv)
      if %sz(1)>=0 then if %nv<>%sz(1) then %nok=%kk,break,end,end
    case 'r  '
      %ierr=execstr('%vv=['+%str(%kk)+']','errcatch');
      if %ierr<>0 then %nok=-%kk;break,end
      if type(%vv)<>16 then %nok=-%kk,break,end
      if typeof(%vv)<>'rational' then %nok=-%kk,break,end
      %sz=%typ(2*%kk);if type(%sz)==10 then %sz=evstr(%sz),end
      [%mv,%nv]=size(%vv(2))
      %ssz=string(%sz(1))+' x '+string(%sz(2))
      if %mv*%nv==0 then
if  %sz(1)>=0&%sz(2)>=0&%sz(1)*%sz(2)<>0 then %nok=%kk,break,end
      else
if %sz(1)>=0 then if %mv<>%sz(1) then %nok=%kk,break,end,end
if %sz(2)>=0 then if %nv<>%sz(2) then %nok=%kk,break,end,end
      end
    else
      error('type non gere :'+%typ(2*%kk-1))
    end
    execstr('%'+string(%kk)+'=%vv')
  end
```

```
  if %nok>0 then
    x_message([’answer given for  ’+%labels(%nok);
              ’has invalid dimension: ’;
              ’waiting for dimension  ’+%ssz])
    %ini=%str
  elseif %nok<0 then
    if %ierr==0 then
      x_message([’answer given for  ’+%labels(-%nok);
’has incorrect type :’+ %typ(-2*%nok-1)])
    else
      x_message([’answer given for  ’+%labels(-%nok);
’is incorrect see error message in scilab window’])
    end
    %ini=%str
  else
    break
  end
end
if %lhs==%nn+2 then
  execstr(’%’+string(%lhs-1)+’=%str’)
end
endfunction

function result=mdialog(title,items,init)
  if argn(2)<1 then
    title=[’this is a demo’;’sdsdfs sdfsddddddddd w"+...
   " errrereeeeeeeeeeeeeeeeeeeee’]
    items=[’asdfaf’;’qwedfeqwfwefffffffffffffffffffff’]
    end
    if argn(2)<3 then
    init=[’asdffasf’;’dfsdfsfsfsdfsdfddddddddddddddddddddddddddd’]
  end
    title=sci2tcl(title);
    for i=1:size(items,’*’)
      items(i)=sci2tcl(items(i))
      init(i)=sci2tcl(init(i))
    end

txt=create_txt(title,items,init);
result=[];

EvalStr_fun(txt)
done=GetVar_fun(’done’)
if done==string(1) then
  for i=1:size(items,’*’)
    execstr(’result(i)=GetVar_fun(’’x’+string(i)+’’’)’)
  end
end


endfunction


function txt=create_txt(title,items,init)
txt=[’set w .form’
’catch {destroy $w}’
’toplevel $w’
’set numx [winfo pointerx .]’
’set numy [winfo pointery .]’
’set title ’"Set Block properties’"’
’wm geometry $w +$numx+$numy’
’wm title $w $title’
’wm iconname $w ’"form’"’
’#positionWindow $w’
’label $w.msg  -wraplength 4i -justify left -text ’"’+title+’’"’
’frame $w.buttons’
’pack $w.buttons -side bottom -fill x -pady 2m’
’button $w.buttons.dismiss -text Dismiss -command {set done 2}’
’button $w.buttons.code -text OK -command {set done 1}’
’pack $w.buttons.dismiss $w.buttons.code -side left -expand 1’];

for i=1:size(items,’*’)
  txt=[txt
       ’frame $w.f’+string(i)+’ -bd 2’
       ’entry $w.f’+string(i)+’.entry -relief sunken -width 40’
       ’label $w.f’+string(i)+’.label’
       ’pack $w.f’+string(i)+’.entry -side right’
       ’pack $w.f’+string(i)+’.label -side left’];
end
for i=1:size(items,’*’)
txt=[txt
     ’$w.f’+string(i)+’.label config -text ’"’+items(i)+’’"’];
end

 for i=1:size(items,’*’)
   txt=[txt
’$w.f’+string(i)+’.entry insert 0 ’"’+init(i)+’’"’
];
   end

tt=’’
for i=1:size(items,’*’)
  tt=tt+’global x’+string(i)+’;set x’+string(i)+’ [$w.f’+string(i)+’.entry get];’
  //tt=tt+’ScilabEval ’"result(’+string(i)+’)=x’+string(i)+’’";’
end
txt=[txt;
     ’proc done1 {w} {’+tt+’}’
]

tt=’’
for i=1:size(items,’*’)
```

```
  tt=tt+'$w.f'+string(i)+' '
end
 txt=[txt;
     'pack $w.msg '+tt+'-side top -fill x'
     'set done 1';'done1 $w']
 //     'focus $w.f1.entry']
 //     'set done 0'
 //     'bind $w <Return> {set done 1}'
 //     'bind $w <Destroy> {set done 2}']
      //'tkwait variable done'
      //'if {$done==1} {done1 $w}']
     //'catch {destroy $w}']
endfunction


function tt=sci2tcl(ttx)
  for t=['\',''"','[',']','{','}']
    ttx=strsubst(ttx,t,'\'+t)
  end
  if size(ttx,'*')<2 then tt=ttx,return,end
  tt=ttx(1)
  ttx=ttx(2:$)';
  for t=ttx
    tt=tt+'\n '+t
  end

endfunction
```

2.13.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.14 Met à jour les paramètres réels d'un bloc dans une liste Info

- **Nom :** put_rpar_block
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.14.1 Séquence d'appel

```
Info = put_rpar_block(Info,name,vec)
```

2.14.2 Paramètres

- **Info :** add here the parameter description
- **name :** add here the parameter description
- **vec :** add here the parameter description
- **Info :** add here the parameter description

2.14.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.14.4 Exemple

```
Add here scilab instructions and comments
```

2.14.5 Contenu du fichier

```
function Info=put_rpar_block(Info,name,vec)
 %cpr=Info(2);
 n=find_num_block(Info,name);
 b=(%cpr.sim("rpptr")(n));
 for i=1:size(vec,1)
  %cpr.sim("rpar")(b+i-1)=vec(i);
 end
 //pause
 Info(2)=%cpr;
endfunction
```

2.14.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.15 Met à jour l'état discret des blocs scicos dans une liste Info

- **Nom :** put_state
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.15.1 Séquence d'appel

```
Info = put_state(Info,n,vec)
```

2.15.2 Paramètres

- **Info :** add here the parameter description
- **n :** add here the parameter description
- **vec :** add here the parameter description
- **Info :** add here the parameter description

2.15.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.15.4 Exemple

```
Add here scilab instructions and comments
```

2.15.5 Contenu du fichier

```
function Info=put_state(Info,n,vec)
 %cpr=Info(2);
 //n=find_num_block(Info2,name);
 b=(%cpr.sim("zptr")(n));
 for i=1:size(vec,1)
  %cpr.state("z")(b+i-1)=vec(i);
 end
 Info(2)=%cpr;
endfunction
```

2.15.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.16 Met à jour l'état discret des blocs scicos dans une liste Info

- **Nom :** put_state_block
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.16.1 Séquence d'appel

```
Info = put_state_block(Info,name,vec)
```

2.16.2 Paramètres

- **Info :** add here the parameter description
- **name :** add here the parameter description
- **vec :** add here the parameter description
- **Info :** add here the parameter description

2.16.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.16.4 Exemple

