RAL-TR 1999068
R3 STORE

CLRC

Technical Report
RAL-TR-1999-068

OSIRIS (Data Reduction)

D Engberg

10th November 1999

COUNCIL FOR THE CENTRAL LABORATORY OF THE RESEARCH COUNCILS

ISSN 1358-6254

OSIRIS (data reduction)

Dennis Engberg, D.Engberg@rl.ac.uk November, 1999

This document contains documentation of OSIRIS programmes and normalisation.
RAL-TR-1999-068

Contents

1 organisation

1.1 Where are the files

The login file for OSIRIS is osiris$disk0:[osiris]login.com and this file just contains a command to run the OSIMGR login file osiris$disk0:[osimgr]login.com.

All the programmes on OSIRIS resides in either the directory OSIRIS$DISK0:[OSIRIS.programmes] or OSIRIS$DISK0:[OSIRIS.alpha]. In these you have some subdirectories

sub-directory	content
com	command files
isis	code not developed on OSIRIS
og	gcl files, source and modules, used with OG
genie	old genie code
src	source and executables, not used within OG
wish	new dashboard
misc	misc

Table 1: The subdirectories of [osiris.programmes] and what they contain.

The raw files created during an experiment is saved in OSIRIS$DISK0:[OSIMGR.DATA] and is later saved in OSIRIS$DISK1:[OSIMGR.DATA] before it is backed up to optical disks. The logical OSIRIS_DATA allows you to access both directories without worrying about where your data is.

1.2 start up files

There are several things set up in the login files but the important command for the analysis programmes is the running of the command file osiris$disk0:[osiris]user_set_up.com. In this file, there are the following lines:

```
$ assign osiris$disk0:[osiris]genieinit.gcl genie_gcl_init
$ opengenie :==@LIB$DISK:[OPENGENIE]GENIE_SETUP -l
$ og        :==@LIB$DISK:[OPENGENIE]GENIE_SETUP -l
$ wg        :==@LIB$DISK:[OPENGENIE]GENIE_SETUP
$
$ drange== "@[osiris.programmes.com]drange.com"
$ ochange=="$osiris$disk0:[osiris.programmes.src]ochange.exe"
$ dashboard:==@OSIRIS$DISK0:[OSIRIS.programmes.WISH]tcp.com
```

All the OpenGenie set-up is in this file so it is enough for a user to put this line in his login file to be able to run all the programmes. The first line is to make sure you use the correct genieinit.gcl file, more about that below. The -l flag on the opengenie, og definitions is to run the command line version of OG as default And wg is then if for some strange reason you would like to run the windowed version. The definitions of the commands at the end will be explained later.

The important set-up file genieinit.gcl should look like this:

```
set/inst "osi"
set/disk "osiris_data:"
set/dir ""
set/ext  "raw"
alter/plotcolour $red

# load commands to analyse diffraction data
load "osiris$disk0:[osiris.programmes.og]osd.gcl"
# load local OG extensions
load "osiris$disk0:[osiris.programmes.og]local.gcl"
# load program to export in gsasformat by Stuart 28/9/98
load "osiris$disk0:[osiris.programmes.isis]gsasgen.gcl"
# load file with alias definitions.
load "osiris$disk0:[osiris.programmes.og]alias.gcl"
#Don't print all that blue information text
toggle/info
```

The first lines set up OG to look in the right directories for the data. As you note the directory is not set. This is because the logical `OSIRIS_DATA` already contain the directory in its definition. Thereafter the reduction programmes are loaded.

2 USER programmes

2.1 command files

drange.com: this is a command file that sets the chopper phases and the TCB for some preset d-range values. It sets the correct chopper phases and loads the corresponding TCB files.

2.2 gcl files

2.2.1 osd.gcl

The `osd.gcl` contain the main reduction programmes for OSIRIS. The code is fairly self explanatory. There are in principle two different way of reducing the data. Firstly, by using the sum_raw programme where you can sum a number of raw files, *then* you focus your data with the focus_norm or focus_norm_bank programme. Secondly you can focus each run separately and thereafter merge the different runs with merge_diffractogram. This is the recommended way of doing it.

2.2.2 gsasgen.gcl,constants.gcl,local.gcl

gsasgen.gcl: contains code for generation of GSAS format files, it is not included since the author did not write it; constants.gcl: contains some physical constants, actually so far it only contains one line but all physical constants will be put in this file as the need arises; local.gcl: contains short procedures that make life a bit easier, as redefinition of assign to allow you to type assign dae, easy ascii export interface, ccsl output generation, see the code for more details.

