Bruker software support is available via phone, fax, e-mail, Internet, or ISDN. Please contact your local office, or directly:

Address: Bruker Analytik GmbH
Software Department
Silberstreifen
D-76287 Rheinstetten
Germany

Phone: +49 (721) 5161 440
Fax: +49 (721) 5161 480
E-mail: nmr-software-support@bruker.de
FTP: ftp.bruker.de / ftp.bruker.com
WWW: www.bruker.de / www.bruker.com
ISDN: on request
## Contents

### Chapter 1  
Introduction ................................................................. 5  
1.1 What are AU programs? ................................................. 5  
1.2 What is new in XWIN-NMR 3.1 ........................................ 5  
1.3 Quick reference to using AU programs .............................. 6  
1.4 Installing and compiling AU programs ............................ 7  
1.5 Executing AU programs ............................................... 8  
1.6 Viewing AU programs ................................................ 8  
1.7 About AU macros ................................................... 9  
1.8 About Bruker library functions ..................................... 9  
1.9 Creating your own AU programs .................................... 9  
1.10 How an AU program is translated into C-code .................. 16  
1.11 Listing of all predefined C statements ........................... 18  

### Chapter 2  
Inventory of AU macros and Bruker library functions .............. 23  
2.1 Naming conventions .................................................. 23  
2.2 Macros for dataset handling ....................................... 24  
2.3 Macros prompting the user for input ............................. 26  
2.4 Macros handling XWIN-NMR parameters .......................... 26  
2.5 Acquisition macros ................................................ 28  
2.6 Macros handling the shim unit and the sample changer ........ 29  
2.7 Macros handling the temperature unit ............................ 30  
2.8 Macros handling the MAS and HPCU unit ........................ 30  
2.9 1D processing macros .......................................... 31  
2.10 Peak picking, integration and miscellaneous macros ........ 32  
2.11 Macros for algebraic operations on datasets .................... 33  
2.12 Bayes, deconvolution and T1/T2 macros .......................... 34  
2.13 2D processing macros .......................................... 35  
2.14 Macros reading and writing projections etc. .................... 37  
2.15 3D processing macros .......................................... 39  
2.16 XWIN-NMR plotting macros ....................................... 39  
2.17 XWIN-PLOT related macros ...................................... 40  
2.18 Macros converting datasets from Aspect 2000/3000 and other vendors ..................  41  
2.19 Macros to execute other AU programs, XWIN-NMR macros or commands ..............  42  
2.20 Bruker library functions ......................................... 42  
2.21 Macros to return from an AU program ........................... 43
<table>
<thead>
<tr>
<th>Chapter 3</th>
<th>General AU macros</th>
<th>45</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 4</td>
<td>Macros changing the current AU dataset</td>
<td>51</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>Macros copying datasets</td>
<td>65</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>Macros handling rows/columns</td>
<td>69</td>
</tr>
<tr>
<td>Chapter 7</td>
<td>Macros converting datasets</td>
<td>81</td>
</tr>
<tr>
<td>Chapter 8</td>
<td>Macros handling XWIN-NMR parameters</td>
<td>89</td>
</tr>
<tr>
<td>Chapter 9</td>
<td>Macros for XWIN-Plot/autoplot</td>
<td>101</td>
</tr>
<tr>
<td>Chapter 10</td>
<td>Macros prompting the user for input</td>
<td>111</td>
</tr>
<tr>
<td>Chapter 11</td>
<td>Bruker library functions</td>
<td>117</td>
</tr>
<tr>
<td>Chapter 12</td>
<td>List of Bruker AU programs</td>
<td>143</td>
</tr>
<tr>
<td>Chapter 13</td>
<td>XWIN-NMR parameter types</td>
<td>159</td>
</tr>
<tr>
<td>13.1</td>
<td>Short description of all Bruker AU programs</td>
<td>143</td>
</tr>
<tr>
<td>13.1</td>
<td>Integer parameters</td>
<td>160</td>
</tr>
<tr>
<td>13.2</td>
<td>Float parameters</td>
<td>161</td>
</tr>
<tr>
<td>13.3</td>
<td>Double parameters</td>
<td>162</td>
</tr>
<tr>
<td>13.4</td>
<td>Character-string parameters</td>
<td>163</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 What are AU programs?

AU programs can be considered as user defined XWIN-NMR commands. Any repetitive task is most effectively accomplished through an AU program. All commands which can be entered on the XWIN-NMR command line can also be entered in an AU program in the form of macros. This includes selecting and changing datasets, reading and setting parameters, starting acquisitions, processing data and plotting the result. A simple AU program is nothing else than a sequence of such macros which execute the corresponding XWIN-NMR commands. However, AU programs may also contain C-language statements. In fact, an AU program is a C-program because all AU macros are translated to C-statements. XWIN-NMR automatically compiles AU programs to executable binaries, using a C-compiler.

XWIN-NMR offers two other ways of creating user defined commands: XWIN-NMR macros (not to be confused with AU macros) and Tcl/Tk scripts. They differ from AU programs in that they do not need to be compiled.

1.2 What is new in XWIN-NMR 3.1

Standard processing AU programs, like proc_1d contain the AUTOPLOT macro for plotting whereas in XWIN-NMR 3.0 and older, the PLOT macro was used. How-
ever, AU programs that contain PLOTX macros; are still used in the original form. The old AU programs, using PLOT macro, are still available under the names p_*.

For example, the standard 1D processing AU program is available as:

- proc_1d - contains the AUTOPL0T macro
- p_1d - contains the PLOT macro

ICON-NMR 3.1 automatically uses AU programs with the AUTOPL0T macro. Processing AU programs that contain the AUTOPL0T macro can be used with one of the options e, h or t. They cause AUTOPL0T to store the plot as a postscript file. For example, the AU program proc_1d can be enter as:

- proc_1d - prints to the printer defined in the layout
- proc_1d e - also prints to a postscript file in the dataset procno
- proc_1d h - also prints to a postscript file in the users home directory
- proc_1d t - also prints to a postscript file in the TEMP directory

(see also the header of the AU program plot_to_file)

ICON-NMR 3.1 can be configured to use the call the AU programs with the e option (Configuration → Automation Driver Engine → Master Switches → Generate Spectrum Print-Out ...). Note that if Datamail in the User Settings → User Manager is checked, the plot is not stored but sent as an Email.