```
Add here scilab instructions and comments
```

2.16.5 Contenu du fichier

```
function Info=put_state_block(Info,name,vec)
 %cpr=Info(2);
 n=find_num_block(Info,name);
 b=(%cpr.sim("zptr")(n));
 for i=1:size(vec,1)
  %cpr.state("z")(b+i-1)=vec(i);
 end
 Info(2)=%cpr;
endfunction
```

2.16.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.17 échange les états discrets de blocs scicos entre deux listes Info

- **Nom :** put_state_to_state
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.17.1 Séquence d'appel

```
Info2 = put_state_to_state(Info1,Info2,n1,n2)
```

2.17.2 Paramètres

- **Info1 :** add here the parameter description
- **Info2 :** add here the parameter description
- **n1 :** add here the parameter description
- **n2 :** add here the parameter description
- **Info2 :** add here the parameter description

2.17.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.17.4 Exemple

```
Add here scilab instructions and comments
```

2.17.5 Contenu du fichier

```
function Info2=put_state_to_state(Info1,Info2,n1,n2)
 //%cpr=Info(2);
 //n=find_num_block(name);
 //pause
 Info2(2).state("z")(Info2(2).sim("zptr")(n2):Info2(2).sim("zptr")(n2+1)-1)=Info1(2).state("z")(Info1(2).sim("zptr")(n1):Info1(2).sim("zptr")(n1+1)-1);
 //b=(%cpr.sim("zptr")(n));
 //for i=1:size(vec,1)
 // %cpr.state("z")(b+i-1)=vec(i);
 //end
 //Info(2)=%cpr;
endfunction
```

2.17.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.18 Retourne les blocs d'une liste scs_m

- **Nom :** return_block_cos
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.18.1 Séquence d'appel

```
tt = return_block_cos(name,libn,dbl)
```

2.18.2 Paramètres

- **name :** add here the parameter description
- **libn :** add here the parameter description
- **dbl :** add here the parameter description
- **tt :** add here the parameter description

2.18.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.18.4 Exemple

```
Add here scilab instructions and comments
```

2.18.5 Contenu du fichier

```
//return_block_cos
//Entrée name : chemin+nom du fichier cos
//             ex : name=MODNUM+'/scs_diagr/dyna/chua/chua.cos'
//      libn : un identifiant de libraire (ex 'mod_num')
//      dbl : un drapeau pour les doublons
//sortie tt : La liste des blocks
function tt=return_block_cos(name,libn,dbl)
  if fileinfo(name)<>[] then
    //Vérifie cohérence des param
    [lsh,rsh]=argn(0)
    if rsh<3 then dbl='', end;
    if rsh<2 then libn='', end;
    if ~exists('libn') then libn='', end
    if ~exists('dbl') then dbl='', end

    //load scicos library
    load SCI/macros/scicos/lib

    //charge le ficier cos
    load(name);

    //appel return_block_name
    tt=return_block_name(scs_m,libn,dbl)

  else
    printf("Unable to find cos file");
```

```
  end
endfunction
```

2.18.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.19 Retourne la description des blocs d'une liste scs_m

- **Nom :** return_block_desc
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.19.1 Séquence d'appel

```
txt = return_block_desc(scs_m,dbl)
```

2.19.2 Paramètres

- **scs_m :** add here the parameter description
- **dbl :** add here the parameter description
- **txt :** add here the parameter description

2.19.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.19.4 Exemple

```
Add here scilab instructions and comments
```

2.19.5 Contenu du fichier

```
//Fonction qui retourne une description des blocks
//d'une liste scs_m
//si dbl==1 alors élimine les doublons

function txt=return_block_desc(scs_m,dbl)
 txt=[]
 [%lhs,%rhs]=argn(0)
 n=lstsize(scs_m.objs) //nbr d'objet dans scs_m
 for i=1:n
  if execstr('scs_m.objs(i).gui','errcatch')==0 then
    ww=whereis(scs_m.objs(i).gui)
    if ww==[]|ww=='' then ww='', end
    txt=[txt;string(scs_m.objs(i).gui) string(i) string(ww) string(scs_m.objs(i).model.sim(1))]
  end
 end

  if exists('dbl') then
   if dbl==1 then //Trouve les doublons
    if txt<>[] then
     tt=txt(1,:)
     for i=1:size(txt,1)
         ok=%t
         for j=1:size(tt,1)
           if tt(j,1)==txt(i,1) then
            ok=%f
           end
         end
         if ok then tt=[tt;txt(i,:)], end
     end
     txt=tt
    end
   end
  end
endfunction
```

2.19.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.20 Retourne les noms des blocs d'une liste scs_m

- **Nom :** return_block_name
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.20.1 Séquence d'appel

```
txt = return_block_name(scs_m,libn,dbl)
```

2.20.2 Paramètres

- **scs_m :** add here the parameter description
- **libn :** add here the parameter description
- **dbl :** add here the parameter description
- **txt :** add here the parameter description

2.20.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.20.4 Exemple

```
Add here scilab instructions and comments
```

2.20.5 Contenu du fichier

```
//Fonction qui retourne les noms des blocks de la libraire
//libn présent dans une liste scs_m
//Entrée libn un identifiant de libraire
//       dbl drapeau pour éliminer les doublons
//si libn est absent retourne la liste de tous les blocks
//si dbl==1 alors élimine les doublons

function txt=return_block_name(scs_m,libn,dbl)
 txt=[]
 [%lhs,%rhs]=argn(0)
 n=lstsize(scs_m.objs) //nbr d'objet dans scs_m
 for i=1:n
  if execstr('scs_m.objs(i).gui','errcatch')==0 then
    if %rhs>1 then
     if exists('libn')&libn<>'' then
     //////Doit faire pour chaque libn
      ww=whereis(scs_m.objs(i).gui)
      if strindex(ww,libn)<>[]
        txt=[txt;scs_m.objs(i).gui]
      end
      /////////
     else
      txt=[txt;scs_m.objs(i).gui]
     end
    else
     txt=[txt;scs_m.objs(i).gui]
    end
  end
 end

 if %rhs>=3 then
  if exists('dbl') then
   if dbl==1 then //Trouve les doublons
    if txt<>[] then
     tt=txt(1)
     for i=1:size(txt,1)
         ok=%t
         for j=1:size(tt,1)
           if tt(j,1)==txt(i,1) then
            ok=%f
           end
         end
         if ok then tt=[tt;txt(i,1)], end
```

```
     end
     txt=tt
    end
   end
  end
 end
endfunction
```

2.20.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.21 Retourne la liste des noms des blocs présents dans un fichier cosf

- **Nom :** return_block_pal
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.21.1 Séquence d'appel

```
tt = return_block_pal(name,libn,dbl)
```

2.21.2 Paramètres

- **name :** add here the parameter description
- **libn :** add here the parameter description
- **dbl :** add here the parameter description
- **tt :** add here the parameter description

2.21.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.21.4 Exemple

```
Add here scilab instructions and comments
```

2.21.5 Contenu du fichier

```
//return_block_pal
//Entrée name : chemin+nom de la palette
//             ex : name=MODNUM+'/macros/scicos_blocks/Tools.cosf'
//       nlib : un identifiant de libraire (ex 'mod_num')
//sortie tt : La liste des blocks
function tt=return_block_pal(name,libn,dbl)
  if fileinfo(name)<>[] then

   //load scicos library
   load SCI/macros/scicos/lib

   exec(name,-1)
   tt=return_block_name(scs_m,libn,dbl)

  else
   printf("Unable to find cosf file");
  end
endfunction
```

2.21.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.22 Retourne le contexte d'un fichier diagramme scicos

- **Nom :** return_context_diagr
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.22.1 Séquence d'appel

```
txt = return_context_diagr(namef)
```

2.22.2 Paramètres

- **namef :** add here the parameter description
- **txt :** add here the parameter description

2.22.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.22.4 Exemple