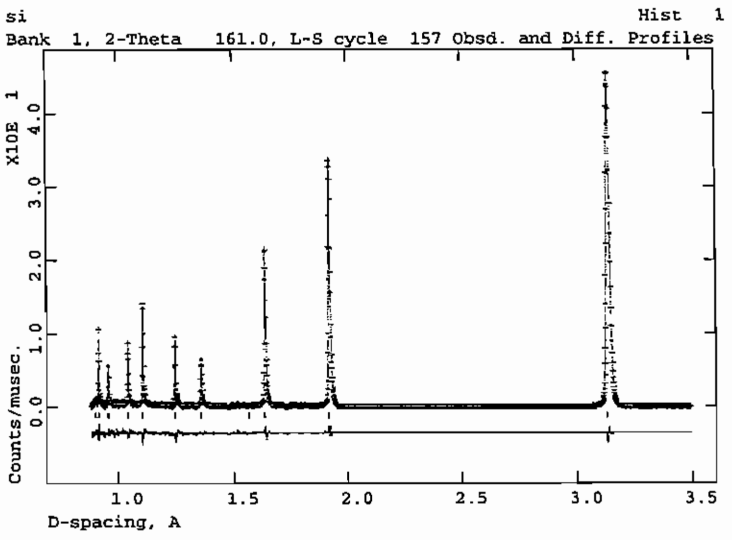

Figure 1: GSAS Si refinement

2.3 fortran

`find_min.f` is a short f77 routine to find the minimum in the monitor spectrum. It is used together with the unwrap routines in osd.gcl

2.4 c

`ochange.c` is a programme to change all the tcb files. On most instrument you would use the `change` editor to change the tcb settings but since on OSIRIS ten or twenty files has to be edited for each user, this is not feasible to do manually for each user, this c programme does it for you.

3 Instrument programmes

3.1 normalisation

Instrument parameter files for GSAS and CCSL have been set up and here is a short description of the present status of the resulting peak shape of OSIRIS. For gsas profile number three was used (exponential pseudovoigt convolution, Von Dreele, 1990 unpublished) and for CCSL the Ikeda-Carpenter was used. The present parameter files are available on osiris in `[osiris.instpar]`/

3.1.1 gsas

The result for the first two modules look good even though there still is some small background problems. The peakshape is well described though not as well as with the CCSL package, see fig. 1 to fig. 4.