The commands wsr and wsc take an extra argument, the experiment number. The corresponding AU macros must be specified as follows:

- WSR(row, procno, expno, name, user, disk)
- WSC(column, procno, expno, name, user, disk)

The new command rserr2d exists. The corresponding AU macro can be used as follows:

- RSER2D(direction, plane, expno, procno)

### 1.3 Quick reference to using AU programs

Bruker delivers a library of standard AU programs with XWIN-NMR. After XWIN-NMR has been installed you must do the following in order to use them:

1. Run expinstall once to install all AU programs
2. Run `compileall` once to compile all AU programs
3. Enter the name of an AU program to execute it

Furthermore, you can write your own AU programs in the following way:

1. Enter `edau <name>`
   The file `<name>` will be opened with a text editor

2. Do one of the following:
   • Write your own AU program from scratch
   • Read in an existing AU program and modify it according to your needs

3. Save the result and exit from the editor.
   Press `return` to answer the question about compilation.

4. Enter the name of the AU program to execute it.

After you have installed a new version of XWIN-NMR, you must run `expinstall` and `compileall` again to install and compile both Bruker’s and your own AU programs.

### 1.4 Installing and compiling AU programs

When you have installed a new version of XWIN-NMR, you must install the library AU programs once by executing the XWIN-NMR command `expinstall`. Your own AU programs which you created under a previous version of XWIN-NMR are still available, they only need to be re-compiled.

After running `expinstall`, there are 5 different commands to compile AU programs:

- `compileall`
  compile all library and user-defined AU programs

- `cplbruk <name>` or `cplbruk all`
  compile one or all Bruker AU programs

- `cpluser <name>` or `cpluser all`
  compile one or all user-defined AU programs

- `edau <name>`
  create (or view) and compile one AU program

- `xau <name>`
  compile and execute one AU program
1.5 Executing AU programs

Once an AU program has been installed, there are 3 different ways to execute it:

1. Enter the name of the AU program. This will work if:
   - The AU program is already compiled
   - No XWIN-NMR command or macro \(^1\) with the same name exist

2. Enter `xau au-program-name`

   If the AU program is not yet compiled, it will first be compiled and then executed. Otherwise, the program is immediately executed.

3. Enter `xau`

   A list of available AU programs will appear. Click on the AU program you want to execute. If it is not yet compiled, it will first be compiled and then executed. Otherwise, the AU program is immediately executed.

1.6 Viewing AU programs

You can view existing AU programs in the following way:

1. View the entire content of one AU program:
   a) Enter `edau`

      A list of all existing AU programs are displayed in two columns. The left column shows the user defined AU programs, the right column the Bruker library AU programs.

   b) Click on an AU program in the list

      When you select a Bruker AU program, it is shown in *view* mode which means you cannot edit it. When you click on a user-defined AU program it is shown in *edit* mode which means you can change it.

2. Enter `listall_au`

   A list and a short description of all library AU programs is stored in the file `listall` in the users home directory. Note that this list is also available in

---

\(^1\) Here we refer to an XWIN-NMR macro created with `edmac`
1.7 About AU macros

We will use the word *macro* rather often throughout this manual referring to AU macros. This should not be confused with XWIN-NMR macros which are files containing a sequence of XWIN-NMR commands. XWIN-NMR macros are created with `edmac` and executed with `xmac`. An AU macro, however, is a statement in an AU program which defines one or more XWIN-NMR commands, library functions or C-language statements. In its simplest form, an AU macro defines one XWIN-NMR command. For example the macros ZG and FT execute the XWIN-NMR commands `zg` and `ft`, respectively. Other macros like FETCHPAR and IEXPNO do not define XWIN-NMR commands, their function is only relevant in the context of an AU program. More complex macros may contain several XWIN-NMR commands and/or C-statements. All macros in AU programs should be written in capital letters. They are automatically translated to the corresponding C-code when the AU program is compiled. AU macros are defined in the file:

```
/xwhome/prog/include/aucmd.h
```

1.8 About Bruker library functions

Bruker library functions are C-functions which are contained in Bruker libraries. They offer several features which are also used in the XWIN-NMR interface, for example the display of a list of datasets from which the user can select one dataset. If you use a Bruker library function in an AU program the corresponding library is automatically linked to the AU program during compilation. The most important and versatile Bruker library functions are described in chapter 9.

1.9 Creating your own AU programs

1.9.1 Writing a simple AU program

Before you start writing an AU program, you might want to check if an AU program already exists which (almost) meets your requirements. If this is not the case, you can write your own AU program in the following way:
1. Enter `edau <au-name>`
   Your preferred XWIN-NMR text editor will be opened

2. Do one of the following:
   - Insert an existing library AU program and modify it to your needs.
   - Write a new AU program using the macros as described in this manual.
   The first macro in an AU program should always be GETCURDATA, the last macro should always be QUIT (or QUITMSG).

3. Save the file and exit from the editor.

4. A dialogue will appear asking you whether you want to compile the AU program, quit without compilation or go back to a listing of all AU programs:
   - Press `Return` to compile the AU program.

1.9.2 Using variables
Since AU programs are C programs you can use C-language variables. Several variables are already predefined for usage in AU programs. In fact, we distinguish three different types of variables: predefined dedicated variables, predefined general variables and user defined variables.

1.9.2.1 Predefined dedicated variables
Predefined dedicated variables have the following properties:
- they do not need to be declared in an AU program
- their declaration is automatically added during compilation
- they are known to the AU main body and to possible subroutines
- they are set implicitly by certain macros, e.g. the variable expno is set by macros like GETCURDATA, DATASET and IEXPNO
- they should not be set explicitly, so do **NOT** use statements like:
  ```c
  expno = 11;
  FETCHPAR("NS", &expno)
  ```
- they can be evaluated in macros or C-statements, e.g.:

1. You can change the XWIN-NMR text editor by entering the command `setres`. 
DATASET(name, expno, 2, disk, "guest")
   i1=expno+1;

   • examples of different types of predefined dedicated variables are:
     char-string: name, disk, user, name2
     integer: expno, procn0, loopcount1, loopcount2, lastparflag

A complete list of all predefined dedicated variables with their types can be found in Chapter 1.11.2

1.9.2.2 Predefined general variables

Predefined general variables have the following properties:

   • they do not need to be declared in an AU program
   • their declaration is automatically added during compilation
   • they are known to the AU main body but not to possible subroutines
   • they can be freely used for various purposes
   • examples of different types of predefined general variables are:
     integer: i1, i2, i3
     float: f1, f2, f3
     double: d1, d2, d3
     char-string: text

A complete list of all predefined general variables with their types and initial values can be found in Chapter 1.11.3.

1.9.2.3 User defined variables

For simple AU programs the number of predefined general variables is sufficient, you do not need to declare any additional variables. For more complex AU programs you might need more variables or you might want to use specific names. In these cases you can define your own variables in the AU program. User defined variables have the following properties:

   • they must be declared at the beginning of an AU program
   • they can be freely used for various purposes
   • they are known to the main AU program but not to possible subroutines
   • examples of declarations are:
int ivar1, ivar2;
float fvar1, fvar2, fvar3;
double dvar1, dvar2, dvar3;
char cstr1[20], cstr2[200];

1.9.3 Using AU macros with arguments

Several AU macros take one or more arguments. Arguments can be constants (values) or variables. In fact, an argument can be specified in four different ways as described here for the macro REXPNO:

- as a constant, e.g.:
  REXPNO(3)
- as a predefined dedicated variable e.g.:
  REXPNO(expno+1)
- as predefined general variable, e.g.:
  i1=6;
  REXPNO(i1)
- as a user defined variable, e.g.:
  int my_exp;
  ....
  my_exp=1;
  REXPNO(my_exp)

It is very important that the arguments are of the correct type. Macros can take arguments of the type integer (like REXPNO), float, double or character-string.