```
Add here scilab instructions and comments
```

2.22.5 Contenu du fichier

```
//return_context_diagr
//fonction qui retourne le contexte
//d'un diagrammme scicos contenue dans
//le fichier 'name'
//Entrée : name le nom du fichier à éexaminer
//         ex : namef=MODNUM+'/scs_diagr/dyna/chua/chua.cos'
function txt=return_context_diagr(namef)

 txt=[]
 if fileinfo(namef)<>[] then
   ierror=execstr('load(namef)','errcatch')
   if ierror==0 then
      ierror2=execstr('txt=scs_m.props(""context"")','errcatch')
      if ierror2<>0 then
        printf("Error while reading context\n");
        txt=[];
      end
   else
     printf("Error while loading %s\n",namef);
   end
 else
  printf("file %s not found\n",namef);
 end

endfunction
```

2.22.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.23 Retourne le titre et le chemin inscrit dans un fichier diagramme scicos

- **Nom :** return_cos_name_cos
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.23.1 Séquence d'appel

```
[ttn,ttp] = return_cos_name_cos(name)
```

2.23.2 Paramètres

- **name :** add here the parameter description
- **ttn :** add here the parameter description
- **ttp :** add here the parameter description

2.23.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.23.4 Exemple

```
Add here scilab instructions and comments
```

2.23.5 Contenu du fichier

```
//return_cos_name_cos
//fonction qui retourne le titre et le
//chemin d'un diagramme scicos inscrit dans une liste scs
//Entrée name : chemin+nom du fichier cos
//              ex : name=MODNUM+'/scs_diagr/dyna/chua/chua.cos'
//sortie tt : le nom du fichier cos
function [ttn,ttp]=return_cos_name_cos(name)
  if fileinfo(name)<>[] then
    //charge le ficier cos
    load(name);

    ttn=scs_m.props("title")(1)
    ttp=scs_m.props("title")(2)

  else
    printf("Unable to find cos file");
  end
endfunction
```

2.23.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.24 Retourne la date

- **Nom :** return_date
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.24.1 Séquence d'appel

```
txt = return_date
```

2.24.2 Paramètres

- **txt :** add here the parameter description

2.24.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.24.4 Exemple

```
Add here scilab instructions and comments
```

2.24.5 Contenu du fichier

```
function txt=return_date()
 w=string(getdate());
 //mprintf("Year:%d,Month:%d,Day:%d",w(1),w(2),w(6));
 txt=w(6)+'/'+w(2)+'/'+w(1)+', '+w(7)+':'+w(8)+':'+w(9);
endfunction
```

2.24.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.25 Retourne la description de la boîte de dialogue d'un bloc scicos

- **Nom :** return_desc_block
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.25.1 Séquence d'appel

```
txt = return_desc_block(name)
```

2.25.2 Paramètres

- **name :** add here the parameter description
- **txt :** add here the parameter description

2.25.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.25.4 Exemple

```
Add here scilab instructions and comments
```

2.25.5 Contenu du fichier

```
function txt=return_desc_block(name)

 //Disable scilab function protection
 prot=funcprot();
 funcprot(0);

 //load scicos libraries
 load SCI/macros/scicos/lib
 exec(loadpallibs,-1)
 %scicos_prob=%f;
 alreadyran=%f
 needcompile=4
 //%zoom=1.8;

 //redefine getvalue
 getvalue=mgetvalue;

 //retrieve labels of getvalue fonction
 global mydesc

 ierror=execstr('blk='+name+'(''define'')','errcatch')
 if ierror<>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 ierror=execstr('blk='+name+'(''set'',blk)','errcatch')
 if ierror <>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end
```

```
 //restore function protection
 funcprot(prot);

 //
 txt=mydesc

 clearglobal mydesc
 clearglobal mylables
 clearglobal mytyp
 clearglobal myini
endfunction
```

2.25.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.26 Retourne la description de la boîte de dialogue d'un bloc scicos

- **Nom :** return_desc_block2
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.26.1 Séquence d'appel

```
txt = return_desc_block2(name,txt_exprs)
```

2.26.2 Paramètres

- **name :** add here the parameter description
- **txt_exprs :** add here the parameter description
- **txt :** add here the parameter description

2.26.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.26.4 Exemple

```
Add here scilab instructions and comments
```

2.26.5 Contenu du fichier

```
//return_desc_block2
//fonction qui retourne le titre d'une boite de dialogue
//Entrée name      : le nom du block
//       txt_exprs : expression à exécuter apres le cas define
function txt=return_desc_block2(name,txt_exprs)

 //Disable scilab function protection
 prot=funcprot();
 funcprot(0);

 //load scicos libraries
 load SCI/macros/scicos/lib
 exec(loadpallibs,-1)
 %scicos_prob=%f;
 alreadyran=%f
 needcompile=4
 //%zoom=1.8;

 //redefine getvalue
 getvalue=mgetvalue2;

 //retrieve labels of getvalue fonction
 global mydesc

 ierror=execstr('blk='+name+'(''define'')','errcatch')
 if ierror<>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
```

```
 end

 execstr(txt_exprs);

 ierror=execstr('blk='+name+'(''set'',blk)','errcatch')
 if ierror <>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 //restore function protection
 funcprot(prot);

 //
 txt=mydesc

 clearglobal mydesc
 clearglobal mylables
 clearglobal mytyp
 clearglobal myini
endfunction
```

2.26.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.27 Retourne les valeurs intiales des paramètres d'une boîte de dialogue d'un bloc scicos

- **Nom :** return_ini_block
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.27.1 Séquence d'appel

```
txt = return_ini_block(name)
```

2.27.2 Paramètres

- **name :** add here the parameter description
- **txt :** add here the parameter description

2.27.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.27.4 Exemple

```
Add here scilab instructions and comments
```

2.27.5 Contenu du fichier

```
function txt=return_ini_block(name)

 //Disable scilab function protection
 prot=funcprot();
 funcprot(0);

 //load scicos libraries
 load SCI/macros/scicos/lib
 exec(loadpallibs,-1)
 %scicos_prob=%f;
 alreadyran=%f
 needcompile=4
 //%zoom=1.8;

 //redefine getvalue
 getvalue=mgetvalue;

 //retrieve labels of getvalue fonction
```

```
 global myini

 ierror=execstr('blk='+name+'(''define'')','errcatch')
 if ierror<>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 ierror=execstr('blk='+name+'(''set'',blk)','errcatch')
 if ierror <>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 //restore function protection
 funcprot(prot);

 //
 txt=myini

 clearglobal mydesc
 clearglobal mylables
 clearglobal mytyp
 clearglobal myini
endfunction
```

2.27.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.28 Retourne les valeurs intiales des paramètres d'une boîte de dialogue d'un bloc scicos

- **Nom :** return_ini_block2
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.28.1 Séquence d'appel

```
txt = return_ini_block2(name,txt_exprs)
```

2.28.2 Paramètres

- **name :** add here the parameter description
- **txt_exprs :** add here the parameter description
- **txt :** add here the parameter description

2.28.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.28.4 Exemple

```
Add here scilab instructions and comments
```

2.28.5 Contenu du fichier

```
function txt=return_ini_block2(name,txt_exprs)

 //Disable scilab function protection
 prot=funcprot();
 funcprot(0);

 //load scicos libraries
 load SCI/macros/scicos/lib
 exec(loadpallibs,-1)
 %scicos_prob=%f;
```

```
 alreadyran=%f
 needcompile=4
 //%zoom=1.8;

 //redefine getvalue
 getvalue=mgetvalue2;

