"Modnum"
Scilab toolbox
for the communication systems
Internals Guide

Draft Version

IRCOM Group
Alan Layec

February 8, 2006

Contents

Part I

Utilities Scilab scripts and macros

Chapter 1

Utilities Scilab scripts

1.1 builder

- **Short description:** builder scilab script of the toolbox

1.1.1 File content

```
//builder.sce
//build mod_num_3 package for scilab-3.0
//both linux and windob(with lcc)
//13-10-2004 Alan
//
//Modification : 4/4/2005
//builder rewritten with scilab function
//Improvment : Build with VC++

// REVISION HISTORY :
// $Log$
//

//def_MODNUM_path()
//Input : nothing
//Output : tt MODNUM path (ex:/home/alan/mod_num_3)
function tt=def_MODNUM_path()
 tt=get_absolute_file_path('builder.sce');
 end_char=part(tt,length(tt));
 if end_char=='/'|end_char=='\' then
  tt=part(tt,1:length(tt)-1);
 end
endfunction

mode(-1);
lines(0);

//define MODNUM path
MODNUM=def_MODNUM_path();

//Search Scilab Version
getf(MODNUM+'/macros/build_util/find_scilab_ver.sci');
[ok]=find_scilab_ver();
if ~ok then
 printf("Mod_num toolbox source version only build with scilab >=3.0\n");
 abort
end

//open loader.sce file for writting
ierror=execstr('u=mopen(MODNUM+""/loader.sce"",""w"");','errcatch');
if ierror<>0 then
 printf("Can''t write loader file.\nVerify your writting access.\n");
end
clear ierror;

//write Header of loader.sce file
getf(MODNUM+'/macros/build_util/write_header.sci');
write_header(u,'builer.sce')

//Build and load util macro library
getf(MODNUM+'/macros/build_util/build_lib.sci');
build_lib('/macros/build_util','mod_num_util');
getf(MODNUM+'/macros/build_util/write_inf_lib.sci');
txt=write_inf_lib(u,'/macros/build_util','mod_num_util',1);
ierr=execstr(txt,'errcatch');

//Build and load all scilab library
libs=['generate_doc';'gen_doc_util';'xmltotex';
      'misc';'signal';'scicos_util';'find_file'];
rep_lib='/macros/'+libs;
lib_name='mod_num_'+libs;
build_lib(rep_lib,lib_name);
txt=write_inf_lib(u,rep_lib,lib_name,1);
ierr=execstr(txt,'errcatch');
```

```
//Build and load palette of scicos_blocks
pal=['Tools';'Sources';'Sinks';'Pll';'NonLinear';'Communication'];
rep_pal='/macros/scicos_blocks/'+pal;
pal_title='mod_num_'+convstr(pal,'l');
lib_name=pal_title+'_pal';
build_lib(rep_pal,lib_name);
fprintf(u,"%s\n",'//Load mod_num scicos_blocks library');
txt=write_inf_lib(u,rep_pal,lib_name,0);
ierr=execstr(txt,'errcatch');

//Generate palette of scicos_blocks
for i=1:size(rep_pal,"*")
  files_sci=basename(listfiles(MODNUM+rep_pal(i)+"/*.sci"));
  generate_palette(files_sci,MODNUM+'/macros/scicos_blocks/',pal(i));
end
txt=write_inf_pal(u,rep_pal+'.cosf',pal_title);
ierr=execstr(txt,'errcatch');

clear txt;clear pal_title;clear lib_name;clear rep;clear rep_pal;
clear ierr;clear files_sci;clear i; clear pal;

//Build routines Library
printf("Build mod_num routines libraries\n");

//Find C compiler
[flagc,ccmd]=find_cmd('c');
//Find Fortran compiler
[flagf,fcmd]=find_cmd('f');
//find witch directory to include for compilation
incl_path=find_incl_path();

//Build and load mod_num_lib routines
printf("Build mod_num_lib C routines\n");
path='/routines/mod_num_lib';
files=basename(listfiles(MODNUM+path+'/*.c'));
libname="libmodnum_lib";
scifunc=["genint";"modpsk";"surecht"];
intname="intmod_num_lib";
if ccmd<>'trash' then
 precompilation(flagc,ccmd,incl_path,path,files);
 create_library(flagc,ccmd,libname,path,files);
 txt=write_inf_rout_lib(u,libname,path,files,0);
 ierr=execstr(txt,'errcatch');
 txt=write_addinter_line(u,libname,path,intname,scifunc);
 ierr=execstr(txt,'errcatch');
end
clear files, clear libname; clear scifunc; clear intname;

//Build and load scicos C routines
printf("Build scicos blocks C routines\n");
path='/routines/scicos';
files_c=basename(listfiles(MODNUM+path+'/*.c'));
libname="libmodnum_c";
if ccmd<>'trash' then
 precompilation(flagc,ccmd,incl_path,path,files_c);
 create_library(flagc,ccmd,libname,path,files_c);
 txt=write_inf_rout_lib(u,libname,path,files_c,1);
 ierr=execstr(txt,'errcatch');
end
clear files_c; clear libname;

//Build and load scicos F routines
printf("Build scicos blocks Fortran routines\n");
files_f=basename(listfiles(MODNUM+path+'/*.f'));
libname="libmodnum_f";
if fcmd<>'trash' then
 precompilation(flagf,fcmd,incl_path,path,files_f);
 create_library(flagf,fcmd,libname,path,files_f);
 txt=write_inf_rout_lib(u,libname,path,files_f,2);
 ierr=execstr(txt,'errcatch');
end
clear path; clear files_f; clear libname;
clear incl_path;clear flagc;clear flagf;
clear ccmd;clear fcmd; clear ierr;

//Build documentation and demos
txt=write_inf_doc(u);
ierr=execstr(txt,'errcatch');
if ierr==0 then
 %helps=%helps;
end
build_demo();
//flag_doc=build_doc();

//Close loader.sce
mclose(u);
printf("Write a loader.sce file\n");

//write a .scilab file
if ~MSDOS then
 txt=mgetl('loader.sce');
 mputl(txt,'.scilab');
 printf("Write a .scilab file\n");
end
clear u; clear txt; clear ierr; clear ok;
```

1.2 load_generate_doc_function

- **Short description:** load global variables of the documentation generator of the toolbox

1.2.1 File content

```
//retrieve default LANGUAGE
if ~exists('LANGUAGE') then
  global LANGUAGE;lang=LANGUAGE;clear LANGUAGE;
else
  lang=LANGUAGE;
end

if lang<>'eng' & lang<>'fr' then
 printf("language %s is not supported, switch to ''eng''\n",lang);
 lang='eng'
end
//////////////////////////
//Define MODNUM directories
//////////////////////////

//man path
man_path  = MODNUM+'/man/';
//data_path = man_path+'data/'+lang+'/';
data_path = man_path+'data/';
//xml_path  = man_path+'xml/'+lang+'/';
xml_path  = man_path+'xml/';
//tex_path  = man_path+'tex/'+lang+'/';
tex_path  = man_path+'tex/';
bib_path  = man_path+'tex/bib/'; //(!!??!)
//html_path = man_path+'htm/'+lang+'/';
html_path = man_path+'htm/';
//pdf_path  = man_path+'pdf/'+lang+'/';
pdf_path  = man_path+'pdf/';
sbeq_path = man_path+'sblock_equiv/';
web_path  = man_path+'web/';


//
pal_path  = MODNUM+'/macros/scicos_blocks/';
simu_path = MODNUM+'/simu/';
//spec_desc_path=MODNUM+'/man/spec_desc/';
//PAS NECESSAIRE->c'est aussi bien dans tex_path... pour l'instant
rout_path = MODNUM+'/routines/scicos/';
mac_path  = MODNUM+'/macros';
low_rout_path=MODNUM+'/routines/mod_num_lib/';
scs_diagr_path=MODNUM+'/scs_diagr/';

//return master list of files and directories
if ~exists('tt_ml') then
 tt_ml=return_master_list(MODNUM);
end

//Define diagram list
//diagr_cs=[];
diagr_cs=[scs_diagr_path+'dyna/lorentz/','lorentz'
          scs_diagr_path+'dyna/van_der_pol/','van_der_pol_trap'
          scs_diagr_path+'dyna/van_der_pol/','van_der_pol_forc_euler'
          scs_diagr_path+'dyna/duf_van_der_pol/','duf_van_der_pol'
          scs_diagr_path+'dyna/rossler/','rossler'
          scs_diagr_path+'dyna/duffing/','duffing'
          scs_diagr_path+'dyna/chua/','chua'
          scs_diagr_path+'dyna/chua/','chua_sub'
          scs_diagr_path+'dyna/chua/','chua_masque'];
//diagr_ds=[];
diagr_ds=[scs_diagr_path+'dyna/logistique/','logistique_bif_2D'
          scs_diagr_path+'dyna/logistique/','logistique_bif_3D'
          scs_diagr_path+'dyna/henon/','henon'
          scs_diagr_path+'dyna/sig_delta/','sig_delta_1st_order'
          scs_diagr_path+'dyna/frey/','frey'
          scs_diagr_path+'dyna/lin_chua/','lin_chua'
          scs_diagr_path+'dyna/lin_chua/','lin_chua_cod_decod'];
//diagr_os=[];
diagr_os=[scs_diagr_path+'pll/vco/','scicos_vco'
          scs_diagr_path+'pll/vco/','discr_vco'];
//diagr_is=[];
diagr_is=[scs_diagr_path+'pll/synthe/','synthe_scicos';
          scs_diagr_path+'pll/synthe/','synthe_eclat';
          scs_diagr_path+'pll/synthe/','synthe_int';
          scs_diagr_path+'pll/synthe/','synthe_interp';];
//diagr_fs=[];
diagr_fs=[scs_diagr_path+'pll/synthe_frac/','synthe_sd_quick'];
//diagr_PSK=[];
diagr_PSK=[scs_diagr_path+'comsys/vectorial/qpsk/','qpsk_teb';
           scs_diagr_path+'comsys/vectorial/qpsk/','qpsk_teb_int';
           scs_diagr_path+'comsys/vectorial/qpsk/','qpsk_etal_teb';
           scs_diagr_path+'comsys/sequential/qam/','qam_seq'];
//diagr_SD=[];
diagr_SD=[scs_diagr_path+'comsys/sequential/sig_delta/','mash_1er_ordre';
          scs_diagr_path+'comsys/sequential/sig_delta/','mash_2eme_ordre';
          scs_diagr_path+'comsys/sequential/sig_delta/','mash_gauss';
          scs_diagr_path+'comsys/sequential/sig_delta/','sig_delta_2';
          scs_diagr_path+'comsys/sequential/sig_delta/','sig_delta_3'];
//diagr_FSK=[];
diagr_FSK=[];
//diagr_FSK_chaos=[];
diagr_FSK_chaos=[scs_diagr_path+'pll/transchaos/','trans_chaos_em';
                 scs_diagr_path+'pll/transchaos/','trans_chaos_em_rec'];
```

```
//diagr_elec=[];
diagr_elec=[scs_diagr_path+'electrical/','atten'];

diagr_all=[diagr_cs;diagr_ds;diagr_os;
           diagr_is;diagr_fs;
           diagr_FSK;diagr_PSK;diagr_SD;diagr_FSK_chaos;
           diagr_elec];

//Define simulation script list
sim_chaos=[simu_path,'lin_chua_sim';
           simu_path,'lin_chua_teb_sim'];
sim_synthe=[simu_path,'synthe_int_sim'
            simu_path,'synthe_int_jit_sim'
            simu_path,'synthe_sd_quick_sim'];
sim_PSK=[simu_path,'mash_gauss_sim';
         simu_path,'qpsk_teb_sim';
         simu_path,'rayleigh_sim'];
sim_all=[sim_chaos;sim_synthe;sim_PSK];
//Define script list
sce_all=[MODNUM+'/','builder';
         mac_path+'/generate_doc/','load_generate_doc_function'];
//Define if simulation of script file are executed
with_sim=%t;
//Define library name for internal section
lib_build=['build_util';'find_file'];
lib_gen_doc=['gen_doc_util';'generate_doc';'xmltotex'];
//Define excluded library
ex_lib_name=['other';'scicos_blocks';lib_build;lib_gen_doc];
//Define interfaced functions of modnum library
mod_num_sci_lib='mod_num_sci_lib';
if ~exists('modnum_sci_func') then
 sci_func=["genint";"modpsk";"surecht"];
end
//Define name of routines library of modnum
mod_num_rout_lib='mod_num_rout_lib';
//Define latex command
latex_cmd='latex -interaction=nonstopmode ';
//Define dvips command
dvips_cmd='dvips -E ';
//Define latex2html command
latex2html_cmd='latex2html -white -info """" -no_navigation -link 0 -split 3 -short_extn -image_type gif -prefix ';
//Define bibtex command
bibtex_cmd='bibtex ';
//Define web browser
wbr_cmd='mozilla';
//Define with_gui flag
with_gui=%t;
//Define xwd command
xwd_cmd='xwd ';
//Define dir commmand
dir_cmd='ls ';
//Define mkdir command (MUST USE scilab function mkdir('')
mkdir_cmd='mkdir ';
//Define move file command
mv_cmd='mv -f ';
//Define remove file command
rm_cmd='rm -fr ';
//Define copy command
cp_cmd='cp -fr ';
//Define scilab browser flag
sci_browser=%t;
//Load file of function
//Disable scilab function protection
prot=funcprot();
funcprot(0);
////////////////

//Build and load generate_doc library
build_lib('/macros/generate_doc','mod_num_generate_doc');
mod_num_generate_doc=lib(MODNUM+'/macros/generate_doc/');
//Build and load xml2tex library
build_lib('/macros/xmltotex','mod_num_xmltotex');
mod_num_xmltotex=lib(MODNUM+'/macros/xmltotex/');
//Build and load generate_doc_util library
build_lib('/macros/gen_doc_util','mod_num_gen_doc_util');
mod_num_gen_doc_util=lib(MODNUM+'/macros/gen_doc_util/');

//Return to original scilab function protection mode
funcprot(prot);

//increase stacksize (for scicos_simulate)
stacksize(60000000);
```

Chapter 2

Library of utilities functions to build the toolbox

2.1 Demonstration file builder of the toolbox

- **Name:** build_demo
- **Library:** build_util - Library of utilities functions to build the toolbox

2.1.1 Calling Sequence

```
build_demo
```

2.1.2 File content

```
//fonction qui construit les fichiers
//demos de la boîte à outils
function build_demo()
  printf("Generate demos of Mod_num...\n");
  exec(MODNUM+'/macros/generate_doc/load_generate_doc_function.sce',-1);
  generate_modnum_dem();
  printf("Done\n");
endfunction
```

2.2 Documentation builder of the toolbox

- **Name:** build_doc
- **Library:** build_util - Library of utilities functions to build the toolbox

2.2.1 Calling Sequence

```
flag = build_doc()
```

2.2.2 Parameters

- **flag :** a flag

2.2.3 File content

```
//buid_doc
//Entrée :
//Sortie : flag
function  [flag]=build_doc()
printf("Build documentation\n");
//if MSDOS then
// disp('need winfig latex2html!')
//else
// exec(MODNUM+'macros/generate_doc/load_generate_doc_function.sce');
// generate_mod_num_html();
// unix_g('mv -f htm/ ./man/');
```

```
//end
 flag=[];
endfunction
```

2.3 Scilab macro libraries builder of the toolbox

- **Name:** build_lib
- **Library:** build_util - Library of utilities functions to build the toolbox

2.3.1 Calling Sequence

```
build_lib(path,tt)
```

2.3.2 Parameters

- **path :** string. path of the directory which contains the scilab macros
- **tt :** string. a library name (for id)

2.3.3 File content

```
//build_lib
//fonction qui crée les libraires des macros scilab
//Entrée : path chemin de la libraire dans MODNUM (ex: macros/util/)
//         tt nom de la librairie (ex:mod_num_util)
//Sortie : néant
// txt : Information utile de chargement
//               ex : mod_num_scicos_utils=lib(MODNUM+'/macros/scicos_util/');
function []=build_lib(path,tt)
 if size(path,'*')==size(tt,'*') then
   for i=1:size(path,'*')
    //Affiche un message
    printf("Build "+tt(i)+" macro library\n");
    //construit la libraire tt dans le chemin path
    str='genlib('''+tt(i)+''',MODNUM+'''+path(i)+''',%t);'
    ierr=execstr(str,'errcatch')
    if ierr<> 0 then
     printf("Build lib : error\n");
     abort
    end
   end
 else
  printf("path and tt must have the same size");
  abort
 end
endfunction
```

2.4 Change the path of the toolbox directory in makefile

- **Name:** change_path_makefile
- **Library:** build_util - Library of utilities functions to build the toolbox

2.4.1 Calling Sequence

```
txt = change_path_makefile(MakeName)
```

2.4.2 Parameters

- **MakeName :** string. path+name of the Makefile
- **txt :** vector of strings. text of the new Makefile file

2.4.3 File content

```
function txt=change_path_makefile(MakeName)

  txt=mgetl(MakeName);
  for j=1:size(txt,1)

   //change scilab path
   if strindex(txt(j),'SCIDIR =')<>[] then
     txt(j)='SCIDIR = '+SCI;
   end

  //change MODNUM path
   if strindex(txt(j),'MODNUMDIR =') <>[] then
    txt(j)='MODNUMDIR = '+MODNUMDIR;
  end
  mputl(txt,MakeName);
 end
endfunction
```

2.5 Posix to lcc makefile converter (OBSOLETE)

- **Name:** convert_makefile_for_lcc
- **Library:** build_util - Library of utilities functions to build the toolbox

2.5.1 Calling Sequence

```
tt = convert_makefile_for_lcc(MakeName)
```

2.5.2 Parameters

- **MakeName :** string. path+name of the Makefile
- **tt :** vector of strings. text of the new Makefile file

2.5.3 File content

```
function tt=convert_makefile_for_lcc(MakeName)

  txt=mgetl(MakeName);
  tt=[];
  SCIDIR=strsubst(SCI,'/','\');
  k=1
  for j=1:size(txt,1)

   //change scilab path
   if strindex(txt(j),'SCIDIR =')<>[] then
     tt(k)='SCIDIR = '+"""+SCIDIR+""";
     k=k+1;
   end

  //change MODNUM path
   if strindex(txt(j),'MODNUMDIR =') <>[] then
    tt(k)='MODNUMDIR = '+"""+MODNUMDIR+""";
    k=k+1;
  end

   //change OBJS extension
   if strindex(txt(j),'OBJS =')<>[] then
    tt(k)=strsubst(txt(j),'.o','.obj');
    k=k+1
   end

   //change OBJSSTAN extension
    if strindex(txt(j),'OBJSSTAN=')<>[] then
      tt(k)=strsubst(txt(j),'.o','.obj');
      k=k+1;
    end

   //change SCILIBS
    if strindex(txt(j),'SCILIBS =')<>[] then
      tt(k)='SCILIBS = '+"""+SCIDIR+"\bin\LibScilablcc.lib"""
      k=k+1;
   end

   //change OTHERLIBS
    if strindex(txt(j),'OTHERLIBS =')<>[] then
     tt(k)=strsubst(txt(j),'/','\');
     tt(k)=strsubst(tt(k),'.o','.obj');
     k=k+1;
   end

   //change CFLAGS
   if strindex(txt(j),'CFLAGS =')<>[] then
      tt(k)='CFLAGS = -I""$(SCIDIR)\routines"" -I""$(SCIDIR)\routines\scicos"" -I""$(SCIDIR)\routines\sun"" -I""$(SCIDIR)\routines\f2c"" -Dmexfunction_=m
```

```
      k=k+1;
   end

   //copy LIBRARY line
   if strindex(txt(j),'LIBRARY =')<>[] then
      tt(k)=txt(j)
      k=k+1;
   end

  end
   //add line for lcc
   tt(k)='CC = lcc'; k=k+1;
   tt(k)='LINKER = lcclnk'; k=k+1;
   tt(k)='LINKER_FLAGS = -dll -nounderscores';k=k+1;
   tt(k)='DUMPEXTS = $(SCIDIR)\bin\dumpexts'; k=k+1;

   //add line for making all
   tt=[tt
       ''
       'all :: $(LIBRARY).dll'
       '$(LIBRARY).dll: $(OBJS)'
       '@echo Creation of dll $(LIBRARY).dll and import lib from ...'
       '@echo $(OBJS)'
       '$(DUMPEXTS) -o ""$*.def"" ""$*.dll"" $(OBJS)'
       '$(LINKER) $(LINKER_FLAGS) $(OBJS) $(OTHERLIBS) $(SCILIBS) $(XLIBSBIN) $(TERMCAPLIB) $*.def -o $(LIBRARY).dll'
       'c.obj:'
       '@echo ------------- Compile file $< --------------'
       '$(CC) $(CFLAGS) $< '
       ]
   mputl(tt,MakeName+'.lcc');

endfunction
```

2.6 Create shared routines library

- **Name:** create_library
- **Library:** build_util - Library of utilities functions to build the toolbox

2.6.1 Calling Sequence

`create_library(flag,cmd,libname,path,listf)`

2.6.2 Parameters

- **flag :** string. compiler flag
  - **'GCC' :** gcc compiler
  - **'LCC' :** lcc-win32 compiler
  - **'VC' :** visual c/c++ compiler
- **cmd :** string. the command compiler (ex :'gcc','cl',..)
- **libname :** string. name of the library to produce (extension free)
- **path :** string. the target path which must be include the source file
- **listf :** vector of string. the list of file to include in library (extension free)

2.6.3 File content

```
//create_library
//Entrée flag : flag compilateur (GCC,LCC,VC)
//       cmd  : commande du compilateur
//       libname :  nom de la libraire (sans extension : ex mylibrary)
//       path  : le chemin du répertoire de compilation
//       listfo : liste des fichiers à inclure sans extensions (ex:mymodule)
function []=create_library(flag,cmd,libname,path,listf)
 printf(" Create shared library...\n");
 select flag

  case 'GCC' then
    CCFLAG=[];
    listf=listf+'.o'
    for i=1:size(listf,'*')
     CCFLAG=CCFLAG+listf(i)+" ";
    end
    CCFLAG=CCFLAG+'-shared -o '+libname+'.so';
```

```
    cmd=cmd+CCFLAG;

  case 'G77' then
    FCFLAG=[] ;
    listf=listf+'.o'
    for i=1:size(listf,'*')
     FCFLAG=FCFLAG+listf(i)+" ";
    end
    FCFLAG=FCFLAG+'-shared -o '+libname+'.so';
    cmd=cmd+FCFLAG;

  case 'LCC' then
    listf=listf+'.obj';
    CCFLAG=[];
    for i=1:size(listf,'*')
     CCFLAG=CCFLAG+listf(i)+" ";
    end
    cmd=[SCI+'\bin\dumpexts -o '+libname+'.def '+libname+'.dll '+CCFLAG];
    //change directory
    rep=pwd();
    chdir(MODNUM+path);
    unix_g(cmd);
    chdir(rep);
    if libname=='libmodnum_c' then
     CCFLAG=CCFLAG+' ..\mod_num_lib\libmodnum_lib.lib';
    end
    sci_lib=[];
    if fileinfo(SCI+'\bin\LibScilab.lib')<>[] then
      sci_lib=sci_lib+SCI+'\bin\LibScilab.lib ';
    end
    if fileinfo(SCI+'\bin\atlas.lib')<>[] then
      sci_lib=sci_lib+SCI+'\bin\atlas.lib ';
    end
    cmd=[SCI+'\lcc\bin\lcclnk -dll -nounderscores '+CCFLAG+' '+sci_lib+libname+'.def -o '+libname+'.dll'];
    cmd=pathconvert(cmd,%f,%t,'w');

  case 'VC' then
    listf=listf+'.obj';
    CCFLAG=[];
    for i=1:size(listf,'*')
     CCFLAG=CCFLAG+listf(i)+" ";
    end
    cmd=[SCI+'\bin\dumpexts -o '+libname+'.def '+libname+'.dll '+CCFLAG];
    //change directory
    rep=pwd();
    chdir(MODNUM+path);
    unix_g(cmd);
    chdir(rep);
    if libname=='libmodnum_c' then
     CCFLAG=CCFLAG+' ..\mod_num_lib\libmodnum_lib.lib';
    end
    sci_lib=[];
    if fileinfo(SCI+'\bin\LibScilab.lib')<>[] then
      sci_lib=sci_lib+SCI+'\bin\LibScilab.lib ';
    end
    if fileinfo(SCI+'\bin\atlas.lib')<>[] then
      sci_lib=sci_lib+SCI+'\bin\atlas.lib ';
    end
    cmd=['link '+CCFLAG+' '+sci_lib+'/dll /out:'+libname+'.dll /def:'+libname+'.def'];
  case 'trash' then
    printf("Compilation aborted\n");
    return;
 end

  //change directory
 rep=pwd();
 chdir(MODNUM+path);
 //link
 unix_g(cmd);
 //change directory
 chdir(rep);
endfunction
```

2.7 Define absolute path of the toolbox

- **Name:** def_MODNUM_path
- **Library:** build_util - Library of utilities functions to build the toolbox

2.7.1 Calling Sequence

```
tt = def_MODNUM_path
```

2.7.2 Parameters

- **tt :** the absolute path of the modnum directory

2.7.3 File content

```
//def_MODNUM_path()
//Entrée : néant
//Sortie : tt chemin de MODNUM (ex:/home/mod_num_3)
function tt=def_MODNUM_path()
 tt=get_absolute_file_path('builder.sce');
 end_char=part(tt,length(tt));
 if end_char=='/'|end_char=='\' then
  tt=part(tt,1:length(tt)-1);
 end
endfunction
```

2.8 Search and find compiler command

- **Name:** find_cmd
- **Library:** build_util - Library of utilities functions to build the toolbox

2.8.1 Calling Sequence

```
[flag,cmd] = find_cmd(lang)
```

2.8.2 Parameters

- **lang :** string. set the programming language
  - **'c' :** for C language
  - **'f' :** for Fortran language
- **flag :** string.compiler flag
  - **'GCC' :** return gcc command line
  - **'LCC' :** return lcc-win32 command line
  - **'VC' :** return vc command line
- **cmd :** the string of the of compiler command line.

2.8.3 File content

```
//find_ccmd
//fonction qui trouve la ligne de commande
//du compilateur en fonction du compilateur choisit
//entrée lang : language 'c' ou 'f'
//sortie flag : flag compilateur (GCC,LCC,VC)
//       cmd  : commande du compilateur
function [flag,cmd]=find_cmd(lang)

 if lang=="c" then
   printf("... Search a C compiler ...\n");
 elseif lang=="f" then
   printf("... Search a Fortran compiler ...\n");
 end

 GCC_FLAG=%f;
 LCC_FLAG=%f;
 VC_FLAG=%f;
 G77_FLAG=%f;

 if MSDOS then //windob
  printf(" Windows platform ?\n");
  if lang=='c' then
   //trouve LCC
//   pause
   if exists('LCC') then
     //trouve LCC
     if LCC==%F then
       stat=unix('cl');
       if stat==0 then
         printf(" Found Microsoft Visual C/C++ Compiler\n %s\n",'Version ?');
         VC_FLAG=%t;
       end
     else
       LCC_FLAG=%t;
       stat=unix(SCI+'\lcc\bin\lcc -v');
       if stat==0 then
         txt=unix_g(SCI+'\lcc\bin\lcc -v');
         printf(" Found lcc-win32\n %s\n",txt(1));
```

```
        LCC_FLAG=%t;
      else
        printf("Your LCC flag is set TRUE, but can''t find lcc compiler");
      end
    end
  else
   stat=unix('cl');
   if stat==0 then
      printf(" Found Microsoft Visual C/C++ Compiler\n %s\n",'Version ?');
      VC_FLAG=%t;
   end
  end
 elseif lang=='f' then
 end
else //unix/linux
 printf(" Posix platform ?\n");
 if lang=='c' then
  stat=unix('gcc --version');
  if stat==0 then
   txt=unix_g('gcc --version');
   printf(" Found gcc\n %s\n",txt(1));
   GCC_FLAG=%t;
  end
 elseif lang=='f' then
  //myvar_tt="stat=unix(''g77 --version'');";
  stat=unix('g77 --version');
  if stat==0 then
   txt=unix_g("g77 --version");
   printf(" Found g77\n %s\n",txt(1));
   G77_FLAG=%t;
  end
 end
end

if GCC_FLAG then
 cmd="gcc ";
 flag="GCC";
elseif G77_FLAG then
 cmd="g77 "
 flag="G77";
elseif LCC_FLAG then
 cmd=pathconvert(SCI,%f,%t,'w')+"\lcc\bin\lcc ";
 flag="LCC";
elseif VC_FLAG then
 cmd="cl ";
 flag="VC";
else
 printf(" Can''t find Compiler\n");
 cmd='trash';
 flag='trash';
end
endfunction
```

2.9 Search path which must be include for building routines of the toolbox

- **Name:** find_incl_path
- **Library:** build_util - Library of utilities functions to build the toolbox

2.9.1 Calling Sequence

```
tt = find_incl_path
```

2.9.2 Parameters

- **tt :** vector of character strings containing the included path at the compilation.

2.9.3 File content

```
//find_incl_path
//besoin machine.h scicos_block.h f2c.h
//       mex.h stack-c.h stack-def.h
//sortie tt est une liste de chaine de caractère contenant
//          les répertoires à inclure lors de la compilation
function tt=find_incl_path()
 printf("... Search Headers of Scilab ...\n");
 //define headers to find (to search!)
 routines_dir=[SCI+"/routines/"];
 routines_head=["machine.h";"mex.h";"stack-c.h";"stack-def.h"];
 routines_flag=[%t;%t;%t;%t];

 scicos_dir=[SCI+"/routines/scicos/"];
 scicos_head=["scicos_block.h"];
 scicos_flag=[%t];

 f2c_dir=[SCI+"/routines/f2c/"];
 f2c_head=["f2c.h"];
```

```
 f2c_flag=[%t];

 dir_flag=[%t;%t;%t]; //3 répertoires

 //define provided headers directory
 prov_dir=MODNUM+"/routines/sci_headers/";

 //incl_list : (1) liste des répertoires de taille n
 //            (2) flags correspondant aux répertoires
 //            (2*i+1..2*i+1+n) fichiers d'en-têtes correspondant
 //            (2*i+2..2*i+2+n) flags correspondant aux fichiers
 incl_list=list([routines_dir;scicos_dir;f2c_dir],..
                dir_flag,..
                routines_head,routines_flag,..
                scicos_head,scicos_flag,f2c_head,f2c_flag);

 if MSDOS then
  incl_list(1)=pathconvert(incl_list(1),%f,%t,'w');
  for i=1:size(incl_list(1),'*')
    incl_list(1)(i)=part(incl_list(1)(i),1:length(incl_list(1)(i))-1);
  end
  prov_dir=pathconvert(prov_dir,%f,%t,'w');
  //doit-être vérifier
  //incl_list(1)="""+pathconvert(incl_list(1),%f,%t,'w')+""";
  //prov_dir="""+pathconvert(prov_dir,%f,%t,'w')+""";
 end

 prov_head_must_be_includ = %f;

 for i=1:size(incl_list(1),'*')
  if fileinfo(incl_list(1)(i))==[] then
   incl_list(2)(i)=%f;
   printf(" Directory %s isn''t found\n",incl_list(1)(i));
   prov_head_must_be_includ = %t;
  else
   num_head=size(incl_list(2*i+1),'*');
   if MSDOS then
     sep="\";
   else
     sep="";
   end
   for j=1:num_head
    if fileinfo(incl_list(1)(i)+sep+incl_list(2*i+1)(j))==[] then
       incl_list(2*i+2)(j)=%f;
       printf(" File %s isn''t found\n",incl_list(2*i+1)(j));
       prov_head_must_be_includ = %t;
    end
   end
   flag_dir=%f;
   for j=1:num_head
    flag_dir=flag_dir | incl_list(2*i+2)(j);
   end
   if ~flag_dir then incl_list(2)(i)=%f; end
  end
 end
 if prov_head_must_be_includ then
  printf(" Use provided header\n");
 else
  printf(" Scilab Source Version found\n");
 end;

 tt=[];k=1;
 for i=1:size(incl_list(1),'*')
  if incl_list(2)(i) then
   tt(k)=incl_list(1)(i);
   k=k+1;
  end
 end
 if prov_head_must_be_includ tt(k)=prov_dir; end;
endfunction
```

2.10 Retrieve major and minor version of scilab

- **Name:** find_scilab_ver
- **Library:** build_util - Library of utilities functions to build the toolbox

2.10.1 Calling Sequence

```
ok = find_scilab_ver
```

2.10.2 Parameters

- **ok :** boolean

2.10.3 File content

```
//find_scilab_ver
//To be done
function ok=find_scilab_ver()
 printf("... Search Scilab version ...\n");
 printf("Version of scilab is : %s\n",getversion());
 //win : scilab-3.0
 ok=%t;
endfunction
```

2.11 Generate text of demonstrations of the toolbox

- **Name:** generate_dem_scicos
- **Library:** build_util - Library of utilities functions to build the toolbox

2.11.1 Calling Sequence

```
txt = generate_dem_scicos(g,listt)
```

2.11.2 Parameters

- **g :** integer. id of the scicos diagram
- **listt :** matrix of strings. (’diagr_list’ or ’sim_list’)
- **txt :** vector of strings. text of the demo to execute

2.11.3 File content

```
//fonction qui genère le texte du fichier
//demo concernant les diagrammes scicos
//rmq : n’est pas stocké dans un fichier
//
//g le numéro dans la liste
//listt : la matrice de chaine de caratère de la liste
//        (diagr_list ou sim_list)
function txt=generate_dem_scicos(g,listt)
    if ~exists(’xml_path’) then
       //retrieve default LANGUAGE
       if ~exists(’LANGUAGE’) then
          global LANGUAGE;lang=LANGUAGE;clear LANGUAGE;
       else
          lang=LANGUAGE;
       end
       if lang<>’eng’ & lang<>’fr’ then
         printf("language %s is not supported, switch to ’’eng’’\n",lang);
         lang=’eng’
       end
       xml_path=MODNUM+’/man/xml/’+lang+’/’;
    end

    if listt==[]|listt=="" then txt=[]; return, end;
    if g==0|g>size(listt,1) then txt=[]; return, end;
    ierror=execstr(’myvar=evstr(listt(g,1))’,’errcatch’);
    if ierror<>0 then txt=[]; return, end;
    if myvar==[] then txt=[]; return, end;

    demos_name=[];
    path_demos=[];
    tt=[];
    tt1=[];
    tt2=[];
    if lang==’fr’ then
       tt_title=’Choisissez une démo’
    else
       tt_title=’Choose a demo’
    end
    for i=1:size(myvar,1)
       demos_name(i)=return_xml_sdesc(xml_path+myvar(i,2)+’.xml’);
       stri=strindex(myvar(i,1),MODNUM);
       //pause
       //if stri<>[] then
       //  path_demos(i)=’MODNUM+’+’’’’+part(myvar(i,1),stri+length(MODNUM):length(myvar(i,1)));
       //end
       //pause
       path_demos(i)=myvar(i,1);
       tt1=[tt1;’’’’+strsubst(demos_name(i),’’’’,’’’’’’)+’’’’];
       tt2=[tt2;’case ’’’+strsubst(demos_name(i),’’’’,’’’’’’)+’’’ then’;
            ’  scicos(’’’+path_demos(i)+myvar(i,2)+’.cos’’);’];
    end
    tt1(size(tt1,1))=tt1(size(tt1,1))+’]’;
    tt1=[’mode(-1)’;’demos_name=[’;tt1];
```

```
    tt2=['if ~exists(''fun'') then'
         '  fun=x_choose'
         '  if exists(''with_tk'') then'
         '    if whereis(''tk_choose'')<>[] then'
         '      if with_tk() then fun=tk_choose, end;'
         '    end'
         '  end'
         'end'
         'while %t do'
         'n=fun(demos_name,'''+tt_title+''')'
         'if n ==0 then break,end'
         'select demos_name(n)'
         tt2
         'end';'end'];
    tt=[tt1;tt2];
    txt=tt;
endfunction
```

2.12 Generate text of scilab simulation script demonstrations of the toolbox

- **Name:** generate_dem_sim
- **Library:** build_util - Library of utilities functions to build the toolbox

2.12.1 Calling Sequence

```
txt = generate_dem_sim(g,listt)
```

2.12.2 Parameters

- **g :** integer. id of the scimulation script
- **listt :** matrix of strings. ('diagr_list' or 'sim_list')
- **txt :** vector of strings. text of the demo to execute

2.12.3 File content

```
//fonction qui genère le texte du fichier
//demo concernant les scripts de simulation
//rmq : n'est pas stocké dans un fichier
//
//g le numéro dans la liste
//listt : la matrice de chaine de caratère de la liste
//        (diagr_list ou sim_list)
function txt=generate_dem_sim(g,listt)

    if ~exists('xml_path') then
       //retrieve default LANGUAGE
       if ~exists('LANGUAGE') then
          global LANGUAGE;lang=LANGUAGE;clear LANGUAGE;
       else
          lang=LANGUAGE;
       end
       if lang<>'eng' & lang<>'fr' then
         printf("language %s is not supported, switch to ''eng''\n",lang);
         lang='eng'
       end
       xml_path=MODNUM+'/man/xml/'+lang+'/';
    end

    if listt==[]|listt=="" then txt=[]; return, end;
    if g==0|g>size(listt,1) then txt=[]; return, end;
    ierror=execstr('myvar=evstr(listt(g,1))','errcatch');
    if ierror<>0 then txt=[]; return, end;
    if myvar==[] then txt=[]; return, end;

    demos_name=[];
    path_demos=[];
    tt=[];
    tt1=[];
    tt2=[];
    if lang=='fr' then
       tt_title='Choisissez une démo'
    else
       tt_title='Choose a demo'
    end
    for i=1:size(myvar,1)
       demos_name(i)=return_xml_sdesc(xml_path+myvar(i,2)+'.xml');
       stri=strindex(myvar(i,1),MODNUM);
       //if stri<>[] then
       //  path_demos(i)='MODNUM+'+'''+part(myvar(i,1),stri+length(MODNUM):length(myvar(i,1)));
       //end
       path_demos(i)=myvar(i,1);
```

```
      tt1=[tt1;''''+strsubst(demos_name(i),'''','''''')+''''];
      tt2=[tt2;'case '''+strsubst(demos_name(i),'''','''''')+''' then';
           '  scipad('''+path_demos(i)+myvar(i,2)+'/'+myvar(i,2)+'.sce'');'];
    end
    tt1(size(tt1,1))=tt1(size(tt1,1))+']';
    tt1=['mode(-1)';'demos_name=[';tt1];
    tt2=['if ~exists(''fun'') then'
         '  fun=x_choose'
         '  if exists(''with_tk'') then'
         '    if whereis(''tk_choose'')<>[] then'
         '      if with_tk() then fun=tk_choose, end;'
         '    end'
         '  end'
         'end'
         'while %t do'
         'n=fun(demos_name,'''+tt_title+''')'
         'if n ==0 then break,end'
         'select demos_name(n)'
         tt2
         'end';'end'];
    tt=[tt1;tt2];
    txt=tt;
endfunction
```

2.13 Create demonstration files of the toolbox

- **Name:** generate_modnum_dem
- **Library:** build_util - Library of utilities functions to build the toolbox

2.13.1 Calling Sequence

`tt = generate_modnum_dem`

2.13.2 Parameters

- **tt :** vector of strings. the output text of main modnum demo file

2.13.3 File content

```
//fonction qui genère les fichiers demo
function tt=generate_modnum_dem()
  if ~exists('demos_path') then
    demos_path=MODNUM+'/man/demos/';
  end

  txt=generate_sim_dem();
  mputl(txt,demos_path+'sim.dem');
  txt=generate_scs_diagr_dem();
  mputl(txt,demos_path+'scs_diagr.dem');

  tt=['if ~exists(''LANGUAGE'') then'
      '   global LANGUAGE;lang=LANGUAGE;clear LANGUAGE;'
      'else'
      '   lang=LANGUAGE;'
      'end'
      ''
      'if lang==''fr'' then'
      '    l_i=2;'
      'else'
      '    l_i=1;'
      'end'
      ''
      'tt_title=[''Scilab simulation scripts'',''Scripts Scilab de simulations'''
      '          ''Scicos Diagrams'',''Diagrammes Scicos'''
      '          ''Choose a demo'',''Choisissez une démo'']'
      'demolist=['
      'tt_title(1,l_i),''sim.dem'';'
      'tt_title(2,l_i),''scs_diagr.dem''];'
      ''
      'demos_path=MODNUM+''/man/demos/''';
      ''
      'fun=x_choose'
      'if exists(''with_tk'') then'
      ' if whereis(''tk_choose'')<>[] then'
      '   if with_tk() then fun=tk_choose, end;'
      ' end'
      'end'
      ''
      'while %t then';
      'num0=fun(demolist(:,1),tt_title(3,l_i));'
      '   if num0==0 then ';
      '      return';
      '   else';
      '      exec(demos_path+demolist(num0,2),-1)';
      '   end';
      'end';
```

```
    ];

  mputl(tt,demos_path+'mod_num.dem');

endfunction
```

2.14 Create palettes of scicos blocks

- **Name:** generate_palette
- **Library:** build_util - Library of utilities functions to build the toolbox

2.14.1 Calling Sequence

```
txt = generate_palette(lisf,path,nameP)
```

2.14.2 Parameters

- **lisf :** vector of strings. list of interfacing function name
- **path :** string. target directory
- **nameP :** string. name of palette file
- **txt :** vector of strings. output text of the palette file

2.14.3 File content

```
////////////////////
//generate_palette
//Version 1 - vendredi 10 sept 2004
//
//Crée un fichier cosf à partir d'une
//liste de fonctions d'interface scicos
//
//lsh
//txt chaine de caractère du fichier cosf
//
//rsh
//lisf  : liste de noms de fonction
//nameP : nom de la palette (sans extension)
//
//Lundi 4 avril 2005
//Rajout du chemin où ecrire le fichier cosf : path


function txt=generate_palette(lisf,path,nameP)
 //Affiche un message
 printf("Generate "+nameP+" palette\n");

 //Déclaration variable locale
 sp_x=30     //espace entre chaque block
 sp_y=22     //espace entre chaque ligne
 nb_r=5     //nombre de block par ligne
 lmax=80;    //longeur max d'une ligne txt du fichier
 j=0;        //compteur colonne
 blk_x=0;    //coordonnée x du block
 blk_y=sp_y; //coordonnée y du block
 blk_y1=0;   //memo de la largeur max d'un block sur une ligne

 //Charge librairies Scicos
 load SCI/macros/scicos/lib
 exec(loadpallibs,-1)

 //charge une structure vide
 scs_m=scicos_diagram()

 //Nomme la fenêtre
 scs_m.props("title")=nameP

 //Ecrit en-tête du fichier cosf
 t=['scicos_ver='"scicos2.7.3'"'
    'scs_m=scicos_diagram()']
 t1=sci2exp(scs_m.props,lmax);
 txt=[t;'scs_m.props='+t1(1);t1(2:$)]

 //Pour chaque fonction
 for l=1:size(lisf,1)
   //execute cas define du bloc l
   ierror=execstr('blk='+lisf(l,1)+'(''define'')','errcatch')

   //redimensionne block
   blk.graphics("sz")(1)=blk.graphics("sz")(1)*20
```

```
   blk.graphics("sz")(2)=blk.graphics("sz")(2)*20

   //Mémorise le bloc le plus haut
   if(blk.graphics("sz")(2)>blk_y1) then
    blk_y1=blk.graphics("sz")(2)
   end

   //incremente compteur colonne
   j=j+1;
   //Test début de colonne
   if j==1 then
    blk_x=sp_x //position x du bloc colonne 1
   end
   //Position du block dans la palette
   blk.graphics("orig")=[blk_x blk_y]

   //Incrémente la position x du block
   blk_x=blk_x+blk.graphics("sz")(1)+sp_x

   //Test fin de colonne
   if j==nb_r
      j=0                    //RAZ compteur colonne
      blk_y=blk_y+blk_y1+sp_y //Incremente position y
      blk_y1=0               //RAZ longueur bloc le plus haut
   end

   //Ecrit scs_m.objs(l)
   lhs='scs_m.objs('+string(l)+')='
   t1=sci2exp(blk,lmax-length(lhs))
   n1=size(t1,1)
   bl1=' ';bl1=part(bl1,1:length(lhs))
   txt=[txt;lhs+t1(1);bl1(ones(n1-1,1))+t1(2:$)]
 end
 //Enregistre txt dans nameP.cosf
 mputl(txt,path+nameP+'.cosf');
endfunction
```

2.15 Generate the text of modnum scicos diagram demonstration

- **Name:** generate_scs_diagr_dem
- **Library:** build_util - Library of utilities functions to build the toolbox

2.15.1 Calling Sequence

```
txt = generate_scs_diagr_dem
```

2.15.2 Parameters

- **txt :** vector of strings. scilab instructions to execute

2.15.3 File content

```
//txt : le texte à executer pour ouvrir
//      le menu de demo des diagrammes scicos
function txt=generate_scs_diagr_dem()

demos_list=['diagr_cs','tt_title(1,l_i)';
            'diagr_ds','tt_title(2,l_i)';
            'diagr_os','tt_title(3,l_i)';
            'diagr_is','tt_title(4,l_i)';
            'diagr_fs','tt_title(5,l_i)';
            'diagr_PSK','tt_title(6,l_i)';
            'diagr_SD','tt_title(7,l_i)';
            'diagr_FSK','tt_title(8,l_i)';
            'diagr_FSK_chaos','tt_title(9,l_i)';
            'diagr_elec','tt_title(10,l_i)';];

  list_demos=list(list());
  for i=1:size(demos_list,1)
     new_tt=evstr(demos_list(i));
     list_demos(i)="";
     if new_tt<>[] & new_tt<>"" then
       new_tt=['''+strsubst(new_tt,'''','''''')+'''];
       for j=1:size(new_tt,1)
         stri=strindex(new_tt(j,1),MODNUM);
         if stri<>[] then
           new_tt(j,1)='MODNUM+'+'''+part(new_tt(j,1),stri+length(MODNUM):length(new_tt(j,1)));
         end
       end
       for j=1:size(new_tt,1)
         if j==1 then
           list_demos(i)=[new_tt(j,1)+','+new_tt(j,2)+';'];
         else
           list_demos(i)=[list_demos(i);new_tt(j,1)+','+new_tt(j,2)+';'];
         end
       end
```

```
    end
    list_demos(i)=[demos_list(i,1)+'=[';list_demos(i);];
    list_demos(i)(size(list_demos(i),1))=list_demos(i)(size(list_demos(i),1))+'];';
  end

  var_tt=['''+strsubst(demos_list(:,1),'''',''''''')+''''+...
          ','+strsubst(demos_list(:,2),'''',''''''')+';'];
  var_tt=['demos_list=[';var_tt];
  var_tt(size(var_tt,1))=var_tt(size(var_tt,1))+'];';
  for i=1:size(demos_list,1)
   var_tt=[var_tt;list_demos(i)];
  end

  var_tt=['if ~exists(''LANGUAGE'') then'
  '   global LANGUAGE;lang=LANGUAGE;clear LANGUAGE;'
  'else'
  '   lang=LANGUAGE;'
  'end'
  ''
  'if lang==''fr'' then'
  '    l_i=2;'
  'else'
  '    l_i=1;'
  'end'
  ''
  'tt_title=[''Chaotic Continous time systems'',''Systèmes chaotiques à temps continu'''
  '          ''Chaotic Discrete time systems'',''Systèmes chaotiques à temps discret'''
  '          ''Open loop models of oscillator'',''Modèles boucle ouverte d''''oscillateur'''
  '          ''Integer N Frequency synthesizers'',''Synthétiseurs de fréquence à rapport de division N entier'''
  '          ''Fractional N/N+1 Frequency synthesizers'',''Synthétiseurs de fréquence à rapport de division fractionnaire'''
  '          ''PSK/QAM Transmission'',''Transmission PSK/QAM'''
  '          ''Delta-Sigma Transmission'',''Transmission Sigma-Delta'''
  '          ''FSK Transmission'',''Transmission FSK'''
  '          ''Chaotic FSK Transmission'',''Transmission chaotique FSK'''
  '          ''Electrical circuits'',''Circuits électriques'''
  '          ''Choose a demo'',''Choisissez une démo''];'
  ''
  var_tt]

  sup_tt=['if ~exists(''fun'') then'
          '  fun=x_choose'
          ' if whereis(''tk_choose'')<>[] then'
          '  if exists(''with_tk'') then'
          '    if with_tk() then fun=tk_choose, end;'
          '  end'
          ' end'
          'end'
          'while %t then'
          '   num=fun(demos_list(:,2),tt_title(11,l_i));'
          '   if num==0 then'
          '      return'
          '   else';
          '      txt=generate_dem_scicos(num,demos_list);'
          '      execstr(txt)';
          '   end;
          'end';
          ];
  txt=[var_tt;sup_tt;]
endfunction
```

2.16 Generate the text of scilab simulation script demonstration

- **Name:** generate_sim_dem
- **Library:** build_util - Library of utilities functions to build the toolbox

2.16.1 Calling Sequence

```
txt = generate_sim_dem
```

2.16.2 Parameters

- **txt :** vector of strings. scilab instructions to execute

2.16.3 File content

```
//txt : le texte à executer pour ouvrir
//      le menu de demo des scripts de simulations
function txt=generate_sim_dem()

demos_list=['sim_chaos','tt_title(1,l_i)';
            'sim_synthe','tt_title(2,l_i)';
            'sim_PSK','tt_title(3,l_i)';];

  list_demos=list(list());
```

```
  for i=1:size(demos_list,1)
     new_tt=evstr(demos_list(i));
     list_demos(i)="";
     if new_tt<>[] & new_tt<>"" then
       new_tt=[''''+strsubst(new_tt,'''',''''''')+''''];
       for j=1:size(new_tt,1)
         stri=strindex(new_tt(j,1),MODNUM);
         if stri<>[] then
           new_tt(j,1)='MODNUM+'+''''+part(new_tt(j,1),stri+length(MODNUM):length(new_tt(j,1)));
         end
       end
       for j=1:size(new_tt,1)
         if j==1 then
           list_demos(i)=[new_tt(j,1)+',+new_tt(j,2)+';'];
         else
           list_demos(i)=[list_demos(i);new_tt(j,1)+',+new_tt(j,2)+';'];
         end
       end
     end
     list_demos(i)=[demos_list(i,1)+'=[';list_demos(i);];
     list_demos(i)(size(list_demos(i),1))=list_demos(i)(size(list_demos(i),1))+'];';
  end

  var_tt=[''''+strsubst(demos_list(:,1),'''',''''''')+''''+...
          ',+strsubst(demos_list(:,2),'''',''''''')+';'];
  var_tt=['demos_list=[';var_tt];
  var_tt(size(var_tt,1))=var_tt(size(var_tt,1))+'];';
  for i=1:size(demos_list,1)
   var_tt=[var_tt;list_demos(i)];
  end

  var_tt=['if ~exists(''LANGUAGE'') then'
  '   global LANGUAGE;lang=LANGUAGE;clear LANGUAGE;'
  'else'
  '   lang=LANGUAGE;'
  'end'
  ''
  'if lang==''fr'' then'
  '    l_i=2;'
  'else'
  '    l_i=1;'
  'end'
  ''
  'tt_title=[''Simulations of chaotic systems'',''Simulations de systèmes chaotiques'''
  '          ''Simulations of oscillators & Phase Locked Loop'',''Simulations d''''oscillateurs et de boucles à verrouillage de phase'''
  '          ''Simulations of communication systems'',''Simulations de systèmes de communication'''
  '          ''Choose a demo'',''Choisissez une démo'']'
  ''
  var_tt]

  sup_tt=['if ~exists(''fun'') then'
          '  fun=x_choose'
          ' if whereis(''tk_choose'')<>[] then'
          '   if exists(''with_tk'') then'
          '     if with_tk() then fun=tk_choose, end;'
          '   end'
          '  end'
          ' end'
          'while %t then'
          '   num=fun(demos_list(:,2),tt_title(4,l_i));'
          '   if num==0 then'
          '      return'
          '   else';
          '      txt=generate_dem_sim(num,demos_list);'
          '      execstr(txt)';
          '   end';
          'end';
          ];
  //pause
  txt=[var_tt;sup_tt;]
endfunction
```

2.17 Compile routine files of the toolbox

- **Name:** precompilation
- **Library:** build_util - Library of utilities functions to build the toolbox

2.17.1 Calling Sequence

`precompilation(flag,cmd,incl_path,path,listf)`

2.17.2 Parameters

- **flag :** string. compiler flag
  - **'GCC' :** gcc compiler

  - **'LCC' :** lcc-win32 compiler
  - **'VC' :** visual c/c++ compiler

- **string. cmd :** string. command-line of the compiler
- **incl_path :** vector of strings. the include paths for the compiler
- **path :** string. directory of the targetted object files
- **listf :** vector of strings. name of object file to compile.

2.17.3 File content

```
//precompilation
//Entrée flag : flag compilateur (GCC,LCC,VC)
//       cmd  : commande du compilateur
//       incl_path :  répertoires à inclure lors de la compilation
//       path  : le chemin du répertoire de compilation
//       listf : liste des fichiers à compiler sans extension (ex:monmodule)
function []=precompilation(flag,cmd,incl_path,path,listf)
  printf(" Make first compilation...\n");
  select flag
  case 'GCC' then
    INCL_PATH=incl_path;
    CCFLAG=[];
    for i=1:size(INCL_PATH,'*')
      CCFLAG=CCFLAG+"-I"+INCL_PATH(i)+" ";
    end
    CCFLAG=CCFLAG+"-c ";
    cmd=cmd+CCFLAG;
    listf=listf+'.c';

  case 'G77' then
    FCFLAG=[];
    FCFLAG=FCFLAG+"-c ";
    cmd=cmd+FCFLAG;
    listf=listf+'.f';

  case 'LCC' then
    INCL_PATH=incl_path;
    CCFLAG=[];
    for i=1:size(INCL_PATH,'*')
      CCFLAG=CCFLAG+"-I"+INCL_PATH(i)+" ";
    end
    CCFLAG=CCFLAG+"-c ";
    cmd=cmd+CCFLAG;
    listf=listf+'.c';

  case 'VC' then
    INCL_PATH=incl_path;
    CCFLAG=[];
    for i=1:size(INCL_PATH,'*')
      CCFLAG=CCFLAG+"/I "+INCL_PATH(i)+" ";
    end
    CCFLAG=CCFLAG+"/c ";
    cmd=cmd+CCFLAG;
    listf=listf+'.c';

  case 'trash' then
   printf("Compilation aborted\n");
   return;
 end

 //change directory
 rep=pwd();
 chdir(MODNUM+path);
 //compilation
 for i=1:size(listf,'*')
  unix_g(cmd+listf(i))
 end
 //change directory
 chdir(rep);
endfunction
```

2.18 Purge directories (make distclean) of the toolbox

- **Name:** purge_modnum
- **Library:** build_util - Library of utilities functions to build the toolbox

2.18.1 Calling Sequence

`purge_modnum(flag)`

2.18.2 Parameters

- **flag :** string. set the type of the purge
  - **'clean' :** erase object file (*.o, *.obj)
  - **'distclean' :** erase object file (*.o, *.obj), libraries(*.dll *.so *.lib ,...), palettes (*.cosf),...