Some macros, for example STOREPAR, take XWIN-NMR parameters as arguments and each parameter is of a certain type. For example, the AU statement

STOREPAR("O1", d1)

stores the value of the variable d1 into the parameter O1. The predefined (double) variable d1 is used since O1 is of the type double. The second argument could also be a constant, e.g.:

STOREPAR("O1", 287.15)

A list of all XWIN-NMR parameters and their type can be found in Chapter 13.
1.9.4 Using C-language statements

AU programs can contain AU macros but also C-language statements like:

- define statements, e.g.: \#define MAXSIZE 32768
- include statements, e.g.: \#include <time.h>
- variable declarations, e.g. int ivar;
- variable assignments, e.g.: ivar = 20;
- loop structures, e.g.: for, while, do
- control structures, e.g.: if-else
- C-functions, e.g.: strcpy, strcmp, sprintf

Important: several C-language statements (including declarations of variables) are already predefined and automatically added during compilation of the AU program.

A example of an AU program using macros and C-statements is:

```c
int eno, pno;
char datapath [500], dataname[50], datauser[50], datadisk[200];

GETCURDATA
(void) strcpy (dataname,name);
(void) strcpy (datauser,user);
(void) strcpy (datadisk,disk);
eno = expno;
pno = procno;
(void) sprintf (datapath,"%s/data/%s/nmr/%s%d/pdata/%d/title",
datatext, datauser, dataname, eno, pno);
if ( (i1 = showfile (datapath)) < 0 )
{
    Proc_err (DEF_ERR_OPT,"Problems with showfile function");
}
QUIT
```

Note that GETCURDATA and QUIT are AU macros, strcpy and sprintf are C-functions and showfile and Proc_err are Bruker library functions.

For an explanation of C-functions and more information on C-language we refer to the literature on C-programming.
1.9.5 Additional hints on C-statements

If you are using C-language code in your AU programs, then there are a few things to be considered.

1. Using C-language header files

Several C-language header files are automatically added to your AU program during compilation. If you are using C-code which requires additional header files you must write your AU program in a special way. The main AU program should be a call to a subroutine which performs the actual task of the AU program. The include statements for the header file must be entered between the main AU program and the subroutine. This gives the following structure:

```c
GETCURDATA
subroutine(curdat, cmd)
QUIT
#include <headerfile.h>
subroutine(curdat, cmd)
char *curdat, *cmd;
{
    MACRO1
    MACRO2
}
or
subroutine(curdat, cmd)
QUIT
#include <headerfile.h>
subroutine(curdat, cmd)
char *curdat, *cmd;
{
    GETCURDATA
    MACRO1
    MACRO2
}
```

Such a structure is used in several Bruker library AU programs (e.g. amplstab, decon_t1, etc.). Several Bruker library functions like `PrintExpTime`, `gethighest`, `getxwinvers`, `pow_next` and `unlinkpr` also require an in-
clude statement in the AU program (see Chapter 11).

2. Some macros, e.g. IEXPNO and IPROCNO change the current AU dataset but
do not make it available for subsequent commands. If they are followed by a
CPR_exec or any C-statement which access the current AU dataset, then you
must precede that statement with SETCURDATA (see also the descriptions of
GETCURDATA, SETCURDATA, IEXPNO etc. in Chapter 4).

3. If you are using C-languages loop statements like for, do or while or control
statements like if, we strongly recommend to always put the body of such state-
ments between {}. If the body only contains simple macros like ZG or FT you
can omit them because these macro definitions already contain {}. However,
more complex macros might internally define C-statements that include loop or
control structures. If such a macro is used within a loop or control structure in
the AU program, then you create nested loops which require the usage of {}.

1.9.6 Viewing Bruker standard AU programs for macro syntax

The syntax of many AU macros is trivial, just enter the XWIN-NMR command in
capital letters. Other macros and especially Bruker library functions are more com-
plex. A detailed description of frequently used AU macros and functions can be
found in subsequent chapters of this manual. Alternatively, you can also look for
an existing AU program containing this macro or function. If, for example, you
want to know the syntax of the macro WRPA you can do the following:

On an UNIX workstation:
  • open a UNIX shell
  • cd /<xwhome>/prog/au/src.exam
  • grep -i wrpa *

where <xwhome> is the directory where XWIN-NMR is installed.

On a Windows PC:
  • Click Start -> Find -> Files or Folders
  • Click Browse and open C:\Bruker\xwin-nmr\prog\au\src.exam
  • Click Advanced, in the field Containing text enter wrpa
  • Click on Find now

assuming XWIN-NMR is installed in C:\Bruker.
1.10 How an AU program is translated into C-code

This paragraph is intended for users who want to get a deeper understanding of the compilation process. If you simply want to write and use AU programs you can skip this paragraph.

XWIN-NMR automatically translates your AU program into C-language and compiles it. Files and directories used during AU program compilation are:

- `<xwhome>/exp/stan/nmr/au/makeau`
- `<xwhome>/exp/stan/nmr/au/vorspann`
- `<xwhome>/exp/stan/nmr/au/mk_AUtable.exe`
- `<xwhome>/prog/include/aucmd.h`
- `<xwhome>/prog/include/inc`

The compilation process is entirely controlled by the script `makeau` which performs the following steps.

1. The file `vorspann` is concatenated with your AU program. This file contains a variety of definitions including
   - the C-program `main` statement
   - `#include` statements of C-header files (which in turn contain other definitions)
   - `#define` statements which define constants
   - predefined dedicated variables, e.g.: `name, disk, user, expno, procno`
   - predefined general variables, e.g.: `text, i1, i2, i3, f1, f2, f3, d1, d2, d3`

2. After `vorspann` and your code have been concatenated, a pre-processor program called `mk_AUtable.exe` scans the file for macro definitions and replaces them. The pre-processor searches for macro definitions in the file `aucmd.h` and in the `inc` directory. All AU macros are defined in the ascii file `aucmd.h`. Additional macros are defined in the files in the `inc` directory. In some cases, the name of the macro is the name of one of the files in `inc` directory and the entire content of the file represents that macro.