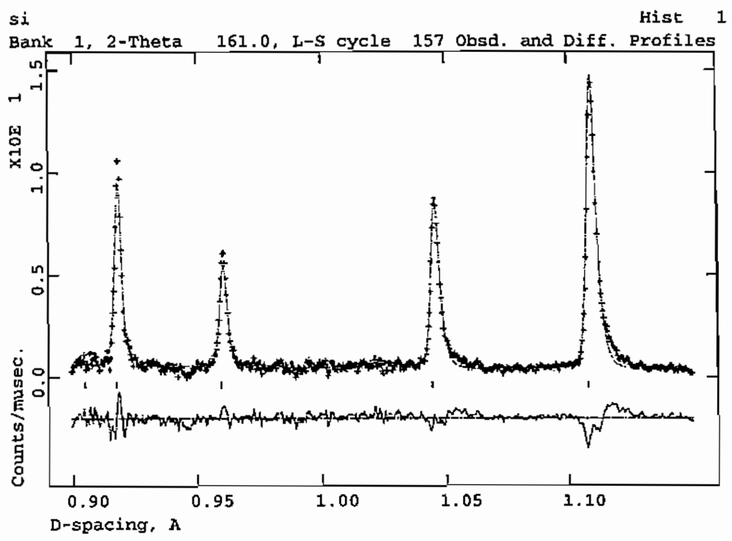

Figure 2: GSAS Si refinement

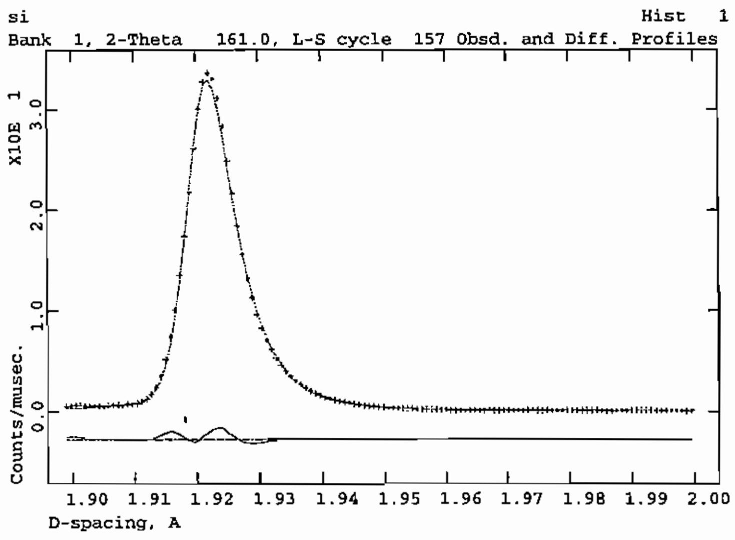

Figure 3: GSAS Si refinement

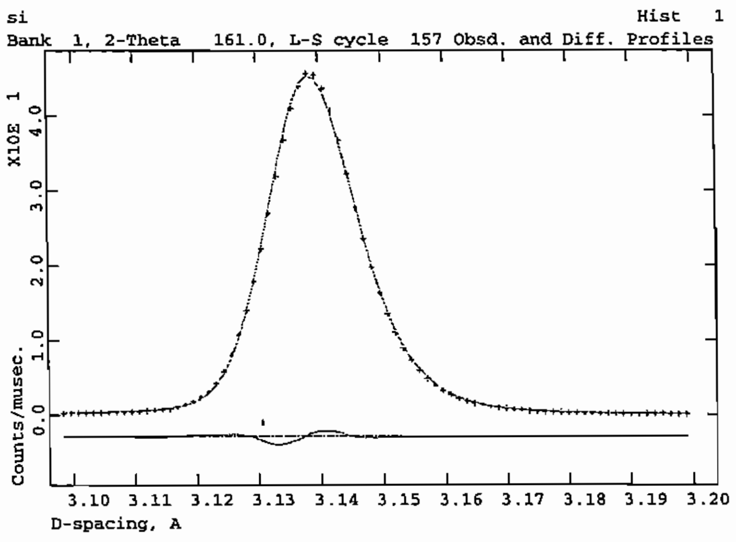

Figure 4: GSAS Si refinement

3.1.2 ccsl

The description of the peakshape is better done by CCSL than GSAS, inspite of this the preferred package among the users has so far been the GSAS one.

4 code listing

4.1 drange.com

```
$!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$! This command file sets the proper values for the standard wavelength!
$! ranges at 25Hz.                                                      !
$! Dennis Engberg 25/11/98                                              !
$!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$
$
$ IF (p1 .eqs. 1)   THEN GOTO d1
$ IF (p1 .eqs. 2)   THEN GOTO d2
$ IF (p1 .eqs. 3)   THEN GOTO d3
$ IF (p1 .eqs. 4)   THEN GOTO d4
$ IF (p1 .eqs. 5)   THEN GOTO d5
$ IF (p1 .eqs. 6)   THEN GOTO d6
$ IF (p1 .eqs. 7)   THEN GOTO d7
$ IF (p1 .eqs. 8)   THEN GOTO d8
$ IF (p1 .eqs. 9)   THEN GOTO d9
$ IF (p1 .eqs. 10)  THEN GOTO d10
```