2.18.3 File content

```
//purge_modnum
//Entrée flag 'clean' pour enlever les fichiers objets (.obj,.o)
//            'distclean' pour enlever
//                        les fichiers objets (.obj .o)
//                        les fichiers binaire .bin
//                        les librairies (name lib .so .dll .lib .def)
//                        les palettes (*.cosf)
//WARNING : i don't find a way to delete dll in windob
//because protected
function []=purge_modnum(flag)
 //set here directories to be explored (in MODNUM)
 macros_dir = '/macros/'+...
             ['scicos_util';
              'signal';
              'misc';
              'generate_doc';
              'gen_doc_util';
              'find_file';
              'xmltotek'
              'build_util';
              'scicos_blocks/Communication';
              'scicos_blocks/NonLinear';
              'scicos_blocks/Pll';
              'scicos_blocks/Skins';
              'scicos_blocks/Sources';
              'scicos_blocks/Tools'];

 routines_dir = '/routines/'+...
               ['mod_num_lib';
                'scicos'];

 pal_dir = ['/macros/scicos_blocks']

 if MSDOS then
  macros_dir=pathconvert(macros_dir,%f,%t,'w');
  routines_dir=pathconvert(routines_dir,%f,%t,'w');
  pal_dir=pathconvert(pal_dir,%f,%t,'w');
  rm_cmd='del /F ';
  obj_ext='*.obj';
  lib_ext='*.dll *.exp *.lib *.def';
 else
  rm_cmd="rm -f ";
  obj_ext='*.o'
  lib_ext='*.so'
 end

 select flag
  case 'clean' then
    cur_rep=pwd();
    for i=1:size(routines_dir,'*')
      chdir(MODNUM+routines_dir(i));
      tt='unix_g(rm_cmd+obj_ext)';
      ierr=execstr(tt,'errcatch');
    end
    chdir(cur_rep);

  case 'distclean' then
    cur_rep=pwd();
    for i=1:size(macros_dir,'*')
     chdir(MODNUM+macros_dir(i));
     tt='unix_g(rm_cmd+''*.bin lib names'')';
     ierr=execstr(tt,'errcatch');
    end
    for i=1:size(routines_dir,'*')
     chdir(MODNUM+routines_dir(i));
     tt='unix_g(rm_cmd+obj_ext)';
     ierr=execstr(tt,'errcatch');
     tt='unix_g(rm_cmd+lib_ext)';
     ierr=execstr(tt,'errcatch');
    end
    for i=1:size(pal_dir,'*')
     chdir(MODNUM+pal_dir(i));
     tt='unix_g(rm_cmd+''*.cosf'')';
     ierr=execstr(tt,'errcatch');
    end
    chdir(MODNUM);
    tt='unix_g(rm_cmd+''loader.sce'')';
    ierr=execstr(tt,'errcatch');
    chdir(cur_rep);

  else printf("Invalid flag\n");
 end
endfunction
```

2.19 Return directories presents in a path

- **Name:** return_dirs
- **Library:** build_util - Library of utilities functions to build the toolbox

2.19.1 Calling Sequence

```
tt = return_dirs(path)
```

2.19.2 Parameters

- **path :** string. the directory to analyse.
- **tt :** vector of strings. the name of the directories

2.19.3 File content

```
//return_dirs
//entrée : path chemin du repertoire racine
//sortie : tt nom des répertoires contenu dans le repertoire racine
function tt=return_dirs(path)
 files=listfiles(path);
 k=1;
 tt=[];
 for i=1:size(files,'*')
  if isdir(path+files(i)) then
   tt(k)=files(i);
   k=k+1;
  end
 end
endfunction
```

2.20 Return addinter line for builder script of the toolbox

- **Name:** write_addinter_line
- **Library:** build_util - Library of utilities functions to build the toolbox

2.20.1 Calling Sequence

```
txt = write_addinter_line(u,libname,path,intname,scifunc)
```

2.20.2 Parameters

- **u :** integer. a file descriptor
- **libname :** string. the name of the library
- **path :** string. the path of the library
- **intname :** string. the name of the interfacing routine
- **scifunc :** vector of strings. the names of the scilab functions
- **txt :** vector of strings. the text of the addinter line

2.20.3 File content

```
//write_addinter_line
//Entrée : u : file descriptor
//         libname : nom de la libraire (sans extension)
//         path : chemin de la libraire dans MODNUM ex /routines/mod_num_lib/
//         intname : nom de la routine d'interface
//         scifunc : liste des noms de fonctions scilab
function txt=write_addinter_line(u,libname,path,intname,scifunc)
 if MSDOS then
  mylibname=libname+'.dll';
 else
  mylibname=libname+'.so';
 end

 tt_loader=['//Link of interfaced modnum functions';]
 myvar_tt="modnum_sci_func=[";
 for i=1:size(scifunc,'*')
  myvar_tt=myvar_tt+"""+scifunc(i)+""";";
 end
 myvar_tt=myvar_tt+"];";
 tt_loader=[tt_loader;myvar_tt;
            'addinter(MODNUM+"""+path+"/"+mylibname+""","""+intname+""",modnum_sci_func);';''];
 //execstr(tt_loader(2:3));
 txt=tt_loader(2:3);
 fprintf(u,"%s\n",tt_loader);
endfunction
```

2.21 Return text of header of loader script for builder script of the toolbox

- **Name:** write_header
- **Library:** build_util - Library of utilities functions to build the toolbox

2.21.1 Calling Sequence

`write_header(u,builder_name)`

2.21.2 Parameters

- **u :** integer. a file descriptor
- **builder_name :** string. the name of the builder script

2.21.3 File content

```
//write_header(u,builder_name)
//Entrée : u file descriptor
//         builder_name : name of builder
//Sortie : néant
function []=write_header(u,builder_name)
 tt_loader=['//Loader Script of mod_num for scilab 3.0'
            '//Generated by '+builder_name
            '//'+date()+' Ircom Group A.Layec';'';
            '//Redefine stacksize'
            'stacksize(30000000);';'';
            '//Define mod_num root path']
 if MSDOS then
   tt_loader=[tt_loader;'MODNUM=get_absolute_file_path(''loader.sce'');';
                        'if part(MODNUM,length(MODNUM))==''\'' then';
                        '   MODNUM=part(MODNUM,1:length(MODNUM)-1);';
                        'end']
   //tt_loader=[tt_loader;'MODNUM=;'+MODNUM;]
   //end
 else
   tt_loader=[tt_loader;'MODNUM='+"""+MODNUM+""";'';]
 end
 fprintf(u,"%s\n",tt_loader);
endfunction
```

2.22 Return text of documentation for builder script of the toolbox

- **Name:** write_inf_doc
- **Library:** build_util - Library of utilities functions to build the toolbox

2.22.1 Calling Sequence

```
txt = write_inf_doc(u)
```

2.22.2 Parameters

- **u :** integer. a file descriptor
- **txt :** vector of strings. the text of the information to load modnum documentation

2.22.3 File content

```
//write_inf_doc
//Entrée : u : file descriptor
//Sortie : txt : Information utile de chargement
//         flag :
function txt=write_inf_doc(u)

  txt=[];
  tt='myhelps=[MODNUM+''/man/htm/''+lang,""Modnum communication toolbox""];';
  tt2='add_demo(''Mod_num'',MODNUM+''/man/demos/mod_num.dem'');'
  if MSDOS then
    tt=pathconvert(tt,%f,%t,'w')
    tt2=pathconvert(tt2,%f,%t,'w')
  end

  tt_loader=['//find the LANGUAGE'
             'if ~exists(''LANGUAGE'') then'
             ' global LANGUAGE;lang=LANGUAGE;clear LANGUAGE;'
             'else'
             ' lang=LANGUAGE;'
             'end'
             'if lang<>''fr''&lang<>''eng'' then'
             '  printf(""Documentation: Unsupported language %s, switch to eng.\n"",lang);'
             '  lang=''eng'';'
             'end'
             '//Add mod_num help chapter'
             tt
             '%helps=[%helps;myhelps(1,:)];'
             '//Add mod_num demo'
             tt2
             'clear myhelps;clear lang'
             ]
  txt=[tt_loader(2:10);tt_loader(12:13);tt_loader(15:16)];
  if exists('u') then
   fprintf(u,"%s\n",tt_loader);
  end

endfunction
```

2.23 Return text of libraries for builder script of the toolbox

- **Name:** write_inf_lib
- **Library:** build_util - Library of utilities functions to build the toolbox

2.23.1 Calling Sequence

```
txt = write_inf_lib(u,path,tt,flag)
```

2.23.2 Parameters

- **u :** integer. a file descriptor
- **path :** string. the path of the target macro library
- **tt :** string. name of the library
- **flag :** integer. a flag to set text header
  - **0 :** no header
  - **1 :** header
- **txt :** vector of strings. the text of the information to load modnum library

2.23.3 File content

```
//write_inf_lib
//Entrée : u file descriptor
//         path chemin de la libraire dans MODNUM (ex: macros/util/)
//         tt nom de la librairie (ex:mod_num_util)
//         flag drapeau pour affichage d'un header dans loader.sce
//              (0 : pas de header; 1 : header)
//sortie : txt : Information utile de chargement
//               ex : mod_num_scicos_utils=lib(MODNUM+'/macros/scicos_util/');
function txt=write_inf_lib(u,path,tt,flag)
 if size(path,'*')==size(tt,'*') then
   if MSDOS then
    path=pathconvert(path,%f,%t,'w')+'\';
   else
    path=path+'/';
   end
   for i=1:size(path,'*')
    if flag then
      tt_loader=['//Load '+tt(i)+' library'
                 tt(i)+'=lib(MODNUM+'''+path(i)+''');';'';
                ]
      txt(i)=tt_loader(2);
    else
     tt_loader=tt(i)+'=lib(MODNUM+'''+path(i)+''');'
     txt(i)=tt_loader;
    end
    fprintf(u,"%s\n",tt_loader);
   end
 else
  //Affiche un message d'erreur
  printf("path and tt must have the same size");
  abort
 end
endfunction
```

2.24 Return text of palettes for builder script of the toolbox

- **Name:** write_inf_pal
- **Library:** build_util - Library of utilities functions to build the toolbox

2.24.1 Calling Sequence

```
txt = write_inf_pal(u,listf,pal_title)
```

2.24.2 Parameters

- **u :** integer. a file descriptor
- **listf :** vector of strings. list of the files of the palettes
- **pal_title :** vector of strings. list of the title of the palettes
- **txt :** vector of strings. the text of the information to load modnum palettes

2.24.3 File content

```
//write_inf_pal
//Entrée : u file descriptor
//         listf : liste de fichiers de palette avec chemin(ex : \macros\scicos_blocks\Pll.cosf)
//         pal_title : liste de titres de la librairie (ex:mod_num_pll)
//Sortie txt : information utile de charegement
//             ex :mod_num_pal=['mod_num_pll',MODNUM+'\macros\scicos_blocks\Pll.cosf];
//                 scicos_pal=[scicos_pal;mod_num_pal];
function txt=write_inf_pal(u,listf,pal_title)
 if size(listf,'*')==size(pal_title,'*') then
  if MSDOS then
   listf=pathconvert(listf,%f,%t,'w');
  end
  fprintf(u,"\n%s\n",'//Add mod_num palette');
  tt_loader=['mod_num_pal=[';]
  for i=1:size(listf,"*")
     tt_loader=[tt_loader;''''+pal_title(i)+''',MODNUM+'''+listf(i)+''';'];
  end
  tt_loader(i+1)=tt_loader(i+1)+'];';
  tt_loader=[tt_loader;'scicos_pal=[scicos_pal;mod_num_pal];';''];
  fprintf(u,"%s\n",tt_loader);
  txt=tt_loader;
 else
  printf("listf and pal_title must have the same size");
  abort
 end
endfunction
```

2.25 Return text of routines for builder script of the toolbox

- **Name:** write_inf_rout_lib
- **Library:** build_util - Library of utilities functions to build the toolbox

2.25.1 Calling Sequence

```
txt = write_inf_rout_lib(u,libname,path,files,flag)
```

2.25.2 Parameters

- **u :** integer. a file descriptor
- **libname :** string. the name of the target routine library
- **path :** string. the path of the target routine library
- **files :** vector of strings. the names of the object files
- **flag :** integer. a flag to set the type of routines
  - **0 :** only for the library
  - **1 :** library + C modules
  - **2 :** library + fortran modules
- **txt :** vector of strings. the text of the information to load modnum routine

2.25.3 File content

```
//write_inf_rout_lib
//Entrée : u : file descriptor
//         libname : nom de la libraire (sans extension)
//         path : chemin de la libraire dans MODNUM ex /routines/mod_num_lib/
//         files : nom des modules à inclure (sans extension)
//         flag : drapeau 0 : seulement la librairie
//                        1 : librairie + modules 'c'
//                        2 : librairie + modules 'f'
function txt=write_inf_rout_lib(u,libname,path,files,flag)
 if MSDOS then
  mylibname=libname+'.dll';
 else
  mylibname=libname+'.so';
 end

 if flag==0 then
   tt_loader=['//Link '+libname+' library'
              'Id_'+libname+'=link(MODNUM+""'+path+'/'+mylibname+'"");';'';
             ]
 elseif flag==1 then
   var=[];
   for i=1:size(files,'*')
    var=var+''''+files(i)+''',';
   end
   var=part(var,1:length(var)-1);
   tt_loader=['//Link '+libname+' library'
            'Id_'+libname+'=link(MODNUM+""'+path+'/'+mylibname+'"",['+var+'],''c'');';
            '';
            ];

 elseif flag==2 then
   var=[];
   for i=1:size(files,'*')
    var=var+''''+files(i)+''',';
   end
   var=part(var,1:length(var)-1);
   tt_loader=['//Link '+libname+' library';
              'Id_'+libname+'=link(MODNUM+""'+path+'/'+mylibname+'"",['+var+']);';
              '';
             ];
 end
 txt=tt_loader(2);
 fprintf(u,"%s\n",tt_loader);
endfunction
```

Chapter 3

File management library

3.1 Purge scicos diagram

- **Name:** purge_diagr
- **Library:** find_file - File management library

3.1.1 Calling Sequence

```
purge_diagr(tt)
```

3.1.2 Parameters

- **tt :** matrix of strings of size(n,2). define the name and directory of the scicos file.
  - **tt(,1) :** the path of the scicos file
  - **tt(,2) :** the name of the scicos file

3.1.3 File content

```
//purge_diagr
//Fonction qui éxecute la fonction purge de scicos
//Entrée : tt un vecteur de chaînes de caractères de taille nx2
//         tt(,1) : le chemin du fichier
//         tt(,2) : le nom du fichier
function purge_diagr(tt)
  ////////////////////////////////
  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) ;
  pal_mode=%f;
  super_block=%f;
  ////////////////////////////////

  for i=1:size(tt,1)
    txt_tmp=tt(i,1)+tt(i,2);
    printf("Load and purge file %s\n",txt_tmp);
    load(txt_tmp);
    scs_m=do_purge(scs_m);
    if tt(i,1)<>scs_m.props.title(2) then
     scs_m.props.title(2)=tt(i,1);
    end
    do_save(scs_m)
  end
endfunction
```

3.2 Return names of .cos file presents in a tt_ml list

- **Name:** return_cos_file
- **Library:** find_file - File management library

3.2.1 Calling Sequence

```
tt = return_cos_file(tt_ml)
```

3.2.2 Parameters

- **tt_ml :** tt_ml master_list. (see return_master_list)
- **tt :** matrix of strings of size (n,2)
  - **tt(,1) :** paths of the files
  - **tt(,2) :** names of the files

3.2.3 File content

```
//return_cos_file
//fonction cherche les fichier d'extension .cos
//dans une liste principale (voir return_master_list)
//Entrée : tt_ml liste principale
//Sortie : tt vecteurs de chaîne de carctère de taille nx2
//         tt(,1) : chemin du fichier
//         tt(,2) : nom du ficher
function tt=return_cos_file(tt_ml)
tt=[]
//tt_ml=return_master_list();
p=size(tt_ml);
l=0
for i=1:p
   for j=1:size(tt_ml(i))
       for k=1:size(tt_ml(i)(j)(2),1)
        if strindex(tt_ml(i)(j)(2)(k),'.cos')<>[] then
          if strindex(tt_ml(i)(j)(2)(k),'.cosf')==[] then
              tt_tmp=[tt_ml(i)(j)(1),tt_ml(i)(j)(2)(k)];
              tt=[tt;tt_tmp];
          end
        end
       end
   end
end
endfunction
```

3.3 Return directories presents in a path in a tt_ml list

- **Name:** return_dir_in_dir
- **Library:** find_file - File management library

3.3.1 Calling Sequence

```
tt = return_dir_in_dir(tt_ml,dirn)
```

3.3.2 Parameters

- **tt_ml :** tt_ml master_list. (see return_master_list)
- **dirn :** string. name of directory to find in the tt_ml list
- **tt :** vector of strings. name of directories contained in the dirn directory.

3.3.3 File content

```
//return_dir_in_dir
//fonction qui cherche les noms de répertoires
//dans un répertoire dir dans la liste principale
//Entrée : tt_ml : une liste principale (voir return_master_list)
//         dirn : un vecteur de nom de répertoire
//               ex : MODNUM+/macros
//Sortie : tt un vecteur de taille 1 contenant
//           les noms de répertoires
function tt=return_dir_in_dir(tt_ml,dirn)

if MSDOS then
   dirn=pathconvert(dirn,%t,%t,'w')
else
```

```
   dirn=pathconvert(dirn,%t,%t,'u')
end

tt=[];
p=size(tt_ml); //cherche dans toute l'arborescence
l=0
for i=1:p
  for j=1:size(tt_ml(i)) //cherche dans tous les répertoires
   if(tt_ml(i)(j)(1)==dirn(1)) then
     for k=1:size(tt_ml(i)(j)(3),1) //uniquement les fichiers
        l=l+1;
        tt=[tt;tt_ml(i)(j)(3)(k)];
     end
     break
    end
  end
 if tt<>[] then break, end;
end
//printf("Found %d file(s) with %s extension in %s\n",l,ext,dirn);
endfunction
```

3.4 Return a list of a directory content

- **Name:** return_dir_list
- **Library:** find_file - File management library

3.4.1 Calling Sequence

```
ttl = return_dir_list(path)
```

3.4.2 Parameters

- **path :** vector of strings. the directories to analyse.
- **ttl :** list.
  - **ttl() :**
    - **ttl()(1) :** string. the string of the absolute path
    - **ttl()(2) :** vector of strings. name of files
    - **ttl()(3) :** vector of strings. name of directories

3.4.3 File content

```
//return_dir_list
//fonction qui retourne le contenu d'un répertoire
//sous formes d'une liste
//Entrée :
//path un vecteur de chaîne de caractère de chemin
//Sortie :
//ttl une liste
//ttl()(1) : le nom du chemin examiné (absolu)
//ttl()(2) : un vecteur de chaînes de caractères des noms de fichiers
//ttl()(3) : un vecteur de chaînes de caractères des noms de répertoires
//          (sans chemin) (ex: ['man';'macros']
function ttl=return_dir_list(path)
ttl=list();
if path<>[] then
 p=size(path,1);
 for j=1:p
     tt_sub=return_dir_name(path(j))
     tt_fil=return_fil_name(path(j))
     if MSDOS then
       path(j)=pathconvert(path(j),%t,%t,'w')
     else
       path(j)=pathconvert(path(j),%t,%t,'u')
     end
     ttl(j)=list(path(j),tt_fil,tt_sub);
 end
end
endfunction
```

3.5 Return directories presents in a path

- **Name:** return_dir_name

- **Library:** find_file - File management library

3.5.1 Calling Sequence

```
tt = return_dir_name(path)
```

3.5.2 Parameters

- **path :** string. the directory to analyse.
- **tt :** vector of strings. the name of the directories

3.5.3 File content

```
//tt une variable txt
//path un chemin
function tt=return_dir_name(path)
 tt=[];
 if MSDOS then
   if part(path,length(path))==’\’ then
      path=part(path,1:length(path)-1);
   end
 end
 if fileinfo(path)<>[] then
   rep=pwd();
   chdir(path);

   if MSDOS then
     k=1;
     tt=[];
     str=’ttf=unix_g(’’dir /B /A:D’’);’;
     execstr(str,’errcatch’);
     if ttf<>[] then
       for j=1:size(ttf,1)
         if isdir(ttf(j)) then
           tt(k)=ttf(j); k=k+1;
         end
       end
     end

   else
     k=1;
     tt=[];
     str=’ttf=unix_g(""ls"")’;
     execstr(str,’errcatch’);
     if ttf<>[] then
       for j=1:size(ttf,1)
         if isdir(ttf(j)) then
           tt(k)=ttf(j); k=k+1;
         end
       end
     end
   end

  chdir(rep);
 end
endfunction
```

3.6 Return file of specified extension presents in a tt_ml list

- **Name:** return_ext_file
- **Library:** find_file - File management library

3.6.1 Calling Sequence

```
tt = return_ext_file(tt_ml,ext)
```

3.6.2 Parameters

- **tt_ml :** tt_ml master_list. (see return_master_list)
- **ext :** string. the extension of file to find in the tt_ml list
- **tt :** vector of strings. the list of the file names of extension ext

3.6.3 File content

```
//return_ext_file
//fonction qui cherche les fichiers d'extension ext
//dans une liste principale
//Entrée : tt_ml : une liste principale (voir return_master_list)
//         ext : un vecteur chaîne d'extensions finales de fichier
//        (ex ext='sci', ext='cos', mais aussi ext='monfichier.sci')
//Sortie : tt un vecteur de taille 2 contenant
//           le chemin des fichiers cherchés et le nom du fichier
function tt=return_ext_file(tt_ml,ext)

//doit faire faire sur type of ext
a=length(ext)

tt=[];
p=size(tt_ml); //cherche dans toute l'arborescence
l=0
for i=1:p
  for j=1:size(tt_ml(i)) //cherche dans tous les répertoires
     for k=1:size(tt_ml(i)(j)(2),1) //uniquement les fichiers
       tt_nam=tt_ml(i)(j)(2)(k);
       //for m=1:size(a,1) //meilleur temps si l'on utilise ext de taille 1
        if length(tt_nam)>(a-1) then
          //ierr=execstr('ok=part(tt_nam,length(tt_nam)-(a-1):length(tt_nam))==ext','errcatch');
          //if ierr==0 & ok==%t then
          if part(tt_nam,length(tt_nam)-(a-1):length(tt_nam))==ext then
          //b=strindex(tt_ml(i)(j)(2)(k),ext);
          //if b(size(b,'*'))==length(tt_ml(i)(j)(2)(k))-(a-1) then
            l=l+1;
            tt_tmp=[tt_ml(i)(j)(1),tt_ml(i)(j)(2)(k)];
            tt=[tt;tt_tmp];
          end
        //end
       end
     end
  end
end
printf("Found %d file(s) with %s extension\n",l,ext);
endfunction
```

3.7 Return file of specified extension in a directory presents in a tt_ml list

- **Name:** return_ext_file_in_dir
- **Library:** find_file - File management library

3.7.1 Calling Sequence

```
tt = return_ext_file_in_dir(tt_ml,dirn,ext)
```

3.7.2 Parameters

- **tt_ml :** tt_ml master_list. (see return_master_list)
- **dirn :** string. name of directory to find in the tt_ml list
- **ext :** string. the extension of file to find in the tt_ml list
- **tt :** vector of strings. the list of the file names of extension ext

3.7.3 File content

```
//return_ext_file_in_dir
//fonction qui cherche les fichiers d'extension ext
//dans un répertoire dir dans une liste principale
//Entrée : tt_ml : une liste principale (voir return_master_list)
//         dirn : un vecteur de nom de répertoire
//              ex : MODNUM+/macros
//         ext : un vecteur chaîne d'extensions finales de fichier
//        (ex ext='sci', ext='cos', mais aussi ext='monfichier.sci')
//Sortie : tt un vecteur de taille 1 contenant
//           les noms de fichiers d'extension ext
function tt=return_ext_file_in_dir(tt_ml,dirn,ext)

if MSDOS then
   dirn=pathconvert(dirn,%t,%t,'w')
else
   dirn=pathconvert(dirn,%t,%t,'u')
end

//doit faire faire sur type of ext
a=length(ext(1))
```

```
tt=[];
p=size(tt_ml); //cherche dans toute l'arborescence
l=0
for i=1:p
  for j=1:size(tt_ml(i)) //cherche dans tous les répertoires
   if(tt_ml(i)(j)(1)==dirn(1)) then
     for k=1:size(tt_ml(i)(j)(2),1) //uniquement les fichiers
       tt_nam=tt_ml(i)(j)(2)(k);
        if length(tt_nam)>(a-1) then
          if part(tt_nam,length(tt_nam)-(a-1):length(tt_nam))==ext then
            l=l+1;
            tt=[tt;tt_ml(i)(j)(2)(k)];
          end
       end
     end
   break;
   end
  end
 if tt<>[] then break, end;
end
//printf("Found %d file(s) with %s extension in %s\n",l,ext,dirn);
endfunction
```

3.8 Return list of files presents in a path

- **Name:** return_fil_name
- **Library:** find_file - File management library

3.8.1 Calling Sequence

```
tt = return_fil_name(path)
```

3.8.2 Parameters

- **path :** string. the directory to analyse.
- **tt :** vector of strings. the name of the directories

3.8.3 File content

```
function tt=return_fil_name(path)
 tt=[];
 if MSDOS then
   if part(path,length(path))==’\’ then
      path=part(path,1:length(path)-1);
   end
 end
 if fileinfo(path)<>[] then
   rep=pwd();
   chdir(path);

   if MSDOS then
     k=1;
     tt=[];
     str=’ttf=unix_g(’’dir /B’’);’;
     execstr(str,’errcatch’);
     if ttf<>[] then
       for j=1:size(ttf,1)
         if ~isdir(ttf(j)) then
           tt(k)=ttf(j); k=k+1;
         end
       end
     end

   else
     k=1;
     tt=[];
     str=’ttf=unix_g(""ls"")’;
     execstr(str,’errcatch’);
     if ttf<>[] then
       for j=1:size(ttf,1)
         if ~isdir(ttf(j)) then
           tt(k)=ttf(j); k=k+1;
         end
       end
     end
   end

   chdir(rep);
 end
endfunction
```

3.9 Return a list which contains all directories and files names of a specified directory

- **Name:** return_master_list
- **Library:** find_file - File management library

3.9.1 Calling Sequence

```
tt_ml = return_master_list(path)
```

3.9.2 Parameters

- **path :** string. the directory to analyse.
- **tt_ml :** a master list.
  - **tt_ml() :** the level in the directory tree
  - **tt_ml()() :** the id of the directory
  - **tt_ml()()() :**
    - **tt_ml()()(1) :** name of the directory
    - **tt_ml()()(2) :** list of files
    - **tt_ml()()(3) :** list of directories

3.9.3 File content

```
//return_master list
//fonction qui examine l'intégralité d'un répertoire
//et qui retourne ses noms de repertoires et fichiers
//dans une liste scilab.
//Entrée : path un vecteur de noms de répertoire à éxaminer
//         de pref en chemin absolu et de taille 1x1
//Sortie tt_ml : liste principale
// tt_ml()()()
//      | | |
//      | | |
//      | | |--> 1 : nom du répertoire
//      | |      2 : liste des fichiers
//      | |      3 : liste des répertoires sous-adjacents
//      | |
//      | |--> indice du n° répertoire
//      |
//      |--> indice du niveau d'arborescence
function tt_ml=return_master_list(path)

 if argn(2)==0 | ~exists('path') then
  path=MODNUM
 elseif typeof(path)<>"string" then
  path=MODNUM
 elseif path=="" then
  path=MODNUM
 end

 printf("Search directories and files in %s... ",path);
 tt_ml=list();k=0;
 while path<>[]
   k=k+1;
  [tt,path]=return_single_list(path);
  tt_ml(k)=tt;
 end
 printf("Done\n");
endfunction
```

3.10 Return the directory of a specified cos file in a tt_ml list

- **Name:** return_path_cos_file
- **Library:** find_file - File management library

3.10.1 Calling Sequence

```
path = return_path_cos_file(name,tt_ml)
```

3.10.2 Parameters

- **name :** string. the name of the scicos diagram to find (extension free)
- **tt_ml :** tt_ml master_list. (see return_master_list)
- **path :** string. path of scicos diagram file

3.10.3 File content

```
function path=return_path_cos_file(name,tt_ml)
tt=return_cos_file(tt_ml);
path=[];
for i=1:size(tt,1)
 c=strindex(tt(i,2),name+'.cos');
 if c<>[] then
  if c==1 then
   path=tt(i,1);
   break;
  end;
 end
end
endfunction
```

3.11 Return Relative Path Of Root Directory Of Ext(specified) File in a tt_ml list

- **Name:** return_rpordoef
- **Library:** find_file - File management library

3.11.1 Calling Sequence

```
tt = return_rpordoef(tt_ml,ext)
```

3.11.2 Parameters

- **tt_ml :** tt_ml master_list. (see return_master_list)
- **ext :** string. the extension of file to find in the tt_ml list
- **tt :** string. the relative path

3.11.3 File content

```
//return_rpordoef
//return Relatif Path Of Root Directory Of ext File
//utilisé pour trouver les noms des palettes et librairies
//fonction qui cherche les fichiers d'extension ext
//dans la liste principale et qui retourne
//le répertoire racine du fichier cherché en chemin relatif
//Entrée : tt_ml : une liste principale (voir return_master_list)
//         ext : un vecteur chaîne d'extensions finales de fichier
//        (ex ext='sci', ext='cos', mais aussi ext='monfichier.sci')
//Sortie : tt un vecteur de taille 1 contenant
//         le chemin relatif des fichiers cherchés
//         ex tt='build_util' ou 'Skins'
function tt=return_rpordoef(tt_ml,ext)
//doit faire faire sur type of ext
a=length(ext)

tt=[];
p=size(tt_ml); //cherche dans toute l'arborescence
l=0
for i=1:p
  for j=1:size(tt_ml(i)) //cherche dans tous les répertoires
     for k=1:size(tt_ml(i)(j)(2),1) //uniquement les fichiers
       tt_nam=tt_ml(i)(j)(2)(k);
        //if length(tt_nam)>(a-1) then
          //if part(tt_nam,length(tt_nam)-(a-1):length(tt_nam))==ext then
          if tt_nam==ext then
            l=l+1;
```

```
            tt_tmp=[tt_ml(i)(j)(1)];
            tt=[tt;tt_tmp];
            break;
          end
        //end
      end
      if tt<>[] then break, end
   end
   if tt<>[] then break, end
end

if tt<>[] then
 for i=1:size(tt,1)
  tt(i)=part(tt(i),1:length(tt(i))-1);
 end
 tt=basename(tt);
end
endfunction
```

3.12 Return a single list of directories and files names of a specified directory

- **Name:** return_single_list
- **Library:** find_file - File management library

3.12.1 Calling Sequence

```
[tt,path] = return_single_list(path)
```

3.12.2 Parameters

- **path :** vector of strings. a directories list
- **tt :** list.
  - **tt() :** id the of the directory
  - **tt()() :**
    - **tt()(1) :** string. the name of the directory
    - **tt()(2) :** vector of strings. the list of the files
    - **tt()(3) :** vector of strings. the list of absolute path
- **path :** vector of strings. the list of absolute path of sub-adjacent directories.

3.12.3 File content

```
//return_single_list
//Entrée path : un vecteur de chaîne de caractère de nom de chemin de taille n
//Sortie tt une liste
//       tt()(1) le nom du répertoire examiné
//       tt()(2) la liste des fichiers
//       tt()(3) la liste des chemins absolus
//               ex : ['/home/man';'/home/macros']
//path : la liste de tous les répertoires sous adjacents aux repertoires
//       du vecteur d'entrée (chemins absolus)
function [tt,path]=return_single_list(path)
tt=return_dir_list(path);
MORE=%F;
for j=1:size(tt)
 if size(tt(j)(3),1)<> 0 then MORE=%T; end;
end
if ~MORE then break; end;
 path=[];
for j=1:size(tt)
 if (tt(j)(3)<>[]) then
   if MSDOS then
     tt(j)(3)=tt(j)(1)+tt(j)(3)+'\';
   else
     tt(j)(3)=tt(j)(1)+tt(j)(3)+'/';
   end
    path=[path;tt(j)(3)];
 end
end
endfunction
```

Part II

Documentation generator macro libraries

Chapter 1

Library of utilities functions for documentation generation

1.0.1 See Also

- IMODULO_f - Integer modulo function block (Scicos Block)
- IMODULOB_f - Integer modulob function block (Scicos Block)
- FMODULOB_f - Float modulob function block (Scicos Block)
- FMODULOC_f - Float moduloc function block (Scicos Block)
- LCMODULO_f - Left shift circulate integer modulo function block (Scicos Block)

1.1 Modify bbl LaTeX file

- **Name:** analyse_bbl_file
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.1.1 Calling Sequence

```
analyse_bbl_file(name)
```

1.1.2 Parameters

- **name :** string. path+name of the bbl to analyse

1.1.3 File content

```
//analyse_bbl_file
//fonction qui modifie un fichier bbl
//produit par bibtex avec IEETransBST
//pour pouvoir correctement etre compiler
//par latex2html
//
//Entrée : name : nom du fichier à analyser.

function analyse_bbl_file(name)
 if fileinfo(name)<>[] then
  tt=mgetl(name);
  a=[];
  b=[];
  for i=1:size(tt,1)
   if strindex(tt(i),'\csname url@rmstyle\endcsname')<>[] then
    a=i;
   end
   if strindex(tt(i),'\bibitem{')<>[] then
    b=i;
    break;
   end
  end
  if a<>[] & b<>[] then
```

```
   new_tt=[tt(1:a-1);tt(b:size(tt,1))];
   mputl(new_tt,name);
  end
 end

endfunction
```

1.2 Analyse a LaTeX file

- **Name:** analyse_tex_file
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.2.1 Calling Sequence

```
analyse_tex_file(rep)
```

1.2.2 Parameters

- **rep :** string. directory of tex files to analyse

1.2.3 Description

This function change a html source file produced from latex2html in a readable man page by the scilab help browser. At this time, scilab (scilab 3-3.1rc) help browser doesn't support figure inserted in html environment <TABLE>.

1.2.4 File content

```
//analyse_tex_file
//fonction qui effectue les changements nécessaires
//dans des fichiers tex pour rendre la page finale
//convertie en html par latex2html correctement affichable
//par le browser de scilab
//Entrée rep : repertoire où sont les fichiers tex
//
function analyse_tex_file(rep)

 lisf_rep=return_fil_name(rep)
 lisf_tex=[];
 k=1
 for i=1:size(lisf_rep,1)
   if strindex(lisf_rep(i),'.tex')<>[] then
    lisf_tex(k)=lisf_rep(i);
    k=k+1
   end
 end

 if lisf_tex<>[] then
  printf("Analysing tex file... ")
  for i=1:size(lisf_tex,1)
   txt_tex=change_capt_tex_file(lisf_tex(i));
   if txt_tex<>[] then
    mputl(txt_tex,lisf_tex(i));
   end
   txt_tex=change_equa_tex_file(lisf_tex(i));
   if txt_tex<>[] then
    mputl(txt_tex,lisf_tex(i));
   end
  end
  printf("Done\n");
 end

endfunction
```

1.3 Change label of bibliography in a html file

- **Name:** change_biblio_line
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.3.1 Calling Sequence

```
tt = change_biblio_line(htmf,lang)
```

1.3.2 Parameters

- **htmf :** add here the parameter description
- **lang :** add here the parameter description
- **tt :** add here the parameter description

1.3.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

1.3.4 Example

```
Add here scilab instructions and comments
```

1.3.5 File content

```
//change_biblio_line
//Fonction qui change le texte "Bibliography"
//d'une page d'aide html
//Entrée : htmf  : un nom de fichier à modifier
//         lang : langue
//Sortie : tt le texte du fichier htm
function tt=change_biblio_line(htmf,lang)
 if fileinfo(htmf(1,1))<>[] then
  tt=mgetl(htmf(1,1));
  if tt<>[] then
   printf("Change Bibliography line... ")
   for i=1:size(tt,1)
    a=strindex(tt(i),"Bibliography</A>");
    //pause
    if a<>[] then
      if lang=='fr' then
       tt(i)=strsubst(tt(i),"Bibliography</A>","Bibliographie</A>")
      end
    end
   end
   printf("Done\n")
  end
 else
  tt=[]
 end
endfunction
```

1.3.6 Used function(s)

Add here the used function name and references

1.3.7 See Also

- add a key here - ()
- add a key here - ()

1.3.8 Bibliography

Add here the function bibliography if any

1.4 Change figure caption of tex file to allow compatibility with scilab browser

- **Name:** change_capt_tex_file
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.4.1 Calling Sequence

```
txt = change_capt_tex_file(file_tex)
```

1.4.2 Parameters

- **file_tex :** string. The path+name of the tex file to analyze
- **txt :** vector of strings. the text of the new tex file

1.4.3 File content

```
//change_capt_tex_file
//fonction qui effectue les changements sur les
//titres des figures dans un fichier tex pour rendre la page finale
//convertie en html par latex2html correctement affichable
//par le browser de scilab
//Entrée file_tex : fichier à analyser
//Sorite txt : texte du nouveau fichier tex

function txt=change_capt_tex_file(file_tex)

 txt_tex_main=mgetl(file_tex)
 num_fig=0;
 tt_fig=list();
 for i=1:size(txt_tex_main,1)
   if strindex(txt_tex_main(i),'\begin{figure}')<>[] then
     num_fig=num_fig+1
     a(num_fig)=i
     tt_fig(num_fig)=""
   elseif strindex(txt_tex_main(i),'\end{figure}')<>[] then
     b(num_fig)=i;
     //cherche la ligne caption
     for j=a(num_fig):b(num_fig)
       if strindex(txt_tex_main(j),'\caption{')<>[] then
         capt=strsubst(txt_tex_main(j),'\caption{','');
         ja=1;
         //trouve la légende
         for k=1:length(capt)
           tt_char=part(capt,k)
           if tt_char=='{' then
            ja=ja+1
           elseif tt_char=='}' then
            ja=ja-1
           end
           if ja==0 then break, end
         end
         capt=part(capt,1:k-1);

         tt_fig(num_fig)=[txt_tex_main(a(num_fig):j-1);
                          '%'+txt_tex_main(j);
                          txt_tex_main(j+1:b(num_fig));
                          '\begin{center}';
                          '\textbf{Figure :} '+capt;
                          '\end{center}']
       end
     end
     if tt_fig(num_fig)=="" then
      tt_fig(num_fig)=txt_tex_main(a(num_fig):b(num_fig));
     end
   end
 end

 if num_fig<>0 then
  txt=[]
  for i=1:num_fig
   if i==1 then
    i_beg=0
   else
    i_beg=b(i-1)
   end
   i_end=a(i)
   txt=[txt;txt_tex_main(i_beg+1:i_end-1);
        tt_fig(i);]
  end
  txt=[txt;txt_tex_main(b(num_fig)+1:size(txt_tex_main,1))]
 else
  txt=[];
 end

endfunction
```

1.5 Change color of subtitles in a html file

- **Name:** change_color_subtitle
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.5.1 Calling Sequence

```
tt = change_color_subtitle(htmf,colorn)
```

1.5.2 Parameters

- **htmf :** string. the path+name of the html file
- **colorn :** string. the color which must be apply
- **tt :** vector of strings. the text of the new html file.

1.5.3 File content

```
//change_font_subtitle_color
//Fonction qui change la couleur des sous-titres
//d'une page d'aide html
//Entrée : htmf  : un nom de fichier à modifier
//         colorn : chaine de caractère la couleur à appliquer
//Sortie : tt le texte du fichier htm
function tt=change_color_subtitle(htmf,colorn)
 if fileinfo(htmf(1,1))<>[] then
  tt=mgetl(htmf(1,1));
  if tt<>[] then
   flagb='<H2>';
   flage='</H2>';
   printf("Change color of subtitles... ")
   for i=1:size(tt,1)
    a=strindex(tt(i),flagb);
    //disp(a)
    //pause
    if a<>[] then
     for j=1:size(a,1)
      //disp(tt(i))
      tt(i)=strsubst(tt(i),flagb,flagb+'<font color="""+colorn+"""">')
      //disp(tt(i))
     end
    end
    b=strindex(tt(i),flage);
    if b<>[] then
     for j=1:size(b,1)
      tt(i)=strsubst(tt(i),flage,'</font>'+flage)
     end
    end
   end
   printf("Done\n")
  end
 else
  tt=[]
 end
endfunction
```

1.6 Change label of content in a html file

- **Name:** change_contents_line
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.6.1 Calling Sequence

```
tt = change_contents_line(htmf,lang)
```

1.6.2 Parameters

- **htmf :** add here the parameter description
- **lang :** add here the parameter description
- **tt :** add here the parameter description

1.6.3 Description

Add here a paragraph of the function description. Other paragraph can be added
Add here a paragraph of the function description

1.6.4 Example

```
Add here scilab instructions and comments
```

1.6.5 File content

```
//change_contents_line
//Fonction qui change le texte "contents"
//d'une page d'aide html
//Entrée : htmf  : un nom de fichier à modifier
//         lang : langue
//Sortie : tt le texte du fichier htm
function tt=change_contents_line(htmf,lang)
 if fileinfo(htmf(1,1))<>[] then
  tt=mgetl(htmf(1,1));
  if tt<>[] then
   printf("Change contents line... ")
   for i=1:size(tt,1)
    a=strindex(tt(i),"Contents</A>");
    if a<>[] then
      if lang=='fr' then
       tt(i)=strsubst(tt(i),"Contents</A>","Contenu</A>")
      end
    end
   end
   printf("Done\n")
  end
 else
  tt=[]
 end
endfunction
```

1.6.6 Used function(s)

Add here the used function name and references

1.6.7 See Also

- add a key here - ()
- add a key here - ()

1.6.8 Bibliography

Add here the function bibliography if any

1.7 Change LaTeX equation to be readable with the scilab browser

- **Name:** change_equa_tex_file
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.7.1 Calling Sequence

```
txt = change_equa_tex_file(file_tex)
```

1.7.2 Parameters

- **file_tex :** string. path+name of the tex file to analyze
- **txt :** vector of strings. the text of the new tex file

1.7.3 File content

```
//change_equa_tex_file
//fonction qui effectue les changements sur les
//equations dans un fichier tex pour rendre la page finale
//convertie en html par latex2html correctement affichable
//par le browser de scilab
//Entrée file_tex : fichier à analyser
//Sortie txt : texte du nouveau fichier tex

function txt=change_equa_tex_file(file_tex)

 txt_tex_main=mgetl(file_tex)

 num_equa=0;
 tt_equa=list();
 for i=1:size(txt_tex_main,1)
   if strindex(txt_tex_main(i),'\begin{eqnarray}')<>[] then
     num_equa=num_equa+1
     a(num_equa)=i
     tt_equa(num_equa)=""
   elseif strindex(txt_tex_main(i),'\begin{equation}')<>[] then
     num_equa=num_equa+1
     a(num_equa)=i
     tt_equa(num_equa)=""
   elseif strindex(txt_tex_main(i),'\end{eqnarray}')<>[] then
     b(num_equa)=i;
     tt_equa(num_equa)=txt_tex_main(a(num_equa):b(num_equa));
   elseif strindex(txt_tex_main(i),'\end{equation}')<>[] then
     b(num_equa)=i;
     tt_equa(num_equa)=txt_tex_main(a(num_equa):b(num_equa));
   end
 end
 if num_equa<>0 then
   for i=1:num_equa
    tt_equa(i)=strsubst(tt_equa(i),'\begin{eqnarray}','$$');
    tt_equa(i)=strsubst(tt_equa(i),'\end{eqnarray}','$$');
    tt_equa(i)=strsubst(tt_equa(i),'\begin{equation}','$$');
    tt_equa(i)=strsubst(tt_equa(i),'\end{equation}','$$');
    tt_equa(i)=strsubst(tt_equa(i),'&=&','\,=\,');
    tt_equa(i)=strsubst(tt_equa(i),'&','\,');
   end
// pause
   new_tt_equa=list("");
   for i=1:num_equa
     k=1;
     for j=1:size(tt_equa(i),1)
       i_break=[];
       i_break=strindex(tt_equa(i)(j),'\\')
       if i_break<>[] then
         for e=1:size(i_break,2)
           if e==1 then i_beg=1, else i_beg=i_break(e-1)+2, end
           i_end=i_break(e)-1;
           new_tt_equa(i)(k)=part(tt_equa(i)(j),i_beg:i_end);k=k+1;
           new_tt_equa(i)(k)="$$";k=k+1;
           new_tt_equa(i)(k)="$$";k=k+1;
         end
       else
         if k==1 then new_tt_equa(i)="", end;
         new_tt_equa(i)(k)=tt_equa(i)(j);
         k=k+1;
       end
     end
   end
 end
 //pause

 if num_equa<>0 then
  tt_equa=new_tt_equa
  txt=[]
  for i=1:num_equa
   if i==1 then
    i_beg=0
   else
    i_beg=b(i-1)
   end
   i_end=a(i)
   txt=[txt;txt_tex_main(i_beg+1:i_end-1);
        tt_equa(i);]
  end
  txt=[txt;txt_tex_main(b(num_equa)+1:size(txt_tex_main,1))]
 else
  txt=[];
 end

 //pause
endfunction
```

1.8 Change font and others in html file

- **Name:** change_font
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.8.1 Calling Sequence

```
tt = change_font(htmf,flag)
```

1.8.2 Parameters

- **htmf :** string. path+name of the html file
- **flag :** string. a flag to set the type of the html man page :
  - **'sci' :** for scilab macro
  - **'rout' :** for routine function
  - **'sce' :** for script
- **tt :** vector of strings. the text of the html file

1.8.3 File content

```
//change_font
//Fonction qui change les fontes
//d'une page d'aide html produits par latex2html
//aux "normes" des pages d'aides scilab
//produites par xmltohtml
//Entrée : htmf  : un nom de fichier à modifier
//         flag : 'block','pal','sci','scilib',....
//Sortie : tt le texte du fichier htm
function tt=change_font(htmf,flag)
 if ~exists('flag') then flag='sci', end;
 ok=%f
 if fileinfo(htmf(1,1))<>[] then
  tt=mgetl(htmf(1,1));
  if tt<>[] then
    printf("Font convertion... ");
    //1ere analyse : ajuste le texte en gras
    flagb='<SPAN  CLASS=""textbf"">';
    flage='</SPAN>';
    flagf='</SPAN>0';
    i=1;

    while i<>size(tt,1)
      a=strindex(tt(i),flagb);
      if a<>[] then
        tt(i)=strsubst(tt(i),flagb,'<b>')
        b=strindex(tt(i),flage);
        if size(b,2)==1 then
         if strindex(tt(i),flagf)<>b then
          ok=%t
          d=b;
         end
        else
         for j=1:size(b,2)
           if(b(j))<>strindex(tt(i),flagf) then
            ok=%t
            d=b(j)
           end
         end
        end
        while ~ok
          i=i+1
          if i>size(tt,1) then
           printf("Warning in %s : change font convertion error\n",htmf(1,1))
           break
          end
          b=strindex(tt(i),flage);
          if size(b,2)==1 then
           if strindex(tt(i),flagf)<>b then
             ok=%t
             d=b;
           end
          else
           for j=1:size(b,2)
            if(b(j))<>strindex(tt(i),flagf) then
             ok=%t
             d=b(j)
            end
           end
          end
        end
        tt(i)=part(tt(i),1:d-1)+strsubst(part(tt(i),d:length(tt(i))),flage,'</b>')
      end
      i=i+1;
    end
    printf("Done\n");

    //2eme analyse : Change la 1ere ligne des fichiers d'aide
    //type 'sci' et 'rout' - enlève le les délimiteurs <H1> </H1>
    if flag=='sci'|flag=='rout'|flag=='sce' then
     if flag=='sci' then
```

```
      printf("Scilab function : change font of first line... ");
     elseif flag==’rout’ then
      printf("Low level routine : change font of first line... ");
     elseif flag==’sce’ then
      printf("Scilab script : change font of first line... ");
     end
     for i=1:size(tt,1)
      tt(i)=strsubst(tt(i),’<H1>’,’<BR>’);
      tt(i)=strsubst(tt(i),’</H1>’,’’);
     end
     printf("Done\n");
    end

    //3eme analyse : change la profondeur des titres et sous-titres
    printf("Change level of subtitles... ")
    for i=1:size(tt,1)
     tt(i)=strsubst(tt(i),’<H2>’,’<H3>’);
     tt(i)=strsubst(tt(i),’</H2>’,’</H3>’);
     tt(i)=strsubst(tt(i),’<H1>’,’<H2>’);
     tt(i)=strsubst(tt(i),’</H1>’,’</H2>’);
    end
    printf("Done\n");

    //4eme analyse : enlève la ligne du bas et passe
    //<BODY > en <BODY bgcolor="#FFFFFF">
    printf("Change body color and remove address line... ")
    for i=1:size(tt,1)
      tt(i)=strsubst(tt(i),’<BODY>’,’<BODY bgcolor=""#FFFFFF"">’);
      tt(i)=strsubst(tt(i),’<BODY >’,’<BODY bgcolor=""#FFFFFF"">’);
      if strindex(tt(i),’<ADDRESS>’)<>[] then
       aa=i
       if strindex(tt(i-1),’<HR>’)<>[] then
         tt(i-1)=strsubst(tt(i-1),’<HR>’,’’);
       end
      end
      if strindex(tt(i),’</ADDRESS>’)<>[] then
       bb=i
      end
    end

    if exists(’aa’)&exists(’bb’) then
     for i=aa:bb
      tt(i)=""
     end
    end
    printf("Done\n");

    //5eme analyse : change les délimiteurs ’<P><A’ en ’<A’
    //et </A></P>
    printf("Verification of labels... ")
    for i=1:size(tt,1)
     if strindex(tt(i),’<P><A’)<>[] then
      tt(i)=strsubst(tt(i),’<P><A’,’<A’);
      tt(i)=strsubst(tt(i),’</A></P>’,’’);
     end
    end
    printf("Done\n");

  end
 else
  tt=[]
 end
endfunction
```

1.9 Change language of title

- **Name:** change_lang_title
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.9.1 Calling Sequence

```
txt = change_lang_title(lg,title)
```

1.9.2 Parameters

- **lg :** string. the desired output language
  - **’fr’ :** for french
  - **title :** vector of strings. the strings to traduce
  - **txt :** vector of strings. the strings traduced

1.9.3 File content

```
//fonction qui change des titres donnés dans
//un vecteur de chaînes de caractères
//suivant le paramètre lang
//entrée : lang : 'fr' pour du francais
//         title : vecteurs de chaines de caratères
function txt=change_lang_title(lg,title)
 txt=[]
 txt=title

 if lg=='fr' then
  txt=strsubst(txt,'Theorical background','Rappel théorique');
  txt=strsubst(txt,'Technical background','Rappel technique');
  txt=strsubst(txt,'Algorithm','Algorithmes');
  txt=strsubst(txt,'Super block equivalent model','Modèle équivalent en Super Bloc');
  txt=strsubst(txt,'Scilab script/function equivalent Model','Script/fonction scilab équivalente');
  txt=strsubst(txt,'Dialog box','Boîte de dialogue');
  txt=strsubst(txt,'Example','Exemple');
  txt=strsubst(txt,'Default properties','Propriétés par défaut');
  txt=strsubst(txt,'Interfacing function','Fonction d''interface');
  txt=strsubst(txt,'Computational function','Fonction de calcul');
  txt=strsubst(txt,'Used functions','Fonction utilisée');
  txt=strsubst(txt,'See also','Voir aussi');
  txt=strsubst(txt,'See Also','Voir aussi');
  txt=strsubst(txt,'Authors','Auteurs');
  txt=strsubst(txt,'Bibliography','Bibliographie');
  txt=strsubst(txt,'Blocks','Blocs');
  txt=strsubst(txt,'Context','Contexte');
  txt=strsubst(txt,'Scope Results','Résultats des oscilloscopes');
  txt=strsubst(txt,'Mod\_num blocks','Bloc Mod\_num');
  txt=strsubst(txt,'Simulation script(s)','Script(s) de simulation');
  txt=strsubst(txt,'Scicos diagram(s)','Diagramme(s) Scicos');
  txt=strsubst(txt,'Scilab function','Fonction Scilab');
  txt=strsubst(txt,'Package','Paquet');
  txt=strsubst(txt,'Library','Librairie');
  txt=strsubst(txt,'Calling Sequence','Séquence d''appel');
  txt=strsubst(txt,'Parameters','Paramètres');
  txt=strsubst(txt,'File content','Contenu du fichier');
  txt=strsubst(txt,'Used function(s)','Fonction(s) utilisée(s)');
  txt=strsubst(txt,'Scicos Block','Bloc Scicos');
  txt=strsubst(txt,'Low level routine','Routine de calcul bas-niveau');
 end

endfunction
```

1.10 Change level of bibliography section

- **Name:** change_level_bib
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.10.1 Calling Sequence

```
txt = change_level_bib(name)
```

1.10.2 Parameters

- **name :** string. path+name of the html file
- **txt :** vector of strings. text of the nex html file

1.10.3 File content

```
//change_level_bib
//fonction qui change le niveau du paragraphe
//bibliographie dans la table des matières
//d'un fichier html produit par latex2html
//
//Entrée : name : nom du fichier html
//Sortie : txt : texte du nouveau fichier html
//
function txt=change_level_bib(name)
 if fileinfo(name)<>[] then
   tt=mgetl(name);
   if tt<>[] then
     a=[];
     b=[];
     for i=1:size(tt,1)
       if strindex(tt(i),'>Bibliography</A>')<>[] then
         a=i;
         if strindex(tt(a-2),'</UL><BR>')<>[] then
           b=a-2;
           break;
         end
       end
```

```
     end
     if a<>[] & b<>[] then
      txt=[tt(1:b-1);tt(b+1:a+1);tt(b);tt(a+2:size(tt,1))];
     else
      txt=[];
     end
    //pause
   else
    txt=[];
   end
 else
  txt=[];
 end
endfunction
```

1.11 Close all scilab figure

- **Name:** del_all_graphics
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.11.1 Calling Sequence

```
del_all_graphics
```

1.11.2 File content

```
//Fonction qui ferme toutes les fenetres
//graphiques ouvertes (old_style)
function del_all_graphics()
while %t
  win=xget("window");
  if win==0 then
    xdel(win);
    win=xget("window");
    if win==0 then
      xdel(win);
      break
    end
  else
  xdel(win)
  end
end
endfunction
```

1.12 Export block figure in eps file

- **Name:** dessin_block
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.12.1 Calling Sequence

```
dessin_block(name,flag)
```

1.12.2 Parameters

- **name :** string. name of an interfacing function of a scicos block
- **flag :** string. option to set the up/down and letf/right margin
  - **'html' :** to produce a figure for html man page
  - **'guide' :** to produce a figure for man page with paper format

1.12.3 File content

```
//dessin_block
//fonction qui charge une structure graphique
//d'un block scicos dans une liste scs_m et qui
//exporte les données graphiques en fichier eps
//grâce à la fonction mdo_export
//Entrée : nom de la fonction d'interface du block
//         flag 'html' ou 'guide'
//         lblock [] ou autre chose (renseigne sur le type de block)
function dessin_block(name,flag,lblock)
  //vérifie présence lblock
  if rsh<3 then
    lblock=[];
  end

  //load scicos variable and library
  bak=get('figure_style');
  set("figure_style","old");
  olds=get('old_style');
  set('old_style','on');

  //load scicos variable and library
  load SCI/macros/scicos/lib
  exec(loadpallibs,-1)
  %scicos_prob=%f;
  alreadyran=%f
  needcompile=4
  %zoom=1.8;
  colmap=xget('colormap');

  scs_m=scicos_diagram()
  ierror=execstr('blk='+name+'(''define'')','errcatch')
  if ierror <>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp('define'+name)
    fct=[]
    return
  end
  blk.graphics.sz=20*blk.graphics.sz;
  scs_m.objs(1)=blk
  if flag=='html' then
     if lblock<>[] then
       newflag='html_lblock'
     else
       newflag='html_block'
     end
  else
     if lblock<>[] then
       newflag='guide_lblock'
     else
       newflag=flag
     end
  end
  mdo_export(scs_m,name,newflag)
  //restore figure_style
  gg=xget('window')  // for bug in figure_style and winsid
  xset('window',0) // for bug in figure_style and winsid
  set('figure_style',bak)
  set('old_style',stripblanks(olds));
  xset('window',gg) // for bug in figure_style and winsid
endfunction
```

1.13 Export palette figure in eps file

- **Name:** dessin_pal
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.13.1 Calling Sequence

`dessin_pal(name,flag)`

1.13.2 Parameters

- **name :** string. path+name of the .cosf file
- **flag :** string. option to set the up/down and letf/right margin
  - **'html' :** to produce a figure for html man page
  - **'guide' :** to produce a figure for man page with paper format

1.13.3 File content

```
//dessin_pal
//fonction qui charge un fichier cosf dans une liste
//scs_m et qui exporte les données graphiques dans un fichier
//eps grâce à mdo_export
//Entrée name : chemin+nom de la palette
//              ex : name=MODNUM+'/macros/scicos_blocks/Tools.cosf'
function dessin_pal(name,flag)
  if fileinfo(name)<>[] then
   //load scicos variable and library
   bak=get('figure_style');
   set("figure_style","old");
   olds=get('old_style');
   set('old_style','on');

   //load scicos variable and library
   load SCI/macros/scicos/lib
   exec(loadpallibs,-1)
   %scicos_prob=%f;
   alreadyran=%f
   needcompile=4
   %zoom=1.8;
   colmap=xget('colormap');

   exec(name,-1)
   if flag=='html' then
    newflag='html_pal'
   else
    newflag=flag
   end
   mdo_export(scs_m,basename(name)+'_cosf',newflag)

   //restore figure_style
   gg=xget('window')  // for bug in figure_style and winsid
   xset('window',0) // for bug in figure_style and winsid
   set('figure_style',bak)
   set('old_style',stripblanks(olds));
   xset('window',gg) // for bug in figure_style and winsid
  else
   printf("Unable to find cosf file");
  end
endfunction
```

1.14 Export figure from a scs_m list in eps file

- **Name:** dessin_scs_m
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.14.1 Calling Sequence

`dessin_scs_m(scs_m,name,flag)`

1.14.2 Parameters

- **scs_m :** a main scicos data structure
- **name :** string. name of the figure
- **flag :** string. option to set the up/down and letf/right margin
  - **'html' :** to produce a figure for html man page
  - **'guide' :** to produce a figure for man page with paper format

1.14.3 File content

```
//dessin_scs_m
//fonction qui dessine le contenu graphique
//d'une structure scs_m dans un fichier .eps
//
//Entrée scs_m : structure de données scicos
//       name : nom du fichier produit
//       flag : 'html' ou 'guide'
function dessin_scs_m(scs_m,name,flag)
  //load scicos variable and library
  bak=get('figure_style');
  set("figure_style","old");
  olds=get('old_style');
  set('old_style','on');

  //load scicos variable and library
```

```
  load SCI/macros/scicos/lib
  exec(loadpallibs,-1)
  %scicos_prob=%f;
  alreadyran=%f
  needcompile=4
  %zoom=1.8;
  colmap=xget('colormap');

  mdo_export(scs_m,name,flag)

  //restore figure_style
  gg=xget('window')  // for bug in figure_style and winsid
  xset('window',0) // for bug in figure_style and winsid
  set('figure_style',bak);
  set('old_style',stripblanks(olds));
  xset('window',gg) // for bug in figure_style and winsid
endfunction
```

1.15 Export figure of scicos diagram in eps file

- **Name:** export_diagr
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.15.1 Calling Sequence

```
export_diagr(path,fileN,flag)
```

1.15.2 Parameters

- **path :** string. the path of the file to export
- **fileN :** string. the name of the file to export
- **flag :** string. set the format of the generated eps file
  - **'html' :** generate figure adapted to html file format
  - **'guide' or 'cosguide' :** for paper format (ps file)

1.15.3 Description

Load a scicos diagram in a scs_m list and call mdo_export to create eps file of the graphic data.

1.15.4 Example

```
export_diagr(MODNUM+'/scs_diagr/dyna/lorentz/','lorentz.cos','guide')
```

1.15.5 File content

```
//export_diagr
//fonction qui charge un fichier scicos dans une liste
//scs_m et qui exporte le contenu graphique dans un
//fichier grâce à la fonction mdo_export
//Entrée : path : le chemin du fchier à exporter
//         fileN : le nom du ficher à exporter
//         flag 'html'
//              'guide'
//              'cosguide'
function export_diagr(path,fileN,flag)
  //load scicos variable and library
  bak=get('figure_style');
  set("figure_style","old");
  olds=get('old_style');
  set('old_style','on');

  //load scicos variable and library
  load SCI/macros/scicos/lib
  exec(loadpallibs,-1)
  %scicos_prob=%f;
  alreadyran=%f
  needcompile=4
  %zoom=1.8;
  colmap=xget('colormap');

  load(path+fileN)
```

```
  name=basename(fileN);
  if strindex(name,'sblock_equiv')<>[] then
    if flag=='guide' then flag='sbeq_guide', end;
  end
  mdo_export(scs_m,name,flag);

  //restore figure_style
  gg=xget('window')  // for bug in figure_style and winsid
  xset('window',0) // for bug in figure_style and winsid
  set('figure_style',bak);
  set('old_style',stripblanks(olds));
  xset('window',gg) // for bug in figure_style and winsid
endfunction
```

1.15.6 See Also

- mdo_export - modified do_export function (Scilab Function)

1.16 Convert string to LaTex string

- **Name:** latexsubst
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.16.1 Calling Sequence

```
fields = latexsubst(fields)
```

1.16.2 Parameters

- **fields :** a matrix of character string.
- **fields :** a matrix of character string.

1.16.3 File content

```
//latexsubst
//Fonction qui effectue la conversion
//de caractères speciaux d'un tableau
//de chaîne de caractères pour un texte latex
//Entrée fields : tableau de chaîne de caractères
//Sortie fields : tableau de chaîne de caractères
function fields=latexsubst(fields)

  //rajout pour compatibilité
  //avec return_xml_desc3
  fields=strsubst(fields,'<VERB>','textbf{')
  fields=strsubst(fields,'</VERB>','}')
  fields=strsubst(fields,'<LINK>','textbf{')
  fields=strsubst(fields,'</LINK>','}')
  fields=strsubst(fields,'<VERBATIM><![CDATA[','\begin{verbatim}')
  fields=strsubst(fields,']]></VERBATIM>','\end{verbatim}')

  fields=strsubst(fields,'ö','\""o')
  fields=strsubst(fields,'_','\_')
  fields=strsubst(fields,'%','\%')
  fields=strsubst(fields,'&','\&')
  fields=strsubst(fields,'^','\^')
  fields=strsubst(fields,'<','$<$')
  fields=strsubst(fields,'>','$>$')
endfunction
```

1.17 Sort resulting figures of scilab simulation script and scicos diagram

- **Name:** make_scope_order
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.17.1 Calling Sequence

```
make_scope_order(path,name,ext)
```

1.17.2 Parameters

- **path :** add here the parameter description
- **name :** add here the parameter description

1.17.3 File content

```
//fonction qui trie des fichiers eps
//issus de la fonction scop_results
//en regardant dans un fichier SPECIALDESC
//Entrée : path : chemin du fichier SPECIALDESC
//         name : nom du fichier à traiter
//         ext : type d'extension du fichier à traiter
function make_scope_order(path,name,ext)
 //trouve l'ordre des scope dans SPECIALDESC
 tt=mgetl(path+'/SPECIALDESC');
 a=[];
 for i=1:size(tt,1)
   if strindex(tt(i),'scope_order')<>[] then
     sorder_equ=strindex(tt(i),'=')
     txt=part(tt(i),sorder_equ+1:length(tt(i)))
     a=evstr(txt)
     break
   end
 end

 //remet les fichiers dans le bon ordre
 if a<>[] then
   for j=1:size(a,1)
     f=cp_cmd+'./'+name+ext+'/'+name+'_scope_'+string(j)+'.eps ./'+name+ext+'/'+name+'_scope_'+string(j)+'_tmp.eps'
     unix_g(f)
   end

   for j=1:size(a,1)
     f=cp_cmd+'./'+name+ext+'/'+name+'_scope_'+string(j)+'_tmp.eps ./'+name+ext+'/'+name+'_scope_'+string(a(j))+'.eps'
     unix_g(f)
   end

   unix_g(rm_cmd+'./'+name+ext+'/*_tmp.eps")
 end

endfunction
```

1.18 Modified do_export function

- **Name:** mdo_export
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.18.1 Calling Sequence

`mdo_export(scs_m,fnameN,flag)`

1.18.2 Parameters

- **scs_m :** a scicos main data list
- **fnameN :** the name of the file to produce
- **flag :** set the type of margin
  - **'hmtl' :** to generate figure adapted to html file format
  - **'guide' or 'cosguide' :** for paper format (ps file)

1.18.3 Description

Export graphic datas of a scs_m list in eps file.