3. After `mk_AUtable.exe` has generated a C program source file, this file is compiled and an executable program is created. The compilation is done with the GNU C-compiler `gcc`. The linking process is done with the native linker which is part of the native C-compiler `cc`. All AU program’s source files reside in:
executables will be stored into:

`/<xwhome>/exp/stan/nmr/au/src`

`/<xwhome>/prog/au/bin`

The following section shows the result of concatenating `vorspann` with the following AU program:

```
GETCURDATA
EFP
APK
SREF
QUIT
```

For better presentation, only a part of `vorspann` is shown. All variables declared in `vorspann` are listed in chapter 1.10.

```c
#include <stdio.h>
#include <stdlib.h>
...........................
main(argc,argv)
int argc;
char **argv;
{
    char curdat[PATH_MAX];
    char arglist[BUFSIZ];
    int modret;
    modret = AU_program(curdat,arglist);
}
............................
AU_program(curdat,cmd)
char *curdat;
char *cmd;
{
    int i1=0,i2=0,i3=0;
    float f1=0,f2=0,f3=0,f998=0,f999=0;
    double d1=0,d2=0,d3=0;
    char text[BUFSIZ/2];
```
GETCURDATA
EFP
APK
SREF
QUIT

Note that the macro QUIT defines the closing C-language ‘}’ statement.

1.11 Listing of all predefined C statements

1.11.1 Including header files

The following C-language header files are automatically included during compilation:

```
stdio.h, stdlib.h, unistd.h, string.h, errno.h, math.h, limits.h, fcntl.h
```

which reside in the following directories:

under UNIX : /usr/include
under Windows: C:\Program Files\Microsoft Visual Studio\VC98\Include

and

```
errort.h, brukdef.h, lib/uni.h, lib/libcb.h, lib/util.h, sample.h, aucmd.h
```

which reside in the directory:

/xwhome/prog/include

Note that the latter group of header files is delivered with XWIN-NMR.

1.11.2 Predefined dedicated variables

The following list contains all predefined dedicated variables, their type and the AU macros by which they are set. Note that most variables are set or modified by several macros and only one or two are listed here.
<table>
<thead>
<tr>
<th>type</th>
<th>variable</th>
<th>set by macros</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>lastparflag</td>
<td>USELASTPARS, USECURPARS</td>
</tr>
<tr>
<td>int</td>
<td>loopcount1</td>
<td>TIMES/END</td>
</tr>
<tr>
<td>int</td>
<td>loopcount2</td>
<td>TIMES2/END</td>
</tr>
<tr>
<td>int</td>
<td>loopcount3</td>
<td>TIMES3/END</td>
</tr>
<tr>
<td>int</td>
<td>loopcountinf</td>
<td>TIMESINFINITE</td>
</tr>
<tr>
<td>char</td>
<td>disk[256]</td>
<td>GETCURDATA</td>
</tr>
<tr>
<td>char</td>
<td>user[64]</td>
<td>GETCURDATA</td>
</tr>
<tr>
<td>char</td>
<td>type[16]</td>
<td>GETCURDATA</td>
</tr>
<tr>
<td>char</td>
<td>name[64]</td>
<td>GETCURDATA</td>
</tr>
<tr>
<td>int</td>
<td>expno</td>
<td>GETCURDATA, IEXPNO</td>
</tr>
<tr>
<td>int</td>
<td>procono</td>
<td>GETCURDATA, IPROCNO</td>
</tr>
<tr>
<td>char</td>
<td>disk2[256]</td>
<td>GETCURDATA2</td>
</tr>
<tr>
<td>char</td>
<td>user2[64]</td>
<td>GETCURDATA2</td>
</tr>
<tr>
<td>char</td>
<td>type2[16]</td>
<td>GETCURDATA2</td>
</tr>
<tr>
<td>char</td>
<td>name2[64]</td>
<td>GETCURDATA2</td>
</tr>
<tr>
<td>int</td>
<td>expno2</td>
<td>GETCURDATA2</td>
</tr>
<tr>
<td>int</td>
<td>procono2</td>
<td>GETCURDATA2</td>
</tr>
<tr>
<td>char</td>
<td>disk3[256]</td>
<td>GETCURDATA3</td>
</tr>
<tr>
<td>char</td>
<td>user3[64]</td>
<td>GETCURDATA3</td>
</tr>
<tr>
<td>char</td>
<td>type3[16]</td>
<td>GETCURDATA3</td>
</tr>
<tr>
<td>char</td>
<td>name3[64]</td>
<td>GETCURDATA3</td>
</tr>
<tr>
<td>int</td>
<td>expno3</td>
<td>GETCURDATA3</td>
</tr>
<tr>
<td>int</td>
<td>procono3</td>
<td>GETCURDATA3</td>
</tr>
<tr>
<td>char</td>
<td>parsettype[10]</td>
<td>PARSETTYP, SETPARSET</td>
</tr>
<tr>
<td>char</td>
<td>namelist[10][64]</td>
<td>SETDATASET</td>
</tr>
<tr>
<td>char</td>
<td>dulist[10][256]</td>
<td>SETDATASET</td>
</tr>
</tbody>
</table>

Table 1.1
1.11.3 Predefined general variables

The following list contains all predefined general variables, their types and initial values:

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>userlist[10][64]</td>
<td>SETDATASET</td>
</tr>
<tr>
<td>char</td>
<td>parsetlist[10][16]</td>
<td>RPARSETLIST</td>
</tr>
<tr>
<td>char</td>
<td>pulproglst[10][16]</td>
<td>RPUPROGLIST</td>
</tr>
<tr>
<td>int</td>
<td>expnolist[15]</td>
<td>SETDATASET</td>
</tr>
<tr>
<td>int</td>
<td>procnolist[15]</td>
<td>SETDATASET</td>
</tr>
<tr>
<td>int</td>
<td>loopcountlist[15]</td>
<td>RLOOPCOUNTLIST</td>
</tr>
<tr>
<td>float</td>
<td>vtlist[128]</td>
<td>RVTLIST</td>
</tr>
<tr>
<td>int</td>
<td>xloopcount</td>
<td>ILOOPCOUNTLIST</td>
</tr>
<tr>
<td>int</td>
<td>xpulprog</td>
<td>IPULPROGLIST</td>
</tr>
<tr>
<td>int</td>
<td>xparset</td>
<td>IPARSETLIST</td>
</tr>
<tr>
<td>int</td>
<td>xdataset</td>
<td>IDATASETLIST</td>
</tr>
<tr>
<td>int</td>
<td>xvt</td>
<td>IVTLIST</td>
</tr>
<tr>
<td>int</td>
<td>listcount1</td>
<td>TIMESLIST</td>
</tr>
<tr>
<td>FILE *</td>
<td>textfilepointer</td>
<td></td>
</tr>
<tr>
<td>FILE *</td>
<td>debug</td>
<td></td>
</tr>
<tr>
<td>char</td>
<td>longpath[PATH_MAX]</td>
<td></td>
</tr>
<tr>
<td>char</td>
<td>Hilfs_string[BUFSIZ/2]</td>
<td></td>
</tr>
</tbody>
</table>

Table 1.1
values:

<table>
<thead>
<tr>
<th>type</th>
<th>variable</th>
<th>initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>i1</td>
<td>0</td>
</tr>
<tr>
<td>int</td>
<td>i2</td>
<td>0</td>
</tr>
<tr>
<td>int</td>
<td>i3</td>
<td>0</td>
</tr>
<tr>
<td>double</td>
<td>d1</td>
<td>0</td>
</tr>
<tr>
<td>double</td>
<td>d2</td>
<td>0</td>
</tr>
<tr>
<td>double</td>
<td>d3</td>
<td>0</td>
</tr>
<tr>
<td>float</td>
<td>f1</td>
<td>0</td>
</tr>
<tr>
<td>float</td>
<td>f2</td>
<td>0</td>
</tr>
<tr>
<td>float</td>
<td>f3</td>
<td>0</td>
</tr>
<tr>
<td>float</td>
<td>f998</td>
<td>0</td>
</tr>
<tr>
<td>float</td>
<td>f999</td>
<td>0</td>
</tr>
<tr>
<td>char</td>
<td>text[BUFSIZ/2]</td>
<td></td>
</tr>
</tbody>
</table>

*Table 1.2*
Chapter 2

Inventory of AU macros and Bruker library functions

2.1 Naming conventions

This chapter lists most AU macros and Bruker library functions that are available for AU programming. Simple macros with their short description are only mentioned in this chapter. More complex macros and AU functions are mentioned here and described more extensively in the following chapters. Table 2.1 explains the
Several AU macros that are described in this chapter require one or more arguments. These arguments can be constants or variables as described in Chapter 1.9.3. It is very important to use the correct type of argument in a macro call. The macros described in the tables of this chapter use the following arguments:

- integer: i1, i2, i3, eno, pno
- float: f1
- double: d1
- char-string: text, cmd, file, flag, mac, parm, parse, prog, shim, typ, dsk, usr, nam

Note that the arguments i1, i2, i3, f1, d1 and text have the same names as the corresponding predefined general variables. The predefined general variables are easy to use because they do not need to be declared. You can, however, use your own variables as macro arguments.

### 2.2 Macros for dataset handling

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GETCURDATA *</td>
<td>The first AU program statement; get the foreground dataset</td>
</tr>
<tr>
<td>SETCURDATA *</td>
<td>Make the current AU dataset available for subsequent AU statements</td>
</tr>
<tr>
<td>GETDATASET *</td>
<td>Prompt the user to specify a new dataset</td>
</tr>
<tr>
<td>Macro</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DATASET(....) *</td>
<td>Set the current AU dataset</td>
</tr>
<tr>
<td>DATASET2(....) *</td>
<td>Set the 2nd dataset (like the XWIN-NMR command edc2)</td>
</tr>
<tr>
<td>DATASET3(....) *</td>
<td>Set the 3rd dataset (like edc2)</td>
</tr>
<tr>
<td>GETCURDATA2</td>
<td>Read the 2nd dataset (like edc2)</td>
</tr>
<tr>
<td>GETCURDATA3</td>
<td>Read the 3rd dataset (like edc2)</td>
</tr>
<tr>
<td>DEXPNO *</td>
<td>Decrease the experiment number by one</td>
</tr>
<tr>
<td>IEXPNO *</td>
<td>Increase the experiment number by one</td>
</tr>
<tr>
<td>REXPNO(i1) *</td>
<td>Set the experiment number to the value of i1</td>
</tr>
<tr>
<td>DPROCNO *</td>
<td>Decrease the processing number by one</td>
</tr>
<tr>
<td>IPROCNO *</td>
<td>Increase the processing number by one</td>
</tr>
<tr>
<td>RPROCNO(i1) *</td>
<td>Set the processing number to the value of i1</td>
</tr>
<tr>
<td>GDATASETLIST</td>
<td>Prompt the user to enter a dataset list filename and read its contents</td>
</tr>
<tr>
<td>GLIST</td>
<td>Prompt the user to enter the dataset list filename and read its contents. In addition to the GDATASETLIST macro, GLIST also expects a pulse program and a parameter set name in the dataset list file.</td>
</tr>
<tr>
<td>DDATASETLIST</td>
<td>Decrement to the previous entry in the dataset list</td>
</tr>
<tr>
<td>IDATASETLIST</td>
<td>Increment to the next entry in the dataset list</td>
</tr>
<tr>
<td>RDATASETLIST(i1)</td>
<td>Read the dataset at position i1 of the dataset list and make it the current AU dataset</td>
</tr>
<tr>
<td>IFEODATASETLIST</td>
<td>Checks if the end of the dataset list is reached. The answer is true if there is no further entry.</td>
</tr>
<tr>
<td>SETDATASET</td>
<td>Set the current AU dataset to the one currently defined by the list file.</td>
</tr>
<tr>
<td>DU(dsk)</td>
<td>Set the disk unit to dsk</td>
</tr>
<tr>
<td>SETUSER(usr)</td>
<td>Set the user name to the user usr</td>
</tr>
<tr>
<td>WRA(i1) *</td>
<td>Copy the raw data to the experiment number i1</td>
</tr>
<tr>
<td>WRP(i1) *</td>
<td>Copy the processed data to the processing number i1</td>
</tr>
</tbody>
</table>
Inventory of AU macros and Bruker library functions

2.3 Macros prompting the user for input

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRPA(...) *</td>
<td>Copy the raw and processed data to the specified dataset</td>
</tr>
<tr>
<td>VIEWDATA *</td>
<td>Show the current AU program dataset in XWIN-NMR</td>
</tr>
</tbody>
</table>