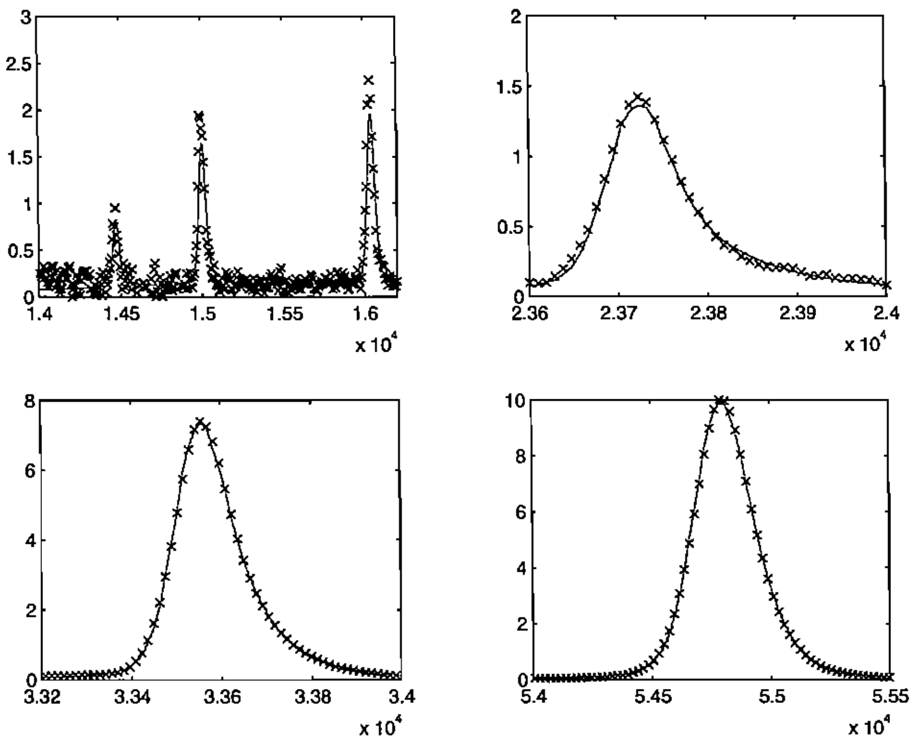

Figure 5: Si refinement using ccsl. Scattered intensity on ordinate and time of flight (micro sec) on the abscissa.

```
$ IF (p1 .eqs. 11)  THEN GOTO d11
$
$ IF (p1 .eqs. 109) THEN GOTO d9a
$ IF (p1 .eqs. 1010) THEN GOTO d10a
$
$!
$ ERROR:
$ WRITE SYS$OUTPUT "Unrecognized parameter option ''P1' "
$ EXIT
$!
$!
$ d1:
$  cset c_ph6 3180
$  cset c_ph10 5056
$  load [osimgr.tcb]12.tcb
$ exit
$
$ d2:
$  cset c_ph6 6370
$  cset c_ph10 10100
$  load [osimgr.tcb]14.tcb
$ exit
$
$ d3:
$  cset c_ph6 9555
$  cset c_ph10 15150
$  load [osimgr.tcb]16.tcb
$ exit
$
$ d4:
$  cset c_ph6 12740
$  cset c_ph10 20200
$  load [osimgr.tcb]18.tcb
$ exit
$
$ d5:
$  cset c_ph6 15900
$  cset c_ph10 25250
$  load [osimgr.tcb]110.tcb
$ exit
$
$ d6:
$  cset c_ph6 19100
$  cset c_ph10 31750
$  load [osimgr.tcb]112.tcb
$ exit
$
```

```
$ d7:
$  cset c_ph6 22250
$  cset c_ph10 36700
$  load [osimgr.tcb]l14.tcb
$ exit
$
$ d8:
$  cset c_ph6 25450
$  cset c_ph10 1790
$  load [osimgr.tcb]l16.tcb
$ exit
$
$ d9:
$  cset c_ph6 28665
$  cset c_ph10 6845
$  load [osimgr.tcb]l18.tcb
$ exit
$
$ d10:
$  cset c_ph6 31850
$  cset c_ph10 11900
$  load [osimgr.tcb]l20.tcb
$ exit
$
$ d11:
$  cset c_ph6 35035
$  cset c_ph10 16955
$  load [osimgr.tcb]l22.tcb
$ exit
$
$
$ d9a:
$  cset c_ph6 30258
$  cset c_ph10 9378
$  load [osimgr.tcb]l9a.tcb
$ exit
$
$ d10a:
$  cset c_ph6 33443
$  cset c_ph10 14434
$  load [osimgr.tcb]l10a.tcb
$ exit
```

4.2 alias.gcl

```
#
# some useful aliases
```

```
#
alias "d" "display"
alias "p" "plot"
alias "a/b" "alter/binning"
alias "old_assign" "assign"
alias "ass" "assign"

# a bug in og, _ is tranlated to U :)
alias "fnb" "focusUnormUbank"
alias "md" "mergeUdiffractogram"
```

4.3 constants.gcl

```
PROCEDURE constants
GLOBAL pi

pi=2*arcsin(1)
ENDPROCEDURE
```

4.4 dd.gcl

```
PROCEDURE dd
PARAMETERS RUN=Integer FIRST=Integer LAST=Integer
GLOBAL dev1
LOCAL ndet t w yindex from to
w=fields()
assign RUN
ndet=get("ndet")
from=FIRST;to=LAST
IF NOT(defined(from));from=3;ENDIF
IF NOT(defined(to));to=ndet ;ENDIF

dev1=device:open("xw")
w=get(as_integer(from):to)
printn w
yindex=dimensions(to-from+1)
fill yindex from*1. 1.
win_twod .1 .9 .1 .9 x=w.x y=yindex
t=colourtable:rainbow(80)
cell/draw values=w.y table=t
# axes/draw
ENDPROCEDURE
```