1.18.4 File content

```
//mdo_export
//fonction qui export le contenu graphique d'une liste
//scs_m dans un fichier .eps
//Entrée : scs_m  : liste de donnée scicos
//         fnameN : nom du ficher à produire
//         flag : html pour produire une figure pour mettre dans du html
//                guide pour du papier
//                cosguide pour du papier

function mdo_export(scs_m,fnameN,flag)
  //Créer un répertoire
  if fileinfo(fnameN+'/')==[] then unix_g("mkdir "+fnameN+"/"), end;

  //Teste le paramètre
  if fnameN==emptystr() then return;end
  fnameN=stripblanks(fnameN);
  ff=str2code(fnameN+'/'+fnameN);
  ff(find(ff==40|ff==53))=36;
  fnameN=code2str(ff);

  //récupère dimension
  rect=dig_bound(scs_m)
  wpar=scs_m.props.wpar
  wa=(rect(3)-rect(1))
  ha=(rect(4)-rect(2))
  if flag=='guide'|flag=='guide_lblock'  then
   %scicos_lr_margin=.2;
   %scicos_ud_margin=.1
  elseif flag=='cosguide' then
   %scicos_lr_margin=.005;
   %scicos_ud_margin=.05
  elseif flag=='html_diagr' then
   %scicos_lr_margin=.15;//.3;
   %scicos_ud_margin=.05//.4;
  elseif flag=='html_block'|flag=='html_lblock' then
   %scicos_lr_margin=.3;//.3;
   %scicos_ud_margin=.35//.4;
  elseif flag=='html_pal' then
   %scicos_lr_margin=.1;//.3;
   %scicos_ud_margin=.05//.4;
  elseif flag=='html' then
   %scicos_lr_margin=.05;//.3;
   %scicos_ud_margin=.05//.4;
  elseif flag=='sbeq_guide' then
   %scicos_lr_margin=.05;//.3;
   %scicos_ud_margin=.02//.4;
  else
   %scicos_lr_margin=.25;//.3;
   %scicos_ud_margin=.35//.4;
  end
  rect(1)=rect(1)-wa*%scicos_lr_margin
  rect(3)=rect(3)+wa*%scicos_lr_margin
  rect(2)=rect(2)-ha*%scicos_ud_margin
  rect(4)=rect(4)+ha*%scicos_ud_margin
  if flag=='html_lblock' then
     rect(2)=rect(2)-ha*%scicos_ud_margin
  elseif flag=='guide_lblock' then
     rect(2)=rect(2)-3.5*ha*%scicos_ud_margin
  elseif flag=='html_pal'
    rect(2)=rect(2)-1.2*ha*%scicos_ud_margin
  end
  wa=(rect(3)-rect(1))
  ha=(rect(4)-rect(2))
  wa=max(60,wa);
  ha=max(40,ha);
  //Initialise drivers graphique
  driver('Pos')
  set_posfig_dim(wa*%zoom/1.8,ha*%zoom/1.8)
  xinit(fnameN),
  xsetech(wrect=[0 0 1 1],frect=rect,arect=[0,0,0,0])
  options=scs_m.props.options
  cmap=options.Cmap
  for k=1:size(cmap,1)
    [mc,kk]=mini(abs(colmap-ones(size(colmap,1),1)*cmap(k,:))*[1;1;1])
    if mc>.0001 then
      colmap=[colmap;cmap(k,:)]
    end
  end
  xset('colormap',colmap)
  xset('pattern',default_color(0));
  //Dessine scs_m
  drawobjs(scs_m),
  set_posfig_dim(0,0),
  xend();

  //convertion finale avec BEpsf
  %scicos_landscape=0
  opt=""
  if %scicos_landscape then opt=" -landscape ";end
  if MSDOS then
    fnameN=pathconvert(fnameN,%f,%t,'w')
    comm=pathconvert(SCI+'\bin\BEpsf',%f,%f,'w')
    rep=unix_g(''"'+comm+'"' '+opt+fnameN)
  else
    rep=unix_g(SCI+'/bin/BEpsf '+opt+fnameN)
  end
  if rep<>[] then
    x_message(['Problem generating ps file.';..
               'perhaps directory not writable'] )
```

```
  end
  driver('Rec')
endfunction
```

1.18.5 See Also

- export_diagr - export figure of scicos diagram in eps file (Scilab Function)

1.19 Probe SPECIALDESC file and return captions of figure

- **Name:** return_capt
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.19.1 Calling Sequence

```
txt = return_capt(namef)
```

1.19.2 Parameters

- **namef :** string. path of the SPECIALDESC file
- **txt :** vector of strings. text of the captions

1.19.3 File content

```
//return_capt
//Fonction qui recherche la présence d'un fichier SPECIALDESC
//dans tex_path et qui retourne le titre des figures
//Entrée namef : nom de la page d'aide
function txt=return_capt(namef)
if fileinfo(namef+'/SPECIALDESC')<>[] then
 txt=[];
 i_equ1=0;i_equ2=0;
 tt=mgetl(namef+'/SPECIALDESC');
 for i=1:size(tt,1)
   if strindex(tt(i),'scope_caption')<>[] then
    i_equ1=i;//strindex(tt(i),'=');
   end
   if (i_equ1<>0 & i>=i_equ1) then
    if strindex(tt(i),']')<>[] then
      i_equ2=i;
      break
    end
   end
 end
 if i_equ1<>0 then
 for i=i_equ1:i_equ2
  txt=txt+tt(i);
 end
 if strindex(txt,'scope_caption')<>[] then
   txt=part(txt,strindex(txt,'=')+1:length(txt));
 end
 txt=evstr(txt);
 end
else
txt=[]
end
endfunction
```

1.20 Probe SPECIALDESC file and return info. on dialogue parameters for doc. generation

- **Name:** return_dial_param
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.20.1 Calling Sequence

```
txt = return_dial_param(name,flag)
```

1.20.2 Parameters

- **name :** string. the name of the interfacing function of a scicos block
- **flag :** string. a flag to set the size of the dialogue parameter window
  - **'html' :** to return size for html man page
  - **'guide' :** to return size for paper format
- **txt :** a list
  - **txt() :** integer. the id of the dialogue parameter window
  - **txt()() :**
    - **txt()(1) :** integer. the id of the parameter which display the sub-adjacent window
    - **txt()(2) :** string. the symbolic expression which display the sub-adjacent window
    - **txt()(3) :** string. the symbolic expression to execute for displaying the sub-adjacent window
    - **txt()(4) :** vector of string. the text of the description of parameters of the dialogue window

1.20.3 File content

```
//fonction qui retourne les info nécessaires contenues dans
//SPECIALDESC pour dessiner les fenetres de dialogues sous-
//adjacentes à la fenetre de dialogue principale
//Entrée : name : nom du fichier (ex :'CONVOLGEN_f')
//         flag : html ou guide
//Sortie : txt : une matrice vide si on ne trouve pas d'info ([])
//              une liste si on trouve des info
function txt=return_dial_param(name,flag)
 txt=[];
 if fileinfo(tex_path+lang+'/'+name+'/SPECIALDESC')<>[] then
   tt=mgetl(tex_path+lang+'/'+name+'/SPECIALDESC');
   a=[];
   b=[];
   j=1;
   for i=1:size(tt,1)
     if strindex(tt(i),'>><<')<>[] then
       a(j)=i;
     elseif strindex(tt(i),'<<>>')<>[] then
       b(j)=i;
       j=j+1;
     end
   end

   if a<>[]&b<>[] then
     txt_temp=[];

     //initialisation de la liste
     txt=list();
     for i=1:j-1
       txt(i)=list()
       txt(i)(1)=0
       txt(i)(2)=""
       txt(i)(3)=""
       txt(i)(4)=[]
       txt(i)(5)=[]
     end

     for i=1:(j-1)
       txt_temp=tt(a(i)+1:b(i)-1);
       execstr(txt_temp);

       if exists('num_param') then //numéro du param
         txt(i)(1)=num_param       //qui fait afficher
       end                      //la boite de dialogue

       if exists('val_param') then //expression symbolique
         txt(i)(2)=val_param       //qui fait afficher
       end                         //la boite de dialogue

       if exists('exp_param') then //expression symbolique
         txt(i)(3)=exp_param       //a executer pour afficher
       end                         //la boite de dialogue

       if exists('txt_param') then //le texte de description
         txt(i)(4)=txt_param       //des paramètres de la
       end                        //boite de dialogue

       if flag=='html' then
         if exists('size_dial_param_html') then //la taille de la
           txt(i)(5)=size_dial_param_html      //boite de dialogue
         end                                   //pour du html
       elseif flag=='guide' then
         if exists('size_dial_param_guide') then //la taille de la
           txt(i)(5)=size_dial_param_guide       //boite de dialogue
         end                                    //pour du papier
```

```
      end

      clear num_param;clear exp_param;clear txt_param;
      clear size_dial_param_html;clear size_dial_param_guide;
    end
  end
 end
endfunction
```

1.21 Return path of a scicos file presents in the diagr_all list

- **Name:** return_dir_cos
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.21.1 Calling Sequence

```
txt = return_dir_cos(name,diagr_all)
```

1.21.2 Parameters

- **name :** string. the name of the scicos diagram to find (extension free)
- **diagr_all :** vector of strings of size(n,2). this list is defined in load_generate_doc_function.sce
- **txt :** string. the path of the scicos diagram

1.21.3 File content

```
//Focntion qui retourne le répertoire d'un
//fichier de simulation en regardant dans la
//liste diagr_all
function txt=return_dir_cos(name,diagr_all)
 txt=[];
 for i=1:size(name,1)
  for j=1:size(diagr_all,1)
    if name(i,1)==diagr_all(j,2) then
      txt=[txt;diagr_all(j,1)]
    end
  end
 end
endfunction
```

1.22 Return the size of the graphic window of a dialog box

- **Name:** return_size_dial
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.22.1 Calling Sequence

```
txt = return_size_dial(name,flag)
```

1.22.2 Parameters

- **name :** string. the name of the interfacing function of a scicos block (extension free)
- **flag :** string. a flag to set the size of the dialogue parameter window
  - **'html' :** to return size for html man page
  - **'guide' :** to return size for paper format
- **txt :** string. the size of the dialogue parameter window

1.22.3 File content

```
//Fonction qui retourne la taille d'une fenêtre
//de dialogue spéciée dans un fichier SPECIALDESC
//Entrée name : nom du fichier (ex :'CONVOLGEN_f')
//       flag : html ou guide
//Sortie txt : nouvelle taille de la figure
function txt=return_size_dial(name,flag)
if fileinfo(tex_path+lang+'/'+name+'/SPECIALDESC')<>[] then
 txt=[];
 h=[];   //height
 w=[];   //width
 s=[];   //scale
 tt=mgetl(tex_path+lang+'/'+name+'/SPECIALDESC');
 for i=1:size(tt,1)
    if flag=='html' then
      if strindex(tt(i),'dial_width_html')<>[] then
         i_equ=strindex(tt(i),'=')
         txt='width='+part(tt(i),i_equ+1:length(tt(i)))
      end
    elseif flag=='guide' then
      if strindex(tt(i),'dial_width_guide')<>[] then
         i_equ=strindex(tt(i),'=')
         txt='[width='+part(tt(i),i_equ+1:length(tt(i)))+']'
      end
    end
 end

else
 txt=[];
end
endfunction
```

1.23 Return the size of the graphic window of a scicos diagram

- **Name:** return_size_scs_diagr
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.23.1 Calling Sequence

```
txt = return_size_scs_diagr(name,flag)
```

1.23.2 Parameters

- **name :** string. the name of the scicos diagram (extension free)
- **flag :** string. a flag to set the size of the scicos diagram
  - **'html' :** to return size for html man page
  - **'guide' :** to return size for paper format
- **txt :** string. the size of the scicos diagram

1.23.3 File content

```
//Fonction qui retourne la taille d'un diagramme
//scicos spéciée dans un fichier SPECIALDESC
//Entrée name : nom du fichier (ex :'CONVOLGEN_f')
//       flag : html ou guide
//       flag2 : 'sbeq' pour un super bloc équivalent
//Sortie txt : nouvelle taille de la figure
function txt=return_size_scs_diagr(name,flag,flag2)

if fileinfo(tex_path+lang+'/'+name+'/SPECIALDESC')<>[] then
 //verifie la présence du paramètre flag2
 [lsh,rsh]=argn(0)
 if rsh<3 then
   flag2=[];
 end

 tt_to_search='scs_diagr_height';
 if flag2=='sbeq' then
   tt_to_search=tt_to_search+'_sbeq';
 elseif flag2=='pal' then
   tt_to_search=tt_to_search+'_pal';
 elseif flag2=='lblock' then
   tt_to_search='is_lblock';
 end

 txt=[];
 h=[];   //height
```

```
 w=[];   //width
 s=[];   //scale
 tt=mgetl(tex_path+lang+'/'+name+'/SPECIALDESC');
 for i=1:size(tt,1)
    //html
    if flag=='html' then
      if strindex(tt(i),tt_to_search+'_html')<>[] then
         if flag2=='lblock' then
           txt=1;
         else
           i_equ=strindex(tt(i),'=')
           txt='height='+part(tt(i),i_equ+1:length(tt(i)))
         end
      end

    //guide
    elseif flag=='guide' then

      if strindex(tt(i),tt_to_search+'_guide')<>[] then
         if flag2=='lblock' then
            txt=1;
         else
            i_equ=strindex(tt(i),'=')
            txt='[height='+part(tt(i),i_equ+1:length(tt(i)))+']'
        end
      end
    end
 end

else
 txt=[];
end
endfunction
```

1.24 Return the size of the graphic window of a scicos diagram

- **Name:** return_size_scs_diagr2
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.24.1 Calling Sequence

```
txt = return_size_scs_diagr2(name,flag)
```

1.24.2 Parameters

- **name :** string. the name of the scicos diagram (extension free)
- **flag :** string. a flag to set the size of the scicos diagram
  - **'html' :** to return size for html man page
  - **'guide' :** to return size for paper format
- **txt :** string. the size of the scicos diagram

1.24.3 File content

```
//Fonction qui retourne la taille d'un diagramme
//scicos spéciée dans un fichier SPECIALDESC
//Entrée name : nom du fichier (ex :'CONVOLGEN_f')
//       flag : html ou guide
//Sortie txt : nouvelle taille de la figure
function txt=return_size_scs_diagr2(name,flag)
if fileinfo(tex_path+lang+'/'+name+'/SPECIALDESC')<>[] then
 txt=[];
 tt=mgetl(tex_path+lang+'/'+name+'/SPECIALDESC');
 for i=1:size(tt,1)
    if flag=='html' then
      if strindex(tt(i),'size_scs_diagr_html')<>[] then
        ierror=execstr(tt(i),'eercatch');
        if ierror<>0 then
          printf("Format error in SPECIALDESC\n");
          break
        else
          txt=size_scs_diagr_html;
        end
      end
    elseif flag=='guide' then
      if strindex(tt(i),'size_scs_diagr_guide')<>[] then
        ierror=execstr(tt(i),'eercatch');
        if ierror<>0 then
          printf("Format error in SPECIALDESC\n");
```

```
          break
        else
          txt=size_scs_diagr_guide;
        end
      end
    end
 end

else
 txt=[];
end
endfunction
```

1.25 Load, run simulation and export figures of a scs_m list

- **Name:** scop_results
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.25.1 Calling Sequence

```
i = scop_results(scs_m)
```

1.25.2 Parameters

- **scs_m :** a main scicos data structure
- **i :** integer. the number of figure display during the simulation

1.25.3 File content

```
//Fonction qui execute une liste scs_m
//et exporte les fenetres graphiques résultantes
//dans des fichiers eps
function i=scop_results(scs_m)
 //Switch to old_mode
 bak=get('figure_style');
 set("figure_style","old");

 //erase all graphics
 del_all_graphics()
 titlef=scs_m.props("title")(1);
 context=scs_m.props("context");
 execstr(context); //c'est dangereux cela
 scs_m.props.tf=Tfin;

 if fileinfo(tex_path+lang+'/'+titlef+'/sim_diagr.sce')<>[] then
   exec(tex_path+lang+'/'+titlef+'/sim_diagr.sce'); //attention
 else
   //ctxt=mlist('')
   ctxt=struct()
   //scs_m.props.context=[];
   Info=list();
   str='Info=scicos_simulate(scs_m,Info,ctxt)';
   ierror=execstr(str,'errcatch');
   if ierror<>0 then
     printf("\n******* Simulation problem *******\n")
   end
 end

 i=0
 while %t
  win=xget("window");
  if win==0 then
    xdel(win);
    win=xget("window");
    if win==0 then
      xdel(win);
      break
    end
  else
   i=i+1;
   //xbasimp(win,titlef+'_scope_'+string(i)+'.ps')
   xbasimp(win,titlef+'_scope_'+string(i))
   //unix_g(SCI+'/bin/BEpsf '+titlef+'_scope_'+string(i)+'.ps'+'.'+string(win))
   unix_g(SCI+'/bin/BEpsf '+titlef+'_scope_'+string(i)+'.'+string(win))
   xdel(win);
  end
 end

 //Retrieve figure_style
 gg=xget('window')  // for bug in figure_style and winsid
 xset('window',0) // for bug in figure_style and winsid
 set('figure_style',bak)
 xset('window',gg) // for bug in figure_style and winsid
endfunction
```

1.26 Load, run simulation and export figures of a scicos diagram

- **Name:** scop_results_cos
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.26.1 Calling Sequence

```
number_scop = scop_results_cos(name)
```

1.26.2 Parameters

- **name :** string. path+name of the scicos diagram file
- **number_scop :** integer. the number of figure display during the simulation

1.26.3 File content

```
//Fonction qui charge un fichier scicos
//dans une liste scs_m et qui execute la simulation
//et exporte les fenetres graphiques résultantes
//dans des fichiers eps grâce à scop_results
//Entrée name : chemin+nom du fichier cos
//              ex : name=MODNUM+'/scs_diagr/dyna/chua/chua.cos'
//Sortie num_cos : nombre de scope eporter
function number_scop=scop_results_cos(name)

 if fileinfo(name)<>[] then
   load(name)
   number_scop=scop_results(scs_m)
 else
  printf("Unable to find %s file\n",name);
  number_scop=[];
 end

endfunction
```

1.27 Load, run simulation and export figures of a scilab simulation script

- **Name:** scop_results_sim
- **Library:** gen_doc_util - Library of utilities functions for documentation generation

1.27.1 Calling Sequence

```
i = scop_results_sim(sim_name)
```

1.27.2 Parameters

- **sim_name :** string. path+name of the scilab simulation script file
- **i :** integer. the number of figure display during the simulation

1.27.3 File content

```
//Fonction qui execute une un script de simulation
//scilab et exporte les fenetres graphiques résultantes
//dans des fichiers eps
//Entrée sim_name : path+nom du fichier script
function i=scop_results_sim(sim_name)
 //Switch to old_mode
 bak=get('figure_style');
 set("figure_style","old");

 //erase all graphics
 del_all_graphics()

 titlef=basename(sim_name);

 str='exec('''+sim_name+''',-1);';
 ierror=execstr(str,'errcatch');
 if ierror<>0 then
     printf("\n******* Simulation problem *******\n")
```

```
 end

 i=0
 while %t
  win=xget("window");
  if win==0 then
    xdel(win);
    win=xget("window");
    if win==0 then
      xdel(win);
      break
    end
  else
   i=i+1;
   //xbasimp(win,titlef+'_scope_'+string(i)+'.ps')
   xbasimp(win,titlef+'_scope_'+string(i))
   //unix_g(SCI+'/bin/BEpsf '+titlef+'_scope_'+string(i)+'.ps'+'.'+string(win))
   unix_g(SCI+'/bin/BEpsf '+titlef+'_scope_'+string(i)+'.'+string(win))
   xdel(win);
  end
 end

 //Retrieve figure_style
 gg=xget('window')  // for bug in figure_style and winsid
 xset('window',0) // for bug in figure_style and winsid
 set('figure_style',bak)
 xset('window',gg) // for bug in figure_style and winsid
endfunction
```

Chapter 2

Main library of the documentation generator

2.1 Export xml paragraph to single data file

- **Name:** export_file_to_data
- **Library:** generate_doc - Main library of the documentation generator

2.1.1 Calling Sequence

```
export_file_to_data(rep_xml,flag,rep_data)
```

2.1.2 Parameters

- **rep_xml :** string. directory of the XML files
- **flag :** string. flag to set the type of data to export
  - **'param' :** export parameters section in MODNUM_data_param
  - **'sdesc' :** export short description in MODNUM_data_sdesc
  - **'see_also' :** export see also section in MODNUM_data_see_also
  - **'authors' :** export authors section in MODNUM_data_authors
  - **'ex' :** export exemples section in MODNUM_data_ex
  - **'desc' :** export description section in MODNUM_data_desc
  - **'biblio' :** export bibliography section in MODNUM_data_biblio
  - **'used_func' :** export used function section in MODNUM_data_used_func
  - **'SPECIALDESC' :** export SPECIALDESC files in MODNUM_data_SPECIALDESC
  - **'all' :** export all sections in data files
- **rep_data :** string. directory of the _data files

2.1.3 File content

```
//export_file_to_data
// flag un vecteur de chaîne de caractères
// 'param'
// 'sdesc'
// 'see_also'
// 'authors'
// 'ex'
// 'desc'
// 'biblio'
// 'used_func'
// 'SPECIALDESC'
// 'all'
// 'call_seq'
// rep_xml :  répertoire de stockage des fichiers
//            xml (ex:rep_xml=xml_path)
// rep_data : répertoire de stockage des fichiers
```

```
//           de données
function export_file_to_data(rep_xml,flag,rep_data)

 //Verifie cohérence des paramètres
 [lsh,rsh]=argn();
 if rsh<3 then
   rep_data=man_path
 else
   rep_data=pathconvert(rep_data,%t)
 end;
 if flag==’all’ then
  flag=[’param’;’sdesc’;’see_also’;
        ’authors’;’ex’;’desc’;’used_func’;
        ’biblio’;’SPECIALDESC’;’call_seq’]
 end

 //def des noms de fichiers de donées
 file_param=’MODNUM_data_param’;
 file_sdesc=’MODNUM_data_sdesc’;
 file_see_also=’MODNUM_data_see_also’;
 file_authors=’MODNUM_data_authors’;
 file_ex=’MODNUM_data_ex’;
 file_desc=’MODNUM_data_desc’;
 file_call_seq=’MODNUM_data_call_seq’;
 file_used_func=’MODNUM_data_used_func’;
 file_biblio=’MODNUM_data_biblio’;
 file_spec_desc=’MODNUM_SPECIALDESC’;

 //cherche tous les fichiers xml présents
 //dans xml_path
 lisf=return_ext_file_in_dir(tt_ml,rep_xml,’.xml’);


 for z=1:size(flag,1)
  flagn=flag(z);

  //param
  if flagn==’param’ then
    tt=[];
    for ij=1:size(lisf,1)
      txt_list=return_xml_param3(rep_xml+lisf(ij,1));
      tt=[tt;’<FILE ’+lisf(ij,1)+’>’];
      tt_param=[];
      if txt_list<>[] then
        //tt=[tt;’<FILE ’+lisf(i,1)+’>’];
        //trouve le nbre de paramètres
        nb_param=0;
        for l=1:size(txt_list)
          nb_param=nb_param+size(txt_list(l)(2),1);
        end
        //initialise une nouvelle liste
        n=list()
        for i=1:nb_param
           n(i)=list();
           n(i)(1)=0;
           n(i)(2)=""
           n(i)(3)=""
        end
        //Creation d’un nouvelle liste
        for i=1:size(txt_list)
           for j=1:size(txt_list(i)(1),1)
              n(txt_list(i)(1)(j,1))(1)=i;
              n(txt_list(i)(1)(j,1))(2)=txt_list(i)(2)(j,1);
              n(txt_list(i)(1)(j,1))(3)=txt_list(i)(2)(j,2);
           end
        end
        //pause
        pre_i=0; //indentation précédente
        for i=1:size(n)
           dif_i=n(i)(1)-pre_i;
           if dif_i<>0 then
              if dif_i>0 then
                 for j=1:dif_i
                    tt_param=[tt_param;’<INDENT>’];
                 end
              elseif dif_i<0 then
                 for j=-1:-1:dif_i
                    tt_param=[tt_param;’</INDENT>’];
                 end
              end
           end
           pre_i=n(i)(1);
           tt_param=[tt_param;
                     ’TITLE=’+n(i)(2);
                     ’DESC=’+n(i)(3)];
        end
        dif_i=-pre_i;
        if dif_i<0 then
           for j=-1:-1:dif_i
              tt_param=[tt_param;’</INDENT>’];
           end
        end
      end
      tt=[tt;tt_param;’</FILE ’+lisf(ij,1)+’>’;’’];
    end
    if tt<>[] then
      printf("Export xml parameters... ");
      mputl(tt,rep_data+file_param);
      printf("Done\n")
    else
      printf("No files found with parameters\n")
```

```
  end

//param_old
elseif flagn==’param_old’ then
  filen=’MODNUM_xml_param_old’
  printf("Export xml parameters... ");
  tt=[];
  for i=1:size(lisf,1)
    tt=[tt;’<FILE ’+lisf(i,1)+’>’];
    txt=return_xml_param(rep_xml+lisf(i,1));
    tt=[tt;string(size(txt,1))];
    tt=[tt;txt(:,1);txt(:,2)];
    tt=[tt;’</FILE ’+lisf(i,1)+’>’;’’];
  end
  mputl(tt,rep_data+filen);
  printf("Done\n")

//sdesc
elseif flagn==’sdesc’
  tt=[];
  for i=1:size(lisf,1)
    txt=return_xml_sdesc(rep_xml+lisf(i,1));
    if txt<>[] then
      tt=[tt;’<FILE ’+lisf(i,1)+’>’];
      tt=[tt;txt]
      tt=[tt;’</FILE ’+lisf(i,1)+’>’;’’];
    end
  end
  if tt<>[] then
    printf("Export xml short descriptions... ");
    mputl(tt,rep_data+file_sdesc);
    printf("Done\n")
  else
    printf("No files found with short description\n")
  end

//see_also
elseif flagn==’see_also’
  tt=[];
  for i=1:size(lisf,1)
    txt=return_xml_see_also(rep_xml+lisf(i,1));
    if txt<>[] then
      tt=[tt;’<FILE ’+lisf(i,1)+’>’];
      tt=[tt;txt]
      tt=[tt;’</FILE ’+lisf(i,1)+’>’;’’];
    end
  end
  if tt<>[] then
    printf("Export xml see_also... ");
    mputl(tt,rep_data+file_see_also);
    printf("Done\n")
  else
    printf("No files found with see also paragraph\n")
  end

//authors
elseif flagn==’authors’ then
  tt=[];
  for i=1:size(lisf,1)
    txt=return_xml_authors2(rep_xml+lisf(i,1));
    if txt<>[] then
      tt=[tt;’<FILE ’+lisf(i,1)+’>’];
      tt=[tt;string(size(txt,1))];
      tt=[tt;txt(:,1);txt(:,2)];
      tt=[tt;’</FILE ’+lisf(i,1)+’>’;’’];
    end
  end
  if tt<>[] then
    printf("Export xml authors... ");
    mputl(tt,rep_data+file_authors);
    printf("Done\n")
  else
    printf("No files found with authors paragraph\n")
  end

//used_func
elseif flagn==’used_func’ then
  tt=[];
  for i=1:size(lisf,1)
    txt=return_xml_used_func(rep_xml+lisf(i,1));
    if txt<>[] then
      tt=[tt;’<FILE ’+lisf(i,1)+’>’];
      tt=[tt;txt(:,1)];
      tt=[tt;’</FILE ’+lisf(i,1)+’>’;’’];
    end
  end
  if tt<>[] then
    printf("Export xml used functions... ");
    mputl(tt,rep_data+file_used_func);
    printf("Done\n")
  else
    printf("No files found with used function paragraph\n")
  end

//biblio
elseif flagn==’biblio’ then
  tt=[];
  for i=1:size(lisf,1)
    name=basename(lisf(i,1))
    if fileinfo(tex_path+lang+’/’+name+’/’+name+’_bib.tex’)<>[]
      txt=mgetl(tex_path+lang+’/’+name+’/’+name+’_bib.tex’);
```

```
      tt=[tt;'<FILE '+lisf(i,1)+'>'];
      tt=[tt;'<LaTeX>';txt(:,1)];
      tt=[tt;'</FILE '+lisf(i,1)+'>';''];
    else
      txt=return_xml_biblio(rep_xml+lisf(i,1));
      if txt<>[] then
        tt=[tt;'<FILE '+lisf(i,1)+'>'];
        tt=[tt;txt(:,1)];
        tt=[tt;'</FILE '+lisf(i,1)+'>';''];
      end
    end
  end
  if tt<>[] then
    printf("Export xml & latex bibliography... ");
    mputl(tt,rep_data+file_biblio);
    printf("Done\n")
  else
    printf("No files found with bibliography\n")
  end

//ex
elseif flagn=='ex' then
  tt=[];
  for i=1:size(lisf,1)
    name=basename(lisf(i,1))
    if fileinfo(tex_path+lang+'/'+name+'/'+name+'_ex.tex')<>[]
      txt=mgetl(tex_path+lang+'/'+name+'/'+name+'_ex.tex');
      tt=[tt;'<FILE '+lisf(i,1)+'>'];
      tt=[tt;'<LaTeX>';txt(:,1)];
      tt=[tt;'</FILE '+lisf(i,1)+'>';''];
    else
      txt=return_xml_ex(rep_xml+lisf(i,1));
      if txt<>[] then
        tt=[tt;'<FILE '+lisf(i,1)+'>'];
        tt=[tt;txt(:,1)];
        tt=[tt;'</FILE '+lisf(i,1)+'>';''];
      end
    end
  end
  if tt<>[] then
    printf("Export xml & latex example... ");
    mputl(tt,rep_data+file_ex);
    printf("Done\n")
  else
    printf("No files found with exemples\n")
  end

//desc
elseif flagn=='desc' then
  tt=[];
  for i=1:size(lisf,1)
    name=basename(lisf(i,1))
    if fileinfo(tex_path+lang+'/'+name+'/'+name+'_long.tex')<>[]
      tt=[tt;'<FILE '+lisf(i,1)+'>'];
      txt=mgetl(tex_path+lang+'/'+name+'/'+name+'_long.tex');
      tt=[tt;'<LaTeX>';txt(:,1)];
      tt=[tt;'</FILE '+lisf(i,1)+'>';''];
    else
      txt_list=return_xml_desc3(rep_xml+lisf(i,1));
      tt=[tt;'<FILE '+lisf(i,1)+'>'];
      if txt_list<>list(list()) then
        tt=[tt;string(size(txt_list))]
        for a=1:size(txt_list)
          tt=[tt;'INDENT='+string(txt_list(a)(1))]
          tt=[tt;'TITLE='+string(txt_list(a)(2))]
          tt=[tt;'<TEXT>';txt_list(a)(3);'</TEXT>']
        end
      end
      tt=[tt;'</FILE '+lisf(i,1)+'>';''];
    end
  end
  if tt<>[] then
    printf("Export xml & latex long description... ");
    mputl(tt,rep_data+file_desc);
    printf("Done\n")
  else
    printf("No files found with long descriptions\n")
  end

//SPECIALDESC
elseif flagn=='SPECIALDESC' then
  printf("Export latex special description... ");
  tt=[];
  for i=1:size(lisf,1)
    tt=[tt;'<FILE '+lisf(i,1)+'>'];
    name=basename(lisf(i,1))
    if fileinfo(tex_path+lang+'/'+name)<>[] then
      if fileinfo(tex_path+lang+'/'+name+'/SPECIALDESC')<>[] then
        txt=mgetl(tex_path+lang+'/'+name+'/SPECIALDESC');
        tt=[tt;txt(:,1)];
      end
    end
    tt=[tt;'</FILE '+lisf(i,1)+'>';''];
  end
  mputl(tt,rep_data+file_spec_desc);
  printf("Done\n")

//call_seq
elseif flagn=='call_seq'
  tt=[];
  for i=1:size(lisf,1)
```

```
      txt=return_xml_call_seq(rep_xml+lisf(i,1));
      if txt<>[] then
        tt=[tt;'<FILE '+lisf(i,1)+'>'];
        tt=[tt;txt]
        tt=[tt;'</FILE '+lisf(i,1)+'>';''];
      end
    end
    if tt<>[] then
      printf("Export xml call_seq... ");
      mputl(tt,rep_data+file_call_seq);
      printf("Done\n")
    else
      printf("No files found with see also paragraph\n")
    end

  end
 end

endfunction
```

2.2 Create auxiliary tex file for the documentation of the toolbox

- **Name:** generate_aux_tex_file
- **Library:** generate_doc - Main library of the documentation generator

2.2.1 Calling Sequence

```
lisf = generate_aux_tex_file(lisf,flag,typdoc,lang)
```

2.2.2 Parameters

- **lisf :** vector of strings. XML file names (extension free)
- **flag :** string. set the type of man page
  - **'block' :** for interfacing function of scicos block
  - **'pal' :** for a palette (.cosf file)
  - **'diagr' :** for a scicos diagram (.cos file)
  - **'scilib' :** for a library of scilab macros
  - **'sci' :** for a scilab macro.
  - **'rout' :** for computational routine
  - **'sim' :** for scilab simulation script (_sim.sce file)
  - **'sce' :** for scilab script (.sce file)
- **typdoc :** string. set the type of tex files to produce
  - **'html' :** to produce tex file for html format
  - **'guide' :** to produce tex for file paper format
- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page

2.2.3 File content

```
//generate_aux_tex_file
//fonction qui génère les fichiers tex auxiliaires nécessaires
//au fichier final tex de documentation dans un répertoire spécifique.
//Dans un premier temps, elle teste la présence d'un fichier .xml correspondant
//au(x) fichiers traités
//
//Une fois le fichier xml trouvé ou crée, elle génère les fichiers tex grâce
//à la fonction xml2tex puis elle regarde la présence d'un répertoire
//man/tex (tex_path) correspondant au(x) fichier(s) traité(s).
//Si un tel répertoire existe alors elle teste la présence
//d'un fichier Makefile, et s'il existe alors elle éxecute
//la tâche 'all'.
//Dans ces deux cas elle recopie l'intégralité des fichiers et
//répertoires dans le répertoire spécifique.
//Si une tâche 'make all' est exécutée alors à l'issue
//de la copie, elle exécute la tâche 'clean' dans le répertoire man/tex.
//
//Entrée : lisf : un vecteur de chaîne de caratères de taille n,1
//                qui contient le nom de fichier xml à traiter
//                (sans extension).
//         flag : 'block' pour une fonction d'interface scicos
//                'pal' pour un fichier palette scicos (cosf)
//                'diagr' pour un diagramme de simulation scicos
//                'scilib' pour une librairie de fonctions scilab
//                'sci' pour une fonction scilab.
//                'rout' pour une routine bas niveau
//                'sim' pour un script de simulation scilab
//                'sce' pour un script scilab
//       typdoc : 'html' (default) pour des pages d'aide html
//                'guide' pour des pages d'aides papier
//       lang   : 'eng' (default) pour de l'anglais
//                'fr' pour du français
//Sortie : lisf : une matrice de chaîne de caratères de taile n,2
//                qui contient le nom de fichier xml traité et sa
//                description courte correspondante.
//
function [lisf]=generate_aux_tex_file(lisf,flag,typdoc,lang)

[lsh,rsh]=argn(0)
if rsh<4 then
  if ~exists('lang') then
    lang='eng'
  elseif lang<>'eng' & lang<>'fr' then
    lang='eng'
  end
end
if rsh<3 then
  typdoc='html'
end

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;

for i=1:size(lisf,1)
  if fileinfo(xml_path+lang+'/'+lisf(i,1)+'.xml')==[] then
     printf("%s.xml not found.\n",lisf(i,1))
     printf("Generate an empty xml file... ");
     txt=generate_xml(lisf(i,1),flag)
     mputl(txt,xml_path+lang+'/'+lisf(i,1)+'.xml')
     printf("Done\n");
  else
     printf("%s.xml found.\n",lisf(i,1))
  end

  if flag=='block'|flag=='pal'|flag=='diagr'|flag=='scilib'|flag=='sci'|flag=='rout'|flag=='sim'|flag=='sce' then
   xml2tex(lisf(i,1),flag,typdoc);
  else
   printf("flag %s is not supported\n",flag);
   abort
  end

  lisf(i,2)=return_xml_sdesc(xml_path+lang+'/'+lisf(i,1)+'.xml');
  lisf(i,3)=return_xml_type(xml_path+lang+'/'+lisf(i,1)+'.xml');

  select flag
    case 'block' then
     ext=''
    case 'pal' then
     ext='_cosf'
    case 'diagr' then
     ext='_cos'
    case 'scilib' then
     ext='_scilib'
    case 'sci' then
     ext='_sci'
    case 'rout' then
     ext='_rout'
    case 'sim' then
     ext=''
    case 'sce' then
     ext='_sce'
  end

  //For block  : generate _dial_box.tex, _sblock_equiv.text
  if flag=='block' then
    ////////////////////////
    //Create  .eps
```

```
//////////////////////
//SPECIALDESC
l_blk=return_size_scs_diagr(lisf(i,1),typdoc,'lblock')
dessin_block(lisf(i,1),typdoc,l_blk)

//////////////////////
//Create  _blocks.tex
//////////////////////
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'.eps')<>[] then //figure block
  txt=['\begin{center}'
       '\epsfig{file='+lisf(i,1)+'.eps,width=90.00pt}'
       '\end{center}']
end
mputl(txt,'./'+lisf(i,1)+'/'+lisf(i,1)+'_blocks.tex');
//////////////////////
//Create  _dial_box.tex
//////////////////////
if with_gui then //Do a capture (need X session)
  ////////////////////////////////
  load SCI/macros/scicos/lib
  exec(loadpallibs,-1)
  %scicos_prob=%f;
  alreadyran=%f
  needcompile=4
  ////////////////////////////////
  prot=funcprot();
  funcprot(0);
  getvalue=mtk_getvalue
  EvalStr_fun(' set title """" ')
  funcprot(prot);
  ierror=execstr('blk='+lisf(i,1)+'(''define'')','errcatch')
  if ierror<>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(lisf(i,1))
    fct=[]
    return
  end
  ierror=execstr('blk='+lisf(i,1)+'(''set'',blk)','errcatch')
  if ierror <>0 then
    x_message(['Error in GUI function';lasterror()] )
    disp(lisf(i,1))
    fct=[]
    return
  end
  //tempo
  time1=getdate();time2=time1;
  while etime(time2,time1)<0.2, time2=getdate(); end;
  //récupère titre de la fenêtre
  title=GetVar_fun('title');
  if title<>"" then
   tt=unix_g('xwd -name ''Set Block properties'' | convert - ./'+lisf(i,1)+'/'+lisf(i,1)+'_gui.eps');

   if tt==[] then
     txt=['\begin{figure}'
          '\begin{center}'
          '\epsfig{file='+lisf(i,1)+'_gui.eps,width=300pt}'
          '\end{center}'
          '\end{figure}']
     if typdoc=='guide' then
       txt(1)=['\begin{figure}[!h]']
     end
      //SPECIALDESC
      size_dial=return_size_dial(lisf(i,1),typdoc)
      if size_dial<>[] then
        txt=strsubst(txt,'width=300pt',size_dial)
      end
   else
     txt=[];
   end
   EvalStr_fun('catch {destroy $w}')
  else
    txt=[]
  end
else //Create a table
  title=latexsubst(return_desc_block(lisf(i,1)))
  ini=return_ini_block(lisf(i,1));
  ini=latexsubst(ini)
  lables=return_lables_block(lisf(i,1))
  lables=latexsubst(lables)
  if ini<>[]&lables<>[] then
   mat=[]
   for j=1:size(ini,2)
     mat=[mat;lables(j)+' & '+ini(j) + '\\']
   end
   txt=['\documentclass[11pt]{article}'
        '\usepackage{times,amsmath,amssymb}'
        '\pagestyle{empty}'
        '\usepackage{color}'
        '\pagecolor{white}'
        '\begin{document}'
        '\begin{center}'
        '\begin{tabular}{|c|}'
        '\hline'
        title+'\\'
        '\hline'
        '\begin{tabular}{c|c}'
        mat
       '\end{tabular} \\'
       '\hline'
       '\end{tabular}'
       '\end{center}'
```

```
          '\end{document}']
          mputl(txt,'.//'+lisf(i,1)+'//'+lisf(i,1)+'_dial_box.tex');
          repaa=pwd()
          chdir('.//'+lisf(i,1));
          unix_g(latex_cmd+lisf(i,1)+'_dial_box.tex');
          unix_g(dvips_cmd+lisf(i,1)+'_dial_box.dvi');
          chdir(repaa)
          txt=['\begin{center}'
               '\epsfig{file='+lisf(i,1)+'_dial_box.ps}'
               '\end{center}']
      else
       txt=[]
      end
    end
    if txt<>[] then
      mputl(txt,'.//'+lisf(i,1)+'//'+lisf(i,1)+'_dial_box.tex');
    end

    ////////////////////////
    //Create  _param.tex
    ////////////////////////
    txt_spec_param=return_dial_param(lisf(i,1),typdoc);
    if txt_spec_param<>[] then
      nb_new_dial=size(txt_spec_param)

      //Ecrit fichier _param.tex
      txt=return_xml_param2(xml_path+lang+'/'+lisf(i,1)+'.xml');
      if txt<>[] then
        txt=latexsubst(txt)
        tt=['\begin{itemize}'];
        for l=1:size(txt,'r')
          if typdoc=='html' then
            tt=[tt;'\item{\textbf{'+txt(l,1)+':}}\linebreak'];
          else
            tt=[tt;'\item{\textbf{'+txt(l,1)+':}}\\'];
          end
          tt=[tt;txt(l,2);''];
          for j=1:nb_new_dial
             if l==txt_spec_param(j)(1) then
               tt=[tt;'If set '+txt_spec_param(j)(2)];
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
              if with_gui then //Do a capture (need X session)
               ////////////////////////////
               load SCI/macros/scicos/lib
               exec(loadpallibs,-1)
               %scicos_prob=%f;
               alreadyran=%f
               needcompile=4
               ////////////////////////////
               prot=funcprot();
               funcprot(0);
               getvalue=mtk_getvalue2
               EvalStr_fun(' set title """" ')
               funcprot(prot);
               ierror=execstr('blk='+lisf(i,1)+'(''define'')','errcatch')
               if ierror<>0 then
                 x_message(['Error in GUI function';lasterror()] )
                 disp(lisf(i,1))
                 fct=[]
                 return
               end
               execstr(txt_spec_param(j)(3))
               ierror=execstr('blk='+lisf(i,1)+'(''set'',blk)','errcatch')
               if ierror <>0 then
                 x_message(['Error in GUI function';lasterror()] )
                 disp(lisf(i,1))
                 fct=[]
                 return
               end
               //tempo
               time1=getdate();time2=time1;
               while etime(time2,time1)<0.2, time2=getdate(); end;
                 //récupère titre de la fenêtre
                 title=GetVar_fun('title');
                 if title<>"" then
                   name_fic_dial=lisf(i,1)+'_'+string(l)+'_'+string(j);
                   titi=unix_g('xwd -name ''Set Block properties'' | convert - ./'+lisf(i,1)+'/'+name_fic_dial+'_gui.eps');
                   if titi==[] then
                     txt2=['\begin{figure}'
                           '\begin{center}'
                           '\epsfig{file='+name_fic_dial+'_gui.eps,width=300pt}'
                           '\end{center}'
                           '\end{figure}']
                     if typdoc=='guide' then
                        txt2(1)=['\begin{figure}[!h]']
                     end
                     if txt_spec_param(j)(5)<>[] then
                       txt2=strsubst(txt2,'width=300pt',txt_spec_param(j)(5))
                     end
                   else
                     txt2=[];
                   end
                   EvalStr_fun('catch {destroy $w}')
                 else
                   txt2=[]
                 end
               else //Create a table
                 title=latexsubst(return_desc_block2(lisf(i,1),txt_spec_param(j)(3)));
                 ini=return_ini_block2(lisf(i,1),txt_spec_param(j)(3));
                 ini=latexsubst(ini);
```

```
                lables=return_lables_block2(lisf(i,1),txt_spec_param(j)(3));
                lables=latexsubst(lables);
                typ=return_typ_block2(lisf(i,1),txt_spec_param(j)(3));
                if ini<>[]&lables<>[] then
                  mat=[];
                  for e=1:size(ini,2)
                    mat=[mat;lables(e)+' & '+ini(e) + '\\'];
                  end
                  txt2=['\documentclass[11pt]{article}'
                        '\usepackage{times,amsmath,amssymb}'
                        '\pagestyle{empty}'
                        '\usepackage{color}'
                        '\pagecolor{white}'
                        '\begin{document}'
                        '\begin{center}'
                        '\begin{tabular}{|c|}'
                        '\hline'
                        title+'\\'
                        '\hline'
                        '\begin{tabular}{c|c}'
                        mat
                        '\end{tabular} \\'
                        '\hline'
                        '\end{tabular}'
                        '\end{center}'
                        '\end{document}'];
                  name_fic_dial=lisf(i,1)+'_'+string(l)+'_'+string(j);
                  mputl(txt2,'./'+lisf(i,1)+'/'+name_fic_dial+'_dial_box.tex');
                  repaa=pwd();
                  chdir('./'+lisf(i,1));
                  unix_g(latex_cmd+name_fic_dial+'_dial_box.tex');
                  unix_g(dvips_cmd+name_fic_dial+'_dial_box.dvi');
                  chdir(repaa);
                  txt2=['\begin{center}'
                        '\epsfig{file='+name_fic_dial+'_dial_box.ps}'
                        '\end{center}'];
                else
                  txt2=[];
                end
              end
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
              tt=[tt;txt2]
              lables=return_lables_block2(lisf(i,1),txt_spec_param(j)(3));
              lables=latexsubst(lables);
              typ=return_typ_block2(lisf(i,1),txt_spec_param(j)(3));
              tt=[tt;'';'\begin{itemize}']
              for e=1:size(lables,1)
                 if typdoc=='html' then
                    tt=[tt;'\item{\textbf{'+lables(e,1)+':}}\linebreak'];
                 else
                    tt=[tt;'\item{\textbf{'+lables(e,1)+':}}\\'];
                 end
               if lang=='eng'
                  tt=[tt;'Type '+typ(e,1)+' of size '+typ(e,2)+'. '+txt_spec_param(j)(4)(e);'']
               elseif lang=='fr'
                  tt=[tt;'Type '+typ(e,1)+' de taille '+typ(e,2)+'. '+txt_spec_param(j)(4)(e);'']
               end
              end
              tt=[tt;'\end{itemize}']
            end
          end
        end
      end
      tt=[tt;'\end{itemize}'];
      mputl(tt,'./'+lisf(i,1)+'/'+lisf(i,1)+'_param.tex');
      clear nb_new_dial;
    end
    clear txt_spec_param;

    ////////////////////////
    //Create  _def_prop.tex
    ////////////////////////
    prop=return_prop_block(lisf(i,1),lang);
    if lang=='fr' then
    txt=['\begin{itemize}'
         '\item \textbf{toujours actif:} '+prop(1)
         '\item \textbf{direct-feedthrough:} '+prop(2)
         '\item \textbf{détection de passage à zéro:} '+prop(3)
         '\item \textbf{mode:} '+prop(4)
         '\item \textbf{nombre/taille des entrées régulières:} '+prop(5)
         '\item \textbf{nombre/taille des sorties sorties régulières:} '+prop(6)
         '\item \textbf{nombre/taille des entrées évènementielles:} '+prop(7)
         '\item \textbf{nombre/taille des sorties évènementielles:} '+prop(8)
         '\item \textbf{possède un état continu:} '+prop(9)
         '\item \textbf{possède un état discret:} '+prop(10)
         '\item \textbf{nom de la fonction de calcul:} {\em '+latexsubst(prop(12))+'}'
         '\end{itemize}'
        ]
    else
    txt=['\begin{itemize}'
         '\item \textbf{always active:} '+prop(1)
         '\item \textbf{direct-feedthrough:} '+prop(2)
         '\item \textbf{zero-crossing:} '+prop(3)
         '\item \textbf{mode:} '+prop(4)
         '\item \textbf{number/sizes of inputs:} '+prop(5)
         '\item \textbf{number/sizes of outputs:} '+prop(6)
         '\item \textbf{number/sizes of activation inputs:} '+prop(7)
         '\item \textbf{number/sizes of activation outputs:} '+prop(8)
         '\item \textbf{continuous-time state:} '+prop(9)
         '\item \textbf{discrete-time state:} '+prop(10)
```

```
        '\item \textbf{name of computational function:} {\em '+latexsubst(prop(12))+'}'
        '\end{itemize}'
       ]
    end
    mputl(txt,'./'+lisf(i,1)+'/'+lisf(i,1)+'_def_prop.tex');

   ////////////////////////////
   //Create  _sblock_equiv.tex
   ////////////////////////////
   if fileinfo(sbeq_path+lisf(i,1)+'_sblock_equiv.cos')<>[] then
     printf("Super block '+lisf(i,1)+' found\n")
     export_diagr(sbeq_path,lisf(i,1)+'_sblock_equiv.cos',typdoc);
     unix_g("mv "+lisf(i,1)+"_sblock_equiv/"+lisf(i,1)+"_sblock_equiv.eps ./"+lisf(i,1))
     unix_g("rm -fr "+lisf(i,1)+"_sblock_equiv/");
     txt=['\begin{center}'
          '\epsfig{file='+lisf(i,1)+'_sblock_equiv.eps,width=400.00pt}'
          '\end{center}']
     //SPECIALDESC
     size_diagr=return_size_scs_diagr(lisf(i,1),typdoc,'sbeq')
     size_diagr=strsubst(size_diagr,'[','');  //for compatibility with \includegraphics
     size_diagr=strsubst(size_diagr,']','');
     if size_diagr~=[] then
       txt=strsubst(txt,'width=400.00pt',size_diagr)
     end
     mputl(txt,'./'+lisf(i,1)+'/'+lisf(i,1)+'_sblock_equiv.tex');
   end

   ////////////////////////////
   //Create  _comput_func.tex
   ////////////////////////////
   comput=[]
   prop=return_prop_block(lisf(i,1));
   func=prop(12);
   if func<>'csuper' then
     if fileinfo(rout_path+func+'.f')<>[] then
       comput=rout_path+func+'.f'
     elseif fileinfo(rout_path+func+'.c')<>[] then
       comput=rout_path+func+'.c'
     elseif strindex(lisf(i,1),'_c')<>[] then
       comput=rout_path+'code_gen/'+func+'/'+func+'.c'
     elseif fileinfo(SCI+'/routines/scicos/'+func+'.f')<>[] then
       comput=SCI+'/routines/scicos/'+func+'.f'
     elseif fileinfo(SCI+'/routines/scicos/'+func+'.c')<>[] then
       comput=SCI+'/routines/scicos/'+func+'.c'

     end
     if comput<>[] then
       //pause
       if type(prop(13))<>10 then
          typb=0
       else
          typb=prop(13)
       end
       if lang=='fr' then
         txt=['\subsection{Fonction de calcul (type '+string(typb)+')}']
       else
         txt=['\subsection{Computational function (type '+string(typb)+')}']
       end
       txt=[txt;'{\tiny'
            '\verbatiminput{'+comput+'}'
           '}']
     end
   else
     tt=return_rpar_block(lisf(i,1))
     dessin_scs_m(tt,lisf(i,1)+'_super',typdoc)
     unix_g("mv "+lisf(i,1)+"_super/"+lisf(i,1)+"_super.eps ./"+lisf(i,1))
     unix_g("rm -fr "+lisf(i,1)+"_super/");
     if lang=='fr' then
       txt=['\subsection{Contenu du super-bloc compilé}'];
     else
       txt=['\subsection{Compiled Super Block content}'];
     end
     txt=[txt;'\epsfig{file='+lisf(i,1)+'_super.eps,width=150.00pt}']
   end
    mputl(txt,'./'+lisf(i,1)+'/'+lisf(i,1)+'_comput_func.tex');
 end

 //For palette
 if flag=='pal' then
   ////////////////////////
   //Create  .eps
   ////////////////////////
   dessin_pal(pal_path+lisf(i,1)+'.cosf',typdoc);

   ////////////////////////
   //Create  _pals.tex
   ////////////////////////
   if fileinfo('./'+lisf(i,1)+'_cosf/'+lisf(i,1)+'_cosf.eps')<>[] then //figure of palette
      txt=['\begin{center}'
           '\epsfig{file='+lisf(i,1)+'_cosf.eps,width=350.00pt}'
           '\end{center}']
      //SPECIALDESC
      size_diagr=return_size_scs_diagr(lisf(i,1)+'_cosf',typdoc,'pal')
      size_diagr=strsubst(size_diagr,'[','');  //for compatibility with \includegraphics
      size_diagr=strsubst(size_diagr,']','');
      if size_diagr~=[] then
        txt=strsubst(txt,'width=350.00pt',size_diagr)
      end
      mputl(txt,'./'+lisf(i,1)+'_cosf/'+lisf(i,1)+'_pals.tex');
   end
   ////////////////////////
```

```
  //Create  _blocks.tex
  ////////////////////////
  mod_num_block=return_block_pal(pal_path+lisf(i,1)+'.cosf','mod_num',1)
  if mod_num_block<>[] then
     txt=['\begin{itemize}']
     for g=1:size(mod_num_block,1)
         txt2=return_xml_sdesc(xml_path+lang+'/'+mod_num_block(g)+'.xml');
         txt=[txt;'\item{\htmladdnormallink{'+latexsubst(mod_num_block(g))+...
         ' - '+txt2+'}{'+mod_num_block(g)+'.htm}}']
     end
     txt=[txt;'\end{itemize}'];
     mputl(txt,'./'+lisf(i,1)+'_cosf/'+lisf(i,1)+'_blocks.tex');
  end

end

//For scicos diagram
if flag=='diagr' then
  //Find path of cos file in diagr_all list
  path=return_dir_cos(lisf(i,1),diagr_all)

  ////////////////////////
  //Create  _diagr.tex
  ////////////////////////
  printf("Export figure of diagram... ")
  export_diagr(path,lisf(i,1)+'.cos',typdoc);
  if fileinfo('./'+lisf(i,1)+'/'+lisf(i,1)+'.eps')<>[] then
    unix_g(mv_cmd+'./'+lisf(i,1)+'/'+lisf(i,1)+'.eps ./'+lisf(i,1)+ext+'/'+lisf(i,1)+ext+'.eps');
    unix_g(rm_cmd+'./'+lisf(i,1));
  end
  printf("Done\n");
  //SPECIALDESC
  size_diagr=return_size_scs_diagr(lisf(i,1),typdoc)
  if size_diagr==[] then size_diagr='width=400pt', end;
  size_diagr=strsubst(size_diagr,'[','');
  size_diagr=strsubst(size_diagr,']','');
  if typdoc=='html' then
    txt=['\begin{center}' //figure block
         '\epsfig{file='+lisf(i,1)+'_cos.eps,'+size_diagr+'}'
         '\end{center}']
  else
    txt=[]; //TO BE DONE
    txt=['\begin{center}' //figure block
         '\epsfig{file='+lisf(i,1)+'_cos.eps,'+size_diagr+'}'
         '\end{center}']
  end
  mputl(txt,'./'+lisf(i,1)+ext+'/'+lisf(i,1)+'_diagr.tex');

  ////////////////////////
  //Create  _context.tex
  ////////////////////////
  printf("Export context of diagram... ");
  context=return_context_diagr(path+lisf(i,1)+'.cos');
  if context<>[] then
    mputl(context,'./'+lisf(i,1)+ext+'/'+lisf(i,1)+'_context.tex');
  end
  printf("Done\n");

  ////////////////////////
  //Create  _block.tex
  ////////////////////////
  //Find mod_num block in scs_m
  mod_num_block=return_block_cos(path+lisf(i,1)+'.cos','mod_num',1);
  if mod_num_block<>[] then
    printf("Export list of mod_num block of diagram... ");
    txt=['\begin{itemize}']
    for g=1:size(mod_num_block,1)
      txt2=return_xml_sdesc(xml_path+lang+'/'+mod_num_block(g)+'.xml');
      if typdoc=='html' then
        txt=[txt;'\item{\htmladdnormallink{'+latexsubst(mod_num_block(g))+...
                 ' - '+latexsubst(txt2)+'}{'+mod_num_block(g)+'.htm}}']
      else
        txt=[txt;'\item '+latexsubst(mod_num_block(g))+' - '+latexsubst(txt2)];
      end
    end
    txt=[txt;'\end{itemize}'];
    mputl(txt,'./'+lisf(i,1)+ext+'/'+lisf(i,1)+'_block.tex');
    printf("Done\n");
  end

  ////////////////////
  //Create  _scop.tex
  ////////////////////
  printf("Launch simulation of diagram... ");
  //Launch simulation
  number_scope=scop_results_cos(path+lisf(i,1)+'.cos');
  printf("Done\n");
  if number_scope<>0 then
    printf("Export figures of scope... ");
    //Define true name of diagram
    CosName=return_cos_name_cos(path+lisf(i,1)+'.cos');
    //mv eps file to documentation directory
    for k=1:number_scope
     unix_g('mv '+CosName+'_scope_'+string(k)+'.eps ./'+...
            lisf(i,1)+ext+'/'+lisf(i,1)+'_scope_'+string(k)+'.eps');
    end
    printf("Done\n");
    //sort scope
    if number_scope>1 then
      //SPECIALDESC
      if fileinfo(tex_path+lang+'/'+lisf(i,1)+'/SPECIALDESC')<>[] then
```

```
        printf("Sort figure of scope... ");
        make_scope_order(tex_path+lang+'/'+lisf(i,1),lisf(i,1),ext)
        printf("Done\n");
      end
    end
    //retrieve caption
    printf("Give caption of scope... ");
    capt=return_capt(tex_path+lang+'/'+lisf(i,1))
    if capt==[] then
      for k=1:number_scope
        if lang=='fr' then
           capt(k)='Résultats des ''scopes'''
        else
           capt(k)='Scope results'
        end
      end
    end
    printf("Done\n");
    //write _scop.tex
    txt=[];
    sub_index=0;
    for k=1:number_scope
       if typdoc=='html' then
          txt=[txt;
               '\begin{figure}'
               '\begin{center}'
               '\epsfig{file='+lisf(i,1)+'_scope_'+string(k)+'.eps,width=300.00pt}'
               '\end{center}'
               '\caption{'+capt(k)+'}'
               '\end{figure}'];
       else
          capt(k)=strsubst(capt(k),'\\',' ');
          if sub_index==0 then
             txt=[txt;'\begin{figure}[!h]';
                  '\centering';]
          end
          txt=[txt;'\subfigure['+capt(k)+']{\epsfig{file='+lisf(i,1)+'_scope_'+string(k)+'.eps,width=230.00pt}}']
          sub_index=sub_index+1;
          if sub_index==2 then
             sub_index=0;
             txt=[txt;'\end{figure}']
          end
       end
    end

    if typdoc=='guide' then
       if sub_index==1 then
          txt=[txt;'\end{figure}']
       end
    end
    mputl(txt,'./'+lisf(i,1)+ext+'/'+lisf(i,1)+'_scop.tex');
  end
end

//For scilab simulation script
if flag=='sim' then
  //define path of the simulation script
  path=simu_path+lisf(i,1)+'/';

  ////////////////////
  //Create _diagr.tex
  ////////////////////

  //export figure of diagram
  lisf_cos=return_ext_file_in_dir(tt_ml,path,'.cos')
  if lisf_cos<>[] then
    printf("Export figure of diagram... ");
    for j=1:size(lisf_cos,1)
      export_diagr(path,lisf_cos(j,1),typdoc);
      filef=basename(lisf_cos(j,1));
      if filef<>lisf(i,1) then
        if fileinfo('./'+filef+'/'+filef+'.eps')<>[] then
          unix_g(mv_cmd+'./'+filef+'/'+filef+'.eps ./'+lisf(i,1));
          unix_g(rm_cmd+'./'+filef);
        end
      end
    end
    printf("Done\n");

    //SPECIALDESC
    size_diagr=return_size_scs_diagr2(lisf(i,1),typdoc);
    if size_diagr==[]|size(size_diagr,1)<>size(lisf_cos,1) then
      for j=1:size(lisf_cos,1) size_diagr(j)='width=400pt'; end;
    end

    //write _diagr.tex file
    txt=[];
    printf("Write a _diagr.tex file... ");
    for j=1:size(lisf_cos,1)
      filef=basename(lisf_cos(j,1));
      if typdoc=='html' then
        txt=[txt;
             '\begin{center}' //figure block
             '\epsfig{file='+filef+'.eps,'+size_diagr(j)+'}'
             '\end{center}'
             '\begin{center}'
             '\textbf{'+lisf_cos(j,1)+'}'
             '\end{center}']
      else
        txt=[]; //TO BE DONE
        txt=[txt;
```

```
           '\begin{center}' //figure block
           '\epsfig{file='+filef+'.eps,'+size_diagr(j)+'}'
           '\end{center}'
           '\begin{center}'
           '\textbf{'+latexsubst(lisf_cos(j,1))+'}'
           '\end{center}']
    end
  end
  mputl(txt,'./'+lisf(i,1)+ext+'/'+lisf(i,1)+'_diagr.tex');
  printf("Done\n");
end

//////////////////////
//Create _context.tex
//////////////////////
lisf_ctxt=return_ext_file_in_dir(tt_ml,path,'_ctxt.sce')
if lisf_ctxt<>[] then
  printf("Write a _context.tex file... ");
  txt=[];
  for j=1:size(lisf_ctxt,1)
    if typdoc=='html' then
      txt=[txt;
           '\verbatiminput{'+path+lisf_ctxt(j,1)+'}'];
    else
      txt=[txt;'\begin{small}'
           '\verbatiminput{'+path+lisf_ctxt(j,1)+'}'
           '\end{small}'];
    end
    txt=[txt;
         '\begin{center}'
         '\textbf{'+latexsubst(lisf_ctxt(j,1))+'}'
         '\end{center}']
  end
  mputl(txt,'./'+lisf(i,1)+ext+'/'+lisf(i,1)+'_context.tex');
  printf("Done\n");
end

////////////////////////
//Create _sim_script.tex
////////////////////////
lisf_sce=return_ext_file_in_dir(tt_ml,path,'.sce')
if lisf_sce<>[] then
  printf("Write a _sim_script.tex file... ");
  txt=[];
  for j=1:size(lisf_sce,1)
    if strindex(lisf_sce(j,1),'_ctxt.sce')==[] then
      if typdoc=='html' then
         txt=[txt;
              '\verbatiminput{'+path+lisf_sce(j,1)+'}']
      else
         txt=[txt;'\begin{small}'
              '\verbatiminput{'+path+lisf_sce(j,1)+'}'
              '\end{small}']
      end
      txt=[txt;'\begin{center}'
           '\textbf{'+latexsubst(lisf_sce(j,1))+'}'
           '\end{center}']
    end
  end
  mputl(txt,'./'+lisf(i,1)+ext+'/'+lisf(i,1)+'_sim_script.tex');
  printf("Done\n");
end

//////////////////
//Create _scop.tex
//////////////////
if fileinfo(path+lisf(i,1)+'.sce')<>[] then
  if with_sim then //test define in loader.sce
    printf("Launch simulation... ");
    number_scope=scop_results_sim(path+lisf(i,1)+'.sce');
    printf("Done\n");
    if number_scope<>0 then
      titlef=basename(path+lisf(i,1)+'.sce')
      printf("Export figures of scope... ");
      //mv eps file to documentation directory
      for k=1:number_scope
        unix_g('mv '+titlef+'_scope_'+string(k)+'.eps ./'+...
           lisf(i,1)+ext+'/'+lisf(i,1)+'_scope_'+string(k)+'.eps');
      end
      printf("Done\n");
      //sort scope
      if number_scope>1 then
        //SPECIALDESC
        if fileinfo(tex_path+lang+'/'+lisf(i,1)+'/SPECIALDESC')<>[] then
          printf("Sort figure of scope... ");
          make_scope_order(tex_path+lang+'/'+lisf(i,1),lisf(i,1),ext)
          printf("Done\n");
        end
      end
      //retrieve caption
      printf("Give caption of scope... ");
      capt=return_capt(tex_path+lang+'/'+lisf(i,1))
      if capt==[] then
        for k=1:number_scope
          if lang=='fr' then
            capt(k)='Résultats des ''scopes'''
          else
            capt(k)='Scope results'
          end
        end
      end
```

```
          printf("Done\n");
          //write _scop.tex
          txt=[];
          sub_index=0;
          for k=1:number_scope
             if typdoc==’html’ then
                txt=[txt;
                     ’\begin{figure}’
                     ’\begin{center}’
                     ’\epsfig{file=’+lisf(i,1)+’_scope_’+string(k)+’.eps,width=300.00pt}’
                     ’\end{center}’
                     ’\caption{’+latexsubst(capt(k))+’}’
                     ’\end{figure}’];
             else
                capt(k)=strsubst(capt(k),’\\’,’ ’);
                if sub_index==0 then
                  txt=[txt;’\begin{figure}[!h]’;
                       ’\centering’;]
                end
                txt=[txt;’\subfigure[’+capt(k)+’]{\epsfig{file=’+lisf(i,1)+’_scope_’+string(k)+’.eps,width=230.00pt}}’]
                sub_index=sub_index+1;
                if sub_index==2 then
                  sub_index=0;
                  txt=[txt;’\end{figure}’]
                end
             end
          end

          if typdoc==’guide’ then
             if sub_index==1 then
                txt=[txt;’\end{figure}’]
             end
          end
          mputl(txt,’./’+lisf(i,1)+ext+’/’+lisf(i,1)+’_scop.tex’);
        end
      end
    end

    ////////////////////
    //Create  _block.tex
    ////////////////////
    lisf_cos=return_ext_file_in_dir(tt_ml,path,’.cos’)
    if lisf_cos<>[] then
      printf("Export list of mod_num block of diagram... ");
      txt=[];
      for j=1:size(lisf_cos,1)
        mod_num_block=return_block_cos(path+lisf_cos(j,1),’mod_num’,1);
        if mod_num_block<>[] then
          for g=1:size(mod_num_block,1)
            txt2=return_xml_sdesc(xml_path+lang+’/’+mod_num_block(g)+’.xml’);
            if typdoc==’html’ then
              txt=[txt;’\item{\htmladdnormallink{’+latexsubst(mod_num_block(g))+...
                       ’ - ’+latexsubst(txt2)+’}{’+mod_num_block(g)+’.htm}}’]
            else
              txt=[txt;’\item ’+latexsubst(mod_num_block(g))+’ - ’+latexsubst(txt2)]
            end
          end
        end
      end
      if txt<>[] then
        //Elimine les doublons
        //TO BE DONE
        txt=[’\begin{itemize}’;txt;’\end{itemize}’];
        printf("Done\n");
        mputl(txt,’./’+lisf(i,1)+ext+’/’+lisf(i,1)+’_block.tex’);
      end
    end

  end

  //For all : look at for a tex directory
  if fileinfo(tex_path+lang+’/’+lisf(i,1)+’/’)<>[] then
   printf("Tex directory ’+lisf(i,1)+’ found : process it\n")

    if fileinfo(tex_path+lang+’/’+lisf(i,1)+’/Makefile’)<>[] then
      printf("Makefile in Tex directory found : process it (all+clean)\n")
      repi=pwd()
      chdir(tex_path+lang+’/’+lisf(i,1)+’/’)
      unix_g(’make all’)
      fc=unix_g(cp_cmd+tex_path+lang+’/’+lisf(i,1)+’/* ’+repi+’/’+lisf(i,1)+ext);
      unix_g(’make clean’)
      chdir(repi)
    else
      fc=unix_g(cp_cmd+tex_path+lang+’/’+lisf(i,1)+’/* ./’+lisf(i,1)+ext);
    end
  end

end
endfunction
```