 //retrieve labels of getvalue fonction
 global myini

 ierror=execstr('blk='+name+'(''define'')','errcatch')
 if ierror<>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 execstr(txt_exprs);

 ierror=execstr('blk='+name+'(''set'',blk)','errcatch')
 if ierror <>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 //restore function protection
 funcprot(prot);

 //
 txt=myini

 clearglobal mydesc
 clearglobal mylables
 clearglobal mytyp
 clearglobal myini
endfunction
```

2.28.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.29 Retourne les labels des paramètres d'une boîte de dialogue d'un bloc scicos

- **Nom :** return_lables_block
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.29.1 Séquence d'appel

```
txt = return_lables_block(name)
```

2.29.2 Paramètres

- **name :** add here the parameter description
- **txt :** add here the parameter description

2.29.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.29.4 Exemple

```
Add here scilab instructions and comments
```

2.29.5 Contenu du fichier

```
function txt=return_lables_block(name)

 //Disable scilab function protection
 prot=funcprot();
```

```
 funcprot(0);

 //load scicos libraries
 load SCI/macros/scicos/lib
 exec(loadpallibs,-1)
 %scicos_prob=%f;
 alreadyran=%f
 needcompile=4
 //%zoom=1.8;

 //redefine getvalue
 getvalue=mgetvalue;

 //retrieve labels of getvalue fonction
 global mylables

 ierror=execstr('blk='+name+'(''define'')','errcatch')
 if ierror<>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 ierror=execstr('blk='+name+'(''set'',blk)','errcatch')
 if ierror <>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 //restore function protection
 funcprot(prot);

 //
 txt=mylables

 clearglobal mydesc
 clearglobal mylables
 clearglobal mytyp
 clearglobal myini

endfunction
```

2.29.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.30 Retourne les labels des paramètres d'une boîte de dialogue d'un bloc scicos

- **Nom :** return_lables_block2
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.30.1 Séquence d'appel

```
txt = return_lables_block2(name,txt_exprs)
```

2.30.2 Paramètres

- **name :** add here the parameter description
- **txt_exprs :** add here the parameter description
- **txt :** add here the parameter description

2.30.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.30.4 Exemple

```
Add here scilab instructions and comments
```

2.30.5 Contenu du fichier

```
function txt=return_lables_block2(name,txt_exprs)

 //Disable scilab function protection
 prot=funcprot();
 funcprot(0);

 //load scicos libraries
 load SCI/macros/scicos/lib
 exec(loadpallibs,-1)
 %scicos_prob=%f;
 alreadyran=%f
 needcompile=4
 //%zoom=1.8;

 //redefine getvalue
 getvalue=mgetvalue2;

 //retrieve labels of getvalue fonction
 global mylables

 ierror=execstr('blk='+name+'(''define'')','errcatch')
 if ierror<>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 execstr(txt_exprs);

 ierror=execstr('blk='+name+'(''set'',blk)','errcatch')
 if ierror <>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 //restore function protection
 funcprot(prot);

 //
 txt=mylables

 clearglobal mydesc
 clearglobal mylables
 clearglobal mytyp
 clearglobal myini

endfunction
```

2.30.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.31 Retourne les propriétés par défaut d'un bloc scicos

- **Nom :** return_prop_block
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.31.1 Séquence d'appel

```
txt = return_prop_block(name,lang)
```

2.31.2 Paramètres

- **name :** add here the parameter description
- **lang :** add here the parameter description
- **txt :** add here the parameter description

2.31.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.31.4 Exemple

Add here scilab instructions and comments

2.31.5 Contenu du fichier

```
//return_prop_block
//fonction qui retourne les propriétés par default
//d'une fonction d'interface d'un bloc scicos
//
//Entrée : name : le nom de la fonction d'interface sans extension
//        lang :  (optionel) 'fr' pour du francais
//                           'eng' et autres pour de l'anglais
//Sortie : txt : liste de texte des propriétés par défaults
function txt=return_prop_block(name,lang)

 //verify the lang parameter
 [lsh,rsh]=argn(0)
 if rsh<2 then
   lang='eng'
 end

 //load scicos libraries
 load SCI/macros/scicos/lib

 //execute define case
 ierror=execstr('blk='+name+'(''define'')','errcatch')
 if ierror<>0 then
    x_message(['Error in case define of GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 prop=[]
 deput=blk.model.dep_ut
 if lang=='fr' then
    if deput(2) then prop(1)='oui', else prop(1)='non', end
    if deput(1) then prop(2)='oui', else prop(2)='non', end
    if blk.model.nzcross<>0 then  prop(3)='oui', else prop(3)='non', end
    if blk.model.nmode<>0 then  prop(4)='oui', else prop(4)='non', end
    if blk.model.state<>[] then prop(9)='oui', else prop(9)='non', end
    if blk.model.dstate<>[] then prop(10)='oui', else prop(10)='non', end
 else
    if deput(2) then prop(1)='yes', else prop(1)='no', end
    if deput(1) then prop(2)='yes', else prop(2)='no', end
    if blk.model.nzcross<>0 then  prop(3)='yes', else prop(3)='no', end
    if blk.model.nmode<>0 then  prop(4)='yes', else prop(4)='no', end
    if blk.model.state<>[] then prop(9)='yes', else prop(9)='no', end
    if blk.model.dstate<>[] then prop(10)='yes', else prop(10)='no', end
 end
 prop(5)=string(size(blk.model.in,'*'))+' / '+strcat(string(blk.model.in),'  ')
 prop(6)=string(size(blk.model.out,'*'))+' / '+strcat(string(blk.model.out),'  ')
 prop(7)=string(size(blk.model.evtin,'*'))+' / '+strcat(string(blk.model.evtin),'  ')
 prop(8)=string(size(blk.model.evtout,'*'))+' / '+strcat(string(blk.model.evtout),'  ')
 prop(11)=name+'.sci'
 if typeof(blk.model.sim)=='list' then
    prop(12)=string(blk.model.sim(1))
    prop(13)=string(blk.model.sim(2))
 else
    prop(12)=string(blk.model.sim(1))
    prop(13)=""
 end
 txt=prop

endfunction
```

2.31.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.32 Retourne les paramètres réels d'un bloc scicos

- **Nom :** return_rpar_block
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.32.1 Séquence d'appel

```
tt = return_rpar_block(name)
```

2.32.2 Paramètres

- **name :** add here the parameter description
- **tt :** add here the parameter description

2.32.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.32.4 Exemple

```
Add here scilab instructions and comments
```

2.32.5 Contenu du fichier

```
function tt=return_rpar_block(name)
 //load scicos libraries
 load SCI/macros/scicos/lib

 //execute define case
 ierror=execstr('blk='+name+'(''define'')','errcatch')
 if ierror<>0 then
    x_message(['Error in case define of GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 tt=blk.model.rpar
endfunction
```

2.32.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.33 Retourne l'état discret d'un bloc scicos

- **Nom :** return_state
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.33.1 Séquence d'appel

```
res = return_state(Info,n)
```

2.33.2 Paramètres

- **Info :** add here the parameter description
- **n :** add here the parameter description
- **res :** add here the parameter description

2.33.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.33.4 Exemple

```
Add here scilab instructions and comments
```

2.33.5 Contenu du fichier

```
function res=return_state(Info,n)
 %cpr=Info(2);
 res=list();
 for i=1:size(n,1)
  if n(i)<>0 then
     res(i)=%cpr.state("z")(%cpr.sim("zptr")(n(i)):%cpr.sim("zptr")(n(i)+1)-1);
  else
     res(i)=[];
  end
 end
endfunction
```

2.33.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.34 Retourne l'état discret d'un bloc scicos

- **Nom :** return_state_block
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.34.1 Séquence d'appel