4.5 local.gcl

```
##############################################################################
```

```
#       Local extensions to OG for OSIRIS

#       Author:  D. Engberg September 1998
##############################################################################

####################################################################
PROCEDURE position
LOCAL i pos
pos=getcursor()
LOOP i FROM 1 TO 100
print "x-coordinate: "
printn pos.w_x
pos=getcursor(pos.w_x,pos.w_y)
IF pos.char="X"
 RETURN
ENDIF
ENDLOOP
ENDPROCEDURE

####################################################################
PROCEDURE dae
set/input "/.:/servers/osiris_dae"
ENDPROCEDURE

####################################################################
alias "oldassign" "assign"

PROCEDURE assign
PARAMETERS RUN
IF NOT(defined(RUN));RUN="dae";ENDIF

IF is_a(RUN,"integer")
 oldassign RUN
ELSE
 set/input "/.:/servers/osiris_dae"
ENDIF
ENDPROCEDURE


####################################################################
PROCEDURE ascii_import
PARAMETERS FIL=String
RESULT w
LOCAL handle

handle=asciifile:open(FIL)
w=asciifile:readfree(handle,"bin,x,y,e", $whitespace,2000)
```

```
asciifile/close handle
ENDPROCEDURE


######################################################################
PROCEDURE ascii_export
PARAMETERS W=Workspace FIL=String
LOCAL handle
handle=asciifile:open(FIL)
asciifile/writefree handle _  _ W.x W.y W.e
asciifile/close handle
ENDPROCEDURE


######################################################################
PROCEDURE ccslgen
PARAMETERS W=Workspace
LOCAL handle fil c1
fil=inquire("Enter file name")
units/t W
c1=max(W.y)
W.y=10.0*W.y/c1
W.e=10.0*W.e/c1
ascii_export W fil
ENDPROCEDURE


######################################################################

PROCEDURE red
alter/plotcolour $red
ENDPROCEDURE

PROCEDURE green
alter/plotcolour $green
ENDPROCEDURE

PROCEDURE yellow
alter/plotcolour $yellow
ENDPROCEDURE

PROCEDURE blue
alter/plotcolour $blue
ENDPROCEDURE

PROCEDURE white
alter/plotcolour $white
```

```
ENDPROCEDURE

######################################################################
PROCEDURE fonster
PARAMETERS new xmin xmax ymin ymax
RESULT dev1
LOCAL picture1

IF NOT device:status()
dev1=device:open("xw",6.5,4.5)
picture1=picture()
win_scaled 0.1 0.9 0.1 0.9 0. 20. 0. 0.02
ENDIF
ENDPROCEDURE

######################################################################
PROCEDURE iris

set/disk "iris$disk0:"
set/directory "[irsmgr.data]"
set/instrument "irs"
######################################################################

PROCEDURE correctL
QUALIFIERS /all
PARAMETERS run=Integer cfil=String
LOCAL last dL i w fil antal
LOCAL handle detdat

detdat=fields()
assign run
last=get("nsp1")
fil="osi"_string(run)&".caw"

IF all
handle=asciifile:open(cfil)
asciifile/skip handle 3
antal=asciifile:lines(handle)
detdat=asciifile:readfree(handle,"Det,delt,L2,code,twotheta",$whitespace,antal)
ENDIF

IF NOT all
dL=get(1,cfil)
w=s(1)
put/new w file=fil
w=s(2)
put w file=fil
```

```
LOOP i FROM 3 TO last
w=s(i)
w.l2=w.l2+dL[i]
put w file=fil
ENDLOOP
ENDIF

IF all
w=s(1)
w.l2=detdat.L2[1]
w.twotheta=detdat.twotheta[1]

put/new w file=fil
LOOP i FROM 3 TO last
w=s(i)
w.l2=detdat.L2[i]
w.twotheta=detdat.twotheta[i]
put w file=fil
ENDLOOP
ENDIF
ENDPROCEDURE
```

4.6 find_min.f

```
subroutine find_min(f1_get,f1_put)
implicit none
integer lpt, i, max_lpt, min
parameter (max_lpt=100000)
real xin(max_lpt), yin(max_lpt), ymin
EXTERNAL F1_GET, F1_PUT

lpt=max_lpt
CALL MODULE_GET_real_array(F1_GET,'x',xin,lpt)
CALL MODULE_GET_real_array(F1_GET,'y',yin,lpt)

ymin=yin(1)
do i=1,lpt
        if(yin(i).lt.ymin) then
          ymin=yin(i)
          min=i
        end if
enddo
CALL MODULE_PUT_INT(f1_put,'min',min)
RETURN
END
```

4.7 ochange.c

```
/* Change the user name, institute and RB number of all the lxx.tcb
   files in the [osiris.tcb] directory.
   As default l2 to l22 is changed, you can use a command line option
   but remember that under vms you have to define a symbol
   symb := $filename.exe, where the important part is the $
*/

#include <stdio.h>
#include <string.h>
int read_local(char local_contact[]);
int read_rb(int *rb);
int read_name(char name[]);
int read_inst(char inst[]);

main(int argc,char *argv[])
{
        FILE *infil, *tmp;
        char rad[500], local_contact[2],name[50], inst[20];
        char *fil, buffert[50][50];
        int i, j, rb, nr;

        if(argc==1){
                nr=13;
                strcpy(buffert[1],"osiris$disk0:[osiris.tcb]l2.tcb");
                strcpy(buffert[2],"osiris$disk0:[osiris.tcb]l4.tcb");
                strcpy(buffert[3],"osiris$disk0:[osiris.tcb]l6.tcb");
                strcpy(buffert[4],"osiris$disk0:[osiris.tcb]l8.tcb");
                strcpy(buffert[5],"osiris$disk0:[osiris.tcb]l10.tcb");
                strcpy(buffert[7],"osiris$disk0:[osiris.tcb]l12.tcb");
                strcpy(buffert[8],"osiris$disk0:[osiris.tcb]l14.tcb");
                strcpy(buffert[9],"osiris$disk0:[osiris.tcb]l16.tcb");
                strcpy(buffert[10],"osiris$disk0:[osiris.tcb]l18.tcb");
                strcpy(buffert[11],"osiris$disk0:[osiris.tcb]l20.tcb");
                strcpy(buffert[12],"osiris$disk0:[osiris.tcb]l22.tcb");
        }
        else{
                nr=argc;
                for(j=1;j<argc;j++){
                        strcpy(buffert[j],argv[j]);
                }
        }


        read_local(local_contact);
        read_rb();
        read_name(name);
        strcat(name,"/");
        strcat(name,local_contact);
        strcat(name,"\n");
        read_inst(inst);
        strcat(inst,"/ISIS\n");

        for(j=1;j<nr;j++){
                fil=buffert[j];
                infil=fopen(fil,"r");
                tmp=fopen("tmp.tmp","w");
                for(i=1;i<5;i++){
                        fgets(rad,500,infil);
                        fputs(rad,tmp);
                }
                fgets(rad,500,infil);
                fprintf(tmp,"%d\n",rb);
                fgets(rad,500,infil);
                fputs(rad,tmp);
                fgets(rad,500,infil);
                fputs(name,tmp);
                fgets(rad,500,infil);
                fputs(rad,tmp);
                fgets(rad,500,infil);
                fputs(inst,tmp);
                while(fgets(rad,500,infil) !=NULL){
                fputs(rad,tmp);
                }
                rename("tmp.tmp",fil);
                fclose(infil);
                fclose(tmp);
        }
}

read_local(char local_contact[])
{
        printf("Type the initials of your local contact: ");
        scanf("%s",local_contact);
}

read_rb(int *rb)
{
        printf("Type your RB number: ");
        scanf("%d",rb);
}

read_name(char name[])
```

```
{
        int i;
        printf("Type the initials of the experimental team (no blanks): ");
        scanf("%s",name);
}

read_inst(char inst[])
{
        printf("Your institute: ");
        scanf("%s",inst);
}
```