2.3 Create main tex file of block

- **Name:** generate_blocks_tex_file
- **Library:** generate_doc - Main library of the documentation generator

2.3.1 Calling Sequence

`generate_blocks_tex_file(lisf,flag,lang)`

2.3.2 Parameters

- **lisf :** string. the name of the interfacing function of a scicos block (extension free)
- **flag :** string. set the type of tex files to produce
  - **'html' :** to produce tex file for html format
  - **'guide' :** to produce tex for file paper format
- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page

2.3.3 File content

```
//generate_blocks_tex_file(lisf(i,1),lisf(i,2),flag)
//
//Fonction qui génère le fichier tex principal
//d'un block scicos
//Entrée : lisf : Nom de la fonction
//         flag : 'html' pour générer une page d'aide html
//              'guide' pour générer un page d'aide ps
//         lang : 'eng pour de l'anglais (default)
//                'fr' pour du français
function generate_blocks_tex_file(lisf,flag,lang)

 //verify the 'lang' right parameter
 [lsh,rsh]=argn(0)
 if rsh<3 then
   if ~exists('lang') then
    lang='eng'
   elseif lang<>'eng' & lang<>'fr' then
    lang='eng'
   end
 end

 //Generate auxiliary tex files
 lisf=generate_aux_tex_file(lisf,'block',flag,lang);

 //define title of paragraph
 tt_title=[
           ''                        //tt1  : header du fichier tex
           ''                        //tt2  : figure du block (.eps)
           'Palette'                 //tt3  : Palette
           'Theorical background'    //tt4  : Theorical background (_theo_bkg)
           'Technical background'    //tt5  : Technical background (_tec_bkg)
           'Description'             //tt6  : Description (_long)
           'Algorithm'               //tt7  : Algorithm (_algo)
           'Super block equivalent model'  //tt8  : Figure du super block equivalent (sblock_equiv)
           'Scilab script/function equivalent Model'  //tt9  : Scilab script equivalent (sci_equiv)
           'Dialog box'              //tt10 : Dialog box (_dial_box,_param)
           'Example'                 //tt11 : Example (_ex)
           'Default properties'      //tt12 : Default properties (_def_prop)
           'Interfacing function'    //tt13 : Interfacing function
           'Computational function' //tt14 : Computational function (_comput_fun)
           'Used functions'          //tt15 : Used functions (_used_func)
           'Pair Block'              //tt16 : Pair Block (_pair_blk)
           'See also'                //tt17 : See Also (_see_also)
           'Authors'                 //tt18 : Authors (_authors)
           'Bibliography'            //tt19 : Bibliography (_bib)
           ''                        //tt20 : End of tex file
          ]

 //change language of title
 if lang=='fr' then
  tt_title=change_lang_title(lang,tt_title);
 end

 //define level of paragraph
 if flag=='html' then
  tex_title='\subsection{'+tt_title+'}'
 else
  tex_title='\subsection{'+tt_title+'}'
 end


 for i=1:size(lisf,1) //for each file
  for j=1:20 execstr('tt'+string(j)+'=[]'),end //for each paragraph

    if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_blocks.tex')<>[] then //figure block
      tt2=['\input{'+lisf(i,1)+'_blocks}']
    end
```

```
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_theo_bkg.tex')<>[] then //Theorical background
   tt4=[tex_title(4)
        '\input{'+lisf(i,1)+'_theo_bkg}']
end

if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_tec_bkg.tex')<>[] then //Technical backgroung
   tt5=[tex_title(5)
        '\input{'+lisf(i,1)+'_tec_bkg}']
end
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_long.tex')<>[] then //Description
   tt6=[tex_title(6)
        ''
        '\input{'+lisf(i,1)+'_long}']
end
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_algo.tex')<>[] then //Algorithm
   tt7=[tex_title(7)
         '\input{'+lisf(i,1)+'_algo}']
end
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_sblock_equiv.tex')<>[] then  //Super Block
   tt8=[tex_title(8)
        '\input{'+lisf(i,1)+'_sblock_equiv}']
end
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_sci_equiv.tex')<>[] then //Scilab script
   tt9=[tex_title(9)
        '\input{'+lisf(i,1)+'_sci_equiv}']
end
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_dial_box.tex')<>[] then //Dialog box
   tt10=[tex_title(10)
         '\input{'+lisf(i,1)+'_dial_box}']
end
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_param.tex')<>[] then //Parameters
   tt10=[tt10;'';'\input{'+lisf(i,1)+'_param}']
end
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_ex.tex')<>[] then //Example
   tt11=[tex_title(11)
        '\input{'+lisf(i,1)+'_ex}']
end
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_comput_func.tex')<>[] then //Computational Function
   tt14=['\input{'+lisf(i,1)+'_comput_func}'] //Warning!
end
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_used_func.tex')<>[] then //Used function
   tt15=[tex_title(15)
        '\input{'+lisf(i,1)+'_used_func}']
end
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_pair_blk.tex')<>[] then //Pair Block
   tt16=[tex_title(16)
        '\input{'+lisf(i,1)+'_pair_blk}']
end
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_see_also.tex')<>[] then //See Also
   tt17=[tex_title(17)
        '\input{'+lisf(i,1)+'_see_also}']
end
if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_bib.tex')<>[] then //bibliography
   tt19=['\input{'+lisf(i,1)+'_bib.tex}']
end

////////////////////
///paper
////////////////////
if flag=='guide' then

  tt1=['\section{'+latexsubst(lisf(i,1))+' - '+...
       latexsubst(lisf(i,2))+'}\label{'+lisf(i,1)+'}'] //Header of tex file

  PalName=return_rpordoef(tt_ml,lisf(i,1)+'.sci')
  if PalName<>[] then //Palette
     tta=return_xml_sdesc(xml_path+lang+'/'+PalName+'.xml')
  else
     tta=[];
  end
  tt3=['\begin{itemize}';
       '\item \textbf{'+tt_title(3)+' :} '+PalName+'.cosf - '+tta;  //Palette on pout rajouter un \ref{} ici!
       '\item \textbf{'+tt_title(13)+' :} \tt '+latexsubst(lisf(i,1))+'.sci'; //fonction d'interface
       '\end{itemize}';]
////////////////////
///html
////////////////////
elseif flag=='html' then

 if lang=='fr' then //Header of tex file
   tt1=['\documentclass[11pt,frenchb]{article}']
 else
   tt1=['\documentclass[11pt]{article}']
 end
 tt1=[tt1;
      '\usepackage{makeidx,graphics,fullpage}'
      '\usepackage{verbatim,times,amsmath,amssymb,epsfig,color}'
      '\usepackage{html}'
      '\begin{document}']
  if lang=='fr' then
    tt1=[tt1;'\begin{center}Bloc Scicos\\']
  else
    tt1=[tt1;'\begin{center}'+lisf(i,3)+'\\']
  end
  tt1=[tt1
       '\htmladdnormallink{eng}{../eng/'+lisf(i,1)+'.htm}\hspace{2mm}-'+...
       '\hspace{2mm}\htmladdnormallink{fr}{../fr/'+lisf(i,1)+'.htm}}'
       '\end{center}'];
  tt1=[tt1;'\section{'+latexsubst(lisf(i,2))+'}\label{'+lisf(i,1)+'}']
```

```
      tt2=[tt2;'\tableofcontents'] //table of contents

      PalName=return_rpordoef(tt_ml,lisf(i,1)+'.sci')
      if PalName<>[] then //Palette
         tta=return_xml_sdesc(xml_path+lang+'/'+PalName+'.xml')
         tt3=[tex_title(3)
             '\begin{itemize}']
         //if flag=='html' then
            tt3=[tt3;'\item{\htmladdnormallink{'+PalName+'.cosf - '+tta+'}{'+PalName+'.htm}}']
        // else
        //    tt3=[tt3;'\item{'+PalName+'.cosf - '+tta+'}']  //on pout rajouter un \ref{} ici!
        // end
         tt3=[tt3;'\end{itemize}']
      end
      if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_def_prop.tex')<>[] then //Defaults Properties
        tt12=[tex_title(12)
              '\input{'+lisf(i,1)+'_def_prop}']
      end
      tt13=[tex_title(13)  //Interfacing Function
            '{\tt '+latexsubst(lisf(i,1))+'.sci}']
      if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_authors.tex')<>[] then //Authors
        tt18=[tex_title(18)
              '\input{'+lisf(i,1)+'_authors}']
      end
      tt20=['\htmlinfo*';'\end{document}']
  end

  //Write the main tex file of block
  txt=[]
  for j=1:20 txt=[txt;evstr('tt'+string(j))], end

  mputl(txt,lisf(i,1)+'/'+lisf(i,1)+'.tex');
end

endfunction
```

2.4 Create main tex file of scicos diagram

- **Name:** generate_diagr_tex_file
- **Library:** generate_doc - Main library of the documentation generator

2.4.1 Calling Sequence

`generate_diagr_tex_file(lisf,flag,lang)`

2.4.2 Parameters

- **lisf :** string. the name of the scicos diagram (extension free)
- **flag :** string. set the type of tex files to produce
  - **'html' :** to produce tex file for html format
  - **'guide' :** to produce tex for file paper format
- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page

2.4.3 File content

```
//generate_diagr
//Fonction qui génère le fichier tex principal
//d'une page de documentation d'un diagramme
//scicos.

function generate_diagr_tex_file(lisf,flag,lang)

 //verify the 'lang' right parameter
 [lsh,rsh]=argn(0)
 if rsh<3 then
   if ~exists('lang') then
     lang='eng'
   elseif lang<>'eng' & lang<>'fr' then
     lang='eng'
   end
 end
```

```
//Generate auxiliary tex files
lisf=generate_aux_tex_file(lisf,'diagr',flag,lang);

//define title of paragraph
tt_title=[
          ''               //tt1  : header du fichier tex
          ''               //tt2  : Diagram (_diagr)
          'Description'    //tt3  : Description (_long)
          'Context'        //tt4  : context (_context)
          'Scope Results'  //tt5  : Scope results (_scop)
          ''               //tt6  : scilab script file (not programmed)
          'Mod\_num blocks' //tt7  : Mod num blocks (_block)
          'See Also'       //tt8  : See Also (_see_also)
          'Authors'        //tt9  : Authors (_authors)
          'Bibliography'   //tt10 : Bibliography (_bib)
          ''               //tt11 : End of tex file
         ]

//change language of title
if lang=='fr' then
 tt_title=change_lang_title(lang,tt_title);
end

for i=1:size(lisf,1)
   for j=1:11 execstr('tt'+string(j)+'=[]'),end
    //define level of paragraph
   //if flag=='html' then
     tex_title='\subsection{'+tt_title+'}';
   //else
     //if grep(lisf(i,1),diagr_elec)<>[] then
       tex_title='\subsection{'+tt_title+'}';
     //else
      // tex_title='\subsection{'+tt_title+'}';
     //end
   //end

   if fileinfo(lisf(i,1)+'_cos'+'/'+lisf(i,1)+'_diagr.tex')<>[] then //figure
     tt2=['\input{'+lisf(i,1)+'_diagr}']
   end

   if fileinfo(lisf(i,1)+'_cos'+'/'+lisf(i,1)+'_long.tex')<>[] then //Description
     tt3=[tex_title(3)
          ''
          '\input{'+lisf(i,1)+'_long}']
   end

   if fileinfo(lisf(i,1)+'_cos'+'/'+lisf(i,1)+'_context.tex')<>[] then //context
     tt4=[tex_title(4)
         '{\tiny'
         '\verbatiminput{'+lisf(i,1)+'_context}'
        '}']
   end

   if fileinfo(lisf(i,1)+'_cos'+'/'+lisf(i,1)+'_scop.tex')<>[] then //scop
     tt5=[tex_title(5)
          '\input{'+lisf(i,1)+'_scop}']
   end

   if fileinfo(lisf(i,1)+'_cos'+'/'+lisf(i,1)+'_block.tex')<>[] then //mod_num block
     tt7=[tex_title(7)
          '\input{'+lisf(i,1)+'_block}']
   end

   if fileinfo(lisf(i,1)+'_cos/'+lisf(i,1)+'_see_also.tex')<>[] then //see also
     tt8=[tex_title(8)
          '\input{'+lisf(i,1)+'_see_also}']
   end

   if fileinfo(lisf(i,1)+'_cos/'+lisf(i,1)+'_bib.tex')<>[] then //bibliography
    tt10=['\input{'+lisf(i,1)+'_bib.tex}']
   end

   if flag=='guide' then
        tt1=['\section{'+latexsubst(lisf(i,2))+'}\label{'+lisf(i,1)+'}'];

   elseif flag=='html' then

   if lang=='fr' then //Header of tex file
     tt1=['\documentclass[11pt,frenchb]{article}']
   else
     tt1=['\documentclass[11pt]{article}']
   end
   tt1=[tt1;
        '\usepackage{makeidx,graphics,fullpage}'
        '\usepackage{verbatim,times,amsmath,amssymb,epsfig,color}'
        '\usepackage{html}'
        '\begin{document}'];
   if lang=='fr' then
     tt1=[tt1;'\begin{center}Diagramme Scicos\\'];
   else
     tt1=[tt1;'\begin{center}Scicos Diagram\\'];
   end
   tt1=[tt1
        '\htmladdnormallink{eng}{../eng/'+lisf(i,1)+'.htm}\hspace{2mm}-'+...
        '\hspace{2mm}\htmladdnormallink{fr}{../fr/'+lisf(i,1)+'.htm}}'
        '\end{center}'];
   tt1=[tt1;'\section{'+latexsubst(lisf(i,2))+'}\label{'+lisf(i,1)+'}'];

   if fileinfo(lisf(i,1)+'_cos/'+lisf(i,1)+'_authors.tex')<>[] then //authors
     tt9=[tex_title(9)
          '\input{'+lisf(i,1)+'_authors}']
```

```
    end

    tt11=['\htmlinfo*';'\end{document}']
    end

    //Generate the main tex file of block
    txt=[]
    for j=1:11 txt=[txt;evstr('tt'+string(j))], end
    mputl(txt,lisf(i,1)+'_cos/'+lisf(i,1)+'.tex')
 end

endfunction
```

2.5 Create man page for scilab html browser

- **Name:** generate_html_file
- **Library:** generate_doc - Main library of the documentation generator

2.5.1 Calling Sequence

`generate_html_file(lisf,flag,lang)`

2.5.2 Parameters

- **lisf :** string. a name of a man page
- **flag :** string. set the type of man page
  - **'block' :** for interfacing function of scicos block
  - **'pal' :** for a palette (.cosf file)
  - **'diagr' :** for a scicos diagram (.cos file)
  - **'scilib' :** for a library of scilab macros
  - **'sci' :** for a scilab macro.
  - **'rout' :** for computational routine
  - **'sim' :** for scilab simulation script (_sim.sce file)
  - **'sce' :** for scilab script (.sce file)
- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page

2.5.3 File content

```
//generate_html_file
//Entrée : lisf est une liste de nom de fichier sans extension: CAN_f, Linear ou synthe
//         flag est un drapeau(pour l'instant de taille 1):
//                'block' pour une fonction d'interface scicos
//                'pal' pour un fichier palette scicos (cosf)
//                'diagr' pour un diagramme de simulation scicos
//                'scilib' pour une librairie de fonctions scilab
//                'sci' pour une fonction scilab.
//                'rout' pour une routines bas niveau
//                'sim' pour un script de simulation scilab
//                'sce' pour un script scilab
// DOIT RAJOUTER lang
function generate_html_file(lisf,flag,lang)

  //verify the 'lang' right parameter
  [lsh,rsh]=argn(0)
  if rsh<3 then
    if ~exists('lang') then
     lang='eng'
    elseif lang<>'eng' & lang<>'fr' then
     lang='eng'
    end
  end

  select flag
```

```
      case 'block'
       ext=''
       func=generate_blocks_tex_file
      case 'pal'
       ext='_cosf'
       func=generate_pals_tex_file
      case 'diagr'
       ext='_cos'
       func=generate_diagr_tex_file
      case 'scilib'
       ext='_scilib'
       func=generate_scilib_tex_file
      case 'sci'
       ext='_sci'
       func=generate_scifun_tex_file
      case 'rout'
       ext='_rout'
       func=generate_rout_tex_file
      case 'sim'
       ext=''
       func=generate_sim_tex_file
      case 'sce'
       ext='_sce'
       func=generate_sce_tex_file
      else
       printf("Try with flag ''block'',''pal'',...
       ''diagr'',''scilib'',''sci'',''rout'',''sim''\n")
       abort
   end

   rep=lisf+ext
   for i=1:size(lisf,1)
      //generate main tex file DOIT RAJOUTER lang
      func(lisf(i,1),'html',lang);
      chdir(rep(i,1));
      //analyse tex files
      analyse_tex_file('.');

      flg_bib=%f;
      //run LaTeX
      if fileinfo(lisf(i,1)+'_bib.tex')<>[] then //bibliography
       printf("Bibliography file found. Run latex...")
       unix_g(latex_cmd+lisf(i,1)+'.tex');
       printf("Done\n")
       flg_bib=%t; //flag bib
      end
      //conversion latex2html
      printf("Convert %s.tex file in %s.html... ",lisf(i,1),lisf(i,1));
      unix_g(latex2html_cmd+lisf(i,1)+'_ '+lisf(i,1));
      printf("Done\n");
      //change color subtitle
      html_txt=change_color_subtitle('./'+lisf(i,1)+'/'+lisf(i,1)+'.htm','blue');
      if html_txt<>[] then mputl(html_txt,'./'+lisf(i,1)+'/'+lisf(i,1)+'.htm'); end;
      //change font
      html_txt=change_font('./'+lisf(i,1)+'/'+lisf(i,1)+'.htm',flag);
      if html_txt<>[] then mputl(html_txt,'./'+lisf(i,1)+'/'+lisf(i,1)+'.htm'); end;
      //change bibliography level
      if flg_bib then
       html_txt=change_level_bib('./'+lisf(i,1)+'/'+lisf(i,1)+'.htm');
      end
      if html_txt<>[] then mputl(html_txt,'./'+lisf(i,1)+'/'+lisf(i,1)+'.htm'); end;
      //change 'Contents' and 'Bibliography' line
      if lang<>'eng' then
       html_txt=change_contents_line('./'+lisf(i,1)+'/'+lisf(i,1)+'.htm',lang);
      end
      if html_txt<>[] then mputl(html_txt,'./'+lisf(i,1)+'/'+lisf(i,1)+'.htm'); end;
      if lang<>'eng' then
       html_txt=change_biblio_line('./'+lisf(i,1)+'/'+lisf(i,1)+'.htm',lang);
      end
      if html_txt<>[] then mputl(html_txt,'./'+lisf(i,1)+'/'+lisf(i,1)+'.htm'); end;

      tt=listfiles("./"+lisf(i,1)+"/");
      htm_f=%f;gif_f=%f;
      for j=1:size(tt,1)
       if strindex(tt(j),'.htm')<>[] then htm_f=%t, end;
       if strindex(tt(j),'.gif')<>[] then gif_f=%t, end;
      end
      //move htm files
      if htm_f then
        //unix_g(mv_cmd+'./'+lisf(i,1)+'/'+lisf(i,1)+'.htm ../htm')
        unix_g(mv_cmd+'./'+lisf(i,1)+'/'+lisf(i,1)+'.htm '+html_path+lang+'/')
      end
      //move gif files
      if gif_f then
        //unix_g(mv_cmd+'./'+lisf(i,1)+'/*.gif ../htm');
        unix_g(mv_cmd+'./'+lisf(i,1)+'/*.gif '+html_path+lang+'/')
      end
      //clean temporary files
      chdir('..')
      unix_g(rm_cmd+rep(i,1))

   end
endfunction
```

2.6 Create paper documentation of the toolbox

- **Name:** generate_mod_num_guide

- **Library:** generate_doc - Main library of the documentation generator

2.6.1 Calling Sequence

`generate_mod_num_guide(flag,lang)`

2.6.2 Parameters

- **flag :** string. a flag to set the type of guide
  - **'user' :** for user's guide
  - **'ref' :** for reference guide
  - **'int' :** for internals guide
- **lang :** string. set the language of the documentation
  - **'eng' :** for english
  - **'fr' :** for french

2.6.3 File content

```
//generate mod_num_guide
//fonction qui crée la documentation papier
//de la boîte à outils
//
//Entrée : flag : un drapeau pour signaler quel type de doc:
//                'user' pour le guide utilisateur
//                'ref'  pour le guide de référence
//                'internal'  pour le guide interne
//         lang : pour choisr le type de language
//                'eng' pour de l'anglais
//                'fr' pour du français
function generate_mod_num_guide(flag,lang)

  [lsh,rsh]=argn(0)
  if rsh<2 then
    if ~exists('lang') then
     lang='eng'
    elseif lang<>'eng' & lang<>'fr' then
     lang='eng'
    end
  end

  //Preambule of each LaTex Guide
  if lang=='fr' then
    tt_head=['\documentclass[11pt,a4paper,frenchb]{report}'];
  else
    tt_head=['\documentclass[11pt,a4paper]{report}'];
  end
  tt_head=[tt_head
           '\usepackage{graphics}'
           '\usepackage{subfigure}'
           '\renewcommand{\thesubfigure}{}'
           '\usepackage{html}'
           '\usepackage[T1]{fontenc}'
           '\usepackage[latin1]{inputenc}'
           '\usepackage{verbatim,times,amsmath,amssymb,epsfig,color}'
           '\usepackage{geometry}']
  if lang=='fr' then
   tt_head=[tt_head
            '\usepackage{babel}'];
  end
  tt_head=[tt_head
           '\geometry{verbose,a4paper,tmargin=2cm,bmargin=2.5cm,lmargin=1.7cm,rmargin=1.7cm,headheight=1cm,footskip=2cm}'
           ''
           '\pagestyle{headings}'
           '\setcounter{tocdepth}{1}'];

  //title of each LaTex Guide
  tt_title=['\author{IRCOM Group \\ Alan Layec}'
            ''
            '\begin{document}'
            '   \maketitle'
            ' \newpage'
            ''
            ' \thispagestyle{empty}'
            ' \null'
            ' \newpage'
            ''']
//   if lang=='fr' then
//      tt_title=[tt_title
//                ' \chapter*{Préface}']
//   else
//      tt_title=[tt_title
//                ' \chapter*{Preface}'];
```

```
//   end
//   tt_title=[tt_title
//             ' \thispagestyle{empty}'
//             ' \newpage'
//             ''
//             ' \thispagestyle{empty}'
//             ' \null'
//             ' \newpage'];

  //contents of each LaTex Guide
  tt_contents=['' 
               ' \setcounter{page}{1}'
               ' \addtocontents{toc}{\protect\thispagestyle{headings}}'
               ' \tableofcontents'
               ' \thispagestyle{headings}'
               ''];

  if flag=='ref' then
  ///////////////
  //Reference guide
  //////////////

  /////////Create LaTeX files//////
  //preface
  tt_i=[];

  //scicos blocks and palette
  tt_sb=[];
  TEXINPUTS='$TEXINPUTS:.';
  PalRep=return_dir_in_dir(tt_ml,pal_path)
  for j=1:size(PalRep,1)
     PalName=basename(part(PalRep(j),1:length(PalRep(j))-1));
     tt_sb=[tt_sb;'\input{'+PalName+'_cosf/'+PalName+'}';'\pagebreak'];
     TEXINPUTS=TEXINPUTS+':./'+PalName+'_cosf/';
     generate_pals_tex_file(PalName,'guide',lang);
     //scicos blocks
     lisf_blocks=return_ext_file_in_dir(tt_ml,PalRep(j),'.sci');
     for i=1:size(lisf_blocks,1)
       name= part(lisf_blocks(i,1),1:length(lisf_blocks(i,1))-4);
       generate_blocks_tex_file(name,'guide',lang)
       tt_sb=[tt_sb;'\input{'+name+'/'+name+'}';'\pagebreak']
       TEXINPUTS=TEXINPUTS+':./'+name+'/'
     end
  end

  //scilab macros
  tt_sim=[];
  Librep=return_dir_in_dir(tt_ml,mac_path)
  for j=1:size(Librep,1)
    LibName=basename(part(Librep(j),1:length(Librep(j))-1));
    if grep(LibName,ex_lib_name)==[] then
      tt_sim=[tt_sim;'\input{'+LibName+'_scilib/'+LibName+'}';];
      TEXINPUTS=TEXINPUTS+':./'+LibName+'_scilib/';
      generate_scilib_tex_file(LibName,'guide',lang);
      //Scilab function
      lisf=return_ext_file_in_dir(tt_ml,Librep(j),'.sci');
      for i=1:size(lisf,1)
         name=basename(lisf(i,1));
         generate_scifun_tex_file(name,'guide',lang);
         tt_sim=[tt_sim;'\input{'+name+'_sci/'+name+'}'];
         TEXINPUTS=TEXINPUTS+':./'+name+'_sci/';
      end
    end
  end
  //Mod_num_sci_lib
  tt_sim=[tt_sim;'\input{'+mod_num_sci_lib+'_scilib/'+mod_num_sci_lib+'}'];
  TEXINPUTS=TEXINPUTS+':./'+mod_num_sci_lib+'_scilib/';
  generate_scilib_tex_file(mod_num_sci_lib,'guide',lang);
  for i=1:size(modnum_sci_func,1)
     generate_scifun_tex_file(modnum_sci_func(i),'guide',lang)
     tt_sim=[tt_sim;'\input{'+modnum_sci_func(i)+'_sci/'+modnum_sci_func(i)+'}'];
     TEXINPUTS=TEXINPUTS+':./'+modnum_sci_func(i)+'_sci/';
  end

  //main tex file
  if lang=='fr' then
    txt=[tt_head;'\title{\Huge ""Modnum""\\'
         'Boîte à outils Scilab\\'
         'pour les systèmes de communication\\'
         'Guide de r\''ef\''erence\\'
         '\small Version provisoire}'];
  else
    txt=[tt_head;'\title{\Huge ""Modnum""\\'
         'Scilab toolbox\\'
         'for the communication systems\\'
         'Reference Guide\\'
         '\small Draft Version}'];
  end
  txt=[txt
       tt_title
       tt_i
       tt_contents];
  if lang=='fr' then
    txt=[txt
         ' %%Blocs Scicos'
         ' \part{Blocs Scicos}'];
  else
    txt=[txt
         ' %%Scicos blocks'
         ' \part{Scicos blocks}'];
  end
```

```
txt=[txt
     ' \null'
     ' \newpage'
     ''
     " "+tt_sb
     ''
     ' \addtocontents{toc}{\protect\pagebreak}'
     ' \setcounter{chapter}{0}'
     ''
     ' \newpage'
     ' \null'
     ' \newpage'
     '']
if lang=='fr' then
  txt=[txt
       ' %%Fonctions Scilab'
       ' \part{Fonctions Scilab}'];
else
  txt=[txt
       ' %%Silab functions'
       ' \part{Silab functions}'];
end
txt=[txt
     '\null'
     '\newpage'
     ''
     " "+tt_sim
     '\end{document}']
mputl(txt,'ref.tex')

////////Compile//////////
//mv report.cls to current directory
unix_g(cp_cmd+man_path+'tex/'+'report.cls'+' .')
//define cmd
latex_cmd='latex -interaction=nonstopmode ref.tex ';
export_cmd='export TEXINPUTS='+TEXINPUTS;
divps_cmd='dvips -o ref.ps ref.dvi';
gv_cmd='gv ref.ps';
//latex+dvips+gv
new_tt=export_cmd+' ; '+latex_cmd+' ; '+latex_cmd+' ; '+latex_cmd+' ; '+divps_cmd+' ; '+gv_cmd;
mputl(new_tt,'compile.sh');
unix_g('chmod a+x compile.sh');
pause
unix_g('./compile.sh');
//ps2pdf
ps2pdf_cmd='ps2pdf14 ref.ps';
unix_g(ps2pdf_cmd);
unix_g('cp ref.pdf modnum_ref.pdf')
unix_g(mv_cmd+'./modnum_ref.pdf '+pdf_path+lang+'/');

////////////Clean//////////
//Erase template LaTeX file
unix_g('rm -f ref.*');
unix_g('rm -f compile.sh');
unix_g(rm_cmd+'report.cls');
//Erase template directories of scicos blocks and palettes
PalRep=return_dir_in_dir(tt_ml,pal_path);
for j=1:size(PalRep,1)
  PalName=basename(part(PalRep(j),1:length(PalRep(j))-1));
  unix_g(rm_cmd+PalName+'_cosf')
  lisf_blocks=return_ext_file_in_dir(tt_ml,PalRep(j),'.sci');
  for i=1:size(lisf_blocks,1)
    name=basename(lisf_blocks(i,1))
    unix_g(rm_cmd+name);
  end
end
//Erase template directories of scilab macros and libraries
Librep=return_dir_in_dir(tt_ml,mac_path)
for j=1:size(Librep,1)
  LibName=basename(part(Librep(j),1:length(Librep(j))-1));
  unix_g(rm_cmd+LibName+'_scilib');
  //Scilab function
  lisf=return_ext_file_in_dir(tt_ml,Librep(j),'.sci');
  for i=1:size(lisf,1)
    name=basename(lisf(i,1));
    unix_g(rm_cmd+name+'_sci');
  end
end
//Mod_num_sci_lib
unix_g(rm_cmd+'mod_num_sci_lib_scilib');
for i=1:size(modnum_sci_func,1)
  unix_g(rm_cmd+modnum_sci_func(i)+'_sci');
end

elseif flag=='user' then
/////////////////
//User's guide
/////////////////

//introduction
tt_i=[];

tt_cs=[]//continous system
TEXINPUTS='$TEXINPUTS:.';
for j=1:size(diagr_cs,1)
   tt_cs=[tt_cs;'\input{'+diagr_cs(j,2)+'_cos/'+diagr_cs(j,2)+'}';'\pagebreak']
   TEXINPUTS=TEXINPUTS+':./'+diagr_cs(j,2)+'_cos/'
   generate_diagr_tex_file(diagr_cs(j,2),'guide',lang);
end

tt_ds=[]//discrete system
```

```
for j=1:size(diagr_ds,1)
   tt_ds=[tt_ds;'\input{'+diagr_ds(j,2)+'_cos/'+diagr_ds(j,2)+'}';'\pagebreak']
   TEXINPUTS=TEXINPUTS+':./'+diagr_ds(j,2)+'_cos/'
   generate_diagr_tex_file(diagr_ds(j,2),'guide',lang);
end

tt_os=[];//Open loop models of oscillator
for j=1:size(diagr_os,1)
   tt_os=[tt_os;'\input{'+diagr_os(j,2)+'_cos/'+diagr_os(j,2)+'}';'\pagebreak']
   TEXINPUTS=TEXINPUTS+':./'+diagr_os(j,2)+'_cos/'
   generate_diagr_tex_file(diagr_os(j,2),'guide',lang);
end

tt_is=[]//integer synthesizer
for j=1:size(diagr_is,1)
   tt_is=[tt_is;'\input{'+diagr_is(j,2)+'_cos/'+diagr_is(j,2)+'}';'\pagebreak']
   TEXINPUTS=TEXINPUTS+':./'+diagr_is(j,2)+'_cos/'
   generate_diagr_tex_file(diagr_is(j,2),'guide',lang);
 end

tt_fs=[]//fractional synthesizer
for j=1:size(diagr_fs,1)
   tt_fs=[tt_fs;'\input{'+diagr_fs(j,2)+'_cos/'+diagr_fs(j,2)+'}';'\pagebreak']
   TEXINPUTS=TEXINPUTS+':./'+diagr_fs(j,2)+'_cos/'
   generate_diagr_tex_file(diagr_fs(j,2),'guide',lang);
end

tt_PSK=[]//PSK Transmission
for j=1:size(diagr_PSK,1)
   tt_PSK=[tt_PSK;'\input{'+diagr_PSK(j,2)+'_cos/'+diagr_PSK(j,2)+'}';'\pagebreak']
   TEXINPUTS=TEXINPUTS+':./'+diagr_PSK(j,2)+'_cos/'
   generate_diagr_tex_file(diagr_PSK(j,2),'guide',lang);
end

tt_SD=[]//Sigma Delta Transmisson
for j=1:size(diagr_SD,1)
   tt_SD=[tt_SD;'\input{'+diagr_SD(j,2)+'_cos/'+diagr_SD(j,2)+'}';'\pagebreak']
   TEXINPUTS=TEXINPUTS+':./'+diagr_SD(j,2)+'_cos/'
   generate_diagr_tex_file(diagr_SD(j,2),'guide',lang);
end

tt_FSK=[]//FSK Transmission

tt_FSK_chaos=[]//FSK Chaos Transmission
for j=1:size(diagr_FSK_chaos,1)
   tt_FSK_chaos=[tt_FSK_chaos;'\input{'+diagr_FSK_chaos(j,2)+'_cos/'+diagr_FSK_chaos(j,2)+'}';'\pagebreak']
   TEXINPUTS=TEXINPUTS+':./'+diagr_FSK_chaos(j,2)+'_cos/'
   generate_diagr_tex_file(diagr_FSK_chaos(j,2),'guide',lang);
end

tt_elec=[] //Electrical circuit
for j=1:size(diagr_elec,1)
   tt_elec=[tt_elec;'\input{'+diagr_elec(j,2)+'_cos/'+diagr_elec(j,2)+'}';'\pagebreak']
   TEXINPUTS=TEXINPUTS+':./'+diagr_elec(j,2)+'_cos/'
   generate_diagr_tex_file(diagr_elec(j,2),'guide',lang);
end

tt_sim_chaos=[] //Simulations of chaotic systems
for j=1:size(sim_chaos,1)
   tt_sim_chaos=[tt_sim_chaos;'\input{'+sim_chaos(j,2)+'/'+sim_chaos(j,2)+'}';'\pagebreak']
   TEXINPUTS=TEXINPUTS+':./'+sim_chaos(j,2)+'/'
   generate_sim_tex_file(sim_chaos(j,2),'guide',lang);
end

tt_sim_synthe=[] //Simulations of oscillators & Phase Locked Loop
for j=1:size(sim_synthe,1)
   tt_sim_synthe=[tt_sim_synthe;'\input{'+sim_synthe(j,2)+'/'+sim_synthe(j,2)+'}';'\pagebreak']
   TEXINPUTS=TEXINPUTS+':./'+sim_synthe(j,2)+'/'
   generate_sim_tex_file(sim_synthe(j,2),'guide',lang);
end

tt_sim_PSK=[] //Simulations of communication systems
for j=1:size(sim_PSK,1)
   tt_sim_PSK=[tt_sim_PSK;'\input{'+sim_PSK(j,2)+'/'+sim_PSK(j,2)+'}';'\pagebreak']
   TEXINPUTS=TEXINPUTS+':./'+sim_PSK(j,2)+'/'
   generate_sim_tex_file(sim_PSK(j,2),'guide',lang);
end

tt_c=[]//general conclusion

//main tex file
if lang=='fr' then
  txt=[tt_head;'\title{\Huge ""Modnum""\\'
       'Boîte à outils Scilab\\'
       'pour les systèmes de communication\\'
       'Guide de l''utilisateur\\'
       '\small Version provisoire}'];
else
  txt=[tt_head;'\title{\Huge ""Modnum""\\'
       'Scilab toolbox\\'
       'for the communication systems\\'
       'User''s Guide\\'
       '\small Draft Version}'];
end

txt=[txt
     tt_title
     tt_i
     tt_contents];

//%%%%%%%%%%%%%
```

```
//Scicos Diagram
//%%%%%%%%%%%%%%
if lang==’fr’ then
  txt=[txt
       ’ %%Diagrammes Scicos’
       ’ \part{Diagrammes Scicos}’];
else
  txt=[txt
       ’ %%Scicos Diagrams’
       ’ \part{Scicos Diagrams}’];
end
txt=[txt
     ’ \null’
     ’ \newpage’
     ’’];

//%%%%%%%%%%%%%%
//Chaotic system
if lang==’fr’ then
       txt=[txt;’ \chapter{Systèmes dynamiques chaotiques}’];
else
       txt=[txt;’ \chapter{Chaotic dynamical systems}’];
end

//Continous time systems
if lang==’fr’ then
  txt=[txt;’ \addcontentsline{toc}{chapter}{Systèmes à temps continu}’];
else
  txt=[txt;’ \addcontentsline{toc}{chapter}{Continous time systems}’];
end
txt=[txt;" "+tt_cs];

//Discrete time systems
if lang==’fr’ then
  txt=[txt;’ \addcontentsline{toc}{chapter}{Systèmes à temps discret}’];
else
  txt=[txt;’ \addcontentsline{toc}{chapter}{Discrete time systems}’];
end
txt=[txt;" "+tt_ds];

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//Oscillators and Phase Locked Loop systems
if lang==’fr’ then
  txt=[txt;’ \chapter{Oscillateurs et boucles à verrouillage de phase}’];
else
  txt=[txt;’ \chapter{Oscillators and Phase Locked Loop systems}’];
end

//Open loop models of oscillators
if lang==’fr’ then
  txt=[txt;’ \addcontentsline{toc}{chapter}{Modèles boucle ouverte d’’oscillateurs}’];
else
  txt=[txt;’ \addcontentsline{toc}{chapter}{Open loop models of oscillators}’];
end
txt=[txt;" "+tt_os];

//Integer N Frequency synthesizers
if lang==’fr’ then
  txt=[txt;’ \addcontentsline{toc}{chapter}{Synthétiseurs de fréquence à rapport de division N entier}’];
else
  txt=[txt;’ \addcontentsline{toc}{chapter}{Integer N Frequency synthesizers}’];
end
txt=[txt;" "+tt_is];

//Fractional N/N+1 Frequency synthesizers
if lang==’fr’ then
  txt=[txt;’ \addcontentsline{toc}{chapter}{Synthétiseurs de fréquence à rapport de division fractionnaire}’];
else
  txt=[txt;’ \addcontentsline{toc}{chapter}{Fractional N/N+1 Frequency synthesizers}’];
end
txt=[txt
     " "+tt_fs
     ’ \addtocontents{toc}{\protect\pagebreak}’];

//%%%%%%%%%%%%%%%%%%%%%%
//Communication systems
if lang==’fr’ then
  txt=[txt;’ \chapter{Systèmes de communication}’];
else
  txt=[txt;’ \chapter{Communication systems}’];
end

//PSK/QAM Transmission
if lang==’fr’ then
  txt=[txt;’ \addcontentsline{toc}{chapter}{Transmission PSK/QAM}’];
else
  txt=[txt;’ \addcontentsline{toc}{chapter}{PSK/QAM Transmission}’];
end
txt=[txt;" "+tt_PSK];

//Delta-Sigma Transmission
if lang==’fr’ then
  txt=[txt;’ \addcontentsline{toc}{chapter}{Transmission Sigma-Delta}’];
else
  txt=[txt;’ \addcontentsline{toc}{chapter}{Delta-Sigma Transmission}’];
end
txt=[txt;" "+tt_SD];

//chaotic FSK Transmission
if lang==’fr’ then
  txt=[txt;’ \addcontentsline{toc}{chapter}{Transmission chaotique FSK}’];
```

```
else
  txt=[txt;' \addcontentsline{toc}{chapter}{FSK chaotic transmission}'];
end
txt=[txt;" "+tt_FSK_chaos];

//%%%%%%%%%%%%%%%%%%%%%%
//Electrical circuits
if lang=='fr' then
  txt=[txt;' \chapter{Circuits électriques}'];
else
  txt=[txt;' \chapter{Electrical circuits}'];
end
txt=[txt
     " "+tt_elec
     ''
     ' \setcounter{chapter}{0}'];

//%%%%%%%%%%%%%%%%%%%%%%%%%
//Simulation Scilab scripts
//%%%%%%%%%%%%%%%%%%%%%%%%%
if lang=='fr' then
 txt=[txt
      ' %%Scripts Scilab de simulations'
      ' \part{Scripts Scilab de simulations}'];
else
 txt=[txt
      ' %%Simulation Scilab scripts'
      ' \part{Simulation Scilab scripts}'];
end
txt=[txt
     ' \null'
     ' \newpage'
     ''];

//
if lang=='fr' then
  txt=[txt;' \chapter{Simulations de systèmes chaotiques}'];
else
  txt=[txt;' \chapter{Simulations of chaotic systems}'];
end
txt=[txt;" "+tt_sim_chaos];

//
if lang=='fr' then
  txt=[txt;' \chapter{Simulations d''oscillateurs et de boucles à verrouillage de phase}'];
else
  txt=[txt;' \chapter{Simulations of oscillators and Phase Locked Loops }'];
end
txt=[txt;" "+tt_sim_synthe];

//
if lang=='fr' then
  txt=[txt;' \chapter{Simulations de systèmes de communication}'];
else
  txt=[txt;' \chapter{Simulations of communication systems }'];
end
txt=[txt;" "+tt_sim_PSK;'\end{document}']

//Write final LaTeX File
mputl(txt,'user.tex')

/////////Compile//////////
//mv report.cls to current directory
unix_g(cp_cmd+man_path+'tex/'+'report.cls'+' .')
//define cmd
latex_cmd='latex -interaction=nonstopmode user.tex ';
export_cmd='export TEXINPUTS='+TEXINPUTS;
divps_cmd='dvips -o user.ps user.dvi';
gv_cmd='gv user.ps';
//latex+dvips+gv
new_tt=export_cmd+' ; '+latex_cmd+' ; '+latex_cmd+' ; '+latex_cmd+' ; '+divps_cmd+' ; '+gv_cmd;
mputl(new_tt,'compile.sh');
unix_g('chmod a+x compile.sh');
pause
unix_g('./compile.sh');
//ps2pdf
ps2pdf_cmd='ps2pdf14 user.ps';
unix_g(ps2pdf_cmd);
unix_g('cp user.pdf modnum_user.pdf')
unix_g(mv_cmd+'./modnum_user.pdf '+pdf_path+lang+'/');

////////////Clean//////////
//Erase template LaTeX file
unix_g('rm -f user.*');
unix_g('rm -f compile.sh');
unix_g(rm_cmd+'report.cls');
//Erase diagram directories
for j=1:size(diagr_cs,1) //continous system
  unix_g(rm_cmd+diagr_cs(j,2)+'_cos');
end
for j=1:size(diagr_ds,1) //discrete system
   unix_g(rm_cmd+diagr_ds(j,2)+'_cos');
end
for j=1:size(diagr_os,1) //Open loop models of oscillator
   unix_g(rm_cmd+diagr_os(j,2)+'_cos');
end
for j=1:size(diagr_is,1) //integer synthesizer
    unix_g(rm_cmd+diagr_is(j,2)+'_cos');
end
for j=1:size(diagr_fs,1) //fractional synthesizer
   unix_g(rm_cmd+diagr_fs(j,2)+'_cos');
```

```
end
for j=1:size(diagr_PSK,1) //PSK Transmission
   unix_g(rm_cmd+diagr_PSK(j,2)+'_cos');
end
for j=1:size(diagr_SD,1) //Sigma Delta Transmisson
   unix_g(rm_cmd+diagr_SD(j,2)+'_cos');
end
for j=1:size(diagr_elec,1) //Electrical circuit
   unix_g(rm_cmd+diagr_elec(j,2)+'_cos');
end
//Erase simulation script directories
for j=1:size(sim_chaos,1) //Simulations of chaotic systems
   unix_g(rm_cmd+sim_chaos(j,2));
end
for j=1:size(sim_synthe,1) //Simulations of oscillators & Phase Locked Loop
   unix_g(rm_cmd+sim_synthe(j,2));
end
for j=1:size(sim_PSK,1) //Simulations of communication systems
   unix_g(rm_cmd+sim_PSK(j,2));
end


elseif flag=='internal' then
//////////////////
//Internal guide
//////////////////

tt_i=[];//introduction

tt_sce=[] //utilities scripts
TEXINPUTS='$TEXINPUTS:.';
 if sce_all<>[] then
  for j=1:size(sce_all,1)
   tt_sce=[tt_sce;'\input{'+sce_all(j,2)+'_sce/'+sce_all(j,2)+'}']
   TEXINPUTS=TEXINPUTS+':./'+sce_all(j,2)+'_sce/';
   generate_sce_tex_file(sce_all(j,2),'guide',lang);
  end
 end

tt_util_sci=[]; //scilab utilities macros
Librep=return_dir_in_dir(tt_ml,mac_path)
for j=1:size(Librep,1)
  LibName=basename(part(Librep(j),1:length(Librep(j))-1));
  if grep(LibName,lib_build)<>[] then
    tt_util_sci=[tt_util_sci;'\input{'+LibName+'_scilib/'+LibName+'}';];
    TEXINPUTS=TEXINPUTS+':./'+LibName+'_scilib/';
    generate_scilib_tex_file(LibName,'guide',lang);
    //Scilab function
    lisf=return_ext_file_in_dir(tt_ml,Librep(j),'.sci');
    for i=1:size(lisf,1)
       name=basename(lisf(i,1));
       generate_scifun_tex_file(name,'guide',lang);
       tt_util_sci=[tt_util_sci;'\input{'+name+'_sci/'+name+'}'];
       TEXINPUTS=TEXINPUTS+':./'+name+'_sci/';
    end
  end
end

tt_doc_gen_sci=[]; //scilab documentation generator libraries
Librep=return_dir_in_dir(tt_ml,mac_path)
for j=1:size(Librep,1)
  LibName=basename(part(Librep(j),1:length(Librep(j))-1));
  if grep(LibName,lib_gen_doc)<>[] then
    tt_doc_gen_sci=[tt_doc_gen_sci;'\input{'+LibName+'_scilib/'+LibName+'}';];
    TEXINPUTS=TEXINPUTS+':./'+LibName+'_scilib/';
    generate_scilib_tex_file(LibName,'guide',lang);
    //Scilab function
    lisf=return_ext_file_in_dir(tt_ml,Librep(j),'.sci');
    for i=1:size(lisf,1)
       name=basename(lisf(i,1));
       generate_scifun_tex_file(name,'guide',lang);
       tt_doc_gen_sci=[tt_doc_gen_sci;'\input{'+name+'_sci/'+name+'}'];
       TEXINPUTS=TEXINPUTS+':./'+name+'_sci/';
    end
  end
end

tt_rout=[]; //low_level routine
generate_scilib_tex_file(mod_num_rout_lib,'guide',lang);
tt_rout=[tt_rout;'\input{'+mod_num_rout_lib+'_scilib/'+mod_num_rout_lib+'}';];
TEXINPUTS=TEXINPUTS+':./'+mod_num_rout_lib+'_scilib/';
lisf_rout=return_ext_file_in_dir(tt_ml,low_rout_path,".c")
for i=1:size(lisf_rout,1)
  name=basename(lisf_rout(i,1));
  generate_rout_tex_file(name,'guide',lang)
  tt_rout=[tt_rout;'\input{'+name+'_rout/'+name+'}'];
  TEXINPUTS=TEXINPUTS+':./'+name+'_rout/';
end

tt_c=[]//conclusion

//main tex file
if lang=='fr' then
  txt=[tt_head;' \title{\Huge Mod\_num\\Guide interne}'];
else
  txt=[tt_head;' \title{\Huge Mod\_num\\Internals Guide}'];
end
if lang=='fr' then
  txt=[tt_head;'\title{\Huge ""Modnum""\\'
       'Boîte à outils Scilab\\'
       'pour les systèmes de communication\\'
```

```
       'Guide interne\\'
       '\small Version provisoire}'];
else
  txt=[tt_head;'\title{\Huge ""Modnum""\\'
       'Scilab toolbox\\'
       'for the communication systems\\'
       'Internals Guide\\'
       '\small Draft Version}'];
end
txt=[txt
     tt_title
     tt_i
     tt_contents];
if lang=='fr' then
  txt=[txt
       ' %%Scripts et macros Scilab utilitaires'
       ' \part{Scripts et macros Scilab utilitaires}'];
else
  txt=[txt
       ' %%Utilities Scilab scripts and macros'
       ' \part{Utilities Scilab scripts and macros}';]
end
txt=[txt
     ' \null'
     ' \newpage'
     ''']
if lang=='fr' then
   txt=[txt;' \chapter{Scripts Scilab utilitaires}'];
else
   txt=[txt;' \chapter{Utilities Scilab scripts}'];
end
txt=[txt
     " "+tt_sce
     " "+tt_util_sci
     ''
     ' \setcounter{chapter}{0}'];
if lang=='fr' then
   txt=[txt
        ' %%Librairies Scilab du générateur de documentation'
        ' \part{Librairies Scilab du générateur de documentation}'];
else
   txt=[txt
        ' %%Documentation generator macro libraries'
        ' \part{Documentation generator macro libraries}'];
end
txt=[txt
     ' \null'
     ' \newpage'
     ''
     " "+tt_doc_gen_sci
     ''
     ' \newpage'
     ' \null'
     ' \newpage'
     ' \setcounter{chapter}{0}'];
if lang=='fr' then
   txt=[txt
        ' %%Routines de calcul bas-niveau'
        ' \part{Routines de calcul bas-niveau}'];
else
   txt=[txt
        ' %%Low level routines'
        ' \part{Low level routines}'];
end
txt=[txt
     ' \null'
     ' \newpage'
     ''']
txt=[txt;" "+tt_rout;'\end{document}'];
mputl(txt,'internals.tex')

////////Compile//////////
//mv report.cls to current directory
unix_g(cp_cmd+man_path+'tex/'+'report.cls'+' .')
//define cmd
latex_cmd='latex -interaction=nonstopmode internals.tex ';
export_cmd='export TEXINPUTS='+TEXINPUTS;
divps_cmd='dvips -o internals.ps internals.dvi';
gv_cmd='gv internals.ps';
//latex+dvips+gv
new_tt=export_cmd+' ; '+latex_cmd+' ; '+latex_cmd+' ; '+latex_cmd+' ; '+divps_cmd+' ; '+gv_cmd;
mputl(new_tt,'compile.sh');
unix_g('chmod a+x compile.sh');
pause
unix_g('./compile.sh');
//ps2pdf
ps2pdf_cmd='ps2pdf14 internals.ps';
unix_g(ps2pdf_cmd);
unix_g('cp internals.pdf modnum_internals.pdf')
unix_g(mv_cmd+'./modnum_internals.pdf '+pdf_path+lang+'/');

///////////Clean//////////
//Erase template LaTeX file
unix_g('rm -f internals.*');
unix_g('rm -f compile.sh');
unix_g(rm_cmd+'report.cls');

//Erase scilab script tex file
if sce_all<>[] then
  for j=1:size(sce_all,1)
    unix_g(rm_cmd+sce_all(j,2)+'_sce');
```

```
    end
  end

  //Erase scilab macro tex files
  Librep=return_dir_in_dir(tt_ml,mac_path)
  for j=1:size(Librep,1)
    LibName=basename(part(Librep(j),1:length(Librep(j))-1));
    if grep(LibName,lib_build)<>[] then
      unix_g(rm_cmd+LibName+'_scilib');
      lisf=return_ext_file_in_dir(tt_ml,Librep(j),'.sci');
      for i=1:size(lisf,1)
         name=basename(lisf(i,1));
         unix_g(rm_cmd+name+'_sci');
      end
    end
  end
  Librep=return_dir_in_dir(tt_ml,mac_path) //scilab documentation generator libraries
  for j=1:size(Librep,1)
    LibName=basename(part(Librep(j),1:length(Librep(j))-1));
    if grep(LibName,lib_gen_doc)<>[] then
      unix_g(rm_cmd+LibName+'_scilib');
      //Scilab function
      lisf=return_ext_file_in_dir(tt_ml,Librep(j),'.sci');
      for i=1:size(lisf,1)
         name=basename(lisf(i,1));
         unix_g(rm_cmd+name+'_sci');
      end
    end
  end

  //Erase routine tex files
  unix_g(rm_cmd+mod_num_rout_lib+'_scilib');
  for i=1:size(lisf_rout,1)
    name=basename(lisf_rout(i,1));
    unix_g(rm_cmd+name+'_rout');
  end

 end
endfunction
```

2.7 Create html documentation of the toolbox

- **Name:** generate_mod_num_html
- **Library:** generate_doc - Main library of the documentation generator

2.7.1 Calling Sequence

`generate_mod_num_html(flag,lang)`

2.7.2 Parameters

- **flag :** string. set the type of man page
  - **'block' :** for interfacing function of scicos block
  - **'pal' :** for a palette (.cosf file)
  - **'diagr' :** for a scicos diagram (.cos file)
  - **'scilib' :** for a library of scilab macros
  - **'sci' :** for a scilab macro.
  - **'rout' :** for computational routine
  - **'sim' :** for scilab simulation script (_sim.sce file)
  - **'sce' :** for scilab script (.sce file)
- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page