2.4 Macros handling XWIN-NMR parameters

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GETDOUBLE(text,d1) *</td>
<td>Prompt the user to enter a double value</td>
</tr>
<tr>
<td>GETFLOAT(text,f1) *</td>
<td>Prompt the user to enter a float value</td>
</tr>
<tr>
<td>GETINT(text,i1) *</td>
<td>Prompt the user to enter an integer value</td>
</tr>
<tr>
<td>GETSTRING(text,nam) *</td>
<td>Prompt the user to enter a text string</td>
</tr>
<tr>
<td>GETPROSOL</td>
<td>Copy the probehead and solvent dependent parameters to the corresponding acquisition parameters</td>
</tr>
<tr>
<td>FETCHPAR(parm,&amp;val) *</td>
<td>Get an acquisition, processing or output parameter</td>
</tr>
<tr>
<td>FETCHPAR1(parm,&amp;val)</td>
<td>Get an F1 dimension parameter (2D acquisition/processing)</td>
</tr>
<tr>
<td>FETCHPAR3(parm,&amp;val)</td>
<td>Get an F1 dimension parameter (3D acquisition/processing)</td>
</tr>
<tr>
<td>FETCHPARS(parm,&amp;val) *</td>
<td>Get a status parameter (acquisition and processing)</td>
</tr>
<tr>
<td>FETCHPAR1S(parm,&amp;val)</td>
<td>Get an F1 dimension status parameter (2D)</td>
</tr>
<tr>
<td>FETCHPAR3S(parm,&amp;val)</td>
<td>Get an F1 dimension status parameter (3D)</td>
</tr>
<tr>
<td>STOREPAR(parm,val) *</td>
<td>Store an acquisition, processing or output parameter</td>
</tr>
<tr>
<td>STOREPAR1(parm,val)</td>
<td>Store an F1 dimension parameter (2D)</td>
</tr>
<tr>
<td>STOREPAR3(parm,val)</td>
<td>Store an F1 dimension parameter (3D)</td>
</tr>
<tr>
<td>STOREPARS(parm,val) *</td>
<td>Store a status parameter (acquisition and processing)</td>
</tr>
<tr>
<td>STOREPAR1S(parm,val)</td>
<td>Store an F1 dimension status parameter (2D)</td>
</tr>
<tr>
<td>Macro</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STOREPAR3S(parm, val)</td>
<td>Store an F1 dimension status parameter (3D)</td>
</tr>
<tr>
<td>FETCHPARM(parm, &amp;val)</td>
<td>Get a tomography measurement parameter</td>
</tr>
<tr>
<td>STOREPARM(parm, val)</td>
<td>Store a tomography measurement parameter</td>
</tr>
<tr>
<td>FETCHPLPAR(parm, &amp;val)</td>
<td>Get a plot parameter</td>
</tr>
<tr>
<td>STOREPLPAR(parm, val)</td>
<td>Store a plot parameter</td>
</tr>
<tr>
<td>FETCHPLWPAR(parm, &amp;val)</td>
<td>Get a white-washed stacked plot parameter</td>
</tr>
<tr>
<td>STOREPLWPAR(parm, val)</td>
<td>Store a white-washed stacked plot parameter</td>
</tr>
<tr>
<td>FETCHPLXPAR(parm, &amp;val)</td>
<td>Get an expansion plot parameter</td>
</tr>
<tr>
<td>STOREPLXPAR(parm, val)</td>
<td>Store an expansion plot parameter</td>
</tr>
<tr>
<td>FETCHT1PAR(parm, val)</td>
<td>Get a T1 parameter</td>
</tr>
<tr>
<td>STORET1PAR(parm, val)</td>
<td>Store a T1 parameter</td>
</tr>
<tr>
<td>FETCHDOSYPAR(parm, &amp;val)</td>
<td>Get a dosy (eddosy) parameter</td>
</tr>
<tr>
<td>STOREDOSYPAR(parm, val)</td>
<td>Store a dosy (eddosy) parameter</td>
</tr>
<tr>
<td>RPAR(parset, typ) *</td>
<td>Read a parameter set to the current dataset</td>
</tr>
<tr>
<td>WPAR(parset, typ) *</td>
<td>Write the current dataset parameters to a parameter set</td>
</tr>
<tr>
<td>DELPAR(parset)</td>
<td>Delete the parameter set parset</td>
</tr>
<tr>
<td>GPARSETLIST</td>
<td>Prompt the user to enter the name of the parameter set list and read its contents</td>
</tr>
<tr>
<td>DPARSETLIST</td>
<td>Decrement to the previous parameter set name in the parameter set list</td>
</tr>
<tr>
<td>IPARSETLIST</td>
<td>Increment to the next parameter set name in the parameter set list</td>
</tr>
<tr>
<td>PARSERTTYP(typ)</td>
<td>Set the type of parameter file (acqu, proc, outd or plot) to be read by SETPARSET or RPARSETLIST</td>
</tr>
<tr>
<td>RPARSETLIST(i1)</td>
<td>Read the parameter set name from position i1 in the parameter set list and then read the parameters from this parameter set</td>
</tr>
</tbody>
</table>
### 2.5 Acquisition macros

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETPARSET</td>
<td>Read the parameters from the parameter set name in position i1 of the parameter list</td>
</tr>
<tr>
<td>IFEOPARSETLIST</td>
<td>Checks if the end of the parameter set list is reached. The answer is true if there is no further entry.</td>
</tr>
<tr>
<td>ZG</td>
<td>Start acquisition; if raw data already exist, they are overwritten</td>
</tr>
<tr>
<td>GO</td>
<td>Continue the acquisition on already existing raw data by adding to them</td>
</tr>
<tr>
<td>II</td>
<td>Initialize acquisition interface</td>
</tr>
<tr>
<td>RGA</td>
<td>Automatic receiver gain adjustment</td>
</tr>
<tr>
<td>MAKE_ZERO_FID</td>
<td>Create an empty FID</td>
</tr>
<tr>
<td>DEG90</td>
<td>Determine 90 degree pulse automatically</td>
</tr>
<tr>
<td>GPULPROGLIST</td>
<td>Prompt the user to enter the name of a pulse program list file and read its contents</td>
</tr>
<tr>
<td>DPULPROGLIST</td>
<td>Decrement to the previous name in the pulse program list</td>
</tr>
<tr>
<td>IPULPROGLIST</td>
<td>Increment to the next name in the pulse program list</td>
</tr>
<tr>
<td>RPULPROGLIST(i1)</td>
<td>Read the pulse program name in position i1 of the pulse program list and write it to the acquisition parameters</td>
</tr>
<tr>
<td>SETPULPROG</td>
<td>Store the current pulse program name from the pulse program list</td>
</tr>
<tr>
<td>IFEOPULPROGLIST</td>
<td>Check if the end of the pulse program list is reached. The answer is true if there is no further entry.</td>
</tr>
</tbody>
</table>
2.6 Macros handling the shim unit and the sample changer

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJ</td>
<td>Eject sample from the magnet</td>
</tr>
<tr>
<td>IJ</td>
<td>Insert sample into the magnet</td>
</tr>
<tr>
<td>ROT</td>
<td>Turn rotation on (use value RO from acquisition parameters)</td>
</tr>
<tr>
<td>ROTOFF</td>
<td>Turn rotation off and wait until rotation was turned off</td>
</tr>
<tr>
<td>LOPO</td>
<td>Set the lock parameters (lock power, lock gain, loop filter, loop time and loop gain)</td>
</tr>
<tr>
<td>LFILTER(i1)</td>
<td>Set the loop filter to the value of i1</td>
</tr>
<tr>
<td>LG</td>
<td>Auto-adjust the lock gain</td>
</tr>
<tr>
<td>LGAIN(f1)</td>
<td>Set the loop gain to the value of f1</td>
</tr>
<tr>
<td>LO(f1)</td>
<td>Set the lock power to the value of f1</td>
</tr>
<tr>
<td>LTIME(f1)</td>
<td>Set the loop time to the value of f1</td>
</tr>
<tr>
<td>LOCK</td>
<td>Lock according to the parameters LOCNUC and SOLVENT using the lock parameters from the edlock table</td>
</tr>
<tr>
<td>RSH(file)</td>
<td>Read the shim values from the specified file</td>
</tr>
<tr>
<td>SETSH(shim,i1)</td>
<td>Set one shim to the value of i1</td>
</tr>
<tr>
<td>WSH(file)</td>
<td>Write the shim values to the specified file</td>
</tr>
<tr>
<td>TUNE(file)</td>
<td>Start autoshrimming with the specified tune file</td>
</tr>
<tr>
<td>TUNESX</td>
<td>Start autoshimming with the tune file defined by the currently defined probehead and solvent</td>
</tr>
<tr>
<td>ZSPOIL(i1)</td>
<td>Set value for Z-spoil gradient pulse from BSMS</td>
</tr>
</tbody>
</table>
2.7 Macros handling the temperature unit