```
res = return_state_block(Info,name)
```

2.34.2 Paramètres

- **Info :** add here the parameter description
- **name :** add here the parameter description
- **res :** add here the parameter description

2.34.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.34.4 Exemple

```
Add here scilab instructions and comments
```

2.34.5 Contenu du fichier

```
function res=return_state_block(Info,name)
  n=find_num_block(Info,name);
  res=return_state(Info,n);
endfunction
```

2.34.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.35 Retourne les types des paramètres d'une boîte de dialogue d'un bloc scicos

- **Nom :** return_typ_block
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.35.1 Séquence d'appel

```
txt = return_typ_block(name)
```

2.35.2 Paramètres

- **name :** add here the parameter description
- **txt :** add here the parameter description

2.35.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.35.4 Exemple

```
Add here scilab instructions and comments
```

2.35.5 Contenu du fichier

```
function txt=return_typ_block(name)

 //Disable scilab function protection
 prot=funcprot();
 funcprot(0);

 //load scicos libraries
 load SCI/macros/scicos/lib
 exec(loadpallibs,-1)
 %scicos_prob=%f;
 alreadyran=%f
 needcompile=4
 //%zoom=1.8;

 //redefine getvalue
 getvalue=mgetvalue;

 //retrieve labels of getvalue fonction
 global mytyp

 ierror=execstr('blk='+name+'(''define'')','errcatch')
 if ierror<>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 ierror=execstr('blk='+name+'(''set'',blk)','errcatch')
 if ierror <>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 //restore function protection
 funcprot(prot);

 if mytyp<>[] then
  for i=1:(size(mytyp)/2)
   txt(i,1)=mytyp(2*i-1);
   txt(i,2)=string(mytyp(2*i));
  end
 else
  txt(1,1)=[]
  txt(1,2)=[]
 end

 clearglobal mydesc
 clearglobal mylables
 clearglobal mytyp
 clearglobal myini
endfunction
```

2.35.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.36 Retourne les types des paramètres d'une boîte de dialogue d'un bloc scicos

- **Nom :** return_typ_block2
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.36.1 Séquence d'appel

```
txt = return_typ_block2(name,txt_exprs)
```

2.36.2 Paramètres

- **name :** add here the parameter description
- **txt_exprs :** add here the parameter description
- **txt :** add here the parameter description

2.36.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.36.4 Exemple

```
Add here scilab instructions and comments
```

2.36.5 Contenu du fichier

```
function txt=return_typ_block2(name,txt_exprs)

 //Disable scilab function protection
 prot=funcprot();
 funcprot(0);

 //load scicos libraries
 load SCI/macros/scicos/lib
 exec(loadpallibs,-1)
 %scicos_prob=%f;
 alreadyran=%f
 needcompile=4
 //%zoom=1.8;

 //redefine getvalue
 getvalue=mgetvalue2;

 //retrieve labels of getvalue fonction
 global mytyp

 ierror=execstr('blk='+name+'(''define'')','errcatch')
 if ierror<>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 execstr(txt_exprs);

 ierror=execstr('blk='+name+'(''set'',blk)','errcatch')
 if ierror <>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(name)
    fct=[]
    return
 end

 //restore function protection
 funcprot(prot);

 if mytyp<>[] then
  for i=1:(size(mytyp)/2)
   txt(i,1)=mytyp(2*i-1);
   txt(i,2)=string(mytyp(2*i));
  end
 else
  txt(1,1)=[]
  txt(1,2)=[]
 end

 clearglobal mydesc
 clearglobal mylables
```

```
 clearglobal mytyp
 clearglobal myini
endfunction
```

2.36.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.37 Substitut du texte dans un contexte scicos

- **Nom :** subst_ctxt
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.37.1 Séquence d'appel

```
ctxt = subst_ctxt(scs_m,str1,str2)
```

2.37.2 Paramètres

- **scs_m :** add here the parameter description
- **str1 :** add here the parameter description
- **str2 :** add here the parameter description
- **ctxt :** add here the parameter description

2.37.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.37.4 Exemple

```
Add here scilab instructions and comments
```

2.37.5 Contenu du fichier

```
function ctxt=subst_ctxt(scs_m,str1,str2)
 ctxt=scs_m.props.context;
 for i=1:size(ctxt,1)
  if strindex(ctxt(i),str1)<>[] then
   ctxt(i)=str2;
  end
 end
endfunction
```

2.37.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.38 Substitut le caractère '%' en caractère '%%'

- **Nom :** subst_strC
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.38.1 Séquence d'appel

```
txt = subst_strC(txt)
```

2.38.2 Paramètres

- **txt :** add here the parameter description
- **txt :** add here the parameter description

2.38.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.38.4 Exemple

```
Add here scilab instructions and comments
```

2.38.5 Contenu du fichier

```
function txt=subst_strC(txt)
 txt=strsubst(txt,'%','%%');
endfunction
```

2.38.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.39 écrit le temps intial d'une simulation itérative dans un fichier log

- **Nom :** wlog_bst
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.39.1 Séquence d'appel

```
wlog_bst(u,j,flag)
```

2.39.2 Paramètres

- **u :** add here the parameter description
- **j :** add here the parameter description
- **flag :** add here the parameter description

2.39.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.39.4 Exemple

```
Add here scilab instructions and comments
```

2.39.5 Contenu du fichier

```
//Begin Simulation Time
function wlog_bst(u,j,flag)
  //doit faire test pour présence argument 2
  tt=">> Begin iterative simulation "+string(j)+" : "+return_date()+"\n";
  mfprintf(u,tt);
  if (flag==1) then
   printf(tt);
  elseif(flag==2) then
   ts=sprintf(tt);
   put_list(ts);
  end
endfunction
```

2.39.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.40 écrit un conexte dans un fichier log

- **Nom :** wlog_ctxt
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.40.1 Séquence d'appel

```
wlog_ctxt(u,flag)
```

2.40.2 Paramètres

- **u :** add here the parameter description
- **flag :** add here the parameter description

2.40.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.40.4 Exemple

```
Add here scilab instructions and comments
```

2.40.5 Contenu du fichier

```
//Save context
function wlog_ctxt(u,flag)
  tt=">> Context of scicos diagram : \n";
  tt=tt+"<scs_m.props.context>\n!\n"
  for i=1:size(scs_m.props.context,1)
    tt=tt+string(subst_strC(scs_m.props.context(i)))+"\n";
  end
  tt=tt+"!\n";
  mfprintf(u,tt);
  if (flag==1) then
   printf(tt);
  elseif(flag==2) then
   ts=msprintf(tt);
   put_list(ts);
  end
endfunction
```

2.40.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.41 écrit le temps final d'une simulation dans un fichier log

- **Nom :** wlog_final
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.41.1 Séquence d'appel

```
wlog_final(u,flag)
```

2.41.2 Paramètres

- **u :** add here the parameter description
- **flag :** add here the parameter description

2.41.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.41.4 Exemple

```
Add here scilab instructions and comments
```

2.41.5 Contenu du fichier

```
//Final simulation data
function wlog_final(u,flag)
  tt=">> End of simulation : "+return_date()+"\n\n";
  mfprintf(u,tt);
  if (flag==1) then
   printf(tt);
  elseif(flag==2) then
   ts=sprintf(tt);
   put_list(ts);
  end
endfunction
```

2.41.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.42 écrit le temps final d'une simulation itérative dans un fichier log

- **Nom :** wlog_fst
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.42.1 Séquence d'appel

```
wlog_fst(u,j,flag)
```

2.42.2 Paramètres

- **u :** add here the parameter description
- **j :** add here the parameter description
- **flag :** add here the parameter description

2.42.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.42.4 Exemple