4.8 osd.gcl

```
###############################################################################
#       Corrections of the first OSIRIS data

#       Author:  D. Engberg November 1999
#################################################################################

# load compiled modules
CASE os()
IS "VMS"
   module/load "OSIRIS$DISK0:[OSIRIS.ALPHA.OG]find_min.so"
IS "LINUX"
   module/load "/home/dennis/code/isis/osiris/diffraction/find_min.so"
ENDCASE


# some procedures needed for the main program

# Rewriting the worksapce_append command to work, the original program tests on
# equlity before appending here you choose your "limit".
PROCEDURE workspace_append; PARAMETERS w1=workspace w2=workspace limit=real;
RESULT wres
LOCAL lenx1 lenx2 leny1 leny2
        lenx1=length(w1.x)
        lenx2=length(w2.x)
        leny1=length(w1.y)
        leny2=length(w2.y)
        IF (abs(max(w1.x)-min(w2.x))<limit)
         wres=w1
         wres.x=w1.x & w2.x[2:lenx2]
         wres.y=w1.y & w2.y
         wres.e=w1.e & w2.e
         wres.ntc=lenx1+lenx2
        ENDIF
```

```
ENDPROCEDURE


PROCEDURE find_min; PARAMETERS w1=workspace; RESULT wres
LOCAL min
        min=module:execute("find_min",w1)
        wres=min.min
ENDPROCEDURE


###################################################################
# Unwrap the monitor
###################################################################
PROCEDURE unwrap
RESULT unwrapped
PARAMETERS monitor=workspace
LOCAL wrap_point channel_width channel_start channel_range tmp
LOCAL last_pt

# The last point is always "wrong", this makes it "more" right
last_pt=monitor.ntc
monitor.y[last_pt]=monitor.y[last_pt-1]

wrap_point=monitor.x[find_min(monitor)]
#1.0 put in to make sure that channel_width is real
channel_width=1.0*(monitor.x[2]-monitor.x[1])/monitor.x[1]

channel_start=min(monitor.x)
channel_range=max(monitor.x)-min(monitor.x)

tmp=rebin:log(monitor,bound1=min(monitor.x),step1=channel_width,bound2=wrap_point)
monitor=rebin:log(monitor,bound1=wrap_point,step1=channel_width,bound2=max(monitor.x))
monitor.x=monitor.x-channel_range
unwrapped=workspace_append(monitor,tmp,channel_width*1.1)

ENDPROCEDURE


###################################################################
# Cut monitor to become of the same range as spectra in lamda and
# create a table that holds the values of the range
###################################################################
PROCEDURE klipp_table
RESULT w
PARAMETERS w1 table nr lowcut highcut manually
LOCAL width xxmin xxmax w11 w22
  w11=w1
  w22=s(3)
  width=(w11.x[2]-w11.x[1])/w11.x[1]

  units/lam w11
  units/lam w22
  IF min(w11.x)<min(w22.x)
    xxmin=min(w22.x)
  ELSE
    xxmin=min(w11.x)
  ENDIF
  IF (max(w11.x)<max(w22.x))
   xxmax=max(w11.x)
  ELSE
   xxmax=max(w22.x)
  ENDIF

  IF manually="y"
    printn "the choise is yes"
  ELSE
  #If cut is set to a valute 0<cut<100% then the spectra will
  #be cut of xxmin+lowcut and xxmax-highcut
  xxmin=xxmin+lowcut*(xxmax-xxmin)
  xxmax=xxmax-highcut*(xxmax-xxmin)
  ENDIF

  w11=rebin(w11,xmin=xxmin,xmax=xxmax)
  units/t w11
  w=w11
  table[1,nr]=xxmin
  table[2,nr]=xxmax
ENDPROCEDURE


###################################################################
# Cut spectra to become of the same range as monitor in lamda
###################################################################
PROCEDURE klipp
RESULT w
PARAMETERS w1 table i
LOCAL width xxmin xxmax w11 w22
  w11=w1
  width=(w11.x[2]-w11.x[1])/w11.x[1]
  units/lam w11
  IF min(w11.x)<table[1,i]
    xxmin=table[1,i]
  ELSE
    xxmin=min(w11.x)
  ENDIF
  IF max(w11.x)<table[2,i]
   xxmax=max(w11.x)
  ELSE
   xxmax= table[2,i]
  ENDIF
  w11=rebin(w11,xmin=xxmin,xmax=xxmax)
  units/t w11
  w=w11
ENDPROCEDURE