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TESET</td>
<td>Set the temperature on the temperature unit to the value of the acquisition parameter TE.</td>
</tr>
<tr>
<td>TEGET</td>
<td>Get the temperature from the temperature unit and store it in the acquisition status parameter TE.</td>
</tr>
<tr>
<td>TE2SET</td>
<td>Set the temperature on the second regulator of the temperature unit to the value of the acquisition parameter TE2.</td>
</tr>
<tr>
<td>TE2GET</td>
<td>Get the temperature from the second regulator of the temperature unit and store it in the acquisition status parameter TE2.</td>
</tr>
<tr>
<td>TEREADY(i1,f1)</td>
<td>After the temperature is set, wait until it is accurate to f1 degrees for at least 10 sec., then wait i1 seconds for stabilization.</td>
</tr>
<tr>
<td>TE2READY(i1,f1)</td>
<td>After the second temperature is set, wait until it is accurate to f1 degrees for at least 10 sec., then wait i1 seconds for stabilization.</td>
</tr>
<tr>
<td>TEPAR(file)</td>
<td>Read a file with parameter settings for the temperature unit</td>
</tr>
<tr>
<td>GVTLIST</td>
<td>Prompt the user to enter the variable temperature list name and read its contents</td>
</tr>
<tr>
<td>RVTLIST</td>
<td>Read the contents of the variable temperature list file defined by the acquisition parameter VTLIST.</td>
</tr>
<tr>
<td>DVTLIST</td>
<td>Decrement to the previous value in the vtlist</td>
</tr>
<tr>
<td>IVTLIST</td>
<td>Increment to the next value in the vtlist</td>
</tr>
<tr>
<td>VT</td>
<td>Read and set the temperature according to the current value of the vtlist</td>
</tr>
</tbody>
</table>

2.8 Macros handling the MAS and HPCU unit

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASE</td>
<td>Eject sample from MAS unit</td>
</tr>
<tr>
<td>MASI</td>
<td>Insert sample into MAS unit</td>
</tr>
<tr>
<td>MASR</td>
<td>Set spinning rate according to the acquisition parameter MASR</td>
</tr>
</tbody>
</table>
### 2.9 1D processing macros

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>Automatic baseline correction (creates intrng file !)</td>
</tr>
<tr>
<td>ABSD</td>
<td>Automatic baseline correction with DISNMR algorithm (creates intrng file !)</td>
</tr>
<tr>
<td>ABSF</td>
<td>Automatic baseline correction between limits ABSF1 and ABSF2</td>
</tr>
<tr>
<td>APK</td>
<td>Automatic phase correction</td>
</tr>
<tr>
<td>APK0</td>
<td>Zero order automatic phase correction</td>
</tr>
<tr>
<td>APK1</td>
<td>First order automatic phase correction</td>
</tr>
<tr>
<td>APKF</td>
<td>Automatic phase correction using the spectral region determined by ABSF2 and ABSF1 for the calculation of the phase values.</td>
</tr>
<tr>
<td>APKS</td>
<td>Automatic phase correction especially suitable for polymer spectra</td>
</tr>
<tr>
<td>BC</td>
<td>Baseline correction of FID (DC correction)</td>
</tr>
<tr>
<td>CONVDTA</td>
<td>Convert digitally filtered FID into analogue (conventional) form</td>
</tr>
<tr>
<td>EF</td>
<td>Exponential window multiplication + Fourier transform</td>
</tr>
<tr>
<td>EFP</td>
<td>Exponential window multiplication + Fourier transform + phase correction using the processing parameters PHC0 and PHC1</td>
</tr>
<tr>
<td>EM</td>
<td>Exponential window multiplication of FID</td>
</tr>
<tr>
<td>FMC</td>
<td>Fourier Transform + magnitude calculation</td>
</tr>
</tbody>
</table>
### 2.10 Peak picking, integration and miscellaneous macros

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP</td>
<td>Peak picking according to currently set processing parameters</td>
</tr>
<tr>
<td>PPH</td>
<td>Like PP, but with a peak histogram along the listing</td>
</tr>
<tr>
<td>PPP</td>
<td>Like PP, but the output is written to the file peaklist in the current processing data directory (PROCNO)</td>
</tr>
</tbody>
</table>
### 2.11 Macros for algebraic operations on datasets

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPJ</td>
<td>Like PP, but store peaks in JCAMP-DX format</td>
</tr>
<tr>
<td>LI</td>
<td>List integrals according to the currently defined intrng file. The macro ABS can be used to create an intrng file.</td>
</tr>
<tr>
<td>LIPP</td>
<td>List integrals and all peaks in the integral ranges</td>
</tr>
<tr>
<td>LIPPF</td>
<td>Like LIPP, but works always on the full spectrum</td>
</tr>
<tr>
<td>NMRQUANT(file)</td>
<td>Do quantitative integration with an nmrquant region file</td>
</tr>
<tr>
<td>RLUT(file)</td>
<td>Read color look-up table for 2D or 3D datasets</td>
</tr>
<tr>
<td>RMISC(typ,file)</td>
<td>Read a file from one of the following list types: base_info, baslpnts, intrng, peaklist or reg</td>
</tr>
<tr>
<td>WMISC(typ,file)</td>
<td>Write a base_info, baslpnts, intrng, peaklist or reg file to its lists directory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Add 2nd and 3rd dataset and put the result into the current dataset. The 3rd dataset is multiplied by DC.</td>
</tr>
<tr>
<td>ADDC</td>
<td>Add the constant DC to the current dataset</td>
</tr>
<tr>
<td>AND</td>
<td>Put logical &quot;and&quot; of 2nd and 3rd dataset into the current dataset</td>
</tr>
<tr>
<td>AT</td>
<td>Same as ADD</td>
</tr>
<tr>
<td>CMPL</td>
<td>Put the &quot;complement&quot; of the 2nd and 3rd dataset into the current dataset</td>
</tr>
<tr>
<td>DIV</td>
<td>Divide 2nd and 3rd dataset and put the result into the current dataset. The 3rd dataset is multiplied by DC.</td>
</tr>
<tr>
<td>DT</td>
<td>Calculate the first derivative of the dataset</td>
</tr>
<tr>
<td>FILT</td>
<td>Apply a software digital filter to the current dataset</td>
</tr>
<tr>
<td>LS</td>
<td>Left shift spectrum or FID by NSP points</td>
</tr>
<tr>
<td>MUL</td>
<td>Multiply 2nd and 3rd dataset and put the result into the current dataset. The 3rd dataset is multiplied by DC.</td>
</tr>
<tr>
<td>MULC</td>
<td>Multiply the current dataset with DC</td>
</tr>
<tr>
<td>NM</td>
<td>Negate current spectrum</td>
</tr>
</tbody>
</table>
### 2.12 Bayes, deconvolution and T1/T2 macros