```
Add here scilab instructions and comments
```

2.42.5 Contenu du fichier

```
//Final Simulation Time
function wlog_fst(u,j,flag)
  //doit faire test pour présence argument 2
  tt=">> End iterative simulation "+string(j)+" : "+return_date()+"\n";
  mfprintf(u,tt);
  if (flag==1) then
   printf(tt);
  elseif(flag==2) then
   ts=sprintf(tt);
   put_list(ts);
  end
endfunction
```

2.42.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.43 écrit le temps inital d'une simulation dans un fichier log

- **Nom :** wlog_init
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.43.1 Séquence d'appel

```
wlog_init(u,flag)
```

2.43.2 Paramètres

- **u :** add here the parameter description
- **flag :** add here the parameter description

2.43.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.43.4 Exemple

```
Add here scilab instructions and comments
```

2.43.5 Contenu du fichier

```
//initial simulation data
function wlog_init(u,flag)
  tt=">> Begin new simulation : "+return_date()+"\n";
  mfprintf(u,tt);
  if (flag==1) then
   printf(tt);
  elseif(flag==2) then
    init_ui();
    ts=sprintf(tt);
    put_list(ts);
  end
endfunction
```

2.43.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.44 écrit les variables calculées en fin de simulation dans un fichier log

- **Nom :** wlog_psv
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.44.1 Séquence d'appel

```
wlog_psv(u,lst,flag)
```

2.44.2 Paramètres

- **u :** add here the parameter description
- **lst :** add here the parameter description
- **flag :** add here the parameter description

2.44.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.44.4 Exemple

```
Add here scilab instructions and comments
```

2.44.5 Contenu du fichier

```
//Save Post-processed results
//lst est une tlist
//ex : tlist(['teb';'sigma'],teb,sig)
function wlog_psv(u,lst,flag)
  tt=">> Save Post processed results :\n";
  for i=1:size(lst(1),1)
   tt=[tt+"<"+lst(1)(i)+">"+"\n!\n"];
   for j=1:size(lst(1+i),1) //doit faire test si matrice
    for l=1:size(lst(1+i),2)
     tt=[tt+string(lst(1+i)(j,l))+" "];
    end
    tt=[tt+"\n"];
   end
   tt=[tt+"!\n"];
  end
  mfprintf(u,tt);
  if (flag==1) then
   printf(tt);
  elseif(flag==2) then
   ts=sprintf(tt);
   put_list(ts);
  end
endfunction
```

2.44.6 Fonction(s) utilisée(s)

Add here the used function name and references

2.45 écrit les variables dans un fichier log

- **Nom :** wlog_sv
- **Librairie :** scicos_util - Librairie de fonctions utilitaires pour manipuler les diagrammes scicos à partir du prompt scilab

2.45.1 Séquence d'appel

```
wlog_sv(u,lst,flag)
```

2.45.2 Paramètres

- **u :** add here the parameter description
- **lst :** add here the parameter description
- **flag :** add here the parameter description

2.45.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

2.45.4 Exemple

```
Add here scilab instructions and comments
```

2.45.5 Contenu du fichier

```
//Save variable
function wlog_sv(u,lst,flag)
  tt=">> Save simulated data :\n";
  for i=1:size(lst(1),1)
   tt=[tt+"<"+lst(1)(i)+">"+"\n!\n"];
   for j=1:size(lst(1+i),1)
    for l=1:size(lst(1+i),2)
     tt=[tt+string(lst(1+i)(j,l))+" "];
    end
    tt=[tt+"\n"];
   end
   tt=[tt+"!\n"];
  end
  mfprintf(u,tt);
  if (flag==1) then
   printf(tt);
  elseif(flag==2) then
   ts=sprintf(tt);
   put_list(ts);
  end
endfunction
```

2.45.6 Fonction(s) utilisée(s)

Add here the used function name and references

Chapitre 3

Librairie signal

3.0.1 Description

Add here a paragraph of the function description.

3.1 Affiche le diagramme de Bode d'une fonction de transfert

- **Nom :** affiche_bode
- **Librairie :** signal - Librairie signal

3.1.1 Séquence d'appel

```
affiche_bode(nwin,b1,b2,num,den)
```

3.1.2 Paramètres

- **nwin :** add here the parameter description
- **b1 :** add here the parameter description
- **b2 :** add here the parameter description
- **num :** add here the parameter description
- **den :** add here the parameter description

3.1.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

3.1.4 Exemple

```
Add here scilab instructions and comments
```

3.1.5 Contenu du fichier

```
//affiche_bode
//fonction qui affiche une fonction de transfert
//dans une fenetre graphique scilab
//Entrée : nwin l'id de la fenetre
//         b1 : borne fréquentielle gauche
//         b2 : borne fréquentielle droite
//         num : numerateur de la fct de trnsfrt
//         num : denominateur de la fct de trnsfrt
function []=affiche_bode(nwin,b1,b2,num,den)
 F=syslin('c',num/den);
 xbasc(nwin);
 xset("window",nwin);
 xset("wdim",300,200);
//title='F(s)=(num/den)';
 bode(F,b1,b2);
endfunction
```

3.1.6 Fonction(s) utilisée(s)

Add here the used function name and references

3.2 Retourne les pôles et les zéros d'une boucle numérique de type 2 du troisième ordre

- **Nom :** calcul_3eme_ordre
- **Librairie :** signal - Librairie signal

3.2.1 Séquence d'appel

```
[tau1,tau,tau2] = calcul_3eme_ordre(fn,phi,kv,icp,N)
```

3.2.2 Paramètres

- **fn :** add here the parameter description
- **phi :** add here the parameter description
- **kv :** add here the parameter description
- **icp :** add here the parameter description
- **N :** add here the parameter description
- **tau1 :** add here the parameter description
- **tau :** add here the parameter description
- **tau2 :** add here the parameter description

3.2.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

3.2.4 Exemple

```
Add here scilab instructions and comments
```

3.2.5 Contenu du fichier

```
//Calcul_3eme_ordre - Scilab function -
//Return the zeros and the pole of a third order digital PLL type 2
//Based on the Rhode's Model
//13 octobre 2003 - IRCOM GROUP - Author : A.Layec
function [tau1,tau,tau2]=calcul_3eme_ordre(fn,phi,kv,icp,N)
  wn=2*%pi*fn;
  tau2=(-tan(phi)+1/cos(phi))/wn;
  tau1=1/(wn*wn*tau2);
  tau9=(kv*icp/(2*%pi))/(N*wn*wn);
  tau=tau9*sqrt((1+wn*wn*tau1*tau1)/(1+wn*wn*tau2*tau2));
  c0=(tau2/tau1)*((kv*icp/(2*%pi))/(wn*wn*N))*sqrt((1+(wn*tau1)^2)/(1+(wn*tau2)^2));
  c1=c0*((tau1/tau2)-1);
  r1=tau1/c1;
endfunction
```

3.2.6 Fonction(s) utilisée(s)

Add here the used function name and references

3.3 Retourne les pôles et les zéros d'une boucle numérique de type 2 du quatrième ordre

- **Nom :** calcul_4eme_ordre
- **Librairie :** signal - Librairie signal

3.3.1 Séquence d'appel

```
[tau1,tau,tau2] = calcul_4eme_ordre(fn,phi,kv,icp,N,fcpf,attn)
```

3.3.2 Paramètres

- **fn :** add here the parameter description
- **phi :** add here the parameter description
- **kv :** add here the parameter description
- **icp :** add here the parameter description
- **N :** add here the parameter description
- **fcpf :** add here the parameter description
- **attn :** add here the parameter description
- **tau1 :** add here the parameter description
- **tau :** add here the parameter description
- **tau2 :** add here the parameter description

3.3.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

3.3.4 Exemple