2.7.3 File content

```
//generate_mod_num_html
//fonction qui génère les man pages de mod_num
//Entrée : flag : 'diagr'
//                'sce'
//                'block'
//                'sci'
//                'rout'
//                'sim'
//                'sce'
//                'what'
//                'all'
//Sortie : néant
function generate_mod_num_html(flag,lang)

[lsh,rsh]=argn(0)
if rsh<2 then
  if ~exists('lang') then
    lang='eng'
  elseif lang<>'eng' & lang<>'fr' then
    lang='eng'
  end
end

flag_diagr=%f;
flag_sce=%f;
flag_block=%f;
flag_sci=%f;
flag_rout=%f;
flag_sim=%f
flag_what=%f;
flag_all=%f;

for i=1:size(flag,1)
 select flag(i)
  case 'diagr'
   flag_diagr=%t;
  case 'sce'
   flag_sce=%t;
  case 'block'
   flag_block=%t;
  case 'sci'
   flag_sci=%t;
  case 'rout'
   flag_rout=%t;
  case 'sim'
   flag_sim=%t;
  case 'what'
   flag_what=%t;
  case 'all'
   flag_diagr=%t;flag_sce=%t;flag_block=%t;
   flag_sci=%t;flag_rout=%t;flag_what=%t;
   flag_sim=%t;flag_all=%t;
  else
   printf("Invalid flag\n")
   abort
 end
end

//générère les fichiers xml
generate_mod_num_xml(flag,lang);

//récupère les données
import_data_to_file(xml_path+lang+'/','all',data_path+lang+'/');

//return master list of files and directories
tt_ml=return_master_list(MODNUM);

//Scicos diagram
if flag_diagr then
 generate_html_file(diagr_all(:,2),'diagr',lang);
end

//Scicos Palette
if flag_block then
 Palrep=return_dir_in_dir(tt_ml,pal_path)
 for j=1:size(Palrep,1)
   PalName=basename(part(Palrep(j),1:length(Palrep(j))-1));
   generate_html_file(PalName,'pal',lang)
   //Scicos block
   lisf=return_ext_file_in_dir(tt_ml,Palrep(j),'.sci');
   for i=1:size(lisf,1)
     name=basename(lisf(i,1));
     generate_html_file(name,'block',lang)
   end
 end
end

 //Scilab macro
if flag_sci then
 Librep=return_dir_in_dir(tt_ml,mac_path)
 for j=1:size(Librep,1)
  LibName=basename(part(Librep(j),1:length(Librep(j))-1));
  generate_html_file(LibName,'scilib',lang)
  //Scilab function
  lisf=return_ext_file_in_dir(tt_ml,Librep(j),'.sci');
  for i=1:size(lisf,1)
     name=basename(lisf(i,1));
     generate_html_file(name,'sci',lang)
  end
```

```
 end
 //Mod_num_sci_lib
 generate_html_file(mod_num_sci_lib,'scilib',lang);
 for i=1:size(modnum_sci_func,1)
  generate_html_file(modnum_sci_func(i),'sci',lang)
 end
end

//low level routines
if flag_rout then
  generate_html_file(mod_num_rout_lib,'scilib',lang);
  lisf_rout=return_ext_file_in_dir(tt_ml,low_rout_path,".c")
  for i=1:size(lisf_rout,1)
    name=basename(lisf_rout(i,1));
    generate_html_file(name,'rout',lang)
  end
end

//Scilab simulation scripts
if flag_sim then
 generate_html_file(sim_all(:,2),'sim',lang);
end

//Scilab scripts
if flag_sce then
 generate_html_file(sce_all(:,2),'sce',lang);
end

//Main html page
if flag_what then
 generate_swhatis(html_path+lang+'/');
end

endfunction
```

2.8 Add short decription here

- **Name:** generate_mod_num_web
- **Library:** generate_doc - Main library of the documentation generator

2.8.1 Calling Sequence

```
generate_mod_num_web(path)
```

2.8.2 Parameters

- **path :** 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 Example

```
Add here scilab instructions and comments
```

2.8.5 File content

```
//generate_mod_num_web
//Fonction qui crée le répertoire
//du sîte web dans de la boîte à outils
//Entrée : path : chemin cible du sîte html

function generate_mod_num_web(path)

   //Test la présence de l'arborescence
   //du répertoire MODNUM+'/../mod_num_web'
   if fileinfo(path)==[] then
      printf('Create '+path+' ...');
      unix_g(mkdir_cmd+path);
      printf(" Done\n");
   end
   if fileinfo(path+'/src')==[] then
      printf('Create '+path+'/src'+' ...');
      unix_g(mkdir_cmd+path+'/src');
      printf(" Done\n");
   end
```

```
if fileinfo(path+'/src/win')==[] then
   printf('Create '+path+'/src/win'+' ...');
   unix_g(mkdir_cmd+path+'/src/win');
   printf(" Done\n");
end
if fileinfo(path+'/src/linux')==[] then
   printf('Create '+path+'/src/linux'+' ...');
   unix_g(mkdir_cmd+path+'/src/linux');
   printf(" Done\n");
end
if fileinfo(path+'/bin')==[] then
   printf('Create '+path+'/bin'+' ...');
   unix_g(mkdir_cmd+path+'/bin');
   printf("Done\n");
end
if fileinfo(path+'/bin/win')==[] then
   printf('Create '+path+'/bin/win'+' ...');
   unix_g(mkdir_cmd+path+'/bin/win');
   printf(" Done\n");
end
if fileinfo(path+'/bin/linux')==[] then
   printf('Create '+path+'/bin/linux'+' ...');
   unix_g(mkdir_cmd+path+'/bin/linux');
   printf(" Done\n");
end
if fileinfo(path+'/web')==[] then
   printf('Create '+path+'/web'+' ...');
   unix_g(mkdir_cmd+path+'/web');
   printf(" Done\n");
end
if fileinfo(path+'/web/eng')==[] then
   printf('Create '+path+'/web/eng'+' ...');
   unix_g(mkdir_cmd+path+'/web/eng');
   printf(" Done\n");
end
if fileinfo(path+'/web/fr')==[] then
   printf('Create '+path+'/web/fr'+' ...');
   unix_g(mkdir_cmd+path+'/web/fr');
   printf(" Done\n");
end
if fileinfo(path+'/man')==[] then
   printf('Create '+path+'/man'+' ...');
   unix_g(mkdir_cmd+path+'/man');
   printf(" Done\n");
end
if fileinfo(path+'/man/htm')==[] then
   printf('Create '+path+'/man/htm'+' ...');
   unix_g(mkdir_cmd+path+'/man/htm');
   printf(" Done\n");
end
if fileinfo(path+'/man/htm/eng')==[] then
   printf('Create '+path+'/man/htm/eng'+' ...');
   unix_g(mkdir_cmd+path+'/man/htm/eng');
   printf(" Done\n");
end
if fileinfo(path+'/man/htm/fr')==[] then
   printf('Create '+path+'/man/htm/fr'+' ...');
   unix_g(mkdir_cmd+path+'/man/htm/fr');
   printf(" Done\n");
end
if fileinfo(path+'/man/pdf')==[] then
   printf('Create '+path+'/man/pdf'+' ...');
   unix_g(mkdir_cmd+path+'/man/pdf');
   printf(" Done\n");
end
if fileinfo(path+'/man/pdf/eng')==[] then
   printf('Create '+path+'/man/pdf/eng'+' ...');
   unix_g(mkdir_cmd+path+'/man/pdf/eng');
   printf(" Done\n");
end
if fileinfo(path+'/man/pdf/fr')==[] then
   printf('Create '+path+'/man/pdf/fr'+' ...');
   unix_g(mkdir_cmd+path+'/man/pdf/fr');
   printf(" Done\n");
end

//Test la présence des fichiers web
//Version anglaise
if fileinfo(web_path+'/eng')<>[] then
   printf('Copy '+web_path+'eng in '+path+'/web/eng'+' ...');
   unix_g(cp_cmd+web_path+'eng/*.* '+path+'/web/eng');
   printf(" Done\n");
end
//Version française
if fileinfo(web_path+'/fr')<>[] then
   printf('Copy '+web_path+'fr in '+path+'/web/fr'+' ...');
   unix_g(cp_cmd+web_path+'fr/*.* '+path+'/web/fr');
   printf(" Done\n");
end

//Test la présence des fichiers readme
//Version anglaise
if fileinfo(MODNUM+'/README_EN')<>[] then
   printf('Copy '+MODNUM+'/README_EN in '+path+' ...');
   unix_g(cp_cmd+MODNUM+'/README_EN '+path);
   printf(" Done\n");
end
//Version française
if fileinfo(MODNUM+'/README_FR')<>[] then
   printf('Copy '+MODNUM+'/README_FR in '+path+' ...');
   unix_g(cp_cmd+MODNUM+'/README_FR '+path);
```

```
      printf(" Done\n");
   end

   //Test la présence des fichiers man/htm
   //Version anglaise
   if fileinfo(html_path+'/eng')<>[] then
      printf('Copy '+html_path+'eng in '+path+'/man/htm/eng'+' ...');
      unix_g(cp_cmd+html_path+'eng/*.* '+path+'/man/htm/eng/');
      printf(" Done\n");
   end
   //Version française
   if fileinfo(html_path+'/fr')<>[] then
      printf('Copy '+html_path+'fr in '+path+'/man/htm/fr'+' ...');
      unix_g(cp_cmd+html_path+'fr/*.* '+path+'/man/htm/fr');
      printf(" Done\n");
   end

   //Test la présence des fichiers man/pdf
   //Version anglaise
   if fileinfo(pdf_path+'eng')<>[] then
      printf('Copy '+pdf_path+'eng in '+path+'/man/pdf/eng'+' ...');
      unix_g(cp_cmd+pdf_path+'eng/*.* '+path+'/man/pdf/eng');
      printf(" Done\n");
   end
   //Version française
   if fileinfo(pdf_path+'fr')<>[] then
      printf('Copy '+pdf_path+'fr in '+path+'/man/pdf/fr'+' ...');
      unix_g(cp_cmd+pdf_path+'fr/*.* '+path+'/man/pdf/fr');
      printf(" Done\n");
   end

endfunction
```

2.8.6 Used function(s)

Add here the used function name and references

2.8.7 See Also

- add a key here - ()
- add a key here - ()

2.8.8 Bibliography

Add here the function bibliography if any

2.9 Create xml documentation files of the toolbox

- **Name:** generate_mod_num_xml
- **Library:** generate_doc - Main library of the documentation generator

2.9.1 Calling Sequence

```
generate_mod_num_xml(flag,lang)
```

2.9.2 Parameters

- **flag :** string. set the type of man page
  - **'block' :** for interfacing function of scicos block
  - **'pal' :** for a palette (.cosf file)
  - **'diagr' :** for a scicos diagram (.cos file)
  - **'scilib' :** for a library of scilab macros
  - **'sci' :** for a scilab macro.
  - **'rout' :** for computational routine
  - **'sim' :** for scilab simulation script (_sim.sce file)

  - **'sce' :** for scilab script (.sce file)
  - **'all' :** for all

- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page

2.9.3 File content

```
//generate_mod_num_xml
//fonction qui génère les man pages de mod_num
//format xml
//Entrée : flag : 'diagr'
//                'sce'
//                'block'
//                'sci'
//                'rout'
//                'all'
//Sortie : néant
function generate_mod_num_xml(flag,lang)

[lsh,rsh]=argn(0)
if rsh<2 then
   if ~exists('lang') then
     lang='eng'
   elseif lang<>'eng' & lang<>'fr' then
     lang='eng'
   end
end

flag_diagr=%f;
flag_sce=%f;
flag_block=%f;
flag_sci=%f;
flag_rout=%f;
flag_sim=%f;
flag_what=%f;

for i=1:size(flag,1)
 select flag(i)
  case 'diagr'
   flag_diagr=%t;
  case 'sce'
   flag_sce=%t;
  case 'block'
   flag_block=%t;
  case 'sci'
   flag_sci=%t;
  case 'rout'
   flag_rout=%t;
  case 'sim'
   flag_sim=%t;
  case 'what'
   flag_what=%t;
  case 'all'
   flag_diagr=%t;flag_sce=%t;flag_block=%t;
   flag_sci=%t;flag_rout=%t;flag_what=%t;
   flag_sim=%t;
  else
   printf("Invalid flag\n")
   abort
 end
end

if flag_diagr then
 //Scicos diagram
 generate_xml_file(diagr_all(:,2),'diagr',lang)
end

if flag_block then
 //Scicos Palette
 Palrep=return_dir_in_dir(tt_ml,pal_path)
 for j=1:size(Palrep,1)
   PalName=basename(part(Palrep(j),1:length(Palrep(j))-1));
   generate_xml_file(PalName,'pal',lang)
   lisf=return_ext_file_in_dir(tt_ml,Palrep(j),'.sci');
   for i=1:size(lisf,1)
     name=basename(lisf(i,1));
     //Scicos block
     generate_xml_file(name,'block',lang)
   end
 end
end

if flag_sci then
 //Scilab library
 Librep=return_dir_in_dir(tt_ml,mac_path)
 for j=1:size(Librep,1)
  LibName=basename(part(Librep(j),1:length(Librep(j))-1));
  generate_xml_file(LibName,'scilib',lang)
  lisf=return_ext_file_in_dir(tt_ml,Librep(j),'.sci');
  for i=1:size(lisf,1)
     name=basename(lisf(i,1));
```

```
    //Scilab function
    generate_xml_file(name,'sci',lang)
  end
 end
 //Mod_num_sci_lib
 generate_xml_file(mod_num_sci_lib,'scilib',lang);
 for i=1:size(modnum_sci_func,1)
  generate_xml_file(modnum_sci_func(i),'sci',lang)
 end
end

//low level routines
if flag_rout then
  generate_xml_file(mod_num_rout_lib,'scilib',lang);
  lisf_rout=return_ext_file_in_dir(tt_ml,low_rout_path,".c")
  for i=1:size(lisf_rout,1)
    name=basename(lisf_rout(i,1));
    generate_xml_file(name,'rout',lang)
  end
end

//scilab simulation scripts
if flag_sim then
  generate_xml_file(sim_all(:,2),'sim',lang)
end

//scilab scripts
if flag_sce then
  generate_xml_file(sce_all(:,2),'sce',lang)
end
endfunction
```

2.10 Create main tex file of palette

- **Name:** generate_pals_tex_file
- **Library:** generate_doc - Main library of the documentation generator

2.10.1 Calling Sequence

```
txt = generate_pals_tex_file(PalName,flag,lang)
```

2.10.2 Parameters

- **PalName :** string. name of the palette
- **flag :** string. set the type of tex files to produce
  - **'html' :** to produce tex file for html format
  - **'guide' :** to produce tex file for paper format
- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page
- **txt :** vector of strings. the output text of the tex file

2.10.3 File content

```
//generate_pals_tex_file
//
//Fonction qui genere le fichier tex principal
//d'une palette scicos
//Entrée : PalName : Nom de la palette
//         flag : 'html' pour générer une page d'aide html
//                'guide' pour générer un page d'aide ps
//         lang : 'eng pour de l'anglais (default)
//                'fr' pour du français
function txt=generate_pals_tex_file(PalName,flag,lang)

  //verify the 'lang' right parameter
  [lsh,rsh]=argn(0)
  if rsh<3 then
    if ~exists('lang') then
     lang='eng'
    elseif lang<>'eng' & lang<>'fr' then
     lang='eng'
```

```
    end
  end

  //Generate auxiliary tex files
  PalName=generate_aux_tex_file(PalName,'pal',flag,lang);

  //define title of paragraph
  tt_title=[
            ''             //tt1  : header du fichier tex
            ''             //tt2  : figure de la palette (.eps)
            'Package'      //tt3  : Package
            'Description'  //tt4  : Description (_long)
            'Blocks'       //tt5  : block contents
            'See Also'     //tt6  : See Also (_see_also)
            'Authors'      //tt7  : Authors (_authors)
            ''             //tt8  : End of tex file
           ]
  //change language of title
  if lang=='fr' then
     tt_title=change_lang_title(lang,tt_title);
  end

  //define level of paragraph
  if flag=='html' then
    tex_title='\subsection{'+tt_title+'}'
  else
    tex_title='\subsection{'+tt_title+'}'
  end

  for i=1:size(PalName,1) //for each file
    for j=1:8 execstr('tt'+string(j)+'=[]'),end //for each paragraph

    if fileinfo(PalName(i,1)+'_cosf/'+PalName(i,1)+'_pals.tex')<>[] then //figure of palette
       tt2=['\input{'+PalName(i,1)+'_pals}']
    end
    if fileinfo(PalName(i,1)+'_cosf/'+PalName(i,1)+'_long.tex')<>[] then //Description
       tt4=[tex_title(4)
            ''
            '\input{'+PalName(i,1)+'_long}']
    end
    if fileinfo(PalName(i,1)+'_cosf/'+PalName(i,1)+'_see_also.tex')<>[] then //see also
       tt6=[tex_title(6)
            '\input{'+PalName(i,1)+'_see_also}']
    end

    if flag=='guide' then
       tt1=['\chapter{'+PalName(i,2)+'}\label{'+PalName(i,1)+'}'] //Header of tex file

    elseif flag=='html' then

       if lang=='fr' then //Header of tex file
         tt1=['\documentclass[11pt,frenchb]{article}']
       else
         tt1=['\documentclass[11pt]{article}']
       end
       tt1=[tt1;
            '\usepackage{makeidx,graphics,fullpage}'
            '\usepackage{verbatim,times,amsmath,amssymb,epsfig,color}'
            '\usepackage{html}'
            '\begin{document}']
       if lang=='fr' then
         tt1=[tt1;'\begin{center}Palette Scicos\\']
       else
         tt1=[tt1;'\begin{center}'+PalName(i,3)+'\\']
       end
       tt1=[tt1
           '\htmladdnormallink{eng}{../eng/'+PalName(i,1)+'.htm}\hspace{2mm}-'+...
           '\hspace{2mm}\htmladdnormallink{fr}{../fr/'+PalName(i,1)+'.htm}}'
           '\end{center}'];

       tt1=[tt1;'\section{'+PalName(i,2)+'}\label{'+PalName(i,1)+'}']

       tt3=[tex_title(3) //Package
            '\begin{itemize}'
            '\item{\htmladdnormallink{Mod\_Num}{whatis.htm}}'
            '\end{itemize}']

       if fileinfo(PalName(i,1)+'_cosf/'+PalName(i,1)+'_blocks.tex')<>[] then //blocks
         tt5=[tex_title(5)
              '\input{'+PalName(i,1)+'_blocks}']
       end

       if fileinfo(PalName(i,1)+'_cosf/'+PalName(i,1)+'_authors.tex')<>[] then //authors
          tt7=[tex_title(7)
               '\input{'+PalName(i,1)+'_authors}']
       end

       tt8=['\htmlinfo*';'\end{document}']
    end
    //Generate the main tex file of block
    txt=[]
    for j=1:8 txt=[txt;evstr('tt'+string(j))], end
  mputl(txt,PalName(i,1)+'_cosf/'+PalName(i,1)+'.tex')
  end

endfunction
```

2.11 Create main tex file of low-level routines

- **Name:** generate_rout_tex_file
- **Library:** generate_doc - Main library of the documentation generator

2.11.1 Calling Sequence

```
txt = generate_rout_tex_file(namef,flag,lang)
```

2.11.2 Parameters

- **namef :** string. name of the routine
- **flag :** string. set the type of tex files to produce
  - **'html' :** to produce tex file for html format
  - **'guide' :** to produce tex file for paper format
- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page
- **txt :** vector of strings. the output text of the tex file

2.11.3 File content

```
//generate_rout_tex_file
//Entrée : namef un vecteur de chaine de caractère
//         flag 'html' ou 'guide'
//txt = le texte du fichier .tex principal
function txt=generate_rout_tex_file(namef,flag,lang)

 //verify the 'lang' right parameter
 [lsh,rsh]=argn(0)
 if rsh<3 then
    if ~exists('lang') then
     lang='eng'
    elseif lang<>'eng' & lang<>'fr' then
     lang='eng'
    end
 end

  //tt1  : header
  //tt2  : library
  //tt3  : calling Sequence
  //tt4  : parameters
  //tt5  : description
  //tt6  : example
  //tt7  : text of function
  //tt8  : used functions
  //tt9  : see also
  //tt10 : Authors
  //tt11 : Bibliography
  //tt12 : end of tex file

  //define title of paragraph
  tt_title=[
            ''                  //tt1  : header du fichier tex
            'Library'           //tt2  : library
            'Calling Sequence'  //tt3  : calling sequence
            'Parameters'        //tt4  : parameters
            'Description'       //tt5  : Description (_long)
            'Example'           //tt6  : Example (_ex)
            'File content'      //tt7  : text of funtion
            'Used function(s)'     //tt8  : Used functions (_used_func)
            'See Also'          //tt9  : See Also (_see_also)
            'Authors'           //tt10 : Authors (_authors)
            ''                  //tt11 : Bibliography (_bib)
            ''                  //tt12 : End of tex file
           ]
 //change language of title
 if lang=='fr' then
  tt_title=change_lang_title(lang,tt_title);
 end

 //define level of paragraph
 if flag=='html' then
  tex_title='\subsection{'+tt_title+'}'
 else
```

```
 tex_title='\subsection{'+tt_title+'}'
end

//Generate auxiliary tex files
namef=generate_aux_tex_file(namef,'rout',flag,lang);

for i=1:size(namef,1)
 for j=1:12 execstr('tt'+string(j)+'=[]'),end

  if fileinfo(namef(i,1)+'_rout/'+namef(i,1)+'_call_seq.tex')<>[] then
     tt3=[tex_title(3) //Calling sequence
          '\input{'+namef(i,1)+'_call_seq}']
   end

   if fileinfo(namef(i,1)+'_rout/'+namef(i,1)+'_param.tex')<>[] then
     tt4=[tex_title(4) //parameters
          '\input{'+namef(i,1)+'_param}']
   end

   if fileinfo(namef(i,1)+'_rout/'+namef(i,1)+'_long.tex')<>[] then
     tt5=[tex_title(5) //description
          ''
          '\input{'+namef(i,1)+'_long}']
   end

   if fileinfo(namef(i,1)+'_rout/'+namef(i,1)+'_ex.tex')<>[] then
     tt6=[tex_title(6) //Example
          '\input{'+namef(i,1)+'_ex}']
   end

   tt_rep=return_ext_file(tt_ml,namef(i,1)+'.c');

   if tt_rep<>[] then
    if size(tt_rep,1)==1 then
     tt7=[tex_title(7) //file content
          '{\tiny'
          '\verbatiminput{'+tt_rep(1)+tt_rep(2)+'}'
          '}']
    elseif size(tt_rep,1)<>1 then
       for l=1:size(tt_rep,1)
          if namef(i,1)+'.c'==tt_rep(l,2) then
             tt7=[tex_title(7) //file content
                  '{\tiny'
                  '\verbatiminput{'+tt_rep(l,1)+tt_rep(l,2)+'}'
                  '}']
             break;
          end
       end
    end
   end

   if fileinfo(namef(i,1)+'_rout/'+namef(i,1)+'_used_func.tex')<>[] then
     tt8=[tex_title(8) //Used function
          '\input{'+namef(i,1)+'_used_func}']
   end

   if fileinfo(namef(i,1)+'_rout/'+namef(i,1)+'_see_also.tex')<>[] then
     tt9=[tex_title(9) //See also
          '\input{'+namef(i,1)+'_see_also}']
   end

   if fileinfo(namef(i,1)+'_rout/'+namef(i,1)+'_bib.tex')<>[] then
     tt11=['\input{'+namef(i,1)+'_bib.tex}'] //bibliography
   end

 if flag=='guide' then
    //TO BE DONE
   tt1=['\section{'+latexsubst(namef(i,1))+'\label{'+namef(i,1)+'}} ';
        '\begin{itemize}';]
   if lang=='fr' then
        tt1=[tt1;'\item \textbf{Description courte:} '+latexsubst(namef(i,2))];
   else
        tt1=[tt1;'\item \textbf{Short description:} '+latexsubst(namef(i,2))];
   end

   LibName=mod_num_rout_lib;
   tta=return_xml_sdesc(xml_path+lang+'/'+LibName+'.xml')
   if tta<>[] then
      tt1=[tt1;'\item \textbf{'+tt_title(2)+':} '+latexsubst(LibName)+...
           ' - '+latexsubst(tta)];
   end
   tt1=[tt1;'\end{itemize}'];

 elseif flag=='html' then
   if lang=='fr' then //Header of tex file
     tt1=['\documentclass[11pt,frenchb]{article}']
   else
     tt1=['\documentclass[11pt]{article}']
   end
   tt1=[tt1;
        '\usepackage{makeidx,graphics,fullpage}'
        '\usepackage{verbatim,times,amsmath,amssymb,epsfig,color}'
        '\usepackage{html}'
        '\begin{document}'];
   if lang<>'fr' then
     tt1=[tt1;'\begin{center}'+latexsubst(namef(i,3))+'\\'];
   else
     tt1=[tt1;'\begin{center}Routine de calcul bas-niveau\\'];
   end
   tt1=[tt1
        '\htmladdnormallink{eng}{../eng/'+namef(i,1)+'.htm}\hspace{2mm}-'+...
```

```
        '\hspace{2mm}\htmladdnormallink{fr}{../fr/'+namef(i,1)+'.htm}}'
        '\end{center}'];

    tt1=[tt1;'\section{\textbf{'+latexsubst(namef(i,1))+...
             '} - '+latexsubst(namef(i,2))+'}\label{'+namef(i,1)+'}'];

    LibName=mod_num_rout_lib;
    tta=return_xml_sdesc(xml_path+lang+'/'+LibName+'.xml')
    if tta<>[] then
       tt2=[tex_title(2) //Library
            '\begin{itemize}'
            '\item{\htmladdnormallink{'+latexsubst(LibName)+...
            ' - '+latexsubst(tta)+'}{'+LibName+'.htm}}'
            '\end{itemize}']
    end

    if fileinfo(namef(i,1)+'_rout/'+namef(i,1)+'_authors.tex')<>[] then
      tt10=[tex_title(10) //authors
            '\input{'+namef(i,1)+'_authors}']
    end

    tt12=['\htmlinfo*';'\end{document}']

  end
   txt=[];
   for j=1:12 txt=[txt;evstr('tt'+string(j))], end;
   mputl(txt,namef(i,1)+'_rout/'+namef(i,1)+'.tex');
end

endfunction
```

2.12 Create main tex file of scilab scripts

- **Name:** generate_sce_tex_file
- **Library:** generate_doc - Main library of the documentation generator

2.12.1 Calling Sequence

```
txt = generate_sce_tex_file(namef,flag,lang)
```

2.12.2 Parameters

- **namef :** add here the parameter description
- **flag :** add here the parameter description
- **lang :** add here the parameter description
- **txt :** add here the parameter description

2.12.3 File content

```
//generate_sce_tex_file
//Entrée : namef un vecteur de chaine de caractère
//         flag 'html' ou 'guide'
//txt = le texte du fichier .tex principal
function txt=generate_sce_tex_file(namef,flag,lang)

 //verify the 'lang' right parameter
 [lsh,rsh]=argn(0)
 if rsh<3 then
   if ~exists('lang') then
    lang='eng'
   elseif lang<>'eng' & lang<>'fr' then
    lang='eng'
   end
 end

  //tt1  : header
  //tt2  : library
  //tt3  : calling Sequence
  //tt4  : parameters
  //tt5  : description
  //tt6  : example
  //tt7  : text of function
  //tt8  : used functions
  //tt9  : see also
  //tt10 : Authors
  //tt11 : Bibliography
  //tt12 : end of tex file

  //define title of paragraph
```

```
 tt_title=[
          ''                  //tt1  : header du fichier tex
          'Description'       //tt2  : Description (_long)
          'File content'      //tt3  : text of funtion
          'Used function'     //tt4  : Used functions (_used_func)
          'See Also'          //tt5  : See Also (_see_also)
          'Authors'           //tt6  : Authors (_authors)
          ''                  //tt7  : Bibliography (_bib)
          ''                  //tt8  : End of tex file
         ]
//change language of title
if lang=='fr' then
 tt_title=change_lang_title(lang,tt_title);
end

//define level of paragraph
if flag=='html' then
 tex_title='\subsection{'+tt_title+'}'
else
 tex_title='\subsection{'+tt_title+'}'
end

//Generate auxiliary tex files
namef=generate_aux_tex_file(namef,'sce',flag,lang);

for i=1:size(namef,1)
 for j=1:8 execstr('tt'+string(j)+'=[]'),end

   if fileinfo(namef(i,1)+'_sce/'+namef(i,1)+'_long.tex')<>[] then
     tt2=[tex_title(2) //description
          ''
          '\input{'+namef(i,1)+'_long}']
   end

   tt_rep=return_ext_file(tt_ml,namef(i,1)+'.sce');

   if tt_rep<>[] then
    if size(tt_rep,1)==1 then
     tt3=[tex_title(3) //file content
          '{\tiny'
          '\verbatiminput{'+tt_rep(1)+tt_rep(2)+'}'
          '}']
    end
   end

   if fileinfo(namef(i,1)+'_sce/'+namef(i,1)+'_used_func.tex')<>[] then
     tt4=[tex_title(4) //Used function
          '\input{'+namef(i,1)+'_used_func}']
   end

   if fileinfo(namef(i,1)+'_sce/'+namef(i,1)+'_see_also.tex')<>[] then
     tt5=[tex_title(5) //See also
          '\input{'+namef(i,1)+'_see_also}']
   end

   if fileinfo(namef(i,1)+'_sce/'+namef(i,1)+'_bib.tex')<>[] then
     tt7=['\input{'+namef(i,1)+'_bib.tex}'] //bibliography
   end

 if flag=='guide' then
    //TO BE DONE
   tt1=['\section{'+latexsubst(namef(i,1))+'\label{'+namef(i,1)+'}} ';
        '\begin{itemize}';]
   if lang=='fr' then
     tt1=[tt1;'\item \textbf{Description courte:} '+latexsubst(namef(i,2))];
   else
     tt1=[tt1;'\item \textbf{Short description:} '+latexsubst(namef(i,2))];
   end

   tt1=[tt1;'\end{itemize}'];

 elseif flag=='html' then
   if lang=='fr' then //Header of tex file
     tt1=['\documentclass[11pt,frenchb]{article}']
   else
     tt1=['\documentclass[11pt]{article}']
   end
   tt1=[tt1;
        '\usepackage{makeidx,graphics,fullpage}'
        '\usepackage{verbatim,times,amsmath,amssymb,epsfig,color}'
        '\usepackage{html}'
        '\begin{document}']
   if lang=='fr' then
     tt1=[tt1;'\begin{center}Script Scilab\\']
   else
     tt1=[tt1;'\begin{center}'+latexsubst(namef(i,3))+'\\']
   end
   tt1=[tt1
        '\htmladdnormallink{eng}{../eng/'+namef(i,1)+'.htm}\hspace{2mm}-'+...
        '\hspace{2mm}\htmladdnormallink{fr}{../fr/'+namef(i,1)+'.htm}}'
        '\end{center}'];

   tt1=[tt1;'\section{\textbf{'+latexsubst(namef(i,1))+...
             '} - '+latexsubst(namef(i,2))+'}\label{'+namef(i,1)+'}']

   if fileinfo(namef(i,1)+'_sce/'+namef(i,1)+'_authors.tex')<>[] then
     tt6=[tex_title(6) //authors
          '\input{'+namef(i,1)+'_authors}']
   end

   tt8=['\htmlinfo*';'\end{document}']
```

```
  end
   txt=[];
   for j=1:8 txt=[txt;evstr('tt'+string(j))], end;
   mputl(txt,namef(i,1)+'_sce/'+namef(i,1)+'.tex');
end

endfunction
```

2.13 Create main tex file of scilab macros

- **Name:** generate_scifun_tex_file
- **Library:** generate_doc - Main library of the documentation generator

2.13.1 Calling Sequence

```
txt = generate_scifun_tex_file(namef,flag,lang)
```

2.13.2 Parameters

- **namef :** string. the name of the scilab macros.
- **flag :** string. set the type of tex files to produce
  - **'html' :** to produce tex file for html format
  - **'guide' :** to produce tex file for paper format
- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page
- **txt :** vector of strings. the output text of the tex file

2.13.3 File content

```
//generate_scifunc_tex_file
//Entrée : namef un vecteur de chaine de caractère
//         flag 'html' ou 'guide'
//txt = le texte du fichier .tex principal
function txt=generate_scifun_tex_file(namef,flag,lang)

 //verify the 'lang' right parameter
 [lsh,rsh]=argn(0)
 if rsh<3 then
    if ~exists('lang') then
     lang='eng'
    elseif lang<>'eng' & lang<>'fr' then
     lang='eng'
    end
 end

 //Generate auxiliary tex files
 namef=generate_aux_tex_file(namef,'sci');

  //tt1  : header
  //tt2  : library
  //tt3  : calling Sequence
  //tt4  : parameters
  //tt5  : description
  //tt6  : remarks (_rmk)
  //tt7  : example
  //tt8  : algorithm (_algo)
  //tt9  : text of function
  //tt10 : used functions
  //tt11 : see also
  //tt12 : Authors
  //tt13 : Bibliography
  //tt14 : end of tex file

 //define title of paragraph
 tt_title=[
           ''                 //tt1  : header du fichier tex
           'Library'          //tt2  : library
           'Calling Sequence' //tt3  : calling sequence
           'Parameters'       //tt4  : parameters
           'Description'      //tt5  : Description (_long)
```

```
          'Remarks'           //tt6  : Remarks
          'Example'           //tt7  : Example (_ex)
          'Algorithm'         //tt8  : Algorithm (_algo)
          'File content'      //tt9  : text of funtion
          'Used function(s)'     //tt10 : Used functions (_used_func)
          'See Also'          //tt11 : See Also (_see_also)
          'Authors'           //tt12 : Authors (_authors)
          'Bibliography'      //tt13 : Bibliography (_bib)
          ''                  //tt14 : End of tex file
         ]

//change language of title
if lang=='fr' then
 tt_title=change_lang_title(lang,tt_title);
end

//define level of paragraph
if flag=='html' then
 tex_title='\subsection{'+tt_title+'}'
else
 tex_title='\subsection{'+tt_title+'}'
end

for i=1:size(namef,1) //for each file
 for j=1:14 execstr('tt'+string(j)+'=[]'),end //for each paragraph

   if fileinfo(namef(i,1)+'_sci/'+namef(i,1)+'_call_seq.tex')<>[] then
     tt3=[tex_title(3) //Calling sequence
          '\input{'+namef(i,1)+'_call_seq}']
   end

   if fileinfo(namef(i,1)+'_sci/'+namef(i,1)+'_param.tex')<>[] then
     tt4=[tex_title(4) //parameters
          '\input{'+namef(i,1)+'_param}']
   end

   if fileinfo(namef(i,1)+'_sci/'+namef(i,1)+'_long.tex')<>[] then
     tt5=[tex_title(5) //description
          ''
          '\input{'+namef(i,1)+'_long}']
   end

   if fileinfo(namef(i,1)+'_sci/'+namef(i,1)+'_rmk.tex')<>[] then
     tt6=[tex_title(6) //Remarks
          '\input{'+namef(i,1)+'_rmk}']
   end

   if fileinfo(namef(i,1)+'_sci/'+namef(i,1)+'_ex.tex')<>[] then
     tt7=[tex_title(7) //Example
          '\input{'+namef(i,1)+'_ex}']
   end

  if fileinfo(namef(i,1)+'/'+namef(i,1)+'_algo.tex')<>[] then //Algorithm
    tt8=[tex_title(8)
         '\input{'+namef(i,1)+'_algo}']
  end

   tt_rep=return_ext_file(tt_ml,namef(i,1)+'.sci');
   if tt_rep<>[] then
    if size(tt_rep,1)==1 then
     tt9=[tex_title(9) //file content
          '{\tiny'
          '\verbatiminput{'+tt_rep(1)+tt_rep(2)+'}'
          '}']
    end
   end

   if fileinfo(namef(i,1)+'_sci/'+namef(i,1)+'_used_func.tex')<>[] then
     tt10=[tex_title(10) //Used function
           '\input{'+namef(i,1)+'_used_func}']
   end

   if fileinfo(namef(i,1)+'_sci/'+namef(i,1)+'_see_also.tex')<>[] then
     tt11=[tex_title(11) //See also
          '\input{'+namef(i,1)+'_see_also}']
   end

   if fileinfo(namef(i,1)+'_sci/'+namef(i,1)+'_bib.tex')<>[] then
     tt13=['\input{'+namef(i,1)+'_bib.tex}']; //bibliography
   end

 if flag==3'guide' then
   //TO BE DONE
   new_title=convstr(part(namef(i,2),1),'u')+part(namef(i,2),2:length(namef(i,2)));
   tt1=['\section{'+latexsubst(new_title)+'\label{'+namef(i,1)+'}} ';
        '\begin{itemize}';]
       if lang=='fr' then
         tt1=[tt1;'\item \textbf{Nom:} '+latexsubst(namef(i,1))];
       else
         tt1=[tt1;'\item \textbf{Name:} '+latexsubst(namef(i,1))];
       end

   LibName=return_rpordoef(tt_ml,namef(i,1)+'.sci');
   //pause
   if LibName==[] then
    for j=1:size(modnum_sci_func,1)
     if namef(i,1)==modnum_sci_func(j) then
       LibName=mod_num_sci_lib;
       break
     end
    end
```

```
    end
    if LibName<>[] then
      if size(LibName,1)==1 then
         tta=return_xml_sdesc(xml_path+lang+'/'+LibName+'.xml')
         if tta<>[] then
           if lang=='fr' then
               tt1=[tt1;'\item \textbf{Librairie:} '+latexsubst(LibName)+...
                    ' - '+latexsubst(tta)]
           else
               tt1=[tt1;'\item \textbf{Library:} '+latexsubst(LibName)+...
                    ' - '+latexsubst(tta)]
           end
         end
      end
    end
    tt1=[tt1;'\end{itemize}'];

  elseif flag=='html' then

    if lang=='fr' then //Header of tex file
      tt1=['\documentclass[11pt,frenchb]{article}']
    else
      tt1=['\documentclass[11pt]{article}']
    end
    tt1=[tt1;
         '\usepackage{makeidx,graphics,fullpage}'
         '\usepackage{verbatim,times,amsmath,amssymb,epsfig,color}'
         '\usepackage{html}'
         '\begin{document}'];
    if lang=='fr' then
       tt1=[tt1;'\begin{center}Fonction Scilab\\'];
    else
       tt1=[tt1;'\begin{center}'+latexsubst(namef(i,3))+'\\'];
    end
    tt1=[tt1
         '\htmladdnormallink{eng}{../eng/'+namef(i,1)+'.htm}\hspace{2mm}-'+...
         '\hspace{2mm}\htmladdnormallink{fr}{../fr/'+namef(i,1)+'.htm}}'
         '\end{center}'];

    tt1=[tt1;'\section{\textbf{'+latexsubst(namef(i,1))+...
             '} - '+latexsubst(namef(i,2))+'}\label{'+namef(i,1)+'}'];

    LibName=return_rpordoef(tt_ml,namef(i,1)+'.sci');
    //pause
    if LibName==[] then
     for j=1:size(modnum_sci_func,1)
      if namef(i,1)==modnum_sci_func(j) then
        LibName=mod_num_sci_lib;
        break
      end
     end
    end
    if LibName<>[] then
      if size(LibName,1)==1 then
         tta=return_xml_sdesc(xml_path+lang+'/'+LibName+'.xml')
         if tta<>[] then
           tt2=[tex_title(2) //Library
                '\begin{itemize}']
           tt2=[tt2;'\item{\htmladdnormallink{'+latexsubst(LibName)+...
                    ' - '+latexsubst(tta)+'}{'+LibName+'.htm}}']
           tt2=[tt2;'\end{itemize}']
         end
      end
    end

    if fileinfo(namef(i,1)+'_sci/'+namef(i,1)+'_authors.tex')<>[] then
      tt12=[tex_title(12) //authors
            '\input{'+namef(i,1)+'_authors}']
    end
    tt14=['\htmlinfo*';'\end{document}']
  end

  //Write the main tex file of scilab macro
  txt=[];
  for j=1:14 txt=[txt;evstr('tt'+string(j))], end;
  mputl(txt,namef(i,1)+'_sci/'+namef(i,1)+'.tex');
end
//   else
//    printf("Bad flag\n");
//    txt=[];
//   end

endfunction
```

2.14 Create main tex file of scilab macro library

- **Name:** generate_scilib_tex_file
- **Library:** generate_doc - Main library of the documentation generator

2.14.1 Calling Sequence

```
txt = generate_scilib_tex_file(LibName,flag,lang)
```

2.14.2 Parameters

- **LibName :** string. the name of the scilab macro libraries
- **flag :** string. set the type of tex files to produce
  - **'html' :** to produce tex file for html format
  - **'guide' :** to produce tex file for paper format
- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page
- **txt :** vector of strings. the output text of the tex file

2.14.3 File content

```
function txt=generate_scilib_tex_file(LibName,flag,lang)

  //verify the 'lang' right parameter
  [lsh,rsh]=argn(0)
  if rsh<3 then
    if ~exists('lang') then
     lang='eng'
    elseif lang<>'eng' & lang<>'fr' then
     lang='eng'
    end
  end

 //Generate auxiliary tex files
 LibName=generate_aux_tex_file(LibName,'scilib',flag,lang);

 //define title of paragraph
 tt_title=[
           ''                //tt1  : header du fichier tex
           'Package'         //tt2  : Package
           'Description'     //tt3  : Description (_long)
           'Scilab function' //tt4  : Scilab functions
           'See Also'        //tt5  : See Also (_see_also)
           'Authors'         //tt6  : Authors (_authors)
           ''                //tt7  : End of tex file
           ]

 //change language of title
 if lang=='fr' then
  tt_title=change_lang_title(lang,tt_title);
 end

 //define level of paragraph
 if flag=='html' then
  tex_title='\subsection{'+tt_title+'}'
 else
  tex_title='\subsection{'+tt_title+'}'
 end
  //tt1 : header
  //tt2 : package
  //tt3 : description
  //tt4 : scilab functions
  //tt5 : see also
  //tt6 : Authors
  //tt7 : end of tex file

 for i=1:size(LibName,1)
   for j=1:7 execstr('tt'+string(j)+'=[]'),end //for each paragraph

      if fileinfo(LibName(i,1)+'_scilib/'+LibName(i,1)+'_long.tex')<>[] then
        tt3=[tex_title(3) //Description
             ''
             '\input{'+LibName(i,1)+'_long}']
      end

      if fileinfo(LibName(i,1)+'_scilib/'+LibName(i,1)+'_see_also.tex')<>[] then
         tt5=[tex_title(5) //see also
             '\input{'+LibName(i,1)+'_see_also}']
      end


      if flag=='guide' then
         tt1=['\chapter{'+latexsubst(LibName(i,2))+'}\label{'+LibName(i,1)+'}']

      elseif flag=='html' then

         if lang=='fr' then //Header of tex file
           tt1=['\documentclass[11pt,frenchb]{article}']
         else
           tt1=['\documentclass[11pt]{article}']
         end
         tt1=[tt1;
```

```
            '\usepackage{makeidx,graphics,fullpage}'
            '\usepackage{verbatim,times,amsmath,amssymb,epsfig,color}'
            '\usepackage{html}'
            '\begin{document}']
        if lang=='fr' then
          tt1=[tt1;'\begin{center}Librairie Scilab\\']
        else
          tt1=[tt1;'\begin{center}Scilab Library\\']
        end
        tt1=[tt1
            '\htmladdnormallink{eng}{../eng/'+LibName(i,1)+'.htm}\hspace{2mm}-'+...
            '\hspace{2mm}\htmladdnormallink{fr}{../fr/'+LibName(i,1)+'.htm}}'
            '\end{center}'];

        tt1=[tt1;'\section{'+latexsubst(LibName(i,2))+'}\label{'+LibName(i,1)+'}']

        tt2=[tex_title(2)
             '\begin{itemize}'
             '\item{\htmladdnormallink{Mod\_Num}{whatis.htm}}'
             '\end{itemize}']


        //Cherche la liste des fichiers sci du rep LibName dans tt_ml
        if LibName(i,1)==mod_num_sci_lib then
          name=modnum_sci_func;
        elseif LibName(i,1)==mod_num_rout_lib then
          lisf_rout=return_ext_file_in_dir(tt_ml,low_rout_path,".c")
          name=basename(lisf_rout);
        else
          sci_files=return_ext_file_in_dir(tt_ml,mac_path+'/'+LibName(i,1),'sci');
          name=basename(sci_files);
        end
        //pause
        if name<>[] then
           tt4=[tex_title(4);'\begin{itemize}']
           for j=1:size(name,1)
             txt2=return_xml_sdesc(xml_path+lang+'/'+name(j)+'.xml');
             txt2=latexsubst(txt2);
             tt4=[tt4;'\item{\htmladdnormallink{'+latexsubst(name(j))+' - '+txt2+'}{'+name(j)+'.htm}}']
           end
           tt4=[tt4;'\end{itemize}'];
        end

    if fileinfo(LibName(i,1)+'_scilib/'+LibName(i,1)+'_authors.tex')<>[] then
      tt6=[tex_title(6)
           '\input{'+LibName(i,1)+'_authors}']
    end

    tt7=['\htmlinfo*';'\end{document}']
  end
  txt=[];
  for j=1:7 txt=[txt;evstr('tt'+string(j))], end;
  mputl(txt,LibName(i,1)+'_scilib/'+LibName(i,1)+'.tex')
 end

endfunction
```

2.15 Create main tex file of simulation script

- **Name:** generate_sim_tex_file
- **Library:** generate_doc - Main library of the documentation generator

2.15.1 Calling Sequence

```
txt = generate_sim_tex_file(lisf,flag,lang)
```

2.15.2 Parameters

- **lisf :** the name of the simulation script
- **flag :** string. set the type of tex files to produce
  - **'html' :** to produce tex file for html format
  - **'guide' :** to produce tex file for paper format
- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page
- **txt :** vector of strings. the output text of the tex file

2.15.3 File content

```
//generate_sim_tex_file
//Fonction qui génère le fichier tex principal
//d'une page de documentation d'un script de
//simulation scilab

function txt=generate_sim_tex_file(lisf,flag,lang)

 //verify the 'lang' right parameter
 [lsh,rsh]=argn(0)
 if rsh<3 then
   if ~exists('lang') then
    lang='eng'
   elseif lang<>'eng' & lang<>'fr' then
    lang='eng'
   end
 end

 //Generate auxiliary tex files
 lisf=generate_aux_tex_file(lisf,'sim',flag,lang);

 //define title of paragraph
 tt_title=[
           ''                       //tt1  : header du fichier tex
           'Description'            //tt2  : Description (_long)
           'Algorithm'              //tt3  : Algorithm (_algo)
           'Simulation script(s)'   //tt4  : simulation script(s) (_sim_script)
           'Scope Results'          //tt5  : Scope results (_scop)
           'Scicos diagram(s)'      //tt6  : scicos diagram(s) (_diagr)
           'Context file(s)'        //tt7  : file of context(s) (_context)
           'Mod\_num blocks'        //tt8  : Mod num blocks (_block)
           'Used function'          //tt9  : used functions (_used_func)
           'See Also'               //tt10 : See Also (_see_also)
           'Authors'                //tt11 : Authors (_authors)
           'Bibliography'           //tt12 : Bibliography (_bib)
           ''                       //tt13 : End of tex file
          ]

 //change language of title
 if lang=='fr' then
  tt_title=change_lang_title(lang,tt_title);
 end

 //define level of paragraph
 if flag=='html' then
  tex_title='\subsection{'+tt_title+'}'
 else
  tex_title='\subsection{'+tt_title+'}'
 end

 for i=1:size(lisf,1)
    for j=1:13 execstr('tt'+string(j)+'=[]'),end

    if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_long.tex')<>[] then //Description
      tt2=[tex_title(2)
           ''
           '\input{'+lisf(i,1)+'_long}']
    end

    if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_algo.tex')<>[] then //algorithm
      tt3=[tex_title(3)
           '\input{'+lisf(i,1)+'_algo}']
    end

    if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_sim_script.tex')<>[] then //Simulation scripts
      tt4=[tex_title(4)
           '\input{'+lisf(i,1)+'_sim_script}']
    end

    if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_scop.tex')<>[] then //scop
      tt5=[tex_title(5)
           '\input{'+lisf(i,1)+'_scop}']
    end

    if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_diagr.tex')<>[] then //Scicos diagram(s)
      tt6=[tex_title(6)
           '\input{'+lisf(i,1)+'_diagr}']
    end

    if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_context.tex')<>[] then //Context file(s)
      tt7=[tex_title(7)
           '\input{'+lisf(i,1)+'_context}']
    end

    if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_block.tex')<>[] then //mod_num block
      tt8=[tex_title(8)
           '\input{'+lisf(i,1)+'_block}']
    end

    if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_used_func.tex')<>[] then //Used function
      tt9=[tex_title(9)
           '\input{'+lisf(i,1)+'_used_func}']
    end

    if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_see_also.tex')<>[] then //see also
      tt10=[tex_title(10)
            '\input{'+lisf(i,1)+'_see_also}']
    end

    if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_bib.tex')<>[] then //bibliography
```

```
      tt12=['\input{'+lisf(i,1)+'_bib.tex}']
    end

    if flag=='guide' then
          tt1=['\section{'+latexsubst(lisf(i,2))+'}\label{'+lisf(i,1)+'}'];

    elseif flag=='html' then

    if lang=='fr' then //Header of tex file
      tt1=['\documentclass[11pt,frenchb]{article}']
    else
      tt1=['\documentclass[11pt]{article}']
    end
    tt1=[tt1;
         '\usepackage{makeidx,graphics,fullpage}'
         '\usepackage{verbatim,times,amsmath,amssymb,epsfig,color}'
         '\usepackage{html}'
         '\begin{document}']
    if lang=='fr' then
      tt1=[tt1;'\begin{center}Script de simulation Scilab\\']
    else
      tt1=[tt1;'\begin{center}'+lisf(i,3)+'\\']
    end
    tt1=[tt1
         '\htmladdnormallink{eng}{../eng/'+lisf(i,1)+'.htm}\hspace{2mm}-'+...
         '\hspace{2mm}\htmladdnormallink{fr}{../fr/'+lisf(i,1)+'.htm}}'
         '\end{center}'];
    tt1=[tt1
         '\section{'+latexsubst(lisf(i,2))+'}\label{'+lisf(i,1)+'}'
         '\tableofcontents'
        ]

    if fileinfo(lisf(i,1)+'/'+lisf(i,1)+'_authors.tex')<>[] then //authors
      tt11=[tex_title(11)
            '\input{'+lisf(i,1)+'_authors}']
    end

    tt13=['\htmlinfo*';'\end{document}']
    end

    //Generate the main tex file of block
    txt=[]
    for j=1:13 txt=[txt;evstr('tt'+string(j))], end
    mputl(txt,lisf(i,1)+'/'+lisf(i,1)+'.tex')
 end

endfunction
```

2.16 Create main whatis.htm file of the html documentation of the toolbox

- **Name:** generate_swhatis
- **Library:** generate_doc - Main library of the documentation generator

2.16.1 Calling Sequence

`generate_swhatis(Path,lang)`

2.16.2 Parameters

- **Path :** string. the target path where whatis.htm file is created.
- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page