#######################################################################
# The routine that merges files
#######################################################################
PROCEDURE merge_spec
PARAMETERS low_detector=workspace high_detector=workspace
RESULT w
LOCAL min1 min2 tmp delta big size x y e
# Check that low_detector has the shorter times
min1=min(low_detector.x)
min2=min(high_detector.x)
IF min1>min2
        min1=min2
ENDIF
    delta=(low_detector.x[2]-low_detector.x[1])/low_detector.x[1]
    big=max(high_detector.x)
    IF big<max(low_detector.x)
        big=max(low_detector.x)
    ENDIF
    high_detector=rebin:log(high_detector,bound1=min1,step1=delta,bound2=big)
    low_detector=rebin:log(low_detector,bound1=min1,step1=delta,bound2=big)
    w=low_detector+high_detector
ENDPROCEDURE



#######################################################################
# The routine checks that variables for sum_raw exists and set them
# to default values if they do not.
#######################################################################
PROCEDURE check_sum_raw
PARAMETERS plowcut=real phighcut=real pmanually=string
GLOBAL lowcut highcut manually

IF NOT(defined(lowcut))
```

```
 lowcut=0.0
ELSE
 lowcut=plowcut
ENDIF
IF NOT(defined(phighcut))
 highcut=0.0
ELSE
 highcut=phighcut
ENDIF
IF NOT(defined(pmanually))
 manually="n"
ELSE
 manually=pmanually
ENDIF
ENDPROCEDURE



######################################################################
# The routine that sums raw files with diffeterent chopper setting
# into one file.
######################################################################
PROCEDURE  sum_raw
PARAMETERS FROM1=Integer TO1=Integer PLOWCUT=Real PHIGHCUT=Real PMANUALLY=string
LOCAL run last_spectra name i j min1 min2 big tmp x y e delta size
LOCAL high_mon high_detector low_mon low_detector detector
LOCAL tmp monitor
LOCAL table mon
GLOBAL lowcut highcut manually

tmp=check_sum_raw(PLOWCUT,PHIGHCUT,PMANUALLY)
table=dimensions(2,TO1-FROM1+1)


set/ext "raw"
assign FROM1
last_spectra=get("nsp1")
name="osi"_string(FROM1)&".raa"

# We start by summing all the monitor into one block. The monitor is cut
# so that it has no point out side the range of the spectra (in lambda).
# We save the monitor range in a table. Later the spectra is also cut,
# before added, not to cover a wider range  than the monitor.

assign run
low_mon=s(1)
monitor=unwrap(low_mon)
```

```
monitor=klipp_table(monitor,table,1,lowcut,highcut,manually)

LOOP run FROM FROM1+1 TO TO1
        i=TO1-run+2
        assign run
        high_mon=s(1)
        high_mon=klipp_table(high_mon,table,i,lowcut,highcut,manually)
        monitor=merge_spec(monitor,high_mon)
ENDLOOP
put/new monitor file=name

# We are now cutting all the spectra to the same range as their
# respective monitor and then merging them into one block
#changed from last_spectra to 21 to make it faster
printn "I am now at spectra:"
LOOP i FROM 2 TO last_spectra
print as_string(i)&" "
assign FROM1
        # Use only the part of the spectra where we have
        # monitor data and spectra
        low_detector=s(i)
        detector=klipp(low_detector,table,1)
        LOOP run FROM FROM1+1 TO TO1
                j=TO1-run+2
                assign run
                high_detector=s(i)
                high_detector=klipp(high_detector,table,j)
                detector=merge_spec(detector,high_detector)
        ENDLOOP
        put detector file=name
ENDLOOP
ENDPROCEDURE


###################################################################
# The routine that focuses and normalises to monitor
###################################################################
PROCEDURE focus_norm
PARAMETERS RUN_NUMBER=Integer FROMM=Integer TTO=Integer FIL=String
RESULT w

GLOBAL monitor