```
Add here scilab instructions and comments
```

3.3.5 Contenu du fichier

```
//Calcul_4eme_ordre - Scilab function -
//Return the zeros and the pole of a fourth order digital PLL type 2
//Based on a NSC's Model
//13 juillet 2004 - IRCOM GROUP - Author : A.Layec
function [tau1,tau,tau2]=calcul_4eme_ordre(fn,phi,kv,icp,N,fcpf,attn)
    wn=2*%pi*fn;
    tau2=(-tan(phi)+1/cos(phi))/wn;
    tau3=sqrt((10^(attn/10)-1)/(2*%pi*fcpf)^2);
    wc=((tan(phi)*(tau2+tau3)/((tau2+tau3)^2+tau2*tau3)))*(sqrt(1+(((tau2+tau3)^2+tau2*tau3))/(tan(phi)*(tau2+tau3)^2))-1);
    tau1=1/((wc^2)*(tau2+tau3));
    c1=(tau2/tau1)*(kv*(icp/(2*%pi))/((wc^2)*N))*sqrt((1+wc^2*tau1^2)/((1+(wc^2)*(tau2^2))*(1+(wc^2)*(tau3^2))));
    c2=c1*(tau1/tau2-1);
    r2=tau1/c2;
    c3=c1/10;
    r3=tau3/c3;
endfunction
```

3.3.6 Fonction(s) utilisée(s)

Add here the used function name and references

3.4 Calcule la taille du tableau de travail nécessaire à la routine de calcul bas niveau dfftmx

- **Nom :** dfftsize
- **Librairie :** signal - Librairie signal

3.4.1 Séquence d'appel

```
[l,ierr] = dfftsize(sizefft)
```

3.4.2 Paramètres

- **sizefft :** add here the parameter description
- **l :** add here the parameter description
- **ierr :** add here the parameter description

3.4.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

3.4.4 Exemple

```
Add here scilab instructions and comments
```

3.4.5 Contenu du fichier

```
////////////////////////////////////
//Function dfftsize
//
//[l,ierr]=dfftsize(sizefft)
//Fonction qui retourne la taille du mot de travail
//nécessaire à la fonction dfftmx pour effectuer une fft.
//Appel la fonction d.f qui est une version raccourcie
//de la fonction SCI/routines/signal/dfftbi.f
//
//sizefft : taille du vecteur d'entrée
//l       : taille du mot(en double) à réserver
//ierr    : numéro erreur
function [l,ierr]=dfftsize(sizefft)
[l,ierr]=call('d',sizefft,1,'i','out',[1,1],2,'i',[1,1],3,'i')
endfunction
```

3.4.6 Fonction(s) utilisée(s)

Add here the used function name and references

3.5 Calcule les coefficients de filtres RIF communément employés en communication numérique

- **Nom :** filter_tap
- **Librairie :** signal - Librairie signal

3.5.1 Séquence d'appel

```
pulse = filter_tap(typ,nb_coef,fe,param,gain)
```

3.5.2 Paramètres

- **typ :** add here the parameter description
- **nb_coef :** add here the parameter description
- **fe :** add here the parameter description
- **param :** add here the parameter description
- **gain :** add here the parameter description
- **pulse :** add here the parameter description

3.5.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

3.5.4 Exemple

Add here scilab instructions and comments

3.5.5 Contenu du fichier

```
//filter_tap macro qui retourne les coefficients d'une
//réponse impulsionnelle pour filtrage RIF
//05-01-2005 Alan Layec -IRCOM Lab-
//
//typ = 1 : Root Raised Cosine (param : alpha=roll-off)
//    = 2 : Raised Cosine (param : alpha=roll-off)
//    = 3 : Gauss (param : beta=BT)
//nb_coef : desired lenght of impulse response
//fe      : sampling frequency
//param   : parameters of response (see typ description)
//gain    : output gain
//
//ex :
//nb_coef=64
//Ne=12
//r=0.35
//gain=1
//[pulse]=filter_tap(1,64,Ne,r,gain)
function [pulse]=filter_tap(typ,nb_coef,fe,param,gain)

if typ<>1&typ<>2&typ<>3 then
 message('Only 1,2 or 3 must be choosen for type of filtering');
 pulse=[];
 abort;
end

if typ==1 then
//RRC
r=param;
t_Ts=1/fe*(-nb_coef/2:nb_coef/2-1)+%eps;
pulse = gain*4*r/%pi*(cos((1+r)*%pi*t_Ts) + (sin((1-r)*%pi*t_Ts)./(4*r*t_Ts)))./(1-(4*r*t_Ts).^2);

elseif typ==2 then
//RC
r=param;
t_Ts=1/fe*(-nb_coef/2:nb_coef/2-1);
h1=cos(%pi*r*t_Ts)./(1-(4*r^2*t_Ts.^2)+(abs(r*t_Ts)==1/2))+(abs(r*t_Ts)==1/2)*%pi/4;
h2=(sin(%pi*t_Ts))./(%pi*t_Ts+(t_Ts==0))+(t_Ts==0);
pulse=gain*(h1.*h2);

elseif typ==3 then
//Gauss
b=param;
t_Ts=1/fe*(-nb_coef/2:nb_coef/2-1);
pulse=1/fe*gain*b*sqrt((2*%pi)/log(2))*exp(-2/log(2)*(b*%pi*t_Ts)^2);
end
//pulse=typ;
endfunction
```

3.5.6 Fonction(s) utilisée(s)

Add here the used function name and references

3.6 Affiche des spectres dans une fenêtre graphique

- **Nom :** plot_spectre
- **Librairie :** signal - Librairie signal

3.6.1 Séquence d'appel

```
plot_spectre(nwin,DSP,step,f1,flength,flag1,flag2,f3)
```

3.6.2 Paramètres

- **nwin :** add here the parameter description
- **DSP :** add here the parameter description
- **step :** add here the parameter description
- **f1 :** add here the parameter description
- **flength :** add here the parameter description
- **flag1 :** add here the parameter description
- **flag2 :** add here the parameter description

– **f3 :** add here the parameter description

3.6.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

3.6.4 Exemple

```
Add here scilab instructions and comments
```

3.6.5 Contenu du fichier

```
////////////////////////////////////
//Function plot_spectre
//
//Fonction qui affiche 10*log10(DSP) dans une fenetre
//
//nwin    : numéro de fenêtre
//DSP     : vecteur de la DSP
//step    : pas fréqentiel
//f1      : fréquence gauche
//flength : largeur fréquentielle de la fenêtre
//flag1   : flag normalisation
//flag2   : flag échelle logarithmique
//f3      : option supplémentaire
//        |--------|--------|
//        f1       f0       f2
//        n1       n0       n2
function []=plot_spectre(nwin,DSP,step,f1,flength,flag1,flag2,f3)
        //Longueur max de la fenetre scilab
        win_len_max=2^19+2^18;

        //Longueur du vecteur DSP
        size_dsp=size(DSP,'*');

        //Calcul bornes
        f2=f1+flength;
        n1=f1/step;
        n2=f2/step;
        //n=size_dsp/2+1+(n1:1:n2);
n=(n1:1:n2);

        //Teste la longueur de la fenetre
if(size(n,'*')>win_len_max) then
          error('Fenêtre trop large pour être affichée');
end;

        //Assigne et crée la fenêtre nwin
        xset("window",nwin);
        xbasc(nwin);
        xset("wdim",300,200);

        //normalise au maximum de la DSP
        if (flag1=="n") then
          DSP=DSP/max(DSP);
        end;