2.16.3 File content

```
//Fonction qui genère le fichier whatis.htm
//des man pages de modnum
//Entrée : Path chemin ou enregister le fichier
function generate_swhatis(Path,lang)

   [lsh,rsh]=argn(0)
   if rsh<2 then
    if ~exists('lang') then
     lang='eng'
    elseif lang<>'eng' & lang<>'fr' then
     lang='eng'
    end
   end
```

```
if lang==’fr’ then
 l_i=2;
else
 l_i=1;
end

tt_tile=[’Scicos Diagrams’,’Diagrammes Scicos’
’Chaotic dynamical systems’,’Systèmes dynamiques chaotiques’
’Continous time systems’,’Systèmes à temps continu’
’Discrete time systems’,’Systèmes à temps discret’
’Oscillators and Phase Locked Loop systems’,’Oscillateurs et boucles à verrouillage de phase’
’Open loop models of oscillators’,’Modèles boucle ouverte d’’oscillateurs’
’Integer N Frequency synthesizers’,’Synthétiseurs de fréquence à rapport de division N entier’
’Fractional N/N+1 Frequency synthesizers’,’Synthétiseurs de fréquence à rapport de division fractionnaire’
’Communication systems’,’Systèmes de communication’
’PSK/QAM Transmission’,’Transmission PSK/QAM’
’Delta-Sigma Transmission’,’Transmission Sigma-Delta’
’FSK chaotic transmission’,’Transmission chaotique FSK’
’Electrical circuits’,’Circuits électriques’
’Simulation Scilab scripts’,’Scripts Scilab de simulations’
’Simulations of chaotic systems’,’Simulations de systèmes chaotiques’
’Simulations of oscillators and Phase Locked Loops’,’Simulations d’’oscillateurs et de boucles à verrouillage de phase’
’Simulations of communication systems’,’Simulations de systèmes de communication’
’Scicos blocks’,’Blocs Scicos’
’Scilab function libraries’,’Fonctions Scilab’
’Communication’,’Communication’
’Modnum internals’,’Guide interne de Modnum’
’Utility Scilab scripts’,’Scripts Scilab utilitaires’
’Utility macros libraries’,’Macros Scilab utilitaires’
’Documentation generator macro libraries’,’Librairies Scilab du générateur de documentation’
’Low level routines’,’Routines de calcul bas-niveau’]

printf("whatis.htm file generation... ")
lines(0);
//---------------------------------------------------------------------------------------
//Header
//---------------------------------------------------------------------------------------
if lang==’fr’ then
  whatis_title=’Documentation Modnum’
else
  whatis_title=’Modnum Documentation’
end
 head=["<html>"
      "<head>"
      "  <meta http-equiv=""Content-Type"" content=""text/html; charset=ISO-8859-1"">"
      "    <title>"+whatis_title+"</title>"
      "</head>"
      "<body bgcolor=""#FFFFFF"">"
      "<DIV ALIGN=""CENTER"">’
      whatis_title
      "<BR>"
      "<A HREF=""../eng/whatis.htm"">eng</A> - "
      "<A HREF=""../fr/whatis.htm"">fr</A>"
      "<font color=""black"">"
      "<BR>"
      "</DIV>"
      "<BR>"];
//---------------------------------------------------------------------------------------
//Scicos diagrams
//---------------------------------------------------------------------------------------
k=1;
line(k)="<BR>";k=k+1;
line(k)="<b><LI>"+tt_tile(1,l_i)+"</b>";k=k+1 //Scicos Diagrams
line(k)="<UL>";k=k+1
line(k)="<BR>";k=k+1;

line(k)="<LI>"+tt_tile(5,l_i);k=k+1 //Oscillators and Phase Locked Loop systems
line(k)="<UL>";k=k+1

if diagr_os<>[] then
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(6,l_i);k=k+1 //Open loop models of oscillators
  for j=1:size(diagr_os,1)
    desc=return_xml_sdesc(xml_path+lang+’/’+diagr_os(j,2)+’.xml’)
    line(k)="<BR><A HREF="""+diagr_os(j,2)+".htm"">"+...
    convstr(part(diagr_os(j,2),1),’u’)+part(diagr_os(j,2),2:length(diagr_os(j,2)))+"</A> - "+...
    desc;k=k+1;
  end
  line(k)="</P>";k=k+1
end

if diagr_is<>[] then
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(7,l_i);k=k+1 //Integer N Frequency synthesizers
  for j=1:size(diagr_is,1)
    desc=return_xml_sdesc(xml_path+lang+’/’+diagr_is(j,2)+’.xml’)
    line(k)="<BR><A HREF="""+diagr_is(j,2)+".htm"">"+...
    convstr(part(diagr_is(j,2),1),’u’)+part(diagr_is(j,2),2:length(diagr_is(j,2)))+"</A> - "+...
    desc;k=k+1;
  end
  line(k)="</P>";k=k+1
end

if diagr_fs<>[] then
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(8,l_i);k=k+1 //Fractional N/N+1 Frequency synthesizers
  for j=1:size(diagr_fs,1)
    desc=return_xml_sdesc(xml_path+lang+’/’+diagr_fs(j,2)+’.xml’)
    line(k)="<BR><A HREF="""+diagr_fs(j,2)+".htm"">"+...
    convstr(part(diagr_fs(j,2),1),’u’)+part(diagr_fs(j,2),2:length(diagr_fs(j,2)))+"</A> - "+...
    desc;k=k+1;
```

```
  end
  line(k)="</P>";k=k+1
end
line(k)="<BR>";k=k+1
line(k)="</UL>";k=k+1//close pll

line(k)="<LI>"+tt_tile(9,l_i);k=k+1 //Communication systems
line(k)="<UL>";k=k+1
if diagr_PSK<>[] then
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(10,l_i);k=k+1 //PSK/QAM Transmission
  for j=1:size(diagr_PSK,1)
    desc=return_xml_sdesc(xml_path+lang+'/'+diagr_PSK(j,2)+'.xml')
    line(k)="<BR><A HREF="""+diagr_PSK(j,2)+".htm"">"+...
    convstr(part(diagr_PSK(j,2),1),'u')+part(diagr_PSK(j,2),2:length(diagr_PSK(j,2)))+"</A> - "+...
    desc;k=k+1;
  end
  line(k)="</P>";k=k+1//close PSK Transmission
end

if diagr_SD<>[] then
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(11,l_i);k=k+1 //Delta-Sigma Transmission
  for j=1:size(diagr_SD,1)
    desc=return_xml_sdesc(xml_path+lang+'/'+diagr_SD(j,2)+'.xml')
    line(k)="<BR><A HREF="""+diagr_SD(j,2)+".htm"">"+...
    convstr(part(diagr_SD(j,2),1),'u')+part(diagr_SD(j,2),2:length(diagr_SD(j,2)))+"</A> - "+...
    desc;k=k+1;
  end
  line(k)="</P>";k=k+1//close Delta-Sigma Transmission
end

if diagr_FSK_chaos<>[] then
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(12,l_i);k=k+1 //chaotic FSK Transmission
  for j=1:size(diagr_FSK_chaos,1)
    desc=return_xml_sdesc(xml_path+lang+'/'+diagr_FSK_chaos(j,2)+'.xml')
    line(k)="<BR><A HREF="""+diagr_FSK_chaos(j,2)+".htm"">"+...
    convstr(part(diagr_FSK_chaos(j,2),1),'u')+part(diagr_FSK_chaos(j,2),2:length(diagr_FSK_chaos(j,2)))+"</A> - "+...
    desc;k=k+1;
  end
  line(k)="</P>";k=k+1//close Delta-Sigma Transmission
end
line(k)="</UL>";k=k+1//close Comm sys

line(k)="<BR>";k=k+1
line(k)="<LI>"+tt_tile(2,l_i);k=k+1 //Chaotic dynamical systems
line(k)="<UL>";k=k+1
if diagr_cs<>[] then
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(3,l_i);k=k+1 //Continous time systems
  for j=1:size(diagr_cs,1)
    desc=return_xml_sdesc(xml_path+lang+'/'+diagr_cs(j,2)+'.xml')
    line(k)="<BR><A HREF="""+diagr_cs(j,2)+".htm"">"+...
    convstr(part(diagr_cs(j,2),1),'u')+part(diagr_cs(j,2),2:length(diagr_cs(j,2)))+"</A> - "+...
    desc;k=k+1;
  end
  line(k)="</P>";k=k+1
end
if diagr_ds<>[] then
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(4,l_i);k=k+1 //Discrete time systems
  for j=1:size(diagr_ds,1)
    desc=return_xml_sdesc(xml_path+lang+'/'+diagr_ds(j,2)+'.xml')
    line(k)="<BR><A HREF="""+diagr_ds(j,2)+".htm"">"+...
    convstr(part(diagr_ds(j,2),1),'u')+part(diagr_ds(j,2),2:length(diagr_ds(j,2)))+"</A> - "+...
    desc;k=k+1;
  end
  line(k)="</P>";k=k+1
end
line(k)="</UL>";k=k+1

if diagr_elec<>[] then
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(13,l_i);k=k+1 //Electrical circuits
  for j=1:size(diagr_elec,1)
    desc=return_xml_sdesc(xml_path+lang+'/'+diagr_elec(j,2)+'.xml')
    line(k)="<BR><A HREF="""+diagr_elec(j,2)+".htm"">"+...
    convstr(part(diagr_elec(j,2),1),'u')+part(diagr_elec(j,2),2:length(diagr_elec(j,2)))+"</A> - "+...
    desc;k=k+1;
  end
  line(k)="</P>";k=k+1//close Electrical circuit
end

line(k)="</UL>";k=k+1 //close Scicos diagram

//-------------------------------------------------------------------------------------
//Scilab simulation scripts
//-------------------------------------------------------------------------------------
line(k)="<BR>";k=k+1;
line(k)="<b><LI>"+tt_tile(14,l_i)+"</b>";k=k+1 //Simulation Scilab scripts
line(k)="<UL>";k=k+1

if sim_chaos<>[] then
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(15,l_i);k=k+1 //Simulations of chaotic systems
  for j=1:size(sim_chaos,1)
    desc=return_xml_sdesc(xml_path+lang+'/'+sim_chaos(j,2)+'.xml')
    line(k)="<BR><A HREF="""+sim_chaos(j,2)+".htm"">"+...
    convstr(part(sim_chaos(j,2),1),'u')+part(sim_chaos(j,2),2:length(sim_chaos(j,2)))+"</A> - "+...
    desc;k=k+1;
```

```
  end
  line(k)="</P>";k=k+1
end

if sim_synthe<>[] then
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(16,l_i);k=k+1 //Simulations of oscillators and Phase Locked Loops
  for j=1:size(sim_synthe,1)
    desc=return_xml_sdesc(xml_path+lang+'/'+sim_synthe(j,2)+'.xml')
    line(k)="<BR><A HREF="""+sim_synthe(j,2)+".htm"">"+...
    convstr(part(sim_synthe(j,2),1),'u')+part(sim_synthe(j,2),2:length(sim_synthe(j,2)))+"</A> - "+...
    desc;k=k+1;
  end
  line(k)="</P>";k=k+1
end

if sim_PSK<>[] then
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(17,l_i);k=k+1 //Simulations of communication systems
  for j=1:size(sim_PSK,1)
    desc=return_xml_sdesc(xml_path+lang+'/'+sim_PSK(j,2)+'.xml')
    line(k)="<BR><A HREF="""+sim_PSK(j,2)+".htm"">"+...
    convstr(part(sim_PSK(j,2),1),'u')+part(sim_PSK(j,2),2:length(sim_PSK(j,2)))+"</A> - "+...
    desc;k=k+1;
  end
  line(k)="</P>";k=k+1
end

line(k)="</UL>";k=k+1 //close Scilab simulation Scripts

//-------------------------------------------------------------------------------------
//Scicos palettes & blocks
//-------------------------------------------------------------------------------------
line(k)="<BR>";k=k+1;
line(k)="<b><LI>"+tt_tile(18,l_i)+"</b>";k=k+1 //Scicos blocks
line(k)="<UL>";k=k+1
Palrep=return_dir_in_dir(tt_ml,pal_path)
for j=1:size(Palrep,1)
  PalName=basename(part(Palrep(j),1:length(Palrep(j))-1));
  desc=return_xml_sdesc(xml_path+lang+'/'+PalName+'.xml')
  line(k)="<P>"; k=k+1;
  line(k)="<LI><A HREF="""+PalName+".htm"">"+PalName+"</A> - "+desc;k=k+1;
  lisf_blocks=return_ext_file_in_dir(tt_ml,Palrep(j),'.sci');
  for i=1:size(lisf_blocks,1)
   name=basename(lisf_blocks(i,1));
   desc=return_xml_sdesc(xml_path+lang+'/'+name+'.xml')
   line(k)="<BR><A HREF="""+name+".htm"">"+name+"</A> - "+desc;
   k=k+1;
  end
  line(k)=["</P>"];k=k+1;
end
line(k)=["</UL>"];k=k+1;

//-------------------------------------------------------------------------------------
//Scilab functions
//-------------------------------------------------------------------------------------
line(k)="<BR>";k=k+1;
line(k)="<b><LI>"+tt_tile(19,l_i)+"</b>";k=k+1  //Scilab function libraries
line(k)="<UL>";k=k+1
Librep=return_dir_in_dir(tt_ml,mac_path)
for j=1:size(Librep,1)
  LibName=basename(part(Librep(j),1:length(Librep(j))-1));
  if grep(LibName,ex_lib_name)==[] then
     desc=return_xml_sdesc(xml_path+lang+'/'+LibName+'.xml')
     line(k)="<P>"; k=k+1;
     line(k)="<LI><A HREF="""+LibName+".htm"">"+...
                         convstr(part(LibName,1),'u')+part(LibName,2:length(LibName))+...
                         "</A> - "+desc;k=k+1;
     lisf_sci=return_ext_file_in_dir(tt_ml,Librep(j),'.sci');
     for i=1:size(lisf_sci,1)
       name=basename(lisf_sci(i,1));
       desc=return_xml_sdesc(xml_path+lang+'/'+name+'.xml');
       line(k)="<BR><A HREF="""+name+".htm"">"+name+"</A> - "+desc;
       k=k+1;
     end
     line(k)=["</P>"];k=k+1;
  end
end
//Mod_num_sci_lib
LibName=mod_num_sci_lib
desc=return_xml_sdesc(xml_path+lang+'/'+LibName+'.xml')
line(k)="<P>"; k=k+1;
line(k)="<LI><A HREF="""+LibName+".htm"">"+tt_tile(20,l_i)+"</A> - "+desc; //Communication function
k=k+1;
for i=1:size(modnum_sci_func,1)
  desc=return_xml_sdesc(xml_path+lang+'/'+modnum_sci_func(i)+'.xml')
  line(k)="<BR><A HREF="""+modnum_sci_func(i)+".htm"">"+modnum_sci_func(i)+"</A> - "+desc;
  k=k+1;
end
line(k)="</P>";k=k+1;
line(k)="</UL>";k=k+1//close Scilab Functions

//-------------------------------------------------------------------------------------
//modnum internals
//-------------------------------------------------------------------------------------
line(k)="<BR>";k=k+1;
line(k)="<LI><b>"+tt_tile(21,l_i)+"</b>";k=k+1 //MODNUM internals
line(k)="<UL>";k=k+1

//-------------------------------------------------------------------------------------
//Utility script
```

```
  //-------------------------------------------------------------------------------------------
  //builder
  //load_generate_doc_function
  line(k)="<P>";k=k+1
  line(k)="<LI>"+tt_tile(22,l_i);k=k+1 //Utility scilab scripts
  if sce_all<>[] then
    for j=1:size(sce_all,1)
      desc=return_xml_sdesc(xml_path+lang+'/'+sce_all(j,2)+'.xml')
      line(k)="<BR><A HREF="""+sce_all(j,2)+".htm"">"+...
      convstr(part(sce_all(j,2),1),'u')+part(sce_all(j,2),2:length(sce_all(j,2)))+"</A> - "+...
      desc;k=k+1;
    end
    line(k)="</P>";k=k+1
  end
  line(k)="<BR>";k=k+1;

  //-------------------------------------------------------------------------------------------
  //Utility macro libraries
  //-------------------------------------------------------------------------------------------
  line(k)="<BR>";k=k+1;
  line(k)="<LI>"+tt_tile(23,l_i);k=k+1 //Utility macros libraries
  line(k)="<UL>";k=k+1
  Librep=return_dir_in_dir(tt_ml,mac_path)
  for j=1:size(Librep,1)
    LibName=basename(part(Librep(j),1:length(Librep(j))-1));
    if grep(LibName,lib_build)<>[] then
       desc=return_xml_sdesc(xml_path+lang+'/'+LibName+'.xml')
       line(k)="<P>"; k=k+1;
       line(k)="<LI><A HREF="""+LibName+".htm"">"+...
                           convstr(part(LibName,1),'u')+part(LibName,2:length(LibName))+...
                           "</A> - "+desc;k=k+1;
       lisf_sci=return_ext_file_in_dir(tt_ml,Librep(j),'.sci');
       for i=1:size(lisf_sci,1)
         name=basename(lisf_sci(i,1));
         desc=return_xml_sdesc(xml_path+lang+'/'+name+'.xml');
         line(k)="<BR><A HREF="""+name+".htm"">"+name+"</A> - "+desc;
         k=k+1;
       end
       line(k)=["</P>"];k=k+1;
    end
  end
  line(k)=["</UL>"];k=k+1;

  //-------------------------------------------------------------------------------------------
  //Documentation generator macro libraries
  //-------------------------------------------------------------------------------------------
  line(k)="<BR>";k=k+1;
  line(k)="<LI>"+tt_tile(24,l_i);k=k+1 //Documentation generator macro libraries
  line(k)="<UL>";k=k+1
  Librep=return_dir_in_dir(tt_ml,mac_path)
  for j=1:size(Librep,1)
    LibName=basename(part(Librep(j),1:length(Librep(j))-1));
    if grep(LibName,lib_gen_doc)<>[] then
       desc=return_xml_sdesc(xml_path+lang+'/'+LibName+'.xml')
       line(k)="<P>"; k=k+1;
       line(k)="<LI><A HREF="""+LibName+".htm"">"+...
                           convstr(part(LibName,1),'u')+part(LibName,2:length(LibName))+...
                           "</A> - "+desc;k=k+1;
       lisf_sci=return_ext_file_in_dir(tt_ml,Librep(j),'.sci');
       for i=1:size(lisf_sci,1)
         name=basename(lisf_sci(i,1));
         desc=return_xml_sdesc(xml_path+lang+'/'+name+'.xml');
         line(k)="<BR><A HREF="""+name+".htm"">"+name+"</A> - "+desc;
         k=k+1;
       end
       line(k)=["</P>"];k=k+1;
    end
  end
  line(k)=["</UL>"];k=k+1;

  //low level routines
  LibName=mod_num_rout_lib
  desc=return_xml_sdesc(xml_path+lang+'/'+LibName+'.xml')
  line(k)="<P>"; k=k+1;
  line(k)="<LI><A HREF="""+LibName+".htm"">"+tt_tile(25,l_i)+"</A> - "+desc;k=k+1 //Low level routines
  lisf_rout=return_ext_file_in_dir(tt_ml,low_rout_path,".c")
  for i=1:size(lisf_rout,1)
     name=basename(lisf_rout(i,1));
     desc=return_xml_sdesc(xml_path+lang+'/'+name+'.xml')
     line(k)="<BR><A HREF="""+name+".htm"">"+name+"</A> - "+desc;
     k=k+1;
  end
  line(k)="</P>";k=k+1;//close Low-level routines

  line(k)="</UL>";k=k+1//close MODNUM internals
  //-------------------------------------------------------------------------------------------
  //Foot
  //-------------------------------------------------------------------------------------------
  text = [head;line;"</dl></body></html>"];
  mputl(text,Path+"whatis.htm");
  printf("Done\n");
endfunction
```

2.17 Create a whatis.htm file (OBSOLETE)

- **Name:** generate_whatis

- **Library:** generate_doc - Main library of the documentation generator

2.17.1 Calling Sequence

```
generate_whatis(Path)
```

2.17.2 Parameters

- **Path :** string. the target path where whatis.htm file is created.

2.17.3 File content

```
//Fonction qui génère un fichier whatis.htm
//à partir de tous les fichiers htm trouvés
//dans le repertoire pointé par Path
function generate_whatis(Path)
   //lisf=return_listfile(Path,'htm')
   lines(0);
   whatis_title='Help chapter'
   head=["<html>"
         "<head>"
         "  <meta http-equiv=""Content-Type"" content=""text/html; charset=ISO-8859-1"">"
         "    <title>"+whatis_title+"</title>"
         "</head>"
         "<body bgcolor=""#FFFFFF"">"
         "<dl>"];
   for i=1:size(lisf,1)
      name=part(lisf(i,1),1:length(lisf(i,1))-4);
      if fileinfo(xml_path+lang+'/'+name+'.xml')==[] then
       lisf(i,2)=name
      else
       lisf(i,2)=return_xml_sdesc(xml_path+lang+'/'+name+'.xml')
      end
      line(i)="<dd><A HREF=""+lisf(i,1)+"">"+name+"</A> - "+lisf(i,2)+"</dd>";
   end
   text = [head;gsort(line,'g','i');"</dl></body></html>"];
   mputl(text,Path+"whatis.htm");
endfunction
```

2.18 Create xml file of a scilab man page

- **Name:** generate_xml_file
- **Library:** generate_doc - Main library of the documentation generator

2.18.1 Calling Sequence

```
generate_xml_file(lisf,flag,lang)
```

2.18.2 Parameters

- **lisf :** string. set the name of the XML file
- **flag :** string. set the type of man page
  - **'block' :** for interfacing function of scicos block
  - **'pal' :** for a palette (.cosf file)
  - **'diagr' :** for a scicos diagram (.cos file)
  - **'scilib' :** for a library of scilab macros
  - **'sci' :** for a scilab macro.
  - **'rout' :** for computational routine
  - **'sim' :** for scilab simulation script (_sim.sce file)
  - **'sce' :** for scilab script (.sce file)
- **lang :** string. set the lang of tex file

- **'eng' :** to produce english man page
- **'fr' :** to produce french man page

2.18.3 File content

```
//generate_xml_file
//fonction qui crée des fichiers d'aide xml
//Entrée : lisf est une liste de nom de fichier sans extension: CAN_f, Linear ou synthe
//         flag est un drapeau(pour l'instant de taille 1):
//                'block' pour une fonction d'interface scicos
//                'pal' pour un fichier palette scicos (cosf)
//                'diagr' pour un diagramme de simulation scicos
//                'scilib' pour une librairie de fonctions scilab
//                'sci' pour une fonction scilab.
//                'sim' pour un script de simulation scilab
//                'rout' pour une routine bas-niveau
function generate_xml_file(lisf,flag,lang)

[lsh,rsh]=argn(0)
if rsh<3 then
  if ~exists('lang') then
   lang='eng'
  elseif lang<>'eng' & lang<>'fr' then
   lang='eng'
  end
end

for i=1:size(lisf,1)
  if fileinfo(xml_path+lang+'/'+lisf(i,1)+'.xml')==[] then
     printf("%s.xml not found.\n",lisf(i,1))
     printf("Generate an empty xml file... ");
     txt=generate_xml(lisf(i,1),flag,lang)
     mputl(txt,xml_path+lang+'/'+lisf(i,1)+'.xml')
     printf("Done\n");
  else
     printf("%s.xml already exists.\n",lisf(i,1))
  end
end

endfunction
```

2.19 Import xml sections from data files

- **Name:** import_data_to_file
- **Library:** generate_doc - Main library of the documentation generator

2.19.1 Calling Sequence

`import_data_to_file(rep_xml,flag,rep_data)`

2.19.2 Parameters

- **rep_xml :** directory of the XML files
- **flag :** flag to set the type of data to import
  - **'param' :** import parameters section from MODNUM_data_param
  - **'sdesc' :** import short description from MODNUM_data_sdesc
  - **'see_also' :** import see also section from MODNUM_data_see_also
  - **'authors' :** import authors section from MODNUM_data_authors
  - **'ex' :** import exemples section from MODNUM_data_ex
  - **'desc' :** import description section from MODNUM_data_desc
  - **'biblio' :** import bibliography section from MODNUM_data_biblio
  - **'used_func' :** import used function section from MODNUM_data_used_func
  - **'SPECIALDESC' :** import SPECIALDESC files from MODNUM_data_SPECIALDESC
  - **'all' :** import all sections from data files
- **rep_data :** directory of the _data files

2.19.3 File content

```
//Fonction qui importe des données à partir des fichiers
//
//dans la base de fichers xml
//
//import_data_to_file
// flag un vecteur de chaîne de caractères
// 'param'
// 'sdesc'
// 'see_also'
// 'authors'
// 'ex'
// 'desc'
// 'used_func'
// 'biblio'
// 'SPECIALDESC'
// 'all'
// 'call_seq'
// rep_xml :  répertoire de stockage des fichiers
//            xml (ex:rep_xml=xml_path)
// rep_data : répertoire de stockage des fichiers
//            de données
//Modif : passez le flag <LaTeX> en <LaTeX force>
//pour mettre votre fichier latex à jour
function import_data_to_file(rep_xml,flag,rep_data)

 //Verifie cohérence des  paramètres
 [lsh,rsh]=argn();
 if rsh<3 then
  rep_data=man_path
  else
   rep_data=pathconvert(rep_data,%t)
 end;

 if flag=='all' then
  flag=['param';'sdesc';'see_also';
        'authors';'ex';'desc';'used_func';
        'biblio';'SPECIALDESC';'call_seq']
 end

 //def des noms de fichiers de données
 file_param='MODNUM_data_param';
 file_sdesc='MODNUM_data_sdesc';
 file_see_also='MODNUM_data_see_also';
 file_authors='MODNUM_data_authors';
 file_ex='MODNUM_data_ex';
 file_desc='MODNUM_data_desc';
 file_call_seq='MODNUM_data_call_seq';
 file_used_func='MODNUM_data_used_func';
 file_biblio='MODNUM_data_biblio';
 file_spec_desc='MODNUM_SPECIALDESC';

 for z=1:size(flag,1)
   flagn=flag(z);

   ////////
   //'param'
   ////////
   if flagn=='param' then
    if fileinfo(rep_data+file_param)<>[] then
      printf("Import xml parameters...\n");
      tt=mgetl(rep_data+file_param);
      i=1;
      k=0;
      while i<size(tt,1)
        if strindex(tt(i),'<FILE ')<>[] then
          a=i;
          namef=strsubst(tt(i),'<FILE ',"");
          namef=strsubst(namef,'>',"");
          i=i+1;
        elseif strindex(tt(i),'</FILE ')<>[] then
          b=i;
          len_param=evstr(b-a-1);
          if len_param>0 then
            tt_temp=tt(a+1:b-1);
            //mon code
            j=0;
            nb_indent=0;
            //Trouve la profondeur max d'indentation
            for l=1:size(tt_temp,1)
              if strindex(tt_temp(l),'<INDENT>')<>[] then
                j=j+1;
              elseif strindex(tt_temp(l),'</INDENT>')<>[] then
                j=j-1;
              end
              if j>nb_indent then nb_indent=j; end
            end
            //initialise la liste
            txt_list=list();
            for l=1:nb_indent
              txt_list(l)=list();
              txt_list(l)(1)=[];
              txt_list(l)(2)=[];
            end
            //Remplissage de la liste
            j=0;
            e=0;
            for l=1:size(tt_temp,1)
              if strindex(tt_temp(l),'<INDENT>')<>[] then
                j=j+1;
```

```
          elseif strindex(tt_temp(l),'TITLE=')<>[] then
            e=e+1;
            txt_list(j)(1)=[txt_list(j)(1);e];
            title_tmp=strsubst(tt_temp(l),'TITLE=','');
          elseif strindex(tt_temp(l),'DESC=')<>[] then
            desc_tmp=strsubst(tt_temp(l),'DESC=','');
            txt_list(j)(2)=[txt_list(j)(2);title_tmp,desc_tmp];
          elseif strindex(tt_temp(l),'</INDENT>')<>[] then
            j=j-1;
          end
        end
      elseif len_param==0 then
         txt_list=list();
      end
      if fileinfo(rep_xml+namef)<>[] then
         printf("Update %s... ",namef);
         new_tt=put_xml_param3(txt_list,rep_xml+namef);
         mputl(new_tt,rep_xml+namef);
         printf("Done\n");
         k=k+1;
      end
      a=0;b=0;
      i=i+1
    else
      i=i+1
    end
  end
  printf("Processed %d files\n",k);
 else
  printf("file %s not found\n",file_param);
 end

//////////
//'sdesc'
////////
elseif flagn=='sdesc'
 if fileinfo(rep_data+file_sdesc)<>[] then
   printf("Import xml short descriptions...\n");
   tt=mgetl(rep_data+file_sdesc);
   i=1;
   k=0;
   while i<size(tt,1)
     if strindex(tt(i),'<FILE ')<>[] then
       namef=strsubst(tt(i),'<FILE ',"");
       namef=strsubst(namef,'>',"");
       i=i+1;
       txt=tt(i);
       if strindex(txt,'</FILE')<>[] then
         txt=""
       end
       if fileinfo(rep_xml+namef)<>[] then
         printf("Update %s... ",namef);
         new_tt=put_xml_sdesc(txt,rep_xml+namef);
         mputl(new_tt,rep_xml+namef);
         printf("Done\n");
         k=k+1;
       end
     else
       i=i+1
     end
   end
   printf("Processed %d files\n",k);
 else
   printf("file %s not found\n",file_sdesc);
 end

//////////
//see_also
//////////
elseif flagn=='see_also'
 if fileinfo(rep_data+file_see_also)<>[] then
   printf("Import xml see_also...\n");
   tt=mgetl(rep_data+file_see_also);
   i=1;
   k=0;
   a=0;
   b=0;
   while i<size(tt,1)
     if strindex(tt(i),'<FILE ')<>[] then
       a=i;
       namef=strsubst(tt(i),'<FILE ',"");
       namef=strsubst(namef,'>',"");
       i=i+1
     elseif strindex(tt(i),'</FILE ')<>[] then
       b=i;
       num_see_also=evstr(b-a-1);
       if num_see_also>=0 then
         if num_see_also==0 then
          txt=""
         else
          txt=tt(i-num_see_also:i-1)
         end
         if fileinfo(rep_xml+namef)<>[] then
           printf("Update %s... ",namef);
           new_tt=put_xml_see_also(txt,rep_xml+namef);
           mputl(new_tt,rep_xml+namef);
           printf("Done\n");
           k=k+1;
         end
       end
       a=0;b=0;
```

```
      i=i+1
    else
     i=i+1
    end
  end
  printf("Processed %d files\n",k);
 else
  printf("file %s not found\n",file_see_also);
 end

//////////
//'authors'
//////////
elseif flagn=='authors' then
 if fileinfo(rep_data+file_authors)<>[] then
  printf("Import xml authors...\n");
  tt=mgetl(rep_data+file_authors);
  i=1;
  k=0;
  while i<size(tt,1)
    if strindex(tt(i),'<FILE ')<>[] then
      namef=strsubst(tt(i),'<FILE ',"");
      namef=strsubst(namef,'>',"");
      i=i+1;
      num_authors=evstr(tt(i));
      if num_authors<>0 then
        txt=emptystr(num_authors,2);
        txt(:,1)=tt(i+1:i+num_authors);
        txt(:,2)=tt(i+1+num_authors:i+2*num_authors);
        i=i+1+2*num_authors;
        if fileinfo(rep_xml+namef)<>[] then
          printf("Update %s... ",namef);
          new_tt=put_xml_authors(txt,rep_xml+namef);
          mputl(new_tt,rep_xml+namef);
          printf("Done\n");
          k=k+1;
        end
      end
    else
      i=i+1
    end
  end
  printf("Processed %d files\n",k);
 else
  printf("file %s not found\n",file_authors);
 end

///////
//'ex'
///////
elseif flagn=='ex' then
 if fileinfo(rep_data+file_ex)<>[] then
  printf("Import xml & latex examples...\n");
  tt=mgetl(rep_data+file_ex);
  i=1;
  k=0;
  while i<size(tt,1)
    if strindex(tt(i),'<FILE ')<>[] then
      a=i;
      namef=strsubst(tt(i),'<FILE ',"");
      namef=strsubst(namef,'>',"");
      i=i+1;
    elseif strindex(tt(i),'</FILE ')<>[] then
      b=i;
      len_ex=evstr(b-a-1);
      if len_ex>=0 then
        if strindex(tt(a+1),'<LaTeX force>')<>[] then //LaTeX
          txt=tt(i-len_ex+1:i-1);
          name=basename(namef);
          file_tex=tex_path+lang+'/'+name+'/'+name+'_ex.tex';
          if fileinfo(file_tex)==[] then
            if fileinfo(tex_path+lang+'/')==[] then
              unix_g(mkdir_cmd+tex_path+lang+'/');
            end
            if fileinfo(tex_path+lang+'/'+name+'/')==[] then
              unix_g(mkdir_cmd+tex_path+lang+'/'+name);
            end
          end
          printf("Update %s... ",name+'_ex.tex');
          mputl(txt,file_tex);
          k=k+1
          printf("Done\n");
        else //XML
          if strindex(tt(a+1),'<LaTeX')==[] then
            if len_ex==0 then
              txt=""
            else
              txt=tt(i-len_ex:i-1)
            end
            if fileinfo(rep_xml+namef)<>[] then
              printf("Update %s... ",namef);
              new_tt=put_xml_ex(txt,rep_xml+namef);
              mputl(new_tt,rep_xml+namef);
              printf("Done\n");
              k=k+1;
            end
          end
        end
      end
      a=0;b=0;
      i=i+1
```

```
      else
        i=i+1
      end
    end
    printf("Processed %d files\n",k);
  else
    printf("file %s not found\n",file_ex);
  end

 /////////
 //'desc'
 /////////
 elseif flagn=='desc' then
  if fileinfo(rep_data+file_desc)<>[] then
    printf("Import xml & latex long description...\n");
    tt=mgetl(rep_data+file_desc);
    i=1;
    k=0;
    while i<size(tt,1)
      if strindex(tt(i),'<FILE ')<>[] then
        a=i;
        namef=strsubst(tt(i),'<FILE ',"");
        namef=strsubst(namef,'>',"");
        i=i+1;
      elseif strindex(tt(i),'</FILE ')<>[] then
        b=i;
        len_desc=evstr(b-a-1);
        if len_desc>1 then
          if strindex(tt(a+1),'<LaTeX force>')<>[] then //LaTeX
            txt=tt(i-len_desc+1:i-1);
            name=basename(namef);
            file_tex=tex_path+lang+'/'+name+'/'+name+'_long.tex';
            if fileinfo(file_tex)==[] then
              if fileinfo(tex_path+lang+'/')==[] then
                unix_g(mkdir_cmd+tex_path+lang+'/');
              end
              if fileinfo(tex_path+lang+'/'+name+'/')==[] then
                unix_g(mkdir_cmd+tex_path+lang+'/'+name);
              end
            end
            printf("Update %s... ",name+'_long.tex');
            mputl(txt,file_tex);
            k=k+1
            printf("Done\n");
          else //XML
            if strindex(tt(a+1),'<LaTeX')==[] then
              txt=tt(i-len_desc:i-1);
              nb_para=evstr(txt(1)) //nb de paragraphe
              txt_list=list() //initialisation de la liste
              for j=1:nb_para
                txt_list(j)=list()
                txt_list(j)(1)=1   //profondeur d'indentation
                txt_list(j)(2)=""  //le nom du paragraphe
                txt_list(j)(3)=""  //texte du paragraphe
              end
              //remplit la profondeur d'indentation et les titres
              //trouve les délimiteurs <TEXT> </TEXT>
              l=1;c=[];d=[];
              for j=2:size(txt,1)
                if strindex(txt(j),'INDENT=')<>[] then
                  txt_list(l)(1)=evstr(strsubst(txt(j),'INDENT=',""))
                elseif strindex(txt(j),'TITLE=')<>[] then
                  txt_list(l)(2)=strsubst(txt(j),'TITLE=',"")
                elseif strindex(txt(j),'<TEXT>')<>[] then
                  c(l)=j;
                elseif strindex(txt(j),'</TEXT>')<>[] then
                  d(l)=j;
                  l=l+1
                end
              end
              //trouve le texte des paragraphes
              for j=1:nb_para
                txt_list(j)(3)=txt(c(j)+1:d(j)-1)
              end
              if fileinfo(rep_xml+namef)<>[] then
                printf("Update %s... ",namef);
                new_tt=put_xml_desc(txt_list,rep_xml+namef);
                mputl(new_tt,rep_xml+namef);
                printf("Done\n");
                k=k+1;
              end
            end
          end
        elseif len_desc==0|len_desc==1 then
          txt_list=list(list());
          if fileinfo(rep_xml+namef)<>[] then
             printf("Update %s... ",namef);
             new_tt=put_xml_desc(txt_list,rep_xml+namef);
             mputl(new_tt,rep_xml+namef);
             printf("Done\n");
             k=k+1;
          end
          //pause
        end
        a=0;b=0;
        i=i+1
      else
        i=i+1
      end
    end
    printf("Processed %d files\n",k);
```

```
 else
   printf("file %s not found\n",file_desc);
 end

//////////////
//'used_func'
//////////////
elseif flagn=='used_func' then
 if fileinfo(rep_data+file_used_func)<>[] then
   printf("Import xml used functions...\n ");
   tt=mgetl(rep_data+file_used_func);
   i=1;
   k=0;
   while i<size(tt,1)
     if strindex(tt(i),'<FILE ')<>[] then
       namef=strsubst(tt(i),'<FILE ',"");
       namef=strsubst(namef,'>',"");
       a=i;
       i=i+1;
     elseif strindex(tt(i),'</FILE ')<>[] then
       b=i;
       len_used_func=evstr(b-a-1);
       if len_used_func<>0 then
         txt=tt(i-len_used_func:i-1);
         if fileinfo(rep_xml+namef)<>[] then
           printf("Update %s... ",namef);
           new_tt=put_xml_used_func(txt,rep_xml+namef);
           mputl(new_tt,rep_xml+namef);
           printf("Done\n");
           k=k+1;
         end
       end
       a=0;b=0;
       i=i+1
     else
       i=i+1
     end
   end
   printf("Processed %d files\n",k);
 else
   printf("file %s not found\n",file_used_func);
 end

//////////
//'biblio'
//////////
elseif flagn=='biblio' then
 if fileinfo(rep_data+file_biblio)<>[] then
   printf("Import xml & latex bibliography...\n ");
   tt=mgetl(rep_data+file_biblio);
   i=1;
   k=0;
   while i<size(tt,1)
     if strindex(tt(i),'<FILE ')<>[] then
       a=i;
       namef=strsubst(tt(i),'<FILE ',"");
       namef=strsubst(namef,'>',"");
       i=i+1;
     elseif strindex(tt(i),'</FILE ')<>[] then
       b=i;
       len_biblio=evstr(b-a-1);
       if len_biblio<>0 then
         if strindex(tt(a+1),'<LaTeX force>')<>[] then //LaTeX
           txt=tt(i-len_biblio+1:i-1);
           name=basename(namef);
           file_tex=tex_path+lang+'/'+name+'/'+name+'_bib.tex';
           if fileinfo(file_tex)==[] then
             if fileinfo(tex_path+lang+'/')==[] then
               unix_g(mkdir_cmd+tex_path+lang+'/');
             end
             if fileinfo(tex_path+lang+'/'+name+'/')==[] then
               unix_g(mkdir_cmd+tex_path+lang+'/'+name);
             end
           end
           printf("Update %s... ",name+'_bib.tex');
           mputl(txt,file_tex);
           k=k+1
           printf("Done\n");
         else //XML
           if strindex(tt(a+1),'<LaTeX')==[] then
             txt=tt(i-len_biblio:i-1)
             if fileinfo(rep_xml+namef)<>[] then
               printf("Update %s... ",namef);
               new_tt=put_xml_biblio(txt,rep_xml+namef);
               mputl(new_tt,rep_xml+namef);
               printf("Done\n");
               k=k+1;
             end
           end
         end
       end
       a=0;b=0;
       i=i+1
     else
       i=i+1
     end
   end
   printf("Processed %d files\n",k);
 else
   printf("file %s not found\n",file_biblio);
 end
```

```
   //////////////
   //SPECIALDESC
   //////////////
   elseif flagn==’SPECIALDESC’ then
    if fileinfo(rep_data+file_spec_desc)<>[] then
      printf("Import latex special description...\n");
      tt=mgetl(rep_data+file_spec_desc);
      i=1;
      k=0;
      while i<size(tt,1)
        if strindex(tt(i),’<FILE ’)<>[] then
          a=i;
          namef=strsubst(tt(i),’<FILE ’,"");
          namef=strsubst(namef,’>’,"");
          i=i+1;
        elseif strindex(tt(i),’</FILE ’)<>[] then
          b=i;
          len_spec_desc=evstr(b-a-1);
          if len_spec_desc<>0 then
              txt=tt(i-len_spec_desc:i-1);
              name=basename(namef);
              file_tex=tex_path+lang+’/’+name+’/SPECIALDESC’;
              if fileinfo(file_tex)==[] then
                if fileinfo(tex_path+lang+’/’)==[] then
                  unix_g(mkdir_cmd+tex_path+lang+’/’);
                end
                if fileinfo(tex_path+lang+’/’+name+’/’)==[] then
                  unix_g(mkdir_cmd+tex_path+lang+’/’+name);
                end
              end
              printf("Update %s... ",name+’/SPECIALDESC’);
              mputl(txt,file_tex);
              k=k+1;
              printf("Done\n");
          end
          a=0;b=0;
          i=i+1;
        else
          i=i+1;
        end
      end
      printf("Processed %d files\n",k);
    else
      printf("file %s not found\n",file_spec_desc);
    end

   //////////////
   //call_seq
   //////////////
   elseif flagn==’call_seq’ then
      if fileinfo(rep_data+file_call_seq)<>[] then
      printf("Import xml call_seq...\n");
      tt=mgetl(rep_data+file_call_seq);
      i=1;
      k=0;
      a=0;
      b=0;
      while i<size(tt,1)
        if strindex(tt(i),’<FILE ’)<>[] then
          a=i;
          namef=strsubst(tt(i),’<FILE ’,"");
          namef=strsubst(namef,’>’,"");
          i=i+1
        elseif strindex(tt(i),’</FILE ’)<>[] then
          b=i;
          num_call_seq=evstr(b-a-1);
          if num_call_seq>=0 then
            if num_call_seq==0 then
             txt=""
            else
             txt=tt(i-num_call_seq:i-1)
            end
            if fileinfo(rep_xml+namef)<>[] then
              printf("Update %s... ",namef);
              new_tt=put_xml_call_seq(txt,rep_xml+namef);
              mputl(new_tt,rep_xml+namef);
              printf("Done\n");
              k=k+1;
            end
          end
          a=0;b=0;
          i=i+1
        else
         i=i+1
        end
      end
      printf("Processed %d files\n",k);
    else
      printf("file %s not found\n",file_call_seq);
    end

   else
    printf("bad flag\n");
   end //fin if
 end //fin for
endfunction
```

Chapter 3

Xml to tex converter library

3.1 Create arbitrary xml file of scilab man page

- **Name:** generate_xml
- **Library:** xmltotex - Xml to tex converter library

3.1.1 Calling Sequence

```
txt = generate_xml(name,typ,lang)
```

3.1.2 Parameters

- **name :** string. name of the XML file
- **typ :** string. set the type of man page
  - **'block' :** for interfacing function of scicos block
  - **'pal' :** for a palette (.cosf file)
  - **'diagr' :** for a scicos diagram (.cos file)
  - **'scilib' :** for a library of scilab macros
  - **'sci' :** for a scilab macro.
  - **'rout' :** for computational routine
  - **'sim' :** for scilab simulation script (_sim.sce file)
  - **'sce' :** for scilab script (.sce file)
- **lang :** string. set the lang of tex file
  - **'eng' :** to produce english man page
  - **'fr' :** to produce french man page
- **txt :** vector of strings. the output text of the XML file

3.1.3 File content

```
//generate_xml
//Fonction qui génère un fichier xml
//Entrée : name : nom du fichier xml
//         typ : type du fichier xml
//               'block'
//               'diagr'
//               'pal'
//               'scilib'
//               'sci'
//               'sim'
//               'sce'
//               'rout'
```

```
//        lang : langue
//               'eng'
//               'fr'
//Sortie : txt : texte du fichier xml à produire
function txt=generate_xml(name,typ,lang)

 [lsh,rsh]=argn(0)
 if rsh<3 then
   if ~exists('LANGUAGE') then
     global LANGUAGE;lang=LANGUAGE;clear LANGUAGE;
   else
     lang=LANGUAGE;
   end
 end
 if lang<>'fr'&lang<>'eng' then
    printf("Documentation: Unsupported language %s, switch to eng.\n",lang);
    lang='eng';
 end

 tt_param=[];
 tt_pair_block=[];
 tt_call_seq=[];
 tt_rmk=[];
 tt_ex=[];
 tt_used_func=[];
 select typ
   case 'block'
        typel='Scicos Block'

        tt_typ=return_typ_block(name);
        tt_lables=return_lables_block(name);

        tt_param=['<PARAM>';'  <PARAM_INDENT>';'']
        if tt_typ<>[] then
           for i=1:size(tt_lables,1)
              tt_param=[tt_param;'  <PARAM_ITEM>';
                        '  <PARAM_NAME>'+tt_lables(i,1)+'</PARAM_NAME>';
                        '  <PARAM_DESCRIPTION>';'    <SP>';];
              if lang=='fr' then
                tt_param=[tt_param;
                        '    : Type '''+tt_typ(i,1)+''' de taille '+tt_typ(i,2)+...
                        '. La description du paramètre '+string(i)+'.';]
              else
                tt_param=[tt_param;
                        '    : Type '''+tt_typ(i,1)+''' of size '+tt_typ(i,2)+...
                        '. The parameter description '+string(i)+'.';]
              end
               tt_param=[tt_param;'    </SP>';'  </PARAM_DESCRIPTION>';'  </PARAM_ITEM>';''];
           end
        end
        tt_param='  '+[tt_param;'</PARAM_INDENT>';'</PARAM>']
        tt_ex='  '+['<EXAMPLE>';'<P>';'Example'
              '</P>';'</EXAMPLE>']
        //tt_pair_block='  '+['<PAIR_BLOCK>';'<PAIR_BLOCK_ITEM> </PAIR_BLOCK_ITEM>';'</PAIR_BLOCK>']
        tt_used_func='  '+['<USED_FUNCTIONS>';'  <SP>';'  </SP>';'</USED_FUNCTIONS>']

   case 'pal'
        typel='Scicos Palette'

   case 'diagr'
        typel='Scicos Diagram'
        tt_ex='  '+['<EXAMPLE>';'<P>';'Example'
              '</P>';'</EXAMPLE>']

   case 'scilib'
        typel='Scilab Library'

   case 'sci'
        prot=funcprot();funcprot(0);
        ierr=execstr('txt=help_skeleton(name);','errcatch');
        funcprot(prot);
        if ierr==0 then
          return
        else
          typel='Scilab Function'
          tt_param='  '+['<PARAM>';'<PARAM_INDENT>';'';'  <PARAM_ITEM>';
                         '  <PARAM_NAME>Name of param 1</PARAM_NAME>';
                         '  <PARAM_DESCRIPTION>';'    <SP>';'    : add here the parameter description';
                         '    </SP>';'  </PARAM_DESCRIPTION>';'  </PARAM_ITEM>';'';'</PARAM_INDENT>';
                        '</PARAM>'];

          tt_call_seq='  '+['<CALLING_SEQUENCE>';
                           '<CALLING_SEQUENCE_ITEM> </CALLING_SEQUENCE_ITEM>';
                           '</CALLING_SEQUENCE>']

          tt_ex='  '+['<EXAMPLE><![CDATA[';
                     ']]></EXAMPLE>']

          tt_used_func='  '+['<USED_FUNCTIONS>';'  <SP>';'  </SP>';'</USED_FUNCTIONS>']
        end

   case 'rout'
        typel='Low level routine'
        tt_param='  '+['<PARAM>';'<PARAM_INDENT>';'';'  <PARAM_ITEM>';
                       '  <PARAM_NAME>Name of param 1</PARAM_NAME>';
                       '  <PARAM_DESCRIPTION>';'    <SP>';'    : add here the parameter description';
                       '    </SP>';'  </PARAM_DESCRIPTION>';'  </PARAM_ITEM>';'';'</PARAM_INDENT>';
                      '</PARAM>'];

        tt_call_seq='  '+['<CALLING_SEQUENCE>';
                         '<CALLING_SEQUENCE_ITEM> </CALLING_SEQUENCE_ITEM>';
```

```
                        '</CALLING_SEQUENCE>']

        tt_ex='  '+['<EXAMPLE><![CDATA[';
                   ']]></EXAMPLE>']

        tt_used_func='  '+['<USED_FUNCTIONS>';'  <SP>';'  </SP>';'</USED_FUNCTIONS>']

   case 'sce'
        typel='Scilab Script'

        tt_used_func='  '+['<USED_FUNCTIONS>';'  <SP>';'  </SP>';'</USED_FUNCTIONS>']

        tt_call_seq='  '+['<CALLING_SEQUENCE>';
                          '<CALLING_SEQUENCE_ITEM> </CALLING_SEQUENCE_ITEM>';
                          '</CALLING_SEQUENCE>']

   case 'sim'
        typel='Scilab simulation Script'

        tt_used_func='  '+['<USED_FUNCTIONS>';'  <SP>';'  </SP>';'</USED_FUNCTIONS>']
 end

 txt=['<?xml version='"1.0'" encoding='"ISO-8859-1'" standalone='"no'"?>'
      '<!DOCTYPE MAN SYSTEM '"'+SCI+'/man/man.dtd'">'
      '<MAN>'
      '  <LANGUAGE>'+lang+'</LANGUAGE>'
      '  <TITLE>'+name+'</TITLE>'
      '  <TYPE>'+typel+'</TYPE>'
      '  <DATE>'+date()+'</DATE>'
      '  <SHORT_DESCRIPTION name='"'+name+''">'+name+' title</SHORT_DESCRIPTION>'
      tt_call_seq
      ''
      '  <DESCRIPTION>'
      '    <DESCRIPTION_INDENT>'
      '    <DESCRIPTION_ITEM>'
      '    <P>'
      '    Add here a paragraph of the function description.'
      '    </P>'
      '    </DESCRIPTION_ITEM>'
      '    </DESCRIPTION_INDENT>'
      '  </DESCRIPTION>'
      ''
      tt_rmk
      ''
      tt_ex
      ''
      tt_param
      ''
      tt_pair_block
      ''
      '  <SEE_ALSO>'
      '  <SEE_ALSO_ITEM> </SEE_ALSO_ITEM>'
      '  </SEE_ALSO>'
      ''
      '  <AUTHORS>'
      '    <AUTHORS_ITEM label=''IRCOM Group''>'
      '    generate_xml'
      '    </AUTHORS_ITEM>'
      '  </AUTHORS>'
      ''
      '  <BIBLIO>'
      '    <SP>'
      '    </SP>'
      '  </BIBLIO>'
      ''
      tt_used_func
      ''
      '</MAN>'
      ]
endfunction
```

3.2 Update authors name and references in xml file

- **Name:** put_xml_authors
- **Library:** xmltotex - Xml to tex converter library

3.2.1 Calling Sequence

```
new_tt = put_xml_authors(txt,filen)
```

3.2.2 Parameters

- **txt :** matrix of strings of size (n,2)
  - **txt(,1) :** name of author
  - **txt(,2) :** reference of author

- **filen :** string. target XML file (path+name)
- **new_tt :** vector of strings. the text of the new XML file

3.2.3 File content

```
//put_xml_authors
//Fonction qui insère un
//dans un fichier xml contenant des délimiteurs
//<AUTHORS> et </AUTHORS>
//Entrée : txt : matrice de chaînes de caractères de taille n,2
//         txt(,1) : Nom de l'auteur
//         txt(,2) : références de l'auteur
//         filen : nom du fichier xml (ex:filen=xml_path+'CAN_f.xml')
function new_tt=put_xml_authors(txt,filen)
 if fileinfo(filen)<>[] then
  if size(txt,2)==2 then
    del1='<AUTHORS>'
    del2='</AUTHORS>'
    tt_sav=mgetl(filen);
    a=0;b=0;new_tt=tt_sav;
    //trouve la position des délimiteurs
    for i=1:size(tt_sav,1)
     if strindex(tt_sav(i),del1)<>[] then a=i, end;
     if strindex(tt_sav(i),del2)<>[] then b=i, end;
    end
    if a<>0&b<>0 then
      tt_authors=[]
      for j=1:size(txt,1)
        tt_authors=[tt_authors;'  <AUTHORS_ITEM label='''+txt(j,1)+'''>';
                    '  '+txt(j,2);'  </AUTHORS_ITEM>']
      end
      //Ecrit la chaine de texte finale
      new_tt=[tt_sav(1:a);tt_authors;tt_sav(b:size(tt_sav,1))]
    end
  else
   printf("Incompatible rsh variable\n");
  end
 else
  printf("File %s not found\n",filen);
  new_tt=[];
 end
endfunction
```

3.3 Update bibliography references in xml file

- **Name:** put_xml_biblio
- **Library:** xmltotex - Xml to tex converter library

3.3.1 Calling Sequence

```
new_tt = put_xml_biblio(txt,filen)
```

3.3.2 Parameters

- **txt :** vector of strings. the vector of reference
- **filen :** string. target XML file (path+name)
- **new_tt :** vector of strings. the text of the new XML file

3.3.3 File content

```
//put_xml_biblio
//Fonction qui insère les références bibliographiques
//dans un fichier xml contenant des délimiteurs
//<BIBLIO> et </BIBLIO>
//Entrée : txt : vecteur de chaînes de caractères de taille n,1
//               contenant les ref. biblio.
//         filen : nom du fichier xml (ex:filen=xml_path+'CAN_f.xml')
function new_tt=put_xml_biblio(txt,filen)
 if fileinfo(filen)<>[] then
  if size(txt,2)==1 then
    del1='<BIBLIO>'
    del2='</BIBLIO>'
    tt_sav=mgetl(filen);
    a=0;b=0;new_tt=tt_sav;
    //trouve la position des délimiteurs
    for i=1:size(tt_sav,1)
     if strindex(tt_sav(i),del1)<>[] then a=i, end;
```

```
     if strindex(tt_sav(i),del2)<>[] then b=i, end;
    end
    if a<>0&b<>0 then
      tt_biblio='  '+['<BIBLIO>'
                       '  <SP>';'  '+txt(:,1);'  </SP>'
                      '</BIBLIO>']

      //Ecrit la chaine de texte finale
      new_tt=[tt_sav(1:a-1);tt_biblio;tt_sav(b+1:size(tt_sav,1))]
    end
  else
   printf("Incompatible rsh variable\n");
  end
 else
  printf("File %s not found\n",filen);
  new_tt=[];
 end
endfunction
```

3.4 Update calling sequence in xml file

- **Name:** put_xml_call_seq
- **Library:** xmltotex - Xml to tex converter library

3.4.1 Calling Sequence

```
new_tt = put_xml_call_seq(txt,filen)
```

3.4.2 Parameters

- **txt :** add here the parameter description
- **filen :** add here the parameter description
- **new_tt :** 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 Example

```
Add here scilab instructions and comments
```

3.4.5 File content

```
//put_xml_call_seq
//Fonction qui insère le paragraph calling sequence
//dans un fichier xml contenant des délimiteurs
//<CALLING_SEQUENCE> et </CALLING_SEQUENCE>
//Entrée : txt : un vecteur de chaînes de caractères de taille n,1
//         filen : nom du fichier xml (ex:filen=xml_path+'CAN_f.xml')
function new_tt=put_xml_call_seq(txt,filen)
 if fileinfo(filen)<>[] then
  if size(txt,2)==1 then
    del1='<CALLING_SEQUENCE>'
    del2='</CALLING_SEQUENCE>'
    tt_sav=mgetl(filen);
    a=0;b=0;new_tt=tt_sav;
    //trouve la position des délimiteurs
    for i=1:size(tt_sav,1)
     if strindex(tt_sav(i),del1)<>[] then a=i, end;
     if strindex(tt_sav(i),del2)<>[] then b=i, end;
    end
    if a<>0&b<>0 then
      tt_see_also=['<CALLING_SEQUENCE>']
      for j=1:size(txt,1)
        tt_see_also=[tt_see_also;
                     '<CALLING_SEQUENCE_ITEM> '+txt(j,1)+' </CALLING_SEQUENCE_ITEM>'];
      end
      tt_see_also=[tt_see_also;'</CALLING_SEQUENCE>']
      //Ecrit la chaine de texte finale
      new_tt=[tt_sav(1:a-1);'  '+tt_see_also;tt_sav(b+1:size(tt_sav,1))]
    end
  else
```

```
   printf("Incompatible rsh variable\n");
  end
 else
  printf("File %s not found\n",filen);
  new_tt=[];
 end
endfunction
```

3.4.6 Used function(s)

Add here the used function name and references

3.4.7 See Also

- add a key here - ()
- add a key here - ()

3.4.8 Bibliography

Add here the function bibliography if any

3.5 Update long description in xml file

- **Name:** put_xml_desc
- **Library:** xmltotex - Xml to tex converter library

3.5.1 Calling Sequence

```
new_tt = put_xml_desc(list_txt,filen)
```

3.5.2 Parameters

- **list_txt :** list
  - **list_txt() :** integer. id of paragraph
  - **list_txt()(1) :** integer. indentation level
  - **list_txt()(2) :** string. title of paragraph
  - **list_txt()(3) :** vector of strings. text of paragraph
- **filen :** string. target XML file (path+name)
- **new_tt :** vector of strings. the text of the new XML file

3.5.3 File content

```
//put_xml_desc
//Fonction qui insère une section description
//dans un fichier xml contenant des délimiteurs
//<DESCRIPTION> et </DESCRIPTION>
//Entrée : list_txt()() : une liste
//                | |
//                | |
//                | |---->  1 : profondeur d'indentation
//                |         2 : le titre du paragraphe
//                |         3 : le texte du paragraphe
//                |---> n° du paragraphe
//         filen : nom du fichier xml (ex:filen=xml_path+'CAN_f.xml')
function new_tt=put_xml_desc(list_txt,filen)
 if fileinfo(filen)<>[] then
   del1='<DESCRIPTION>'
   del2='</DESCRIPTION>'
   tt_sav=mgetl(filen);
   a=0;b=0;new_tt=tt_sav;
   //trouve la position des délimiteurs
   for i=1:size(tt_sav,1)
```

```
    if strindex(tt_sav(i),del1)<>[] then a=i, end;
    if strindex(tt_sav(i),del2)<>[] then b=i, end;
   end
   if size(list_txt(1))==3 then
     if a<>0&b<>0 then
       n=1; //profondeur d'indentation
       tt_long=[];
       for i=1:size(list_txt)
         if list_txt(i)(1)>n then
           for j=1:list_txt(i)(1)-n
             tt_long=[tt_long;'<DESCRIPTION_INDENT>'}
           end
           n=list_txt(i)(1);
         end

         if i>1 & list_txt(i)(1)<n then
           for j=1:n-list_txt(i)(1)
             tt_long=[tt_long;'</DESCRIPTION_INDENT>'}
           end
           n=list_txt(i)(1);
         end

         if list_txt(i)(2)<>"" then
           tt_long=[tt_long;'<DESCRIPTION_ITEM label='+list_txt(i)(2)+'>';
               '<P>';list_txt(i)(3);'</P>';'</DESCRIPTION_ITEM>']
         else
           tt_long=[tt_long;'<DESCRIPTION_ITEM>';
               '<P>';list_txt(i)(3);'</P>';'</DESCRIPTION_ITEM>']
         end
       end
       if n>1 then
         for j=1:(n-1)
           tt_long=[tt_long;'</DESCRIPTION_INDENT>'}
         end
       end
       tt_long="  "+tt_long;
       //Ecrit la chaine de texte finale
       new_tt=[tt_sav(1:a);tt_long;tt_sav(b:size(tt_sav,1))]
     end
   elseif size(list_txt(1))==0 then
     if a<>0&b<>0 then
       tt_long=[];
       new_tt=[tt_sav(1:a);tt_long;tt_sav(b:size(tt_sav,1))]
     end
     //pause
   else
    printf("Incompatible rsh variable\n");
    new_tt=[];
   end
 else
  printf("File %s not found\n",filen);
  new_tt=[];
 end
endfunction
```

3.6 Update examples in xml file

- **Name:** put_xml_ex
- **Library:** xmltotex - Xml to tex converter library

3.6.1 Calling Sequence

```
new_tt = put_xml_ex(txt,filen)
```

3.6.2 Parameters

- **txt :** vector of strings. text of example paragraph
- **filen :** string. target XML file (path+name)
- **new_tt :** vector of strings. the text of the new XML file

3.6.3 File content

```
//put_xml_ex
//Fonction qui insère une liste d'exemple
//dans un fichier xml contenant des délimiteurs
//<EXAMPLE> et </EXAMPLE>
//Entrée : txt : matrice de chaînes de caractères de taille n,1
//         filen : nom du fichier xml (ex:filen=xml_path+'CAN_f.xml')
function new_tt=put_xml_ex(txt,filen)
 if fileinfo(filen)<>[] then
  flag_block=%f;
```

```
  if size(txt,2)==1 then
    del1='<EXAMPLE>'
    del2='</EXAMPLE>'
    tt_sav=mgetl(filen);
    a=0;b=0;new_tt=tt_sav;
    //trouve la position des délimiteurs
    for i=1:size(tt_sav,1)
     if strindex(tt_sav(i),'Scicos Block')<>[] then flag_block=%t, end;
     if strindex(tt_sav(i),del1)<>[] then a=i, end;
     if strindex(tt_sav(i),del2)<>[] then b=i, end;
    end
    if a<>0&b<>0 then
     //Crée le nouveau paragraphe d'exemple
     if flag_block then
      tt_ex=['<EXAMPLE>';'<P>']
     else
      tt_ex=['<EXAMPLE><![CDATA[']
     end
     tt_ex=[tt_ex;txt(:,1)]
     if flag_block then
       tt_ex=[tt_ex;'</P>';'</EXAMPLE>']
     else
       tt_ex=[tt_ex;']]></EXAMPLE>']
     end
     //Ecrit la chaine de texte finale
     new_tt=[tt_sav(1:a-1);'  '+tt_ex;tt_sav(b+1:size(tt_sav,1))]
    end
  else
   printf("Incompatible rsh variable\n");
  end
 else
  printf("File %s not found\n",filen);
  new_tt=[];
 end
endfunction
```

3.7 Update description of parameters in xml file

- **Name:** put_xml_param
- **Library:** xmltotex - Xml to tex converter library

3.7.1 Calling Sequence

```
new_tt = put_xml_param(txt,filen)
```

3.7.2 Parameters

- **txt :** matrix of strings of size (n,2)
  - **txt(,1) :** name of parameter
  - **txt(,2) :** description of parameter
- **filen :** string. target XML file (path+name)
- **new_tt :** vector of strings. the text of the new XML file

3.7.3 File content

```
//put_xml_param
//Fonction qui insère une liste de paramètres
//dans un fichier xml contenant des délimiteurs
//<PARAM> et </PARAM>
//Entrée : txt : matrice de chaînes de caractères de taille n,2
//         txt(,1) : le nom du paramètre
//         txt(,2) : la description du paramètre
//         filen : nom du fichier xml (ex:filen=xml_path+'CAN_f.xml')
//Sortie : new_tt : texte du fichier xml de sortie
function new_tt=put_xml_param(txt,filen)
 if fileinfo(filen)<>[] then
  if size(txt,2)==2 then
    del1='<PARAM>'
    del2='</PARAM>'
    tt_sav=mgetl(filen);
    a=0;b=0;new_tt=tt_sav;
    //trouve la position des délimiteurs
    for i=1:size(tt_sav,1)
     if strindex(tt_sav(i),del1)<>[] then a=i, end;
     if strindex(tt_sav(i),del2)<>[] then b=i, end;
    end
    if a<>0&b<>0 then
      //crée la nouvelle liste des paramètres
```

```
      tt_param=['<PARAM_INDENT>';'']
      for i=1:size(txt,1)
        tt_param=[tt_param;'<PARAM_ITEM>';'<PARAM_NAME>'+txt(i,1)+'</PARAM_NAME>';
                  '<PARAM_DESCRIPTION>';'<SP>';
                  ': '+txt(i,2);
                  '</SP>';'</PARAM_DESCRIPTION>';'</PARAM_ITEM>';'']
      end
      tt_param=[tt_param;'</PARAM_INDENT>']

      //Ecrit la chaine de texte finale
      new_tt=[tt_sav(1:a);tt_param;tt_sav(b:size(tt_sav,1))];
    end
  else
   printf("Incompatible rsh variable\n");
  end
 else
  printf("File %s not found\n",filen);
  new_tt=[];
 end
endfunction
```

3.8 Update description of parameters in xml file

- **Name:** put_xml_param2
- **Library:** xmltotex - Xml to tex converter library

3.8.1 Calling Sequence

```
new_tt = put_xml_param2(txt_list,filen)
```

3.8.2 Parameters

- **txt_list :** list
  - **list_txt() :** integer. indentation level
  - **list_txt()(1) :** integer. id of parameter
  - **list_txt()(2) :** matrix of string of size(n,2)
    - **list_txt()(2)(,1) :** string. name of parameter
    - **list_txt()(2)(,2) :** string. description of parameter
- **filen :** string. target XML file (path+name)
- **new_tt :** vector of strings. the text of the new XML file

3.8.3 File content

```
//put_xml_param
//Fonction qui insère une liste de paramètres
//dans un fichier xml contenant des délimiteurs
//<PARAM> et </PARAM>
//Entrée : txt : matrice de chaînes de caractères de taille n,2
//         txt(,1) : le nom du paramètre
//         txt(,2) : la description du paramètre
//         filen : nom du fichier xml (ex:filen=xml_path+'CAN_f.xml')
//Sortie : new_tt : texte du fichier xml de sortie
//Rmq : ce code est incompréhensible et est trop compliqué.
function new_tt=put_xml_param2(txt_list,filen)
 if fileinfo(filen)<>[] then
  //if size(txt,2)==2 then
    del1='<PARAM>'
    del2='</PARAM>'
    tt_sav=mgetl(filen);
    a=0;b=0;new_tt=tt_sav;
    //trouve la position des délimiteurs
    for i=1:size(tt_sav,1)
     if strindex(tt_sav(i),del1)<>[] then a=i, end;
     if strindex(tt_sav(i),del2)<>[] then b=i, end;
    end

    if a<>0&b<>0 then

      if txt_list<>list() then
        nb_indent=size(txt_list);
        nb_param=0
        for i=1:size(txt_list)
          nb_param=nb_param+size(txt_list(i)(2),1);
        end
```

```
    //Creation d'un nouvelle liste
    param_list=list();
    for i=1:nb_param
      param_list(i)=list();
      param_list(i)=[];
    end
    //Remplissage de la liste
    i=nb_indent;
    for i=nb_indent:-1:2
     for j=1:size(txt_list(i)(1),1)
       //Premier élément de la liste
       if j==1 then
          param_list(txt_list(i)(1)(j))=['<PARAM_INDENT>';
                '<PARAM_ITEM>';'<PARAM_NAME>'+txt_list(i)(2)(j,1)+'</PARAM_NAME>';
                '<PARAM_DESCRIPTION>';'<SP>';': '+txt_list(i)(2)(j,2);
                '</SP>']
          //cherche la postion de la fin de l'item
          if param_list(txt_list(i)(1)(j)+1)==[] then
            param_list(txt_list(i)(1)(j))=[param_list(txt_list(i)(1)(j));'</PARAM_DESCRIPTION>';'</PARAM_ITEM>';''];
          else
             zz=txt_list(i)(1)(j)+1;
             while param_list(zz)<>[]
               zz=zz+1;
               if zz==nb_param+1 then break, end
             end
             param_list(zz-1)=[param_list(zz-1);'</PARAM_DESCRIPTION>';'</PARAM_ITEM>';''];
          end
          if size(txt_list(i)(1),1)==1 then
             param_list(txt_list(i)(1)(j))=[param_list(txt_list(i)(1)(j));'</PARAM_INDENT>';''];
          end
       //Dernier élement de la liste
       elseif j==size(txt_list(i)(1),1)
          param_list(txt_list(i)(1)(j))=[
               '<PARAM_ITEM>';'<PARAM_NAME>'+txt_list(i)(2)(j,1)+'</PARAM_NAME>';
               '<PARAM_DESCRIPTION>';'<SP>';
               ': '+txt_list(i)(2)(j,2);
               '</SP>';'</PARAM_DESCRIPTION>';'</PARAM_ITEM>';'</PARAM_INDENT>';''];

      else
  //Teste la discontinuité dans la liste
         if txt_list(i)(1)(j)<>txt_list(i)(1)(j-1)+1 then
           if param_list(txt_list(i)(1)(j-1)+1)==[] then
             param_list(txt_list(i)(1)(j-1))=[param_list(txt_list(i)(1)(j-1));'</PARAM_INDENT>'];
           else
             zz=txt_list(i)(1)(j-1)+1;
             while param_list(zz)<>[]
                zz=zz+1;
                if zz==nb_param+1 then break, end
             end
             param_list(zz-1)=[param_list(zz-1);'</PARAM_INDENT>'];
           end
           param_list(txt_list(i)(1)(j))=['<PARAM_INDENT>';
                '<PARAM_ITEM>';'<PARAM_NAME>'+txt_list(i)(2)(j,1)+'</PARAM_NAME>';
                '<PARAM_DESCRIPTION>';'<SP>';
                ': '+txt_list(i)(2)(j,2);
                '</SP>']
           if param_list(txt_list(i)(1)(j)+1)==[] then
             param_list(txt_list(i)(1)(j))=[param_list(txt_list(i)(1)(j));'</PARAM_DESCRIPTION>';'</PARAM_ITEM>';''];
           else
             zz=txt_list(i)(1)(j)+1;
             while param_list(zz)<>[]
               zz=zz+1;
               if zz==nb_param+1 then break, end
             end
             param_list(zz-1)=[param_list(zz-1);'</PARAM_DESCRIPTION>';'</PARAM_ITEM>';''];
           end

         else
           param_list(txt_list(i)(1)(j))=[
                '<PARAM_ITEM>';'<PARAM_NAME>'+txt_list(i)(2)(j,1)+'</PARAM_NAME>';
                '<PARAM_DESCRIPTION>';'<SP>';
                ': '+txt_list(i)(2)(j,2);
                '</SP>']
           if param_list(txt_list(i)(1)(j)+1)==[] then
             param_list(txt_list(i)(1)(j))=[param_list(txt_list(i)(1)(j));'</PARAM_DESCRIPTION>';'</PARAM_ITEM>';''];
           else
              zz=txt_list(i)(1)(j)+1;
              while param_list(zz)<>[]
                zz=zz+1;
                if zz==nb_param+1 then break, end
              end
              param_list(zz-1)=[param_list(zz-1);'</PARAM_DESCRIPTION>';'</PARAM_ITEM>';''];
           end
         end
       end
     end
    end
    //pause
    //Niveau 1
    for i=1:size(txt_list(1)(1),1)
      my_tt=['<PARAM_ITEM>';
             '<PARAM_NAME>'+txt_list(1)(2)(i,1)+'</PARAM_NAME>';
             '<PARAM_DESCRIPTION>';'<SP>';': '+txt_list(1)(2)(i,2);'</SP>'];
      param_list(txt_list(1)(1)(i))=my_tt;
      if i<=size(txt_list(1)(1),1) then
        if (txt_list(1)(1)(i))<nb_param then

          if param_list(txt_list(1)(1)(i)+1)==[] then
             param_list(txt_list(1)(1)(i))=[param_list(txt_list(1)(1)(i));'</PARAM_DESCRIPTION>';'</PARAM_ITEM>';''];
          else
```

```
              zz=txt_list(1)(1)(i)+1;
              while param_list(zz)<>[]
                zz=zz+1;
                if zz==nb_param+1 then break, end
              end
              param_list(zz-1)=[param_list(zz-1);'</PARAM_DESCRIPTION>';'</PARAM_ITEM>';''];
            end

           //pause
           elseif (txt_list(1)(1)(i))==nb_param then
             param_list(txt_list(1)(1)(i))=[param_list(txt_list(1)(1)(i));'</PARAM_DESCRIPTION>';'</PARAM_ITEM>';''];
           end
         end
       end
       //pause
       //crée la chaine de texte des paramètres
       tt_param=['<PARAM_INDENT>']
       for i=1:nb_param
        tt_param=[tt_param;param_list(i)];
       end
       tt_param=[tt_param;'</PARAM_INDENT>'];

     else
      tt_param=[];
     end
     //Ecrit la chaine de texte finale
     new_tt=[tt_sav(1:a);tt_param;tt_sav(b:size(tt_sav,1))];

   end
  //else
   //printf("Incompatible rsh variable\n");
  //end
 else
  printf("File %s not found\n",filen);
  new_tt=[];
 end
endfunction
```