LOCAL w ww number run tmp wrap_point channel_width channel_start
LOCAL channel_range first_spec last_spec range
LOCAL factor

run=RUN_NUMBER
```

```
first_spec=FROMM
last_spec=TTO
range=last_spec-first_spec
ww=fields()

# Decide what sort of file we are goin to read
IF NOT(defined(FIL))
 set/file/input "osi"_string(RUN_NUMBER)&".raa"
ELSEIF as_variable(FIL)="raa"
  set/file/input "osi"_string(RUN_NUMBER)&".raa"
ELSEIF as_variable(FIL)="caw"
  set/file/input "osi"_string(RUN_NUMBER)&".caw"
ELSE
 set/ext FIL
 assign RUN_NUMBER
ENDIF

monitor=s(1)
monitor=unwrap(monitor)

# Correct for monitor efficiency. The engineering value 0.21 is
# adjusted from the measured values of silicon
units/lam monitor
factor=(1-exp(-(0.2066-0.1)*monitor.x))
monitor.y=monitor.y/factor
monitor.e=monitor.e/factor

# normalise all spectra to monitor (units wavelength)---------
ww=s((first_spec))
units/d ww
fill ww.y 0.
fill ww.e 0.

printn " "; printn "I am at detector: "

LOOP i FROM (first_spec) TO last_spec
 print i " "
 w=s(i)
 units/lam w
# Correction for detector efficency. The detectors are of
# the same material but of a different thickness and shape than the
# monitor. The exponent is therefore different.
 factor=(1-exp(-0.5596*w.x))
 w.y=w.y/factor
 w.e=w.e/factor
 w=rebin(w,monitor.x)/monitor
 units/d w
 w=rebin(w,ww.x)
 ww=ww+w
ENDLOOP
printn " "
 w=ww

w.spec_no=as_string(first_spec)&"-"_string(last_spec)
# Always focus to the average angle 161.02
w.twotheta=161.02
ENDPROCEDURE

######################################################################

# focus_norm_bank, uses focus_norm to focus all or high

######################################################################
PROCEDURE focus_norm_bank
PARAMETERS RUN_NUMBER=Integer BANK=String FIL=String
RESULT w
LOCAL w1 w2 ndet
IF NOT(defined(BANK)); BANK="all"; ENDIF
IF NOT(defined(FIL));  FIL="raw" ; ENDIF
assign RUN_NUMBER
ndet=get("NDET")

IF BANK="all"
 w=focus_norm(RUN_NUMBER,3,ndet,FIL)
 RETURN
ENDIF

IF BANK="high"
 w1=focus_norm(RUN_NUMBER,5,20,FIL)
 w2=focus_norm(RUN_NUMBER,125,140,FIL)
 w=w1+rebin(w2,w1.x)
 w.spec_no="high res"
 w.twotheta=172.8
ENDIF

ENDPROCEDURE

######################################################################

# The routine that splines the focused different diffraction spectra

######################################################################
PROCEDURE merge_diffractogram
PARAMETERS first=workspace last=workspace next1=workspace &
```

```
           next2=workspace next3=workspace next4=workspace &
           next5=workspace next6=workspace next7=workspace &
           next8=workspace next9=workspace next10=workspace
RESULT w
LOCAL low_limit high_limit a b b1 b2 c
LOCAL i continue picture1 dev1 tmp1 tmp2
LOCAL falt

falt=dimensions(10)
falt[1]=next1;falt[2]=next2;falt[3]=next3;falt[4]=next4;falt[5]=next5
falt[6]=next6;falt[7]=next7;falt[8]=next8;falt[9]=next9;falt[10]=next10

IF NOT device:status()
dev1=device:open("xw",6.5,4.5)
picture1=picture()
win_scaled 0.1 0.9 0.1 0.9 0. 1. 0. 1.
ENDIF

LOOP i FROM 1 TO 10
 tmp1=rebin(last,min(last.x),max(first.x))
 tmp2=1.2*(max(tmp1.y))
 display first min(last.x) max(first.x) ymax=tmp2
 plot last
 printn "High limit for low spectra"
 high_limit=getcursor()
 IF (low_limit.char="X")
  w=first
  RETURN
 ENDIF
 printn "Low limit for high spectra"
 low_limit=getcursor(low_limit.w_x,low_limit.w_y)
 a=rebin(first,min(first.x),low_limit.w_x)
 b1=rebin(first,low_limit.w_x,high_limit.w_x)
 b2=rebin(last,b1.x)
 c=rebin(last,high_limit.w_x,max(last.x))
 b=b1
 b.y=(b1.y+b2.y)*0.5
 b.e=sqrt(b1.e^2+b2.e^2)*0.5
 w=workspace_append(a,b,1.)
 w=workspace_append(w,c,1.)

 continue=defined(falt[i])
 IF continue
  last=falt[i]
  first=w
 ENDIF
 EXITIF NOT(continue)
```

```
ENDLOOP
ENDPROCEDURE
```