        //Affiche la DSP
        if(flag2=="ln") then
          //Calcul la puissance de 10
  //supérieure à step
  if(~exists("f3")) then
   i=0;j=1;
   while (i==0) then
     if(int(step/10^j)<>0) then
        j=j+1;
     else
        i=1;
     end;
   end;
          b1=10^j;
          else
   b1=step;
  end;
   f=(f1:step:f2)-f1+1;
           plot2d("ln",f,10*log10(DSP(n))-10*log10(step),...
                  ,"011",rect=[b1,min(10*log10(DSP(n+2))-10*log10(step)),...
                  max(f),max(10*log10(DSP(n+2))-10*log10(step))]);
        else
          f=f1:step:f2;
          plot2d(f,10*log10(DSP(n)),...
                  ,"011",rect=[min(f),min(10*log10(DSP(n))),max(f),max(10*log10(DSP(n)))]);
        end;
endfunction
```

3.6.6 Fonction(s) utilisée(s)

Add here the used function name and references

3.7 Calcule et affiche des spectres dans une fenêtre graphique d'une forme temporelle avec translation de fréquence

- **Nom :** plot_spectre2
- **Librairie :** signal - Librairie signal

3.7.1 Séquence d'appel

```
step = plot_spectre2(vart,Tse,Tacqui,Fd,l_win,nwin,flag)
```

3.7.2 Paramètres

- **vart :** add here the parameter description
- **Tse :** add here the parameter description
- **Tacqui :** add here the parameter description
- **Fd :** add here the parameter description
- **l_win :** add here the parameter description
- **nwin :** add here the parameter description
- **flag :** add here the parameter description
- **step :** add here the parameter description

3.7.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

3.7.4 Exemple

```
Add here scilab instructions and comments
```

3.7.5 Contenu du fichier

```
//////////////
//
//plot_spectre_2: fonction qui affiche le spectre d'une forme temporelle sous-échantillonnée
//                par translation de spectre
//
//Entrées
//vart : vecteur de la forme temporelle de taille (Nsav,1)
//Tse : pas temporel d'echantillonnage
//Tacqui : temps initial au début de l'analyse
//Fd : Fréquence centrale du spectre à observer
//l_win : taille de la fenêtre spectrale désirée (en nbr d'échantillons)
//nwin  : numéro de fenêtre
//flag  : drapeau échelle logarithmique flag==0 echelle linéaire
//                                      flag==1 échelle logarithmique
//
function [step]=plot_spectre2(vart,Tse,Tacqui,Fd,l_win,nwin,flag)

   //Calcul la longueur du vecteur de la forme temporelle
   Nsav=size(vart,'*');

   //Calcul du pas fréquentiel
   step=1/(Nsav*Tse);

   //Calcul de la date finale de l'analyse
   Tfin=Tacqui+Nsav*Tse;

   //Calcul du vecteur temps pour translation
   t=Tacqui:Tse:Tfin-Tse;

   //Réalise translation
   vart_t=vart'.*cos(2*%pi*Fd*t);

   //Calcul du spectre (non normalisé)
   DSP=10*log10((1/(Nsav)*abs(fftshift(fft(vart_t,-1))))^2)-10*log10(step);

   if flag==0
    //Calcul du vecteur échantillon
    vec=(Nsav-l_win)/2+1:1:(Nsav+l_win)/2;

    //Calcul du vecteur fréquentiel
    vec_f=(vec-Nsav/2)*step+Fd-step;
```

```
    //Ouvre la fenetre graphique
    scf(nwin);
    f=scf(nwin);
    f.figure_size=[300,350];

    //
    plot2d(vec_f,DSP(vec),rect=[min(vec_f),min(DSP(vec)),max(vec_f),max(DSP(vec))+5]);

   elseif flag==1
    //Calcul du vecteur échantillon
    vec=Nsav/2:1:(Nsav-1)-Nsav/2+l_win;

    //Calcul du vecteur fréquentiel
    vec_f=(vec-Nsav/2)*step;

    //Ouvre la fenetre graphique
    scf(nwin);
    f=scf(nwin);
    f.figure_size=[300,350];

    //
    plot2d("ln",vec_f+2*step,DSP(vec+2),,"011",rect=[step,min(DSP(vec+2)),max(vec_f+2*step),max(DSP(vec+2))]);
   end

endfunction
```

3.7.6 Fonction(s) utilisée(s)

Add here the used function name and references

3.8 Affiche une courde de TEB dans une fenêtre graphique

- **Nom :** plot_teb
- **Librairie :** signal - Librairie signal

3.8.1 Séquence d'appel

```
plot_teb(nwin,sig_log,teb,flag)
```

3.8.2 Paramètres

- **nwin :** add here the parameter description
- **sig_log :** add here the parameter description
- **teb :** add here the parameter description
- **flag :** add here the parameter description

3.8.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

3.8.4 Exemple

```
Add here scilab instructions and comments
```

3.8.5 Contenu du fichier

```
////////////////////////////////////
//Function plot_teb
//
//Fonction qui affiche une courbe de TEB dans une fenetre graphique
//
//nwin    : numéro de fenêtre
//sig_log : le vecteur de l'axe x
//teb     : le vecteur de l'axe y
//flag    : option 1 : dessine une grille
function []=plot_teb(nwin,sig_log,teb,flag)
        xset("window",nwin);
        xset("wdim",300,200);
        xbasc();
        for i=1:size(teb,1)
          if teb(i)==0 then teb(i)=%eps,end;
        end
```

```
        plot2d("nl",sig_log,teb,-6);
        plot2d("nl",sig_log,teb);
        if flag=="1" then
          xgrid();
        end
endfunction
```

3.8.6 Fonction(s) utilisée(s)

Add here the used function name and references

3.9 Fonction polyfit mtlb

- **Nom :** polyfit
- **Librairie :** signal - Librairie signal

3.9.1 Séquence d'appel

```
p = polyfit(x,y,n,s)
```

3.9.2 Paramètres

- **x :** add here the parameter description
- **y :** add here the parameter description
- **n :** add here the parameter description
- **s :** add here the parameter description
- **p :** add here the parameter description

3.9.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

3.9.4 Exemple

```
Add here scilab instructions and comments
```

3.9.5 Contenu du fichier

```
function [p]=polyfit(x, y, n, s)
// return coefficient vector or poly if fourth string argument given
[lhs, rhs] = argn(0)
x = x(:); y = y(:)
m = length(x)
if length(y) <> m, error('x and y must have same length'), end
v = ones(m,n+1)
for i=2:n+1, v(:,i) = x.*v(:,i-1), end
p = (v\y)'
if rhs > 3, p = poly(p, s, 'coeff'), end
endfunction
```

3.9.6 Fonction(s) utilisée(s)

Add here the used function name and references

Chapitre 4

Libraires de fonctions Scilab Communication

4.0.1 Description

Add here a paragraph of the function description.

4.1 Générateur de nombre entier aléatoire

- **Nom :** genint
- **Librairie :** mod_num_sci_lib - Libraires de fonctions Scilab Communication

4.1.1 Paramètres

- **Name of param 1 :** add here the parameter description

4.1.2 Description

Add here a paragraph of the function description.

4.2 Modulateur par états De Phase M-aires

- **Nom :** modpsk
- **Librairie :** mod_num_sci_lib - Libraires de fonctions Scilab Communication

4.2.1 Paramètres

- **Name of param 1 :** add here the parameter description

4.2.2 Description

Add here a paragraph of the function description.

4.3 élévateur de cadence

- **Nom :** surecht
- **Librairie :** mod_num_sci_lib - Libraires de fonctions Scilab Communication

4.3.1 Paramètres

- **Name of param 1 :** add here the parameter description

4.3.2 Description

Add here a paragraph of the function description.