3.9 Update description of parameters in xml file

- **Name:** put_xml_param3
- **Library:** xmltotex - Xml to tex converter library

3.9.1 Calling Sequence

```
new_tt = put_xml_param3(txt_list,filen)
```

3.9.2 Parameters

- **txt_list :** list
  - **list_txt() :** integer. indentation level
  - **list_txt()(1) :** integer. id of parameter
  - **list_txt()(2) :** matrix of string of size(n,2)
    - **list_txt()(2)(,1) :** string. name of parameter
    - **list_txt()(2)(,2) :** string. description of parameter
- **filen :** string. target XML file (path+name)
- **new_tt :** vector of strings. the text of the new XML file

3.9.3 File content

```
//put_xml_param3
//Fonction qui insère une liste de paramètres
//dans un fichier xml contenant des délimiteurs
//<PARAM> et </PARAM>
//Entrée : txt : matrice de chaînes de caractères de taille n,2
//         txt(,1) : le nom du paramètre
//         txt(,2) : la description du paramètre
//         filen : nom du fichier xml (ex:filen=xml_path+'CAN_f.xml')
//Sortie : new_tt : texte du fichier xml de sortie
function new_tt=put_xml_param3(txt_list,filen)

    if fileinfo(filen)<>[] then
       del1='<PARAM>'
       del2='</PARAM>'
```

```
      tt_sav=mgetl(filen);
      a=0;b=0;new_tt=tt_sav;
      //trouve la position des délimiteurs
      for i=1:size(tt_sav,1)
         if strindex(tt_sav(i),del1)<>[] then a=i, end;
         if strindex(tt_sav(i),del2)<>[] then b=i, end;
      end
      //si les délimiteurs ont été trouvés
      if a<>0&b<>0 then
         //si la chaine txt n'es pas vide
         if txt_list<>list() then
            //trouve le nbre de paramètres
            nb_para=0;
            for i=1:size(txt_list)
               for j=1:size(txt_list(i)(1),1)
                  nb_para=nb_para+1;
               end
            end
            //initialise une nouvelle liste
            n=list()
            for i=1:nb_para
               n(i)=list();
               n(i)(1)=0;
               n(i)(2)=""
               n(i)(3)=""
            end
            //met la liste dans l'ordre des paragraphes
            for i=1:size(txt_list)
              for j=1:size(txt_list(i)(1),1)
                 n(txt_list(i)(1)(j,1))(1)=i;
                 n(txt_list(i)(1)(j,1))(2)=txt_list(i)(2)(j,1);
                 n(txt_list(i)(1)(j,1))(3)=txt_list(i)(2)(j,2);
              end
            end
            //crée la liste des param au format xml
            tt_param=[];
            pre_i=0; //indentation précédente
            for i=1:size(n)
               dif_i=n(i)(1)-pre_i;
               if dif_i<>0 then
                  if dif_i>0 then
                     for j=1:dif_i
                       tt_param=[tt_param;'  <PARAM_INDENT>'];
                     end
                  elseif dif_i<0 then
                     for j=-1:-1:dif_i
                       tt_param=[tt_param;'  </PARAM_INDENT>'];
                     end
                  end
               end
               pre_i=n(i)(1);
               tt_param=[tt_param;'  <PARAM_ITEM>';
                         '  <PARAM_NAME>'+n(i)(2)+'</PARAM_NAME>';
                         '  <PARAM_DESCRIPTION>';
                         '  <SP>';
                         '  '+n(i)(3);
                         '  </SP>';
                         '  </PARAM_DESCRIPTION>';
                         '  </PARAM_ITEM>';];
            end
            dif_i=-pre_i;
            if dif_i<0 then
               for j=-1:-1:dif_i
                  tt_param=[tt_param;'  </PARAM_INDENT>'];
               end
            end
         else
            tt_param=[];
         end
         //Ecrit la chaine de texte finale
         new_tt=[tt_sav(1:a);tt_param;tt_sav(b:size(tt_sav,1))];
      end
    else
      printf("File %s not found\n",filen);
      new_tt=[];
    end

endfunction
```

3.10 Update short description in xml file

- **Name:** put_xml_sdesc
- **Library:** xmltotex - Xml to tex converter library

3.10.1 Calling Sequence

```
new_tt = put_xml_sdesc(txt,filen)
```

3.10.2 Parameters

- **txt :** string. short description
- **filen :** string. target XML file (path+name)
- **new_tt :** vector of strings. the text of the new XML file

3.10.3 File content

```
//put_xml_sdesc
//Fonction qui insère une description courte
//dans un fichier xml contenant des délimiteurs
//<SHORT_DESCRIPTION et </SHORT_DESCRIPTION>
//Entrée : txt : chaînes de caractères de taille 1
//              contenant la description courte
//         filen : nom du fichier xml (ex:filen=xml_path+'CAN_f.xml')
function new_tt=put_xml_sdesc(txt,filen)
 if fileinfo(filen)<>[] then
  [p,q]=size(txt)
  if p==1 & q==1 then
    del1='<SHORT_DESCRIPTION'
    del2='</SHORT_DESCRIPTION>'
    tt_sav=mgetl(filen);
    a=0;b=0;new_tt=tt_sav;
    //trouve la position des délimiteurs
    for i=1:size(tt_sav,1)
     if strindex(tt_sav(i),del1)<>[] then a=i, end;
     if strindex(tt_sav(i),del2)<>[] then b=i, end;
    end
    if a<>0&b<>0 then
      name=basename(filen)
      tt_sdesc=['  <SHORT_DESCRIPTION name='"'+name+''">'+txt+'</SHORT_DESCRIPTION>']
      if a==b then
        new_tt=[tt_sav(1:a-1);tt_sdesc;tt_sav(b+1:size(tt_sav,1))]
      end
    end
  else
   printf("Incompatible rsh variable\n");
  end
 else
  printf("File %s not found\n",filen);
  new_tt=[];
 end
endfunction
```

3.11 Update see also section in xml file

- **Name:** put_xml_see_also
- **Library:** xmltotex - Xml to tex converter library

3.11.1 Calling Sequence

```
new_tt = put_xml_see_also(txt,filen)
```

3.11.2 Parameters

- **txt :** vector of strings. see_also item
- **filen :** string. target XML file (path+name)
- **new_tt :** vector of strings. the text of the new XML file

3.11.3 File content

```
//put_xml_see_also
//Fonction qui insère un
//dans un fichier xml contenant des délimiteurs
//<SEE_ALSO> et </SEE_ALSO>
//Entrée : txt : un vecteur de chaînes de caractères de taille n,1
//         filen : nom du fichier xml (ex:filen=xml_path+'CAN_f.xml')
function new_tt=put_xml_see_also(txt,filen)
 if fileinfo(filen)<>[] then
  if size(txt,2)==1 then
    del1='<SEE_ALSO>'
    del2='</SEE_ALSO>'
    tt_sav=mgetl(filen);
    a=0;b=0;new_tt=tt_sav;
```

```
    //trouve la position des délimiteurs
    for i=1:size(tt_sav,1)
     if strindex(tt_sav(i),del1)<>[] then a=i, end;
     if strindex(tt_sav(i),del2)<>[] then b=i, end;
    end
    if a<>0&b<>0 then
      tt_see_also=['<SEE_ALSO>']
      for j=1:size(txt,1)
        tt_see_also=[tt_see_also;
                     '<SEE_ALSO_ITEM> <LINK>'+txt(j,1)+'</LINK> </SEE_ALSO_ITEM>'];
      end
      tt_see_also=[tt_see_also;'</SEE_ALSO>']
      //Ecrit la chaine de texte finale
      new_tt=[tt_sav(1:a-1);'  '+tt_see_also;tt_sav(b+1:size(tt_sav,1))]
    end
  else
   printf("Incompatible rsh variable\n");
  end
 else
  printf("File %s not found\n",filen);
  new_tt=[];
 end
endfunction
```

3.12 Skeleton function for update of xml file

- **Name:** put_xml_sk
- **Library:** xmltotex - Xml to tex converter library

3.12.1 File content

```
//put_xml_
//Fonction qui insère un
//dans un fichier xml contenant des délimiteurs
//<> et </>
//Entrée : txt : matrice de chaînes de caractères de taille n,
//
//
//         filen : nom du fichier xml (ex:filen=xml_path+'CAN_f.xml')
function new_tt=put_xml_(txt,filen)
 if fileinfo(filen)<>[] then
  if size(txt,2)== then
    del1='<>'
    del2='</>'
    tt_sav=mgetl(filen);
    //trouve la position des délimiteurs
    for i=1:size(tt_sav,1)
     if strindex(tt_sav(i),del1)<>[] then a=i, end;
     if strindex(tt_sav(i),del2)<>[] then b=i, end;
    end
  else
   printf("Incompatible rsh variable\n");
  end
 else
  printf("File %s not found\n",filen);
  new_tt=[];
 end
endfunction
```

3.13 Update used functions in xml file

- **Name:** put_xml_used_func
- **Library:** xmltotex - Xml to tex converter library

3.13.1 Calling Sequence

```
new_tt = put_xml_used_func(txt,filen)
```

3.13.2 Parameters

- **txt :** vector of strings. the text of the used function paragraph
- **filen :** string. target XML file (path+name)
- **new_tt :** vector of strings. the text of the new XML file

3.13.3 File content

```
//put_xml_used_func
//Fonction qui insère un vecteur chaîne de caractère
//dans un fichier entre les délimiteurs
//<USED_FUNCTIONS> et </USED_FUNCTIONS>
//Entrée : txt : matrice de chaînes de caractères de taille n,1
//         filen : nom du fichier xml (ex:filen=xml_path+'CAN_f.xml')
function new_tt=put_xml_used_func(txt,filen)
 if fileinfo(filen)<>[] then
  if size(txt,2)==1 then
    del1='<USED_FUNCTIONS>'
    del2='</USED_FUNCTIONS>'
    tt_sav=mgetl(filen);
    a=0;b=0;new_tt=tt_sav;
    //trouve la position des délimiteurs
    for i=1:size(tt_sav,1)
     if strindex(tt_sav(i),del1)<>[] then a=i, end;
     if strindex(tt_sav(i),del2)<>[] then b=i, end;
    end
    if a<>0&b<>0 then
      tt_used_func='  '+['<USED_FUNCTIONS>'
                         '  <SP>';'  '+txt(:,1);'  </SP>'
                        '</USED_FUNCTIONS>']

      //Ecrit la chaine de texte finale
      new_tt=[tt_sav(1:a-1);tt_used_func;tt_sav(b+1:size(tt_sav,1))]
    end
  else
   printf("Incompatible rsh variable\n");
  end
 else
  printf("File %s not found\n",filen);
  new_tt=[];
 end
endfunction
```

3.14 Convert extra strings of a xml file

- **Name:** retrieve_char
- **Library:** xmltotex - Xml to tex converter library

3.14.1 Calling Sequence

```
txt = retrieve_char(txt)
```

3.14.2 Parameters

- **txt :** vector of strings. input text
- **txt :** vector of strings. output text

3.14.3 File content

```
//retrieve_char
//fonction qui convertit les caractères particuliers
//des pages d'aides scilab xml en caractères
//normaux
//Entrée : vecteur de chaînes de caractères
//Sortie : vecteur de chaîne de caractères

function txt=retrieve_char(txt)

  txt=strsubst(txt,'&apos;',''''');
  txt=strsubst(txt,'&lt;','<');
  txt=strsubst(txt,'&gt;','>');
  txt=strsubst(txt,'&quot;','""');

endfunction
```

3.15 Return authors name and references of a xml file

- **Name:** return_xml_authors
- **Library:** xmltotex - Xml to tex converter library

3.15.1 Calling Sequence

```
txt = return_xml_authors(fname)
```

3.15.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** vector of strings. name of authors and references

3.15.3 File content

```
//return_xml_authors
//fonction qui retourne le texte placé entre
//les deux premiers drapeaux <AUTHOR_ITEM>
//et </AUTHOR_ITEM> trouvés dans le fichier fname
//ex : txt=return_xml_authors(MODNUM+'/man/xml/CAN_f.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt : tableau de chaines de caractères
function txt=return_xml_authors(fname)
txt_temp=mgetl(fname);
txt=[]
a=[]
j=1;
if txt_temp<>[] then
 for i=1:size(txt_temp,'*')
  if strindex(txt_temp(i),'<AUTHOR_ITEM>')<>[] then
   a(j,1)=i;
  end
  if strindex(txt_temp(i),'</AUTHOR_ITEM>')<>[] then
   a(j,2)=i;
   j=j+1;
  end
 end

 for i=1:size(a,'r')
  for j=a(i,1):a(i,2)
   txt(i)=txt(i)+txt_temp(j)
  end
  txt(i)=strsubst(txt(i),('<AUTHOR_ITEM>'),'')
  txt(i)=strsubst(txt(i),('</AUTHOR_ITEM>'),'')
  while part(txt(i),1)==' '
   txt(i)=part(txt(i),2:length(txt(i)));
  end
 end
end
endfunction
```

3.16 Return authors name and references of a xml file

- **Name:** return_xml_authors2
- **Library:** xmltotex - Xml to tex converter library

3.16.1 Calling Sequence

```
txt = return_xml_authors2(fname)
```

3.16.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** matrix of strings of size (n,2)
  - **txt(,1) :** name of author
  - **txt(,2) :** reference of author

3.16.3 File content

```
//return_xml_authors2
//fonction qui retourne le texte placé entre
//les deux premiers drapeaux <AUTHOR_ITEM>
//et </AUTHOR_ITEM> trouvés dans le fichier fname
//compatible help skeleton
//ex : txt=return_xml_authors2(MODNUM+'/man/xml/CAN_f.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt : tableau de chaines de caractères de taile n,2
//             txt(1,n) : nom des auteurs
//             txt(2,n) : références des auteurs
function txt=return_xml_authors2(fname)
txt_temp=mgetl(fname);
txt=[]
a=[]
j=1;
if txt_temp<>[] then
 for i=1:size(txt_temp,'*')
  if strindex(txt_temp(i),'<AUTHORS_ITEM ')<>[] then
   a(j,1)=i;
  end
  if strindex(txt_temp(i),'</AUTHORS_ITEM>')<>[] then
   a(j,2)=i;
   j=j+1;
  end
 end

 for i=1:size(a,'r')
   //pour chaque bloc
   txt(i,2)="";
   for j=a(i,1):a(i,2)
    //Trouve le nom de l'auteur
    if(strindex(txt_temp(j),'<AUTHORS_ITEM label='))<>[] then
     txt(i,1)=txt_temp(j);
     b=strindex(txt(i,1),'<AUTHORS_ITEM label=''')
     c=strindex(txt(i,1),'''>')
     txt(i,1)=part(txt(i,1),b:c-1);
     txt(i,1)=strsubst(txt(i,1),'<AUTHORS_ITEM label=''',"");
     txt(i,1)=strsubst(txt(i,1),'''>',"")
     //Enlève les blancs au début
     txt(i,1)=stripblanks_begin(txt(i,1));
     //Enlève les blancs de la fin
     txt(i,1)=stripblanks_end(txt(i,1));
    end

    txt(i,2)=txt(i,2)+txt_temp(j);
   end

   //Trouve les références de l'auteur
   txt(i,2)=strsubst(txt(i,2),'</AUTHORS_ITEM>',"");
   b=0;
   if strindex(txt(i,2),'<AUTHORS_ITEM label=')<>[] then
    b=strindex(txt(i,2),'<AUTHORS_ITEM label=')
   end
   c=0;
   if strindex(txt(i,2),'>')<>[] then
    c=strindex(txt(i,2),'>');
   end
   if b<>0&c<>0 then
    txt(i,2)=part(txt(i,2),c+1:length(txt(i,2)))
    //Enlève les blancs au début
    txt(i,2)=stripblanks_begin(txt(i,2));
    //Enlève les blancs de la fin
    txt(i,2)=stripblanks_end(txt(i,2));
   end
   if(part(txt(i,2),1))==',' then
    txt(i,2)=part(txt(i,2),2:length(txt(i,2)));
    //Enlève les blancs au début
    txt(i,2)=stripblanks_begin(txt(i,2));
   end
 end
end
endfunction
```

3.17 Return bibliography references of a xml file

- **Name:** return_xml_biblio
- **Library:** xmltotex - Xml to tex converter library

3.17.1 Calling Sequence

```
txt = return_xml_biblio(fname)
```

3.17.2 Parameters

- **fname :** string. source XML file (path+name)

- **txt :** vector of strings. the text of references

3.17.3 File content

```
//return_xml_biblio
//fonction qui retourne le texte placé entre
//les drapeaux <BIBLIO> et </BIBLIO> trouvés dans
//le fichier fname
//compatible help skeleton
//ex : txt=return_xml_biblio(MODNUM+'/man/xml/CAN_f.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt : tableau de chaines de caractères de taile n,1
//             txt(1,n) : chaine de la biblio

function txt=return_xml_biblio(fname)
 txt_temp=mgetl(fname);
 txt=[]
 j=1;
 a=[];
 if txt_temp<>[] then
   for i=1:size(txt_temp,'*')
     if strindex(txt_temp(i),'<BIBLIO>')<>[] then
       a(j,1)=i;
     end
     if strindex(txt_temp(i),'</BIBLIO>')<>[] then
       a(j,2)=i;
       j=j+1;
     end
   end

   for i=1:size(a,'r')
     for j=a(i,1):a(i,2)
       txt(i)=txt(i)+stripblanks_begin(txt_temp(j))+" "
     end
     txt(i)=strsubst(txt(i),('<BIBLIO>'),'')
     txt(i)=strsubst(txt(i),('</BIBLIO>'),'')
     txt(i)=strsubst(txt(i),('<SP>'),'')
     txt(i)=strsubst(txt(i),('</SP>'),'')
     //Enlève les blancs du début
     txt(i)=stripblanks_begin(txt(i));
     //Enlève les blancs placés à la fin
     txt(i)=stripblanks_end(txt(i));
   end

 else
   disp(fname+" not found.")
   txt=[];
 end
endfunction
```

3.18 Return calling sequence of a xml file

- **Name:** return_xml_call_seq
- **Library:** xmltotex - Xml to tex converter library

3.18.1 Calling Sequence

```
txt = return_xml_call_seq(fname)
```

3.18.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** vector of strings. the text of the calling sequence paragraph

3.18.3 File content

```
//return_xml_call_seq
//fonction qui retourne le texte placé entre
//les deux premiers drapeaux <CALLING_SEQUENCE>
//et </CALLING_SEQUENCE> trouvés dans le fichier fname
//compatilble avec help_skeleton
//ex : txt=return_xml_call_seq(SCI+'/man/eng/nonlinear/intc.xml')
//Entrée fname : chemin+nom du fichier xml
function txt=return_xml_call_seq(fname)
 txt_temp=mgetl(fname)
 txt=[]
 j=1;
 a=[];
 if txt_temp<>[] then
  for i=1:size(txt_temp,'*')
   if strindex(txt_temp(i),'<CALLING_SEQUENCE_ITEM>')<>[] then
```

```
    a(j,1)=i;
   end;
   if strindex(txt_temp(i),'</CALLING_SEQUENCE_ITEM>')<>[] then
    a(j,2)=i;
    j=j+1;
   end
  end
  if a<>[] then
   for i=1:size(a,'r')
      txt(i)="";
      //pour chaque bloc
      for j=a(i,1):a(i,2)
       txt(i)=txt(i)+txt_temp(j)
      end
   end

   txt=strsubst(txt,'<CALLING_SEQUENCE_ITEM>',"");
   txt=strsubst(txt,'</CALLING_SEQUENCE_ITEM>',"");
   txt=retrieve_char(txt);
   txt=stripblanks_begin(txt);
   txt=stripblanks_end(txt);
   if txt=="" then txt=[], end;
  end
 else
  txt=[];
 end
endfunction
```

3.19 Return long description of a xml file (obsolete)

- **Name:** return_xml_desc
- **Library:** xmltotex - Xml to tex converter library

3.19.1 Calling Sequence

```
txt = return_xml_desc(fname)
```

3.19.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** vector of strings. the text of the description paragraph.

3.19.3 File content

```
//return_xml_desc
//fonction qui retourne le texte placé entre
//les deux premiers drapeaux <DESCRIPTION>
//et </DESCRIPTION> trouvés dans le fichier fname
//ex : txt=return_xml_desc(SCI+'/man/eng/nonlinear/intc.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt : tableau de chaines de caractères
function txt=return_xml_desc(fname)
 txt_temp=mgetl(fname)
 txt=[]
 if txt_temp<>[] then
  for i=1:size(txt_temp,'*')
   if strindex(txt_temp(i),'<DESCRIPTION>')<>[] then a=i, end;
   if strindex(txt_temp(i),'</DESCRIPTION>')<>[] then b=i, end;
  end
  j=1
  for i=a:b
   if strindex(txt_temp(i),'<DESCRIPTION>')<>[] then
     txt_temp(i)=strsubst(txt_temp(i),'<DESCRIPTION>','');
   end
   if strindex(txt_temp(i),'</DESCRIPTION>')<>[] then
     txt_temp(i)=strsubst(txt_temp(i),'</DESCRIPTION>','');
   end
   if strindex(txt_temp(i),'<P>')<>[] then
     txt_temp(i)=strsubst(txt_temp(i),'<P>','');
   end
   if strindex(txt_temp(i),'</P>')<>[] then
    txt_temp(i)=strsubst(txt_temp(i),'</P>','');
   end
   txt(j)=txt_temp(i);
   j=j+1;
  end
 end
endfunction
```

3.20 Return long description of a xml file

- **Name:** return_xml_desc2
- **Library:** xmltotex - Xml to tex converter library

3.20.1 Calling Sequence

```
txt = return_xml_desc2(fname)
```

3.20.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** vector of strings. the text of the description paragraph.

3.20.3 File content

```
//return_xml_desc
//fonction qui retourne le texte placé entre
//les deux premiers drapeaux <DESCRIPTION>
//et </DESCRIPTION> trouvés dans le fichier fname
//compatilble avec help_skeleton
//ex : txt=return_xml_desc2(SCI+'/man/eng/nonlinear/intc.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt : tableau de chaines de caractères
function txt=return_xml_desc2(fname)
 txt_temp=mgetl(fname)
 txt=[]
 if txt_temp<>[] then
  for i=1:size(txt_temp,'*')
    if strindex(txt_temp(i),'<DESCRIPTION>')<>[] then a=i, end;
    if strindex(txt_temp(i),'</DESCRIPTION>')<>[] then b=i, end;
  end

  j=1
  for i=a:b
    txt_temp(i)=strsubst(txt_temp(i),'<DESCRIPTION_INDENT>','');
    txt_temp(i)=strsubst(txt_temp(i),'</DESCRIPTION_INDENT>','');
    txt_temp(i)=strsubst(txt_temp(i),'<DESCRIPTION>','');
    txt_temp(i)=strsubst(txt_temp(i),'</DESCRIPTION>','');
    txt_temp(i)=strsubst(txt_temp(i),'<DESCRIPTION_ITEM>','');
    txt_temp(i)=strsubst(txt_temp(i),'</DESCRIPTION_ITEM>','');
    txt_temp(i)=strsubst(txt_temp(i),'<P>','');
    txt_temp(i)=strsubst(txt_temp(i),'<SP>','');
    txt(j)=txt_temp(i);
    j=j+1;
  end
 end

 //Enlève les blancs du début
 txt=stripblanks_begin(txt);

 //Nettoie les lignes vides
 tt=[]
 k=1;
 for i=1:size(txt,1)
  if length(txt(i))<>0 then
 //  tt(k)=strsubst(txt(i),"</P>","");
 //  tt(k)=strsubst(txt(i),"</SP>","");
   tt(k)=txt(i);
   k=k+1;
  end
 end

 tt=strsubst(tt,"</P>","");
 tt=strsubst(tt,"</SP>","");

 //Nettoie la dernière ligne
 k=size(tt,1);
 tt(k)=stripblanks_begin(tt(k));

 if length(tt(k))==0 then tt=tt(1:k-1), end;
 txt=retrieve_char(tt);
endfunction
```

3.21 Return long description of a xml file

- **Name:** return_xml_desc3
- **Library:** xmltotex - Xml to tex converter library

3.21.1 Calling Sequence

```
new_tt = return_xml_desc3(fname)
```

3.21.2 Parameters

- **fname :** string. source XML file (path+name)
- **new_tt :** list
  - **list_txt() :** integer. id of paragraph
  - **list_txt()(1) :** integer. indentation level
  - **list_txt()(2) :** string. title of paragraph
  - **list_txt()(3) :** vector of strings. text of description paragraph

3.21.3 File content

```
//return_xml_desc3
//fonction qui retourne le texte placé entre
//les deux premiers drapeaux <DESCRIPTION>
//et </DESCRIPTION> trouvés dans le fichier fname
//compatilble avec help_skeleton
//ex : new_tt=return_xml_desc3(SCI+'/man/eng/nonlinear/intc.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie new_tt()() : une liste
//             | |
//             | |
//             | |----->  1 : profondeur d'indentation
//             |         2 : le titre du paragraphe
//             |         3 : le texte du paragraphe
//             |---> n° du paragraphe
function new_tt=return_xml_desc3(fname)
 if fileinfo(fname)<>[] then
   txt_temp=mgetl(fname)
   //txt=[]
   if txt_temp<>[] then
   nb_indent=1
   nb_para=1
   n=1
   for i=1:size(txt_temp,'*')
      if strindex(txt_temp(i),'<DESCRIPTION>')<>[] then a=i,
      elseif strindex(txt_temp(i),'</DESCRIPTION>')<>[] then b=i,
      elseif strindex(txt_temp(i),'<DESCRIPTION_INDENT>')<>[] then
       nb_indent=nb_indent+1;
       nb_para=nb_para+1
      elseif strindex(txt_temp(i),'<DESCRIPTION_ITEM')<>[] then
       nb_para=nb_para+1
      elseif strindex(txt_temp(i),'</DESCRIPTION_ITEM')<>[] then
       nb_para=nb_para+1
      elseif strindex(txt_temp(i),'</DESCRIPTION_INDENT>')<>[] then
       nb_para=nb_para+1
       nb_indent=nb_indent-1
      end
      if nb_indent>n then n=nb_indent, end;
   end

   //initialisation de la liste de sortie
   tt=list()
   for i=1:nb_para
    tt(i)=list()
    tt(i)(1)=1   //profondeur d'indentation
    tt(i)(2)=""  //le nom du paragraphe
    tt(i)(3)=""  //texte du paragraphe
   end

   //remplissage de la liste
   nb_indent=1
   nb_para=1
   for i=a:b
     if strindex(txt_temp(i),'<DESCRIPTION_INDENT>')<>[] then
      nb_indent=nb_indent+1;
      nb_para=nb_para+1
      tt(nb_para)(1)=nb_indent;
      tt(nb_para)(3)=[tt(nb_para)(3);txt_temp(i)];
     elseif strindex(txt_temp(i),'</DESCRIPTION_INDENT>')<>[] then
      tt(nb_para)(1)=nb_indent;
      tt(nb_para)(3)=[tt(nb_para)(3);txt_temp(i)];
      nb_indent=nb_indent-1
      nb_para=nb_para+1
     elseif strindex(txt_temp(i),'<DESCRIPTION_ITEM')<>[] then
       nb_para=nb_para+1
       tt(nb_para)(1)=nb_indent;
       if strindex(txt_temp(i),'<DESCRIPTION_ITEM label')<>[] then
        tt(nb_para)(2)=txt_temp(i);
       else
        tt(nb_para)(3)=[tt(nb_para)(3);txt_temp(i)];
       end
     elseif strindex(txt_temp(i),'</DESCRIPTION_ITEM')<>[] then
```

```
      tt(nb_para)(1)=nb_indent;
      tt(nb_para)(3)=[tt(nb_para)(3);txt_temp(i)];
      nb_para=nb_para+1
    else
      tt(nb_para)(1)=nb_indent;
      tt(nb_para)(3)=[tt(nb_para)(3);txt_temp(i)];
    end
  end

  //nettoyage de la liste
  for i=1:nb_para
   tt(i)(3)=strsubst(tt(i)(3),'<DESCRIPTION>',"");
   tt(i)(3)=strsubst(tt(i)(3),'</DESCRIPTION>',"");
   tt(i)(3)=strsubst(tt(i)(3),'<DESCRIPTION_INDENT>',"");
   tt(i)(3)=strsubst(tt(i)(3),'</DESCRIPTION_INDENT>',"");
   tt(i)(3)=strsubst(tt(i)(3),'<DESCRIPTION_ITEM>',"");
   tt(i)(3)=strsubst(tt(i)(3),'</DESCRIPTION_ITEM>',"");
   tt(i)(3)=strsubst(tt(i)(3),'<P>',"");
   tt(i)(3)=strsubst(tt(i)(3),'</P>',"");
   tt(i)(3)=strsubst(tt(i)(3),'<P>',"");
   tt(i)(3)=strsubst(tt(i)(3),'<SP>',"");
   tt(i)(3)=strsubst(tt(i)(3),'</SP>',"");
   tt(i)(2)=strsubst(tt(i)(2),'<DESCRIPTION_ITEM label="">',"");
   tt(i)(2)=strsubst(tt(i)(2),'"">',"");
   tt(i)(3)=retrieve_char(tt(i)(3));
   tt(i)(2)=retrieve_char(tt(i)(2));
  end

  //Enlève les blancs du début
  for i=1:nb_para
   tt(i)(3)=stripblanks_begin(tt(i)(3));
   tt(i)(2)=stripblanks_begin(tt(i)(2));
  end

  //Nettoie les lignes vides
  for i=1:nb_para
   tt_txt=[]
   k=1;
   for j=1:size(tt(i)(3),1)
     if length(tt(i)(3)(j))<>0 then
       tt_txt(k)=tt(i)(3)(j);
       k=k+1;
     end
   end
   tt(i)(3)=tt_txt;
  end

  //Enlève les paragraphe vide de la liste
  new_tt=list(list())
  k=1;
  for i=1:nb_para
   if tt(i)(3)<>[] then
    new_tt(k)=tt(i)
    k=k+1
   end
  end
  clear tt

  else
   new_tt=list();
   printf("File %s is empty\n",fname)
  end

 else
  new_tt=list();
  printf("File %s not found\n",fname)
 end
endfunction
```

3.22 Return exemples of a xml file

- **Name:** return_xml_ex
- **Library:** xmltotex - Xml to tex converter library

3.22.1 Calling Sequence

```
txt = return_xml_ex(fname)
```

3.22.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** vector of strings. the text of the example paragraph.

3.22.3 File content

```
//return_xml_ex
//fonction qui retourne les exemples
//d'aide scilab construit avec help_skeleton
//ex : return_xml_ex((SCI+'/man/fr/nonlinear/ode.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt : tableau de chaines de caractères de taille n,2
//             colonne 1 : nom du paramètre
function txt=return_xml_ex(fname)
txt_temp=mgetl(fname);
txt=[]
j=1;
a=[]
b=[]
//Cherche les bornes <EXAMPLE> et </EXAMPLE>
if txt_temp<>[] then
  for i=1:size(txt_temp,'*')
    if strindex(txt_temp(i),'<EXAMPLE>')<>[] then
      a=i;
    end;
    if strindex(txt_temp(i),'</EXAMPLE>')<>[] then
      b=i;
    end
  end

  if a<>[] & b<>[] then
    txt=txt_temp(a:b)
    txt=strsubst(txt,'<EXAMPLE>',"")
    txt=strsubst(txt,'<P>',"")
    txt=strsubst(txt,'<![CDATA[',"")
    txt=strsubst(txt,']]>',"")
    txt=strsubst(txt,'</P>',"")
    txt=strsubst(txt,'</EXAMPLE>',"")
    //Enlève les blancs du début
    txt=stripblanks_begin(txt);
    //Nettoie les lignes vides
    tt=[]
    k=1;
    emptyf=%t; //Empty flag
    for i=1:size(txt,1)
      if length(txt(i))<>0 then
        tt(k)=txt(i);
        k=k+1;
        emptyf=%f;
      end
    end
    if emptyf then txt=""
    else txt=tt;
    end
  end
else
 txt=[]
end
endfunction
```

3.23 Return pair block section of a xml file(not yet implemented)

- **Name:** return_xml_pair_blk
- **Library:** xmltotex - Xml to tex converter library

3.23.1 Calling Sequence

```
txt = return_xml_pair_blk(fname)
```

3.23.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** vector of strings. the text of the pair block paragraph.

3.23.3 File content

```
//return_xml_pair_blk
//fonction qui retourne le texte placé entre
//tous les drapeaux <PAIR_BLOCK>
//et </PAIR_BLOCK> trouvés dans le fichier fname
//ex : txt=return_pair_blk(MODNUM+'/man/xml/CNA_f.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt : tableau de chaines de caractères
function txt=return_xml_pair_blk(fname)
txt_temp=mgetl(fname);
txt=[]
a=[]
j=1;
```

```
if txt_temp<>[] then
 for i=1:size(txt_temp,'*')
  if strindex(txt_temp(i),'<PAIR_BLOCK_ITEM>')<>[] then
   a(j,1)=i;
  end
  if strindex(txt_temp(i),'</PAIR_BLOCK_ITEM>')<>[] then
   a(j,2)=i;
   j=j+1;
  end
 end

 for i=1:size(a,'r')
  for j=a(i,1):a(i,2)
   txt(i)=txt_temp(j)
  end
  txt(i)=strsubst(txt(i),('<PAIR_BLOCK_ITEM>'),'')
  txt(i)=strsubst(txt(i),('</PAIR_BLOCK_ITEM>'),'')
  txt(i)=strsubst(txt(i),('<LINK>'),'')
  txt(i)=strsubst(txt(i),('</LINK>'),'')
  txt(i)=stripblanks(txt(i));
 end
 ok=%t
 for i=1:size(txt,1)
     if stripblanks(txt(i))=='' then ok=%f, end
 end
 if ok==%f then txt=[], end
end
endfunction
```

3.24 Return parameters section of a xml file (obsolete)

- **Name:** return_xml_param
- **Library:** xmltotex - Xml to tex converter library

3.24.1 Calling Sequence

```
txt = return_xml_param(fname)
```

3.24.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** string of vectors of size(n,2)
  - **txt(,1) :** string. the title of the paremeter
  - **txt(,2) :** string. the description of the paremeter

3.24.3 File content

```
//return_xml_param
//fonction qui retourne le texte placé entre
//tous les drapeaux <ITEM label=,..> et </ITEM>
//trouvés dans le fichier fname
//ex : return_param(MODNUM+'/man/xml/CAN_f.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt : tableau de chaines de caractères
function txt=return_xml_param(fname)
txt_temp=mgetl(fname);
txt=[]
j=1;
a=[]
if txt_temp<>[] then
 for i=1:size(txt_temp,'*')
  if strindex(txt_temp(i),'<ITEM label=')<>[] then
   a(j,1)=i;
  end;
  if strindex(txt_temp(i),'</ITEM>')<>[] then
   a(j,2)=i;
   j=j+1;
  end
 end
end

for i=1:size(a,'r')
    txt(i,2)=''
    for j=a(i,1):a(i,2)
     if strindex(txt_temp(j),'<ITEM label=')<>[] then
      txt(i,1)=strsubst(txt_temp(j),'<ITEM label=''','')
      txt(i,1)=strsubst(txt(i,1),'''>','')
      if part(txt(i,1),1)==[' '] then
       txt(i,1)=part(txt(i,1),2:length(txt(i,1)))
      end
```

```
     end
     if strindex(txt_temp(j),'<ITEM label=')==[] then
       if strindex(txt_temp(j),'</ITEM>')==[] then
              txt(i,2)=txt(i,2)+txt_temp(j);
       end
     end
    end
end

for j=1:size(a,'r')
    while part(txt(j,1),1)==' '
     txt(j,1)=part(txt(j,1),2:length(txt(j,1)));
    end
    while part(txt(j,2),1)==' '
     txt(j,2)=part(txt(j,2),2:length(txt(j,2)));
    end
    txt(j,2)=strsubst(txt(j,2),'<P>','\\');
    txt(j,2)=strsubst(txt(j,2),'</P>','');
    if part(txt(j,2),1)==':' then
     txt(j,2)=part(txt(j,2),2:length(txt(j,2)));
    end
    while part(txt(j,2),1)==' '
     txt(j,2)=part(txt(j,2),2:length(txt(j,2)));
    end
end
endfunction
```

3.25 Return parameters section of a xml file

- **Name:** return_xml_param2
- **Library:** xmltotex - Xml to tex converter library

3.25.1 Calling Sequence

```
txt = return_xml_param2(fname)
```

3.25.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** string of vectors of size(n,2)
  - **txt(,1) :** string. the title of the paremeter
  - **txt(,2) :** string. the description of the paremeter

3.25.3 File content

```
//return_xml_param2
//fonction qui retourne les paramères dun fichier
//d'aide scilab construit avec help_skeleton
//ex : return_xml_param2(MODNUM+'/man/xml/CAN_f.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt : tableau de chaines de caractères de taille n,2
//             colonne 1 : nom du paramètre
//             colonne 2 : description du paramètre
function txt=return_xml_param2(fname)
txt_temp=mgetl(fname);
txt=[]
j=1;
a=[]
b=[]
c=[]

//Cherche les bornes <PARAM_ITEM> et </PARAM_ITEM>
if txt_temp<>[] then
 for i=1:size(txt_temp,'*')
  if strindex(txt_temp(i),'<PARAM_ITEM>')<>[] then
   a(j,1)=i;
  end;
  if strindex(txt_temp(i),'</PARAM_ITEM>')<>[] then
   a(j,2)=i;
   j=j+1;
  end
 end

 for i=1:size(a,'r')
     //pour chaque bloc
     for j=a(i,1):a(i,2)
        //Trouve les bornes du nom du paramètre
        if strindex(txt_temp(j),'<PARAM_NAME>')<>[] then
           b(i,1)=j;
        end
```

```
        if strindex(txt_temp(j),'</PARAM_NAME>')<>[] then
           b(i,2)=j;
        end

        //Trouve les bornes de la description du paramètre
        if strindex(txt_temp(j),'<PARAM_DESCRIPTION>')<>[] then
           c(i,1)=j;
        end
        if strindex(txt_temp(j),'</PARAM_DESCRIPTION>')<>[] then
           c(i,2)=j;
        end
     end
 end

 for i=1:size(a,'r')
   txt(i,1)="";
   if b<>[] then
     for j=b(i,1):b(i,2)
       txt(i,1)=txt(i,1)+txt_temp(j);
     end
     //Enlève les délimiteurs <PARAM_NAME> et </PARAM_NAME>
     txt(i,1)=strsubst(txt(i,1),'<PARAM_NAME>',"");
     txt(i,1)=strsubst(txt(i,1),'</PARAM_NAME>',"");
     //Enlève les blancs du début
     txt(i,1)=stripblanks_begin(txt(i,1));
     //Enlève les blancs placés à la fin
     txt(i,1)=stripblanks_end(txt(i,1));
   end

   txt(i,2)="";
   if c<>[] then
     for j=c(i,1):c(i,2)
       txt(i,2)=txt(i,2)+txt_temp(j);
     end
     //Enlève les délimiteurs <PARAM_DESCRIPTION> et </PARAM_DESCRIPTION>
     txt(i,2)=strsubst(txt(i,2),'<PARAM_DESCRIPTION>',"");
     txt(i,2)=strsubst(txt(i,2),'</PARAM_DESCRIPTION>',"");
     //Enlève les délimiteurs <SP> et </SP>
     if strindex(txt(i,2),'<SP>')<>[] then
       txt(i,2)=strsubst(txt(i,2),'<SP>',"");
     end
     if strindex(txt(i,2),'</SP>')<>[] then
       txt(i,2)=strsubst(txt(i,2),'</SP>',"");
     end
     //Enlève les blancs du début
     txt(i,2)=stripblanks_begin(txt(i,2));
     //Enlève les blancs placés à la fin
     txt(i,2)=stripblanks_end(txt(i,2));
     //Enlève les : du début
     if part(txt(i,2),1)==':' then
       txt(i,2)=part(txt(i,2),2:length(txt(i,2)));
     end
     //Enlève les blancs du début
     txt(i,2)=stripblanks_begin(txt(i,2));

     txt(i,2)=retrieve_char(txt(i,2))
   end
 end

else
 txt=[]
end
endfunction
```

3.26 Return parameters section of a xml file

- **Name:** return_xml_param3
- **Library:** xmltotex - Xml to tex converter library

3.26.1 Calling Sequence

```
txt_list = return_xml_param3(fname)
```

3.26.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt_list :** list
  - **list_txt() :** integer. indentation level
  - **list_txt()(1) :** integer. id of parameter
  - **list_txt()(2) :** matrix of string of size(n,2)

- **list_txt()(2)(,1) :** string. name of parameter
- **list_txt()(2)(,2) :** string. description of parameter

3.26.3 File content

```
//return_xml_param3
//fonction qui retourne les paramères dun fichier xml
//d'aide scilab
//ex : return_xml_param2(MODNUM+'/man/xml/CAN_f.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt_list : une liste
//                  txt_list()()
//                          | |
//                          | |------> 1 : numero paramètre
//                          |          2 : tableau de chaines de caractères de taille n,2
//                          |              colonne 1 : nom du paramètre
//                          |              colonne 2 : description du paramètre
//                          |
//                          |----> profondeur d'indentation

function txt_list=return_xml_param3(fname)
 txt_temp=mgetl(fname);
 txt_list=[]

 a=[]
 b=[]
 c=[]

 if txt_temp<>[] then
   //Cherche les bornes <PARAM> et </PARAM>
   for i=1:size(txt_temp,'*')
     if strindex(txt_temp(i),'<PARAM>')<>[] then
       a=i;
     end;
     if strindex(txt_temp(i),'</PARAM>')<>[] then
       b=i;
     end
   end

   if a<>[]&b<>[] then
     nb_indent=0;
     nb_max_indent=0;
     //Cherche le nbre de <PARAM_INDENT>
     for i=a:b
      if strindex(txt_temp(i),'<PARAM_INDENT>')<>[] then
         nb_indent=nb_indent+1;
      elseif strindex(txt_temp(i),'</PARAM_INDENT>')<>[] then
         nb_indent=nb_indent-1;
      end
      if nb_indent>nb_max_indent then nb_max_indent=nb_indent, end
     end

     //initalise la liste
     txt_list=list();
     for i=1:nb_max_indent+1
       txt_list(i)=list();
       txt_list(i)(1)=[]; //le numero param
       txt_list(i)(2)=""; //le texte
     end
     //Remplit la liste
     j=0;
     e=0;
     f=1;
     for i=a:b
       if strindex(txt_temp(i),'<PARAM_INDENT>')<>[] then
         j=j+1;
       elseif strindex(txt_temp(i),'</PARAM_INDENT>')<>[] then
         j=j-1;
       elseif strindex(txt_temp(i),'<PARAM_ITEM>')<>[] then
         e=e+1;
         txt_list(j+1)(1)=[txt_list(j+1)(1);e];
         txt_list(j+1)(2)=[txt_list(j+1)(2);txt_temp(i)];
       else
         txt_list(j+1)(2)=[txt_list(j+1)(2);txt_temp(i)];
       end
     end

     for ij=1:nb_max_indent+1
       a=[];
       b=[];
       j=1;
       txt=[];
       //Cherche les bornes <PARAM_ITEM> et </PARAM_ITEM>
       for i=1:size(txt_list(ij)(2),'*')
         if strindex(txt_list(ij)(2)(i),'<PARAM_ITEM>')<>[] then
           a(j,1)=i;
         end;
         if strindex(txt_list(ij)(2)(i),'</PARAM_ITEM>')<>[] then
           a(j,2)=i;
           j=j+1;
         end
       end

       for i=1:size(a,'r')
       //pour chaque bloc
         for j=a(i,1):a(i,2)
           //Trouve les bornes du nom du paramètre
           if strindex(txt_list(ij)(2)(j),'<PARAM_NAME>')<>[] then
```

```
          b(i,1)=j;
        end
        if strindex(txt_list(ij)(2)(j),'</PARAM_NAME>')<>[] then
          b(i,2)=j;
        end
        //Trouve les bornes de la description du paramètre
        if strindex(txt_list(ij)(2)(j),'<PARAM_DESCRIPTION>')<>[] then
          c(i,1)=j;
        end
        if strindex(txt_list(ij)(2)(j),'</PARAM_DESCRIPTION>')<>[] then
          c(i,2)=j;
        end
      end
    end

    for i=1:size(b,'r')
      txt(i,1)="";
      if b<>[] then
        for j=b(i,1):b(i,2)
          txt(i,1)=txt(i,1)+txt_list(ij)(2)(j);
        end
        //Enlève les délimiteurs <PARAM_NAME> et </PARAM_NAME>
        txt(i,1)=strsubst(txt(i,1),'<PARAM_NAME>',"");
        txt(i,1)=strsubst(txt(i,1),'</PARAM_NAME>',"");
        //Enlève les blancs du début
        txt(i,1)=stripblanks_begin(txt(i,1));
        //Enlève les blancs placés à la fin
        txt(i,1)=stripblanks_end(txt(i,1));
      end

      txt(i,2)="";
      if c<>[] then
        for j=c(i,1):c(i,2)
          txt(i,2)=txt(i,2)+txt_list(ij)(2)(j);
        end
        //Enlève les délimiteurs <PARAM_DESCRIPTION> et </PARAM_DESCRIPTION>
        txt(i,2)=strsubst(txt(i,2),'<PARAM_DESCRIPTION>',"");
        txt(i,2)=strsubst(txt(i,2),'</PARAM_DESCRIPTION>',"");
        //Enlève les délimiteurs <SP> et </SP>
        if strindex(txt(i,2),'<SP>')<>[] then
          txt(i,2)=strsubst(txt(i,2),'<SP>',"");
        end
        if strindex(txt(i,2),'</SP>')<>[] then
          txt(i,2)=strsubst(txt(i,2),'</SP>',"");
        end
        //Enlève les blancs du début
        txt(i,2)=stripblanks_begin(txt(i,2));
        //Enlève les blancs placés à la fin
        txt(i,2)=stripblanks_end(txt(i,2));
        //Enlève les : du début
        if part(txt(i,2),1)==':' then
          txt(i,2)=part(txt(i,2),2:length(txt(i,2)));
        end
        //Enlève les blancs du début
        txt(i,2)=stripblanks_begin(txt(i,2));
        txt(i,2)=retrieve_char(txt(i,2))
      end
    end
    txt_list(ij)(2)=txt
  end

  //Enlève la première liste si vide
  if txt_list(1)(1)==[] then
    nb_indent=size(txt_list);
    if nb_indent<>1 then
      new_list=list();
      for i=1:nb_indent-1
        new_list(i)=list()
        new_list(i)=txt_list(i+1);
      end
      txt_list=new_list;
    else
      txt_list=[]
    end
  end
 end
endfunction
```

3.27 Return short description of a xml file

- **Name:** return_xml_sdesc
- **Library:** xmltotex - Xml to tex converter library

3.27.1 Calling Sequence

```
txt = return_xml_sdesc(fname)
```

3.27.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** string. short description

3.27.3 File content

```
//return_xml_sdesc
//fonction qui retourne le texte placé entre
//les deux premiers drapeaux <SHORT_DESCRIPTION>
//et </SHORT_DESCRIPTION> trouvés dans le fichier fname
//ex : txt=return_xml_sdesc(SCI+'/man/eng/nonlinear/intc.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt : tableau de chaines de caractères
function txt=return_xml_sdesc(fname)
if fileinfo(fname)<>[] then
 txt_temp=mgetl(fname);
 txt=[]
 j=1;
 a=[];
 if txt_temp<>[] then
  for i=1:size(txt_temp,'*')
   if strindex(txt_temp(i),'<SHORT_DESCRIPTION')<>[] then
    a(j,1)=i;
   end
   if strindex(txt_temp(i),'</SHORT_DESCRIPTION>')<>[] then
    a(j,2)=i;
    j=j+1;
   end
  end

  for i=1:size(a,'r')
   for j=a(i,1):a(i,2)
    txt(i)=txt(i)+txt_temp(j)
   end
   txt(i)=part(txt(i),strindex(txt(i),''">')+2:length(txt(i)));
   txt(i)=strsubst(txt(i),('</SHORT_DESCRIPTION>'),'')
//    while part(txt(i),1)==' '
//     txt(i)=part(txt(i),2:length(txt(i)));
//    end
   txt(i)=stripblanks_begin(txt(i))
  end
 end

else
 printf("Warning : %s not found.\n",fname)
 txt=[];
end
txt=retrieve_char(txt)
endfunction
```

3.28 Return see also section of a xml file

- **Name:** return_xml_see_also
- **Library:** xmltotex - Xml to tex converter library

3.28.1 Calling Sequence

```
txt = return_xml_see_also(fname)
```

3.28.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** vector of strings. see_also item

3.28.3 File content

```
//return_xml_see_also
//fonction qui retourne le texte placé entre
//tous les drapeaux <SEE_ALSO_ITEM>
//et </SEE_ALSO_ITEM> trouvés dans le fichier fname
//ex : txt=return_xml_see_also(MODNUM+'/man/xml/CNA_f.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt : tableau de chaines de caractères
function txt=return_xml_see_also(fname)
txt=[]
if fileinfo(fname)<>[] then
  txt_temp=mgetl(fname);
```

```
  a=[]
  j=1;
  if txt_temp<>[] then
    for i=1:size(txt_temp,'*')
      if strindex(txt_temp(i),'<SEE_ALSO_ITEM>')<>[] then
        a(j,1)=i;
      end
      if strindex(txt_temp(i),'</SEE_ALSO_ITEM>')<>[] then
       a(j,2)=i;
       j=j+1;
      end
    end

    for i=1:size(a,'r')
      for j=a(i,1):a(i,2)
        txt(i)=txt_temp(j)
      end
      txt(i)=strsubst(txt(i),('<SEE_ALSO_ITEM>'),'')
      txt(i)=strsubst(txt(i),('</SEE_ALSO_ITEM>'),'')
      txt(i)=strsubst(txt(i),('<LINK>'),'')
      txt(i)=strsubst(txt(i),('</LINK>'),'')
      txt(i)=stripblanks(txt(i));
    end
  end
else
  printf("File %s not found\n",fname);
end
endfunction
```

3.29 Return type of a xml file

- **Name:** return_xml_type
- **Library:** xmltotex - Xml to tex converter library

3.29.1 Calling Sequence

```
txt = return_xml_type(fname)
```

3.29.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** string. the type of the scilab xml man page

3.29.3 File content

```
//return_xml_type
//fonction qui retourne le texte placé entre
//les drapeaux <TYPE>
//et </TYPE> trouvés dans le fichier fname.xml
//ex : txt=return_xml_type(SCI+'/man/eng/nonlinear/intc.xml')
//Entrée fname : chemin+nom du fichier xml
//Sortie txt : tableau de chaines de caractères
function txt=return_xml_type(fname)
if fileinfo(fname)<>[] then
 txt_temp=mgetl(fname);
 txt=[]
 j=1;
 a=[];
 if txt_temp<>[] then
  for i=1:size(txt_temp,'*')
   if strindex(txt_temp(i),'<TYPE>')<>[] then
    a(j,1)=i;
   end
   if strindex(txt_temp(i),'</TYPE>')<>[] then
    a(j,2)=i;
    j=j+1;
   end
  end

  for i=1:size(a,'r')
   for j=a(i,1):a(i,2)
    txt(i)=txt(i)+txt_temp(j)
   end
  end
 end

else
 printf("Warning : %s not found.\n",fname)
 txt=[];
end
txt=strsubst(txt,'<TYPE>',"");
txt=strsubst(txt,'</TYPE>',"");
txt=stripblanks_begin(txt);
txt=stripblanks_end(txt);
```

```
txt=retrieve_char(txt)
endfunction
```

3.30 Return used function paragraph of a xml file

- **Name:** return_xml_used_func
- **Library:** xmltotex - Xml to tex converter library

3.30.1 Calling Sequence

```
txt = return_xml_used_func(fname)
```

3.30.2 Parameters

- **fname :** string. source XML file (path+name)
- **txt :** vector of strings. the text of the used function paragraph

3.30.3 File content

```
//return_xml_used_func
//fonction qui retourne le texte placé entre
//les deux premiers drapeaux <USED_FUNCTIONS>
//et </USED_FUNCTIONS> trouvés dans le fichier fname
//compatilble avec help_skeleton
//ex : txt=return_xml_used_func(SCI+'/man/eng/nonlinear/intc.xml')
//Entrée fname : chemin+nom du fichier xml
function txt=return_xml_used_func(fname)
 txt_temp=mgetl(fname)
 txt=[]
 a=[]
 b=[]
 if txt_temp<>[] then
  for i=1:size(txt_temp,'*')
    if strindex(txt_temp(i),'<USED_FUNCTIONS>')<>[] then a=i, end;
    if strindex(txt_temp(i),'</USED_FUNCTIONS>')<>[] then b=i, end;
  end

  if a<>[] & b<>[] then
    txt=txt_temp(a:b)
    txt=strsubst(txt,'<USED_FUNCTIONS>',"")
    txt=strsubst(txt,'</USED_FUNCTIONS>',"")
    txt=strsubst(txt,'<SP>',"")
    txt=strsubst(txt,'</SP>',"")
    //Enlève les blancs du début
    txt=stripblanks_begin(txt);
    //Nettoie les lignes vides
    tt=[]
    k=1;
    emptyf=%t; //Empty flag
    for i=1:size(txt,1)
      if length(txt(i))<>0 then
        tt(k)=txt(i);
        k=k+1;
        emptyf=%f;
      end
    end
    if emptyf then txt=""
    else txt=tt;
    end
  end
 else
  txt=[]
 end
endfunction
```

3.31 Strip blanks at the beginning of a character string

- **Name:** stripblanks_begin
- **Library:** xmltotex - Xml to tex converter library

3.31.1 Calling Sequence

```
txt = stripblanks_begin(txt)
```

3.31.2 Parameters

- **txt :** vector of strings. the input text.
- **txt :** vector of strings. the output text.

3.31.3 File content

```
//stripblanks_begin
//fonction qui enlève le caractère " "
//du début de chaque élément du vecteur
//chaîne de caractères présenté en entrée.
//Entrée : txt un vecteur de chaîne de caractères
//Sortie : txt un vecteur de chaîne de caractères 'nettoyé'
function txt=stripblanks_begin(txt)
 if txt<>[] then
   for i=1:size(txt,1)
      while part(txt(i),1)==' '|part(txt(i),1)==code2str(-40)
        if length(txt(i))==0 then
          break
        else
          txt(i)=part(txt(i),2:length(txt(i)));
        end
      end
   end
 end
endfunction
```

3.32 Strip blanks at the end of a character string

- **Name:** stripblanks_end
- **Library:** xmltotex - Xml to tex converter library

3.32.1 Calling Sequence

```
txt = stripblanks_end(txt)
```

3.32.2 Parameters

- **txt :** vector of strings. the input text.
- **txt :** vector of strings. the output text.

3.32.3 File content

```
//stripblanks_end
//fonction qui enlève le caractère " "
//à la fin de chaque élément du vecteur
//chaîne de caractère présenté en entrée.
//Entrée : txt un vecteur de chaîne de caractères
//Sortie : txt un vecteur de chaîne de caractères 'nettoyé'
function txt=stripblanks_end(txt)
 for i=1:size(txt,1)
  if length(txt(i))<>0 then
   while part(txt(i),length(txt(i)))==' '|part(txt(i),length(txt(i)))==code2str(-40)
     if length(txt(i))==0 then
        break
      else
       txt(i)=part(txt(i),1:length(txt(i))-1);
     end
   end
  end
 end
endfunction
```

3.33 Xml file to tex file convertor

- **Name:** xml2tex
- **Library:** xmltotex - Xml to tex converter library

3.33.1 Calling Sequence

`xml2tex(namef,flag,typdoc)`

3.33.2 Parameters

- **namef :** string. name of the XML source file to be converted
- **flag :** string. set the type of man page
  - **'block' :** for interfacing function of scicos block
  - **'pal' :** for a palette (.cosf file)
  - **'diagr' :** for a scicos diagram (.cos file)
  - **'scilib' :** for a library of scilab macros
  - **'sci' :** for a scilab macro.
  - **'rout' :** for computational routine
  - **'sim' :** for scilab simulation script (_sim.sce file)
  - **'sce' :** for scilab script (.sce file)
- **typdoc :** string. a flag to set the type of the documentation to produce
  - **'html' :** for html format
  - **'guide' :** for paper format

3.33.3 File content

```
//xml2tex
//fonction qui convertit un ficher d'aide scilab xml qui
//se trouve dans man/xml (xml_path) en un ensemble de fichier latex :
//fichier namef_call_seq.tex :
//fichier namef_long.tex : fichier tex correspondant à la description longue
//fichier namef_param.tex : fichier tex correspondant aux paramètres
//fichier namef_ex.tex
//fichier namef_see_also.tex : fichier tex correspondant aux pages "voir aussi"
//fichier namef_authors.tex : fichier correspondant aux auteurs du fichier xml
//fichier _bib.tex :
//fichier _used_func.tex :
//
//Entrée namef : nom du fichier xml à convertir sans extension
//               (ex:'CAN_f')
//       flag : 'block' pour une fonction d'interface scicos
//              'pal' pour un fichier palette scicos (cosf)
//              'diagr' pour un diagramme de simulation scicos
//              'scilib' pour une librairie de fonctions scilab
//              'sci' pour une fonction scilab
//              'rout' pour une routine bas-niveau
//              'sim' pour un script de simulation scilab
//     typdoc : 'html' (default)
//              'guide' pour du papier
//Sortie néant
function xml2tex(namef,flag,typdoc)

 if rsh<3 then
   typdoc='html'
 end

 //choix de l'extension du répertoire
 select flag
  case 'block'
   ext=''
  case 'pal'
   ext='_cosf'
  case 'diagr'
   ext='_cos'
  case 'scilib'
   ext='_scilib'
  case 'sci'
   ext='_sci'
  case 'rout'
   ext='_rout'
  case 'sim'
   ext=''
  case 'sce'
   ext='_sce'
 end

 //Pour chaque nom
 for ij=1:size(namef,1)
```

```
name=namef(ij)+ext
//test présence fichier source .xml
if fileinfo(xml_path+lang+'/'+namef(ij)+'.xml')<>[] then
  //crée un repertoire si non existant
  if fileinfo(name+'/')==[] then unix_g("mkdir "+name+"/"), end;
  printf("Generate %s.tex from %s.xml... ",namef(ij),namef(ij));

  //Ecrit fichier _call_seq.tex
  txt=return_xml_call_seq(xml_path+lang+'/'+namef(ij)+'.xml');
  if txt<>[] then
    tt=['\begin{verbatim}';txt];
    tt=[tt;'\end{verbatim}'];
    mputl(tt,'./'+name+'/'+namef(ij)+'_call_seq.tex');
  end

  //Ecrit fichier _param.tex
  //txt=return_xml_param2(xml_path+namef(ij)+'.xml');
  txt_list=return_xml_param3(xml_path+lang+'/'+namef(ij)+'.xml');
  if txt_list<>[] then
     //trouve la profondeur d'indentation max.
    nb_indent=size(txt_list);
    //trouve le nombre de paramètres
    nb_param=0;
    for i=1:size(txt_list)
      nb_param=nb_param+size(txt_list(i)(2),1);
    end
    //pause
    //initialise une nouvelle liste
    n=list();
    for i=1:nb_param
      n(i)=list();
      n(i)(1)=0;
      n(i)(2)="";
      n(i)(3)="";
    end
    //met la liste dans l'ordre des paragraphes
    for i=1:size(txt_list)
       for j=1:size(txt_list(i)(1),1)
           n(txt_list(i)(1)(j,1))(1)=i;
           n(txt_list(i)(1)(j,1))(2)=txt_list(i)(2)(j,1);
           n(txt_list(i)(1)(j,1))(3)=txt_list(i)(2)(j,2);
        end
    end
    //crée la liste des param au format tex
    tt_param=[];
    pre_i=0; //indentation précédente
    for i=1:size(n)
       dif_i=n(i)(1)-pre_i;
       if dif_i<>0 then
         if dif_i>0 then
           for j=1:dif_i
             tt_param=[tt_param;'  \begin{itemize}'];
           end
         elseif dif_i<0 then
           for j=-1:-1:dif_i
             tt_param=[tt_param;'  \end{itemize}'];
           end
         end
       end
       pre_i=n(i)(1);
       titlep=retrieve_char(n(i)(2));
       descp=retrieve_char(n(i)(3));
       if flag=='block' then
          tt_param=[tt_param;
                    '  \item{\textbf{'+latexsubst(titlep)+' :}}\linebreak';
                    latexsubst(descp);];
       else
          tt_param=[tt_param;
                    '  \item{\textbf{'+latexsubst(titlep)+' :}} '+latexsubst(descp);];
      end
    end
    dif_i=-pre_i;
    if dif_i<0 then
      for j=-1:-1:dif_i
        tt_param=[tt_param;'  \end{itemize}'];
      end
    end
    if typdoc=='guide' then
     tt_param=strsubst(tt_param,'\linebreak','\\');
    end
    mputl(tt_param,'./'+name+'/'+namef(ij)+'_param.tex');
  end

  //Ecrit fichier _long.tex
  list_txt=return_xml_desc3(xml_path+lang+'/'+namef(ij)+'.xml');
  if list_txt<>list(list()) then
   n=1; //profondeur d'indentation
   tt=[];
   for i=1:size(list_txt)
     if list_txt(i)(1)>n then
       for j=1:list_txt(i)(1)-n
        tt=[tt;'\begin{quotation}'}
       end
       n=list_txt(i)(1);
     end

     if i>1 & list_txt(i)(1)<n then
       for j=1:n-list_txt(i)(1)
        tt=[tt;'\end{quotation}'}
       end
       n=list_txt(i)(1);
```

```
        end

        if list_txt(i)(2)<>"" then
         tt=[tt;'\textbf{'+latexsubst(list_txt(i)(2))+' }'+...
             latexsubst(list_txt(i)(3));'']
        else
         tt=[tt;latexsubst(list_txt(i)(3));'']
        end
       end
       if n>1 then
          for j=1:(n-1)
            tt=[tt;'\end{quotation}']
          end
       end
       mputl(tt,'./'+name+'/'+namef(ij)+'_long.tex');
      end

      //Ecrit fichier _ex.tex
      txt=return_xml_ex(xml_path+lang+'/'+namef(ij)+'.xml');
      if txt<>[]&txt<>"" then
        if flag=='sci'|flag=='rout' then
          tt=['\begin{verbatim}';txt];
          tt=[tt;'\end{verbatim}'];
        elseif flag=='block' then
          tt=latexsubst(txt);
        end
        mputl(tt,'./'+name+'/'+namef(ij)+'_ex.tex');
      end

      //Ecrit fichier _see_also.tex
      txt=return_xml_see_also(xml_path+lang+'/'+namef(ij)+'.xml');
      if txt<>[]&txt<>"" then
        emptyf=%t;
        for i=1:size(txt,'r')
          if txt(i,1)<>"" then
            emptyf=%f
            break
          end
        end
        if ~emptyf then
          tt=['\begin{itemize}'];
          for i=1:size(txt,'r')
            txt1=latexsubst(txt(i,1));
            txt2=[];
            txt2=return_xml_sdesc(xml_path+lang+'/'+txt(i,1)+'.xml');
            txt3=return_xml_type(xml_path+lang+'/'+txt(i,1)+'.xml');
            txt3=change_lang_title(lang,txt3);
            txt2=latexsubst(txt2)
            if typdoc=='html' then
              tt=[tt;'\item{\htmladdnormallink{'+latexsubst(txt(i,1))+...
                  ' - '+txt2+' ('+txt3+')}{'+txt(i,1)+'.htm}}'];
            else
              tt=[tt;'\item{'+txt1+' - '+txt2+' ('+txt3+')}'];  //on pout rajouter un \ref{} ici!
            end
          end
          tt=[tt;'\end{itemize}'];
          mputl(tt,'./'+name+'/'+namef(ij)+'_see_also.tex');
        end
      end

      //Ecrit fichier _bib.tex
      txt=return_xml_biblio(xml_path+lang+'/'+namef(ij)+'.xml');
      if txt<>[]&txt<>"" then
        txt=latexsubst(txt)
        tt=['\begin{thebibliography}{}';txt;'\end{thebibliography}']
        mputl(tt,'./'+name+'/'+namef(ij)+'_bib.tex');
      end

      //Ecrit fichier _authors.tex
      txt=return_xml_authors2(xml_path+lang+'/'+namef(ij)+'.xml')
      if txt<>[]&txt<>"" then
       txt=latexsubst(txt)
       mputl(txt,'./'+name+'/'+namef(ij)+'_authors.tex');
      end

      //Ecrit fichier _used_func.tex
      txt=return_xml_used_func(xml_path+lang+'/'+namef(ij)+'.xml')
      if txt<>[]&txt<>"" then
       txt=latexsubst(txt)
       mputl(txt,'./'+name+'/'+namef(ij)+'_used_func.tex');
      end

      printf("Done\n");
    end
  end
endfunction
```

Part III

Low level routines

Chapter 1

library of low-level computational routines

1.1 chargepump_c

- **Short description:** charge pump computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.1.1 Parameters

- **n :** size of vectors
- **up :** address of up vectors.
- **down :** address of down vectors.
- **Io :** address of Io vectors.
- **typ_leak :** type of leakage current
  - **0 :** no leak
  - **1 :** constant leak
  - **2 :** noisy leak
- **Ileak_m :** address of the mean of the current vector or constant current vector
- **Ileak_d :** address of the variance of the current vector
- **Icp :** address of the output vector

1.1.2 File content

```
/* chargepump_c subroutine
 * ideal and noisy charge pump computation
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* chargepump_c routine de calcul d'une pompe de charge
 *
 * Entrées :
 * n    : taille des vecteurs
 * up   : adresse de départ des vecteurs up
 * down : adresse de départ des vecteurs down
 * Io   : adresse de départ des vecteurs Io
 * typ_leak : type de courant fuite
 *               0: pas de fuite
 *               1: fuite constante
 *               2: fuite bruit normal
 * Ileak_m : adresse de départ des vecteurs du courants de fuite moyen
 * Ileak_d : adresse de départ des vecteurs de la déviation du courant de fuite
 *
```

```
 * Sorties :
 * Icp : adresse de départ du vecteur de sortie
 */

void chargepump_c(int *n,double *up,double *down,double *Io, int *typ_leak,double *Ileak_m,double *Ileak_d,double *Icp)
{
 /*déclaration*/
 int i;

 /*test sur type de courant de fuite*/
 switch (*typ_leak)
 {
  case 0 : /*cas pas de fuite*/
  {
   for(i=0;i<(*n);i++) Icp[i]=Io[i]*(up[i]-down[i]);
   break;
  }
  case 1 : /*cas fuite constante*/
  {
   for(i=0;i<(*n);i++) Icp[i]=Io[i]*(up[i]-down[i])+Ileak_m[i];
   break;
  }
  case 2 :
  {
   /*Appel noiseblk_c*/
   noiseblk_c(n,(i=1,&i),&Ileak_d[0],&Ileak_m[0],&Icp[0]);
   for(i=0;i<(*n);i++) Icp[i]=Io[i]*(up[i]-down[i])+Icp[i];
   break;
  }
  /*cas other*/
  for(i=0;i<(*n);i++) Icp[i]=Io[i]*(up[i]-down[i]);
  break;
 }
 return;
}
```

1.2 cmplxa_c

- **Short description:** complex addition computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.2.1 Parameters

- **n :** size of vectors
- **sig :** addition or substraction (1 or -1)
- **[z1_r;z1_i] :** address of complex vector 1
- **[z2_r;z2_i] :** address of complex vector 2
- **[y_r;y_i] :** address of resulting complex vector

1.2.2 File content

```
/* cmplxa_c subroutine
 * complex addition computation
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* cmplxa_c routine de calcul d'addition-soustraction de vecteurs complexes
 *
 * n           : la taille des vecteurs
 * sig         : signe de l'opération
 * [z1_r;z1_i] : adresses de départ du vecteur complexe 1
 * [z2_r;z2_i] : adresses de départ du vecteur complexe 2
 * [y_r;y_i]   : adresses de départ du vecteur complexe résultat
 *
 * rmq : doit exister en version BLAS(!?)
 */

void cmplxa_c(int *n,int *sig,double *z1_r,double *z1_i,double *z2_r,double *z2_i,double *y_r,double *y_i)
{
 /*déclaration des variables*/
 int i;

 /*réalise multiplication complex*/
 for(i=0;i<(*n);i++)
```

```
 {
  y_r[i]=z1_r[i]+(*sig)*z2_r[i];
  y_i[i]=z1_i[i]+(*sig)*z2_i[i];
 }

 return;
}
```

1.3 cmplxm_c

- **Short description:** complex multiplication computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.3.1 Parameters

- **n :** size of vectors
- **[z1_r;z1_i] :** address of complex vector 1
- **[z2_r;z2_i] :** address of complex vector 2
- **[y_r;y_i] :** address of resulting complex vector

1.3.2 File content

```
/* cmplxm_c subroutine
 * complex multiplication computation
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/*
 * cmplxm_c routine de calcul de multiplication de vecteurs complexes
 *
 * n           : la taille des vecteurs
 * [z1_r;z1_i] : adresses de départ du vecteur complexe 1
 * [z2_r;z2_i] : adresses de départ du vecteur complexe 2
 * [y_r;y_i]   : adresses de départ du vecteur complexe résultat
 *
 * rmq : doit exister en version BLAS(!?)
 */

void cmplxm_c(int *n,double *z1_r,double *z1_i,double *z2_r,double *z2_i,double *y_r,double *y_i)
{
 /*déclaration des variables*/
 int i,l,n1,m;

 /*réalise multiplication complex*/
 /*F2C(wmmul)(&z1_r[0],&z1_i[0],(n1=1,&n1),&z2_r[0],&z2_i[0],(m=1,&m),&y_r[0],&y_i[0],(m=1,&m),n,(m=1,&m),n);*/
 for(i=0;i<(*n);i++)
 {
  y_r[i]=z1_r[i]*z2_r[i]-z1_i[i]*z2_i[i];
  y_i[i]=z1_r[i]*z2_i[i]+z1_i[i]*z2_r[i];
 }

 return;
}
```

1.4 codinser_c

- **Short description:** sequence by symbol multiplication computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.4.1 Parameters

- **nuc :** size of sequence vector
- **nus :** size of symbol vector

- **c :** address of sequence vector
- **s :** address of symbol vector
- **y :** address of output vector

1.4.2 File content

```
/* codinser_c subroutine
 * symbol by chip multlication
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* codinser_c routine de multiplication code par symbole
 * nuc : taille du vecteur code
 * nus : taille du vecteur symbole
 * c : adresse de départ du vecteur code
 * s : adresse de départ du vecteur symbole
 * y : adresse de départ du vecteur résultat
 */

void codinser_c(int *nuc,int *nus,double *c,double *s,double *y)
{
 /*Déclaration des variables compteur*/
 int i,j;

 /*Réalise multiplication code par symbole*/
 for(i=0;i<(*nus);i++)
  {
   for(j=0;j<(*nuc);j++)
   {
    y[i*(*nuc)+j]=s[i]*c[j];
   }
  }
}
```

1.5 comp_c

- **Short description:** sign comparison computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.5.1 Parameters

- **n :** size of vectors
- **amp :** output level (scalar)
- **u :** input vector
- **y :** output vector

1.5.2 File content

```
/* comp_c subroutine
 * sign comparison computation
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* comp_c routine de calcul de comparaison de signe
 * Entrées :
 * n    : taille des vecteurs
 * amp  : amplitude des sorties (scalaire)
 * u    : vecteur d'entrée
 * Sorties :
 * y   : vecteur de sortie
 */

void comp_c(int *n,double *ampl,double *u,double *y)
{
```

```
 int i;

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

1.6 convolfft_c

- **Short description:** signal by impulse response multiplication with fft method computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.6.1 Parameters

- **m1 :** size of vectors
- **z1_r :** address of real parts of the first time domain input vector
- **z1_i :** address of imaginary parts of the first time domain input vector
- **z2_r :** address of real parts of the second time domain input vector
- **z2_i :** address of imaginary parts of the second time domain input vector
- **y_r :** address of real parts of the time domain output vector
- **y_i :** address of imaginray parts of the time domain output vector

1.6.2 File content

```
/* convolfft_c subroutine
 * FFT convolution computation
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* convolfft_c routine de calcul de filtre à réponse impulsionnelle finie
 * par la méthode de convolution par fft
 *
 * m1   : taille des vecteurs
 * z1_r : adresse de départ de la partie réelle du vecteur temporel 1 d'entrée
 * z1_i : adresse de départ de la partie imaginaire du vecteur temporel 1 d'entrée
 * z2_r : adresse de départ de la partie réelle du vecteur temporel 2 d'entrée
 * z2_i : adresse de départ de la partie imaginaire du vecteur temporel 2 d'entrée
 * y_r  : adresse de départ de la partie réelle du vecteur temporel résultat
 * y_i  : adresse de départ de la partie imaginaire du vecteur temporel résultat
 *
 * utilise : fft842 (signal)
 *           complx_c
 *
 * rmq m1 doit-être de taille en puissance de 2
 */

void convolfft_c(int *m1,double *z1_r,double *z1_i,double *z2_r,double *z2_i,double *y_r,double *y_i)
{
 /*déclaration*/
 int k,i,ierr;

 /*appel fft*/
 F2C(fft842)((k=0,&k),m1,&z1_r[0],&z1_i[0],&ierr);

 /*appel fft*/
 F2C(fft842)((k=0,&k),m1,&z2_r[0],&z2_i[0],&ierr);

 /*Réalise la multiplication vectorielle complexe*/
 cmplxm_c(m1,&z1_r[0],&z1_i[0],&z2_r[0],&z2_i[0],&y_r[0],&y_i[0]);

 /*appel fft-1*/
 F2C(fft842)((k=1,&k),m1,&y_r[0],&y_i[0],&ierr);

 return;
}
```

1.7 convolr_c

- **Short description:** finite impulse response filter by fft and overlap-add method computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.7.1 Parameters

- **n :** size of the original vector
- **nb_coef :** length of the impulse response vector
- **m1 :** size of the fft vector ($2\hat{x}$)
- **[z1_r;z1_i] :** address of the input complex vector 1 (signal)
- **[z2_r;z2_i] :** address of the input complex vector 2 (impulse reponse)
- **[y_r;y_i] :** address of the output complex vector
- **z_r :** address of the overlaped word

1.7.2 File content

```
/* convolr_c subroutine
 * FIR computation
 * with FFT convolution
 * and overlap-ad method
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* convolr_c routine qui  realise un filtre FIR
 * par multiplication dans le domaine frequentiel
 * avec mise mémoire .
 *
 * n           : taille du mot original
 * nb_coef     : longueur de la réponse impulsionnelle
 * m1          : taille de la fft
 * [z1_r;z1_i] : adresses de départ du vecteur complexe 1
 * [z2_r;z2_i] : adresses de départ du vecteur complexe 2
 * [y_r;y_i]   : adresses de départ du vecteur complexe résultat
 * z_r         : adresse de départ du mot mémoire
 *
 * utilise : dset (BLAS)
 *           convolfft_c
 *           overlapadr_c
 */

void convolr_c(int *n,int *nb_coef,int *m1,double *z1_r,double *z1_i,double *z2_r,double *z2_i,double *y_r,double *y_i,double *z_r)
{
 /*déclaration*/
 int i,l,k;

 /*ajoute les zéros au vecteur z1_r*/
 F2C(dset)((k=(*m1)-(*n),&k),(l=0,&l),&z1_r[(*n)],(i=1,&i));

 /*place valeur img z1_i a zero*/
 F2C(dset)((k=(*m1),&k),(l=0,&l),&z1_i[0],(i=1,&i));

 /*ajoute les zéros au vecteur z2_r*/
 F2C(dset)((k=(*m1)-(*nb_coef),&k),(l=0,&l),&z2_r[(*nb_coef)],(i=1,&i));

 /*place valeur img z2_i a zero*/
 F2C(dset)((k=(*m1),&k),(l=0,&l),&z2_i[0],(i=1,&i));

 /*Appel convolfft_c*/
 convolfft_c(m1,&z1_r[0],&z1_i[0],&z2_r[0],&z2_i[0],&y_r[0],&y_i[0]);

 /*Appel overlapad_c*/
 overlapadr_c(m1,n,nb_coef,&y_r[0],&z_r[0]);

 return;
}
```

1.8 cpf_c

- **Short description:** Ideal tri-state phase/frequency comparator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.8.1 Parameters

- **n :** size of vectors
- **nev :** flip-flop clock inputs
  - **1 :** flip-flop 1
  - **2 :** flip-flop 2
  - **3 :** flip-flop 1&2 (rmq : not programmed)
  - **4 :** RAZ input of flip-flops
- **y1 :** output state of flip-flop 1
- **y2 :** output state of flip-flop 2
- **z1 :** preceding output state of flip-flop 1
- **z2 :** preceding output state of flip-flop 2

1.8.2 File content

```
/* cpf_c subroutine
 * Ideal discrete D Flip-Flop
 * Phase/Frequency Detector
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* cpf_c routine de calcul d'un comparateur phase fréquence trois états idéal
 *
 * Entrées :
 * n   : taille des vecteurs
 * nev : entrée d'activation des horloges des bascules
 *      1: bascule 1
 *      2: bascule 2
 *      3: bascule 1&2 (rmq : non programmé)
 *      4: entrée RAZ des deux bascules
 *
 * Sorties :
 * y1 : etat de sortie bascule 1
 * y2 : etat de sortie bascule 2
 * Entrées/sorties :
 * z1 : état de sortie précédent bascule 1
 * z2 : etat de sortie précédent bascule 2
 */

void cpf_c(int *n,int *nev,double *z1,double *z2,double *y1,double *y2)
{
 /*déclaration*/
 int i;

 for (i=0;i<(*n);i++)
 {
  switch (nev[i])
   {
    case 1 :
    {
     if ((z1[i]==0) && (z2[i]==0)) {y1[i] = 1; y2[i] = 0;}
     else if (z1[i] == 1) {y1[i] = 1; y2[i] = 0;}
     else if (z2[i] == 1) {y1[i] = 0; y2[i] = 0;}
     else {y1[i] = 1; y2[i] = 0;}
     break;
    }

    case 2 :
    {
     if ((z1[i] == 0) && (z2[i] == 0)) {y1[i] = 0; y2[i] = 1;}
     else if (z1[i] == 1) {y1[i] = 0; y2[i] = 0;}
     else if (z2[i] == 1) {y1[i] = 0; y2[i] = 1;}
     else {y1[i] = 0; y2[i] = 1;}
     break;
```

```
    }

    case 4 :
    {
     y1[i]=0;
     y2[i]=0;
     break;
    }

    y1[i] = 0 ;
    y2[i] = 0;
    break;
   }
  /*Met en mémoire les états*/
  z1[i]=y1[i];
  z2[i]=y2[i];
 }
 return;
}
```

1.9 cpft_c

- **Short description:** Ideal tri-state phase/frequency comparator with delay computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.9.1 Parameters

- **n :** size of vectors
- **nev :** flip-flop clock inputs
  - **1 :** flip-flop 1
  - **2 :** flip-flop 2
  - **3 :** flip-flop 1&2 (rmq : not programmed)
  - **4 :** RAZ input of flip-flops
- **flag :** a flag to set the sate or the date computation
  - **1 :** compute the states of the comparator
  - **3 :** compute the date of the Return At Zero of the comparator
- **delay :** delay of the Return At Zero
- **y1 :** output state of flip-flop 1
- **y2 :** output state of flip-flop 2
- **evout :** date of the Return At Zero
- **z1 :** preceding output state of flip-flop 1
- **z2 :** preceding output state of flip-flop 2
- **sh :** state of comparator
  - **0 :** enable
  - **1 :** disable

1.9.2 File content

```
/* cpft_c subroutine
 * Ideal discrete D Flip-Flop
 * Phase/Frequency Detector
 * with delay on RAZ state
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* cpft_c routine de calcul d'un comparateur phase fréquence trois états idéal
 * Entrées :
 * n    : taille des vecteurs
 * nev  : entrée d'activation des horloges des bascules
 *       1: bascule 1
 *       2: bascule 2
 *       3: bascule 1&2 (rmq : non programmé)
 *       4: entrée RAZ des deux bascules
 * flag  : Tache à effectuer
 *       1: calcul des etats de sorties
 *       3: calcul de la date de remise à zéro
 * delay : retard de la remise à zéro
 * Sorties :
 * y1    : etat de sortie bascule 1
 * y2    : etat de sortie bascule 2
 * evout : date de la remise à zéro
 * Entrées/sorties :
 * z1 : état de sortie précédent bascule 1
 * z2 : etat de sortie précédent bascule 2
 * sh : etat du comparateur
 *     0 : activé
 *     1 : desactivé
 */

void cpft_c(int *n,int *nev,int *flag,double *delay,double *z1,double *z2,double *evout,double *y1,double *y2,double *sh)
{
 /*déclaration*/
 int i;

 for (i=0;i<(*n);i++)
 {
  if(flag[i]==1)
  {
   switch (nev[i])
   {
    case 1 :
    {
     if(sh[i]!=1)
     {
      if ((z1[i]==0) && (z2[i]==0)) {y1[i] = 1; y2[i] = 0;}
      else if (z1[i] == 1) {y1[i] = 1; y2[i] = 0;}
      else {y1[i] = 1; y2[i] = 0;}
     }
     break;
    }
    case 2 :
    {
     if(sh[i]!=1)
     {
      if ((z1[i] == 0) && (z2[i] == 0)) {y1[i] = 0; y2[i] = 1;}
      else if (z2[i] == 1) {y1[i] = 0; y2[i] = 1;}
      else {y1[i] = 0; y2[i] = 1;}
     }
     break;
    }
    case 4 :
    {
     y1[i]=0;
     y2[i]=0;
     sh[i]=0;
     break;
    }
    break;
   }
  }
  else if(flag[i]==3)
  {
   switch (nev[i])
   {
    case 1 :
    {
     if(z2[i] == 1) {evout[i]=delay[i];}
     sh[i]=1;
     break;
    }
    case 2 :
    {
     if(z1[i] == 1) {evout[i]=delay[i];}
     sh[i]=1;
     break;
    }
    break;
   }
  }
  /*Met en mémoire les états*/
  z1[i]=y1[i];
```

```
  z2[i]=y2[i];
 }
 return;
}
```

1.10 cw_c

- **Short description:** noisy constant wave generator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.10.1 Parameters

- **n :** size of vectors
- **ampl :** level of output wave
- **opt :** routine options
  - **1 :** cosine output
  - **2 :** sine output
  - **3 :** both cosine and sine outputs
- **theta_i :** incremental step of angulary position
- **y1 :** address of real part of output vector
- **y2 :** address of imaginary part of output vector

1.10.2 File content

```
/* cw_c subroutine
 * Noisy Constant Wave computation
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

 #include "mod_num_lib.h"

/* cw_c : routine de calcul d'une onde à fréquence constante (1 ou 2 composantes)
 *
 * Entrées :
 * n       : longueur du vecteur
 * ampl    : amplitude de l'onde
 * opt     : option de la routine (1:cos, 2:sin, 3:cos et sin)
 * theta_i : pas d'incrémentation de l'angle
 *
 * Sorties :
 * y1 : adresse de départ du vecteur de sortie composante réelle
 * y2 : adresse de départ du vecteur de sortie composante imaginaire
 *
 * Entrées/Sorties :
 * theta   : angle instantanée
 *
 * Dépendances :
 * math.h
 *
 */

void cw_c(int *n,double *ampl,int *opt,double *theta_i,double *y1,double *y2,double *theta)
{
 /*déclaration*/
 int i;

 for (i=0;i<(*n);i++)
 {
  switch (*opt)
  {
   case 1 :
   {
    y1[i]=(*ampl)*cos((*theta));
    break;
   }
   case 2 :
   {
    y2[i]=(*ampl)*sin((*theta));
    break;
```

```
   }
   case 3 :
   {
    y1[i]=(*ampl)*cos((*theta));
    y2[i]=(*ampl)*sin((*theta));
    break;
   }
  }
  *theta=(*theta)+(*theta_i);
 }
 return;
}
```

1.11 decod_c

- **Short description:** sequence by symbol multiplication computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.11.1 Parameters

- **nuc :** size of sequence vector
- **nus :** size of symbol vector
- **c :** address of sequence vector
- **s :** address of symbol vector
- **y :** address of output vector

1.11.2 File content

```
/* decod_c subroutine
 * Symbol by chip demodulation
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* decod_c routine de multiplication  code par symbole
 * nuc : taille du vecteur code
 * nus : taille du vecteur symbole
 * c : adresse de départ du vecteur code
 * s : adresse de départ du vecteur symbole
 * y : adresse de départ du vecteur résultat
 */

void decod_c(int *nuc,int *nus,double *c,double *s,double *y)
{
 /*Déclaration des variables compteur*/
 int i,j;

 /*Réalise multiplication code[j] par symbole[i]*/
 for (i=0;i<(*nus);i++)
 {
  for(j=0;j<(*nuc);j++)
  {
   y[i*(*nuc)+j]=s[i*(*nuc)+j]*c[j];
  }
 }
}
```

1.12 demodpsk_c

- **Short description:** M-ary Phase Shift Keying demodulator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.12.1 Parameters

- **n :** size of vectors
- **m :** number of state (scalaire)
- **i_c :** address of input vector of I component
- **q_c :** address of input vector of Q component
- **y :** address of output vector of the number of symbol

1.12.2 File content

```
/* demodpsk_c subroutine
 * Phase Shift Keying demodulator
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

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

void demodpsk_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)/(2*M_PI));
 }
 return;
}
```

1.13 demodqam_c

- **Short description:** Quadrature amplitude demodulator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.13.1 Parameters

- **n :** size of vectors
- **m :** length of the binary words in number of bits (scalar)
- **i_c :** address of input vector of I component
- **q_c :** address of input vector of Q component
- **y :** address of output vector of the number of symbol

1.13.2 File content

```
/* demodqam_c subroutine
 * Quadrature Amplitude Modulation demodulator
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* demodqam_c routine de calcul d'un démodulataur mQAM
 *
 * Entrées :
 * n :longueur des vecteurs d'entrées
 * m :longueurs des mots binaires en nombre de bits (scalaire)
 * 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 demodqam_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++)
 {
  /*Calcul des sélecteurs binaires (c'est maladroit!!)*/
  nd=(1<<(*m)/2)-1;
  ng=(1<<(*m))-1-nd;

  /*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)/2);

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

1.14 genint_c

- **Short description:** random integer number generator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.14.1 Parameters

- **n :** size of vectors
- **m :** address of length vector of the binary words in number of bits (scalar)
- **typ :** address of type of binary generators (scalar)
  - **0 :** Generate random Non Return Zero (NRZ) single bits.
  - **1 :** Generate random Return Zero (RZ) single bits or unsigned integers.
  - **2 :** Generate random signed integers.
- **y :** address of output vector

1.14.2 File content

```
/* genint_c subroutine
 * Random Integer Number Generator
 * IRCOM GROUP - Author : A.Layec
 */

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

```
 */

#include "mod_num_lib.h"

/* genint_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 (scalaire)
 * typ : type des générateurs binaires (scalaire)
 * (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 genint_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)
  {
   /*Type 0 -> dédié signal NRZ*/
   case 0:
   {
    y[i]=(int)((rand()&1)*2-1);
    break;
   }
   /*Type 1 -> mot non signé*/
   case 1 :
   {
    k=1;
    k=k<<(*m);
    j=rand();
    y[i]=(j&(k-1));
    break;
   }
   /*Type 2 -> mot codé code complémént à 2*/
   case 2 :
   {
    j=rand();
    j -= 2<<((*m)-2);
    j &= (2<<((*m)-1)) - 1;
    j -= 2<<((*m)-2);
    y[i]=j;
    break;
   }
   break;
  }
 }
 return;
}
```

1.15 gold_c

- **Short description:** Gold sequence generator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.15.1 Parameters

- **N :** size of registers 1 and 2
- **ny :** size of desired output vector
- **y :** address of results vector
- **reg1 :** address of registre 1 vector
- **reg2 :** address of registre 2 vector
- **coef1 :** address of coefficient 1 vector
- **coef2 :** address of coefficient 2 vector

1.15.2 File content

```
/* gold_c subroutine
 * Gold Sequence generator
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* gold_c routine de calcul dynamique de séquences de gold
 * N     : longueur des registres 1 et 2
 * ny    : longueur du vecteur de sortie désirée
 * y     : adresse de départ du vecteur résultat y[0..ny-1]
 * reg1  : registre 1
 * reg2  : registre 2
 * coef1 : coef du registre 1
 * coef2 : coef du registre 2
 */

void gold_c(int *N, int *ny,double *y,int *reg1, int *reg2,int *coef1, int *coef2)
{
 /*Déclaration des variables*/
 int i,j;
 int j1,j2;

 /* Délivre la valeur du dernier bit sur le registre y[]
  * réalise un XOR à la sortie des deux registres
  */
 for(j=0;j<(*ny);j++)
  {
   if(((*reg1)&1)==((*reg2)&1)) y[j]=-1; /*attention ici -1 et pas 0*/
   else y[j]=1;

   /*Calcul de la nouvelle valeur du bit de poid fort des registres*/
   for(i=0;i<(*N);i++)
   {
    /*Test sur la valeur du coefficient i du registre 1*/
    if(((*coef1)&(1<<i))!=0)
    {
     /*Cas c_i=1*/
     if(i!=0)
     {
      /*Réalise opération XOR*/
      if((((*reg1)>>i)&1)==j1) j1=0;
      else j1=1;
     }
    else j1=(*reg1)&1;
    }

    /*Test sur la valeur du coefficient i du registre 2*/
    if(((*coef2)&(1<<i))!=0)
    {
     /*Cas c_i=1*/
     if(i!=0)
     {
      /*Réalise opération XOR*/
      if((((*reg2)>>i)&1)==j2) j2=0;
      else j2=1;
     }
     else j2=(*reg2)&1;
    }
   }
   /*Décale le registre 1 et 2 de 1 bit vers la droite*/
   (*reg1) = (*reg1)>>1;
   (*reg2) = (*reg2)>>1;

   /*Ajoute le bit de poid fort*/
   (*reg1) += (j1<<(*N-1));
   (*reg2) +=  (j2<<(*N-1));
  }
return;
}
```

1.16 intmod_num_lib

- **Short description:** interfacing routine of low-level routine communication library
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.16.1 File content

```
/* intmod_num_lib.c scilab interface routine
 * IRCOM GROUP - Author : A.Layec
 */

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

```
#include "stack-c.h"
#include "mex.h"

void genint_c __PARAMS((int *n, int *m, int *type, double *y));
void surecht_c __PARAMS((int *opt,int *n,int *nech,int *init_c,double *u,double *y));
void modpsk_c __PARAMS((int *n,int *m,double *u,double *i_c, double *q_c));

int intgenint_c(char *fname)
{
  static int l1, m1, n1; /*entrée 1*/
  static int l2, m2, n2; /*entrée 2*/
  static int l3, m3, n3; /*entrée 3*/
  static int l4; /*sortie 1*/
  static int minlhs=1, maxlhs=1, minrhs=3, maxrhs=3;

  /* Check number of inputs and outputs (lhs=1) */
  CheckRhs(minrhs,maxrhs) ;
  CheckLhs(minlhs,maxlhs) ;

  /*Get input and create output*/
  GetRhsVar(1,"i",&m1,&n1,&l1);
  GetRhsVar(2,"i",&m2,&n2,&l2);
  GetRhsVar(3,"i",&m3,&n3,&l3);
  CreateVar(4,"d",istk(l1),&n1,&l4);

  /*Check dimensions*/
  /*TO BE DONE*/

  /*Appel bin_c*/
  genint_c(istk(l1),istk(l2),istk(l3),stk(l4));

  /*Return output variable*/
  LhsVar(1) = 4;

  return 0;
}

int intsurecht_c(char *fname)
{
  static int l1,m1,n1; /*entrée 1*/
  static int l2,m2,n2; /*entrée 2*/
  static int l3,m3,n3; /*entrée 3*/
  static int l4,m4,n4; /*entrée 4*/
  static int l5; /*sortie 1*/
  static int minlhs=1, maxlhs=1,minrhs=4,maxrhs=4;
  static int k;

  /* Check number of inputs and outputs (lhs=1) */
  CheckRhs(minrhs,maxrhs);
  CheckLhs(minlhs,maxlhs);

  /*Get input and create output*/
  GetRhsVar(1,"i",&m1,&n1,&l1); /*opt : option -scalaire*/
  GetRhsVar(2,"i",&m2,&n2,&l2); /*nech : facteur de surechantillonnage -scalaire-*/
  GetRhsVar(3,"i",&m3,&n3,&l3); /*init_c : valeur du compteur initial -scalaire-*/
  GetRhsVar(4,"d",&m4,&n4,&l4); /*u : vecteur d'entrée*/
  k=*istk(l2)*m4;
  sciprint("k=%d\r\n",k);
  CreateVar(5,"d",&k,&n4,&l5); /*y : vecteur de sortie*/

  /*Check dimensions*/
  /*TO BE DONE*/


  /*Appel surech_c*/
  surecht_c(istk(l1),&m4,istk(l2),istk(l3),stk(l4),stk(l5));

  /*Return output variable*/
  LhsVar(1) = 5;

  return 0;
}

int intmodpsk_c(char *fname)
{
  static int l1,m1,n1; /*entrée 1*/
  static int l2,m2,n2; /*entrée 2*/
  static int l3; /*sortie 2*/
  static int l4; /*sortie 1*/
  static int minlhs=2, maxlhs=2,minrhs=2,maxrhs=2;

  /* Check number of inputs and outputs (lhs=1) */
  CheckRhs(minrhs,maxrhs) ;
  CheckLhs(minlhs,maxlhs) ;

  /*Get input and create output*/
  GetRhsVar(1,"i",&m1,&n1,&l1); /*nbr d'états -scalaire-*/
  GetRhsVar(2,"d",&m2,&n2,&l2); /*vecteur d'entrée*/
  CreateVar(3,"d",&m2,&n2,&l3); /*vecteur i*/
  CreateVar(4,"d",&m2,&n2,&l4); /*vecteur q*/

  /*Check dimensions*/
  /*TO BE DONE*/

  /*Appel bin_c*/
  modpsk_c(&m2,istk(l1),stk(l2),stk(l3),stk(l4));

  /*Return output variable*/
  LhsVar(1) = 3;
  LhsVar(2) = 4;
  return 0;
```

```
}

static GenericTable Tab[]={
{(Myinterfun)sci_gateway, intgenint_c,"error msg"},
{(Myinterfun)sci_gateway, intmodpsk_c,"error msg"},
{(Myinterfun)sci_gateway, intsurecht_c,"error msg"},
};

int C2F(intmod_num_lib)()
{  Rhs = Max(0, Rhs);
(*(Tab[Fin-1].f))(Tab[Fin-1].name,Tab[Fin-1].F);
  return 0;
}
```

1.17 intsym_c

- **Short description:** discret symbol integrator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.17.1 Parameters

- **n :** size of vectors (scalar)
- **nech :** number of sample per symbol (scalar)
- **count :** initial value of counter (scalar)
- **step :** time integration step (scalar)
- **u :** address of input vector (vector)
- **y :** address of output vector (scalar)
- **z :** address of memory word (scalar)

1.17.2 File content

```
#include "mod_num_lib.h"
/*
intsym_c routine de calcul d'intégration symbole
Entrées :
n     : taille des vecteurs (scalaire)
nech  : nombre d'échantillons par symbole (scalaire)
count : valeur initiale du compteur (scalaire)
step  : pas d'intégration (scalaire)
u     : vecteur à intégrer (vecteur)
Sorties :
y : vecteur inégrer (scalaire)
Entrée/Sortie :
z : echantillon mémoire (scalaire)
*/
void intsym_c(int *n,int *nech,int *init_c,double *step,double *u,double *y,double *z)
{
 /*Déclaration des variables compteurs*/
 int i;
 int count;
 /*récupère valeur initiale*/
 count=*init_c;

 /*pour tous les échantillons du vecteur d'entrée*/
 for(i=0;i<(*n);i++)
 {
  if(i==count)
  {
   count += *nech;
   y[i]=0;
  }
  else
  {
   if(i==0) y[i]=2/(*step)*(u[i]+u[i-1])+(*z);
   else if(i==(*n)-1)
   {
    y[i]=2/(*step)*(u[i]+u[i-1])+y[i-1];
    (*z)=y[i];
   }
   else y[i]=2/(*step)*(u[i]+u[i-1])+y[i-1];
  }
 }

 return;
}
```

1.18 mash1_c

- **Short description:** first order delta-sigma modulator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.18.1 Parameters

- **n :** size of input vector (scalar)
- **m :** gain of modulator (scalar)
- **u :** address of input vector
- **e :** integrator state (scalar)
- **z :** output state +m;-m (scalar)
- **y :** address of output vector (+1;-1)
- **q :** address of error quantification output vector

1.18.2 File content

```
/* mash1_c subroutine
 * First order MASH Sigma-Delta Modulator
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* mash1_c routine de calcul d'un modulateur sigma-delta du 1er ordre type MASH
 *
 * entrées :
 * n   : longueur du vecteur d'entrée (scalaire)
 * m   : gain du modulateur (scalaire)
 * u   : adresse de départ du vecteur à convertir
 * entrées/sorties
 * e   : intégral du signal d'erreur du dernier élément (scalaire)
 * z   : etat de sortie du dernier élément +m;-m (scalaire)
 * y   : adresse de départ du vecteur de sortie (+1;-1)
 * sorties :
 * q   : adresse de départ du vecteur de sortie de l'erreur de quantification
 */

void mash1_c(int *n,double *m,double *u,double *e,double *z,double *y,double *q)
{
 /*déclaration*/
 int i;

 for(i=0;i<(*n);i++)
 {
  /*Calcul de l'intégrale du signal d'erreur*/
  *e+=u[i]-(*z);

  /*Réalisation de la comparaison de signe*/
  if((*e)>0) y[i]=1;
  else y[i]=-1;

  /*mise en mémoire de la sortie*/
  *z = *m*y[i];

  /*Calcul du bruit de quantification*/
  q[i]=(*z)-(*e);
 }
 return;
}
```

1.18.3 See Also

- mash2_c - second order delta-sigma modulator computational routine (Low level routine)
- mash3_c - third order delta-sigma modulator computational routine (Low level routine)

1.19 mash2_c

- **Short description:** second order delta-sigma modulator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.19.1 Parameters

- **n :** size of input vector (scalar)
- **m :** gain of modulator (scalar)
- **u :** address of input vector
- **e :** integrator state (scalar)
  - **e[0] :** integrator state of modulator 1
  - **e[1] :** integrator state of modulator 2
- **z :** output state +m;-m (scalar)
  - **z[0] :** output state of modulator 1
  - **z[1] :** output state of modulator 2
  - **z[2] :** output state of the second order mash modulator
- **y :** address of output vector (+1;-1)
- **q :** address of error quantification output vector
- **w :** address of working vector

1.19.2 File content

```
/* mash2_c subroutine
 * Second order MASH Sigma-Delta Modulator
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* mash2_c routine de calcul d'un modulateur sigma-delta du 2eme ordre type MASH
 *
 * entrées :
 * n   : longueur du vecteur d'entrée (scalaire)
 * m   : gain du modulateur (scalaire)
 * u   : adresse de départ du vecteur à convertir
 * entrées/sorties
 * e   : intégral du signal d'erreur du dernier élément (vecteur)
 *       e[0]     : modulateur 1
 *       e[1]   : modulateur 2
 * z   : etat de sortie du dernier élément +m;-m (scalaire)
 *       z[0]     : modulateur 1
 *       z[1]     : modulateur 2
 *       z[2]     : etat mémoire du mash2
 * y   : adresse de départ du vecteur de sortie (+1;-1)
 * q   : adresse de départ du vecteur de sortie de l'erreur de quantification
 * w   : adresse de départ du vecteur de travail
 *
 * dépendances :
 * mash1_c
 */

void mash2_c(int *n,double *m,double *u,double *e,double *z,double *y,double *q,double *w)
{
 /*déclaration*/
 int i;

 /*Appel Mash1*/
 mash1_c(n,m,&u[0],&e[0],&z[0],&y[0],&q[0]);

 /*Appel Mash1*/
 mash1_c(n,m,&q[0],&e[1],&z[1],&w[0],&q[0]);

 /*réalise opération sur bruit*/
```

```
 for(i=0;i<(*n);i++)
 {
  y[i]=(y[i]+z[2]-w[i]);
  /*mise en mémoire de l'etat mémoire du mash2*/
  z[2]=w[i];
 }
 return;
}
```

1.19.3 See Also

- mash1_c - first order delta-sigma modulator computational routine (Low level routine)
- mash3_c - third order delta-sigma modulator computational routine (Low level routine)

1.20 mash3_c

- **Short description:** third order delta-sigma modulator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.20.1 Parameters

- **n :** size of input vector (scalar)
- **m :** gain of modulator (scalar)
- **u :** address of input vector
- **e :** integrator state (scalar)
  - **e[0] :** integrator state of modulator 1
  - **e[1] :** integrator state of modulator 2
  - **e[2] :** integrator state of modulator 3
- **z :** output state +m;-m (scalar)
  - **z[0] :** output state of modulator 1
  - **z[1] :** output state of modulator 2
  - **z[2] :** output state of the second order mash modulator
  - **z[3] :** output state of modulator 3
  - **z[4],z[5] :** output states of the third order mash modulator
- **y :** address of output vector (+1;-1)
- **q :** address of error quantification output vector
- **w :** address of working vector

1.20.2 File content

```
/* mash3_c subroutine
 * Third order MASH Sigma-Delta Modulator
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* mash3_c routine de calcul d'un modulateur sigma-delta du 3eme ordre type MASH
 *
 * entrées :
 * n   : longueur du vecteur d'entrée (scalaire)
```

```
 * m   : gain du modulateur (scalaire)
 * u   : adresse de départ du vecteur à convertir
 * entrées/sorties
 * e   : intégral du signal d'erreur du dernier élément (vecteur)
 *      e[0]     : modulateur 1
 *      e[1]     : modulateur 2
 *      e[2]     : modulateur 3
 * z   : etat de sortie du dernier élément +m;-m (scalaire)
 *      z[0]     : modulateur 1
 *      z[1]     : modulateur 2
 *      z[2]     : etat mémoire du mash2
 *      z[3]     : modulateur 3
 *      z[4],z[5]: etats mémoire du mash2
 * y   : adresse de départ du vecteur de sortie (+1;-1)
 * q   : adresse de départ du vecteur de sortie de l'erreur de quantification
 * w   : adresse de départ du vecteur de travail
 *
 * dépendances :
 * mash1_c
 * mash2_c
 */

void mash3_c(int *n,double *m,double *u,double *e,double *z,double *y,double *q,double *w)
{
 /*déclaration*/
 int i;

 /*Appel mash2*/
 mash2_c(n,m,&u[0],&e[0],&z[0],&y[0],&q[0],&w[0]);

 /*Appel mash1*/
 mash1_c(n,m,&q[0],&e[2],&z[3],&w[0],&q[0]);

 /*réalise opération sur bruit*/
 for(i=0;i<(*n);i++)
 {
  y[i]=(y[i]+(w[i]-2*z[4]+z[5]));
  /*mise en mémoire des états mémoires du mash3*/
  z[5]=z[4];
  z[4]=w[i];
 }
 return;
}
```

1.20.3 See Also

- mash1_c - first order delta-sigma modulator computational routine (Low level routine)
- mash2_c - second order delta-sigma modulator computational routine (Low level routine)

1.21 mllsrs_c

- **Short description:** Pseudo Noise random sequences generator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.21.1 Parameters

- **N :** length of registers
- **ny :** desired length of output vector
- **y :** address of resulting output vector
- **reg :** address of register input vector
- **coef :** address of coefficient input vector

1.21.2 File content

```
/* mllsrs_c subroutine
 * Maximal Linear Length Shift Register Sequence Generator
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"
```

```
/* mllsrs_c routine de calcul dynamique de séquence à longueur max.
 * N     : longueur du registre
 * ny    : longueur du vecteur de sortie désirée
 * y     : adresse de départ du vecteur résultat y[0..ny-1]
 * reg   : registre
 * coef  : coef du registre
 */

void mllsrs_c(int *N, int *ny,double *y,int *reg, int *coef)
{
 /*Déclaration des variables*/
 int i,j,l;

 for(l=0;l<(*ny);l++)
    {
     /*Delivre la valeur du dernier bit sur le registre y[]*/
     if((*reg)&1==1) y[l]=1;
     else y[l]=-1; /*attention ici -1 et pas 0*/

     /*Calcul de la nouvelle valeur du bit de poid fort*/
     for(i=0;i<(*N);i++)
     {
      /*Test sur la valeur du coefficient i*/
      if(((*coef)&(1<<i))!=0)
      {
       /*Cas c_i=1*/
       if(i!=0)
       {
        /*Réalise opération XOR*/
        if((((*reg)>>i)&1)==j) j=0;
        else j=1;
       }
      else j=(*reg)&1;
      }
     }
     /*Décale le registre de 1 bit vers la droite*/
     (*reg) = (*reg)>>1;

     /*Ajoute le bit de poid fort*/
     (*reg) += (j<<((*N)-1));
    }
 return;
}
```

1.22 modpsk_c

- **Short description:** M-ary phase shift keying modulator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.22.1 Parameters

- **n :** size of vectors (scalaire)
- **m :** number of state (scalaire)
- **u :** address of symbol number input vector
- **i_c :** address of I component ouput vector
- **q_c :** address of Q component ouput vector

1.22.2 File content

```
/* modpsk_c subroutine
 * Phase Shift Keying Modulator
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* modpsk_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 scalaire
 * 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 sortie 2)
```

```
 *
 * Dépendances :
 * math.h
 */

void modpsk_c(int *n,int *m,double *u,double *i_c, double *q_c)
{
 /*Déclaration des variables compteurs*/
 int k,i;

 for(i=0;i<(*n);i++)
 {
  /*récupération de la valeur du port d'entrée*/
  k=(int)u[i];
  /*Calcul des composantes i et q*/
  i_c[i]=cos(M_PI*(2*k+1)/(*m));
  q_c[i]=-sin(M_PI*(2*k+1)/(*m));
 }
 return;
}
```

1.23 modqam_c

- **Short description:** Quadrature amplitude modulator computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.23.1 Parameters

- **n :** size of input vectors
- **m :** number of bits per symbol (scalar)
- **u :** address of symbol input vector
- **i_c :** address of I component output vector.
- **q_c :** address of Q component output vector.

1.23.2 File content

```
/* modqam_c subroutine
 * Quadrature Amplitude Modulation Modulator
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* modqam_c routine de calcul d'un modulataur mQAM
 *
 * Entrées :
 * n :longueur du vecteur d'entrée
 * m :longueurs des mots binaires en nombre de bits (scalaire)
 * 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 modqam_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++)
 {
  /*Calcul des sélecteurs binaires (c'est maladroit!!)*/
  nd=(1<<(*m)/2)-1;
  ng=(1<<(*m))-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)/2))*2)-nd;
 }
 return;
}
```

1.24 nfilter_c

- **Short description:** finite impulse response filter by discrete time convolution method computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.24.1 Parameters

- **n :** size of input vector
- **nbcoef :** length of the impulse response
- **u :** address of signal input vector
- **pulse :** address of impulse response input vector
- **y :** address of signal output vector
- **z :** address of discrete state of the filter

1.24.2 File content

```
/* nfilter_c subroutine
 * FIR computation
 * with classic discrete convolution method
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* nfilter_c : routine de calcul d'un filtre à réponse impulsionnelle fini
 *             par convolution numérique
 *
 * Entrées :
 * n : longueur du vecteur d'entrée à filtrer
 * nbcoef : longueur de la réponse impulsionelle
 * u : adresse de départ du vecteur d'entrée à filtrer
 * pulse : adresse de départ de la réponse impulsionelle
 *
 * Sorties :
 * y : adresse de départ du vecteur filtré
 *
 * Entrées/sorties :
 * z : adresse de départ du vecteur mémoire du filtre
 */

void nfilter_c(int *n,int *nbcoef,double *u,double *pulse,double *y,double *z)
{
 /*déclaration*/
 double somme;
 int i,j;

 for(j=0;j<(*n);j++)
 {
  /*calcul sortie*/
  somme=0;
  for(i=0;i<(*nbcoef)-1;i++)
  {
   somme=somme+z[(*nbcoef)-1-i]*pulse[i];
  }
  somme=somme+u[j]*pulse[*nbcoef-1];

  /*calcul mémoire*/
  for(i=0;i<(*nbcoef)-1;i++)
  {
   z[(*nbcoef)-i-1]=z[(*nbcoef)-i-2];
  }
  z[0]=u[j];

  /*recopie sortie dans y*/
  y[j]=somme;
 }

 return;
}
```

1.25 noiseblk_c

- **Short description:** white gaussian noise generator computational routine

- **Library:** mod_num_rout_lib - library of low-level computational routines

1.25.1 Parameters

- **n :** size of vectors
- **typ :** type of output (0:cos/1:sin)
- **sig :** variance of gaussian noise generator (scalar)
- **mean :** mean of gaussian noise generator (scalar)
- **y :** address of noisy output vector

1.25.2 File content

```
/* noiseblk_c subroutine
 * Gaussian Noise generator
 * with Box Muller Law method
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* noiseblk_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  : variance (scalaire)
 * mean : moyenne (scalaire)
 * Sorties :
 * y    : vecteur des sorties
 *
 * dépendances
 * math.h
 *
 */

void noiseblk_c(int *n,int *typ,double *sig,double *mean,double *y)
{
 /*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)*sqrt(-log(rand1));
  phase=2*M_PI*rand2;

  /*Calcul y*/
  if((*typ)==0)
  y[i]=(*mean)+ampl*cos(phase);
  else if((*typ)==1)
  y[i]=(*mean)+ampl*sin(phase);
 }
 return;
}
```

1.26 noiseiq_c

- **Short description:** white gaussian noise generator for quadrature signals computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.26.1 Parameters

- **n :** size of vectors
- **sig :** variance of gaussian noise generator (scalar)
- **mean :** mean of gaussian noise generator (scalar)
- **i_c :** address of I component output vector.
- **q_c :** address of Q component output vector.

1.26.2 File content

```
/* noiseiq_c subroutine
 * Gaussian Noise generator
 * with Box Muller Law method
 * for complex values
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* noiseiq_c routine de calcul d'échantillons bruités par la méthode "Box Muller Law"
 *
 * Entrées :
 * n    : taille des vecteurs
 * sig  : variance (scalaire)
 * mean : moyenne (scalaire)
 * Sorties :
 * i_c : vecteur des composantes I
 * i_q : vecteur des composantes Q
 *
 * dépendances
 * math.h
 */

void noiseiq_c(int *n,double *sig,double *mean,double *i_c,double *i_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)*sqrt(-log(rand1));
       phase=2*M_PI*rand2;

       /*Calcul i_c et i_q*/
       i_c[i]=(*mean)+ampl*cos(phase);
       i_q[i]=(*mean)+ampl*sin(phase);
      }
 return;
}
```

1.27 overlapadr_c

- **Short description:** Overlap computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.27.1 Parameters

- **m1 :** size of the input vector

- **n :** size of the memory word
- **nb_coef :** size of exess word.
- **u_r :** address of the input vector
- **z_r :** address of the output vector

1.27.2 File content

```
/* overlapdr_c subroutine
 * Orverlapping operation
 * for real value
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* overlapadr_c routine de calcul du mot overlappé partie réelle
 *
 * m1      : taille du vecteur d'entrée
 * n       : longeur du mot à conserver
 * nb_coef : taille du mot excédentaire
 * u_r     : vecteur d'entrée de taille m1 (m1>n)
 * z_r     : vecteur de sortie de taille nb_coef
 *
 * utilise : dcopy (BLAS)
 *
 */

void overlapadr_c(int *m1,int *n,int *nb_coef,double *u_r,double *z_r)
{
 /*déclaration*/
 int i,l,k;

 /*Ajoute les nz éléments précédents au début du vecteur y_r*/
 for(i=0;i<(*nb_coef);i++) u_r[i]=u_r[i]+z_r[i];

 /*Recopie les nz éxcédentaires de z_res_r dans z*/
 F2C(dcopy)((l=(*nb_coef),&l),&u_r[(*n)],(k=1,&k),&z_r[0],(k=1,&k));

 return;
}
```

1.28 overlaprsr_c

- **Short description:** vectorial right shift register with memory computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.28.1 Parameters

- **n :** size of vectors
- **m :** number of shift (set the size of the memory word)
- **u :** address of the input vector
- **y :** address of the output vector.
- **z :** address of the memory word

1.28.2 File content

```
/* overlaprsr_c subroutine
 * Vectorial Right Shift Register
 * with memory word
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"
```

```
/* overlaprsr_c routine de calcul de décalage vectoriel à droite avec
 * mot mémoire
 *
 * Entrées :
 * n : taille du vecteur
 * m : longueur du décalage (taille du mot mémoire)
 * u : adresse de départ du vecteur d'entrée
 * Sorties :
 * y : adresse de départ du vecteur de sortie
 * Entrée/Sortie :
 * z : adresse de départ du vecteur mémoire
 */

void overlaprsr_c(int *n,int *m,double *u,double *y,double *z)
{
 /*Déclaration des variables compteurs*/
 int i;

 for(i=0;i<(*m);i++)
 {
    y[i]=z[i];
    z[i]=u[((*n)-(*m))+i];
 }
 for(i=(*m);i<(*n);i++) y[i]=u[i-(*m)];

 return;
}
```

1.29 sousecht_c

- **Short description:** down sample computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.29.1 Parameters

- **n :** size of vectors.
- **nech :** down-sample factor
- **init_c :** initial value of counter
- **u :** address of input vector
- **y :** address of output vector

1.29.2 File content

```
/* sousecht_c subroutine
 * Down-Sampling Computation
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* sousech_c routine de calcul de sous-échantillonnage en temporel
 *
 * Entrées :
 * n      : taille du vecteur original
 * nech   : facteur de sous-échantillonnage
 * init_c : valeur initiale du compteur
 * u      : vecteur d'entrée à sous-échantillonner
 * Sorties :
 * y      : vecteur de sortie
 *
 */

void sousecht_c(int *n,int *nech,int *init_c,double *u,double*y)
{
 /*Déclaration des variables compteurs*/
 int i,j;
 int count;

 /*Récupère valeur initiale du compteur*/
 count=*init_c;
 /*raz j*/
 j=0;

 /*Pour tous les échantillons du vecteur d'entrée*/
 for(i=0;i<(*n);i++)
```

```
 {
  /*if(i==(count-1))*/
  if(i==count)
  {
   y[j]=u[i];
   count += (*nech);
   j++;
  }
 }
 return;
}
```

1.30 surecht_c

- **Short description:** up sample computational routine
- **Library:** mod_num_rout_lib - library of low-level computational routines

1.30.1 Parameters

- **opt :** option
  - **0 :** re-sample
  - **1 :** up-sample with zero pading
- **n :** size of vectors
- **nech :** up-sample vector
- **init_c :** address of the counter vector
- **u :** address of the input vector (of size n)
- **y :** address of the output vector (of size n*nech)

1.30.2 File content

```
/* surecht_c subroutine
 * Up-Sampling Computation
 * IRCOM GROUP - Author : A.Layec
 */

/* REVISION HISTORY :
 * $Log$
 */

#include "mod_num_lib.h"

/* surecht_c routine de calcul de sur-échantillonnage en temporel
 *
 * Entrées :
 * opt    : option 0: sur-échantillonnage classique
 *                 1: sur-échantillonnage avec insertion de zéros(Dirac)
 * n      : taille du vecteur original
 * nech   : facteur de sur-échantillonnage
 * init_c : adresse du compteur
 * u      : adresse du vecteur d'entrée (de taille n)
 * Sorties :
 * y      : adresse du vecteur de sortie (de taille n*nech)
 *
 */

void surecht_c(int *opt,int *n,int *nech,int *init_c,double *u,double *y)
{
 /*déclaration des variables compteur*/
 int i,j;
 int counter;

 switch(*opt)
 {
  /*Suréchantillonnage classique*/
  case 0 : {
            for(j=0;j<(*n);j++)
            {
             for(i=0;i<(*nech);i++) y[j*(*nech)+i]=u[j];
            }
            break;
           }

  /*Suréchantillonnage avec insertion zéro*/
  case 1 : {
```

```
            counter=(*init_c);
            i=0;
            for(j=0;j<(*n)*(*nech);j++)
            {
             y[j]=0;
             if(j==counter)
             {
              counter=counter+(*nech);
              y[j]=u[i];
              i++;
             }
            }
            break;
           }
  /*other case*/
  break;
 }
 return;
}
```