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS</td>
<td>Right shift spectrum or FID by NSP points</td>
</tr>
<tr>
<td>RV</td>
<td>Reverse the spectrum</td>
</tr>
<tr>
<td>ZF</td>
<td>Zero the spectrum (1r,1i)</td>
</tr>
<tr>
<td>ZP</td>
<td>Zero the first NZP points of the spectrum or FID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAYX</td>
<td>Execute Bayesian calculation</td>
</tr>
<tr>
<td>BAYED</td>
<td>Edit the Bayes parameters</td>
</tr>
<tr>
<td>BAYEX</td>
<td>Edit the Bayes parameters and then run the calculation</td>
</tr>
<tr>
<td>LIBAY</td>
<td>List the result of the Bayesian calculation</td>
</tr>
<tr>
<td>VIBAY</td>
<td>Display the result of the Bayesian calculation</td>
</tr>
<tr>
<td>GDCON</td>
<td>Gaussian deconvolution of the peaks automatically picked according to the currently set processing parameters</td>
</tr>
<tr>
<td>LDCON</td>
<td>Lorentzian deconvolution of the peaks automatically picked according to the currently set processing parameters</td>
</tr>
<tr>
<td>MDON</td>
<td>Mixed Gaussian/Lorentzian deconvolution of the peaks in the peaklist file. The peaklist file can be created with the <code>ppp</code> command and it can be modified using the <code>edmisc</code> command.</td>
</tr>
<tr>
<td>CT1</td>
<td>Calculate the T1 value of the current peak</td>
</tr>
<tr>
<td>DAT1</td>
<td>Calculate T1 value for all peaks picked with PD</td>
</tr>
<tr>
<td>DAT2</td>
<td>Calculate T2 value for all peaks picked with PD</td>
</tr>
<tr>
<td>PD</td>
<td>Pick data points for relaxation analysis</td>
</tr>
<tr>
<td>PD0</td>
<td>Pick data points for relaxation analysis at constant peak position</td>
</tr>
<tr>
<td>PF</td>
<td>Peak pick points from a series of FIDs in a 2D ser file</td>
</tr>
<tr>
<td>PFT2</td>
<td>Peak pick points from a single FID</td>
</tr>
<tr>
<td>SIMFIT</td>
<td>Simplex fit; multiple component relaxation analysis of one peak</td>
</tr>
</tbody>
</table>
## 2.13 2D processing macros

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS1</td>
<td>Baseline correction in F1 dimension</td>
</tr>
<tr>
<td>ABS2</td>
<td>Baseline correction in F2 dimension</td>
</tr>
<tr>
<td>ABS1D</td>
<td>Baseline correction in F1 dimension using the DISNMR algorithm</td>
</tr>
<tr>
<td>ABS2D</td>
<td>Baseline correction in F2 dimension using the DISNMR algorithm</td>
</tr>
<tr>
<td>ABSOT1</td>
<td>Trapezoidal baseline correction in F1 dimension using a slightly different algorithm than abst1</td>
</tr>
<tr>
<td>ABSOT2</td>
<td>Trapezoidal baseline correction in F2 dimension using a slightly different algorithm than abst2</td>
</tr>
<tr>
<td>ABST1</td>
<td>Trapezoidal baseline correction in F1 dimension using the processing parameters ABSF1, ABSF2, SIGF1, SIGF2</td>
</tr>
<tr>
<td>ABST2</td>
<td>Trapezoidal baseline correction in F2 dimension using the processing parameters ABSF1, ABSF2, SIGF1, SIGF2</td>
</tr>
<tr>
<td>ADD2D</td>
<td>Add the 2nd dataset to the current dataset.</td>
</tr>
<tr>
<td>BCM1</td>
<td>Baseline correction of all columns using the coefficients that were obtained with a manual 1D baseline correction</td>
</tr>
<tr>
<td>BCM2</td>
<td>Baseline correction of all rows using the coefficients that were obtained with a manual 1D baseline correction</td>
</tr>
<tr>
<td>INVSF</td>
<td>Interchange the frequencies of the two dimensions</td>
</tr>
<tr>
<td>LEVCALC</td>
<td>Calculate the levels for the contour representation of the 2D matrix</td>
</tr>
<tr>
<td>PTILT</td>
<td>Tilt the 2D matrix by an arbitrary angle</td>
</tr>
<tr>
<td>PTILT1</td>
<td>Tilt the 2D matrix along its central vertical line</td>
</tr>
<tr>
<td>REV1</td>
<td>Reverse the spectrum in F1 dimension</td>
</tr>
<tr>
<td>REV2</td>
<td>Reverse the spectrum in F2 dimension</td>
</tr>
</tbody>
</table>
## Inventory of AU macros and Bruker library functions

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB1</td>
<td>Subtract 1D spectrum in F1 dimension (no change in sign !)</td>
</tr>
<tr>
<td>SUB2</td>
<td>Subtract 1D spectrum in F2 dimension (no change in sign !)</td>
</tr>
<tr>
<td>SUB1D1</td>
<td>Subtract 1D spectrum in F1 dimension</td>
</tr>
<tr>
<td>SUB1D2</td>
<td>Subtract 1D spectrum in F2 dimension</td>
</tr>
<tr>
<td>SYM</td>
<td>Symmetrize COSY spectrum</td>
</tr>
<tr>
<td>SYMA</td>
<td>Symmetrize phase sensitive COSY spectrum</td>
</tr>
<tr>
<td>SYMJ</td>
<td>Symmetrize J-resolved spectrum</td>
</tr>
<tr>
<td>TILT</td>
<td>Tilt J-resolved spectrum by an internally calculated angle</td>
</tr>
<tr>
<td>XF1</td>
<td>Fourier transform in F1 dimension</td>
</tr>
<tr>
<td>XF1P</td>
<td>Phase correction in F1 dimension using the processing parameters PHC0 and PHC1</td>
</tr>
<tr>
<td>XF2</td>
<td>Fourier transform in F2 dimension</td>
</tr>
<tr>
<td>XF2P</td>
<td>Phase correction in F2 dimension using the processing parameters PHC0 and PHC1</td>
</tr>
<tr>
<td>XFB</td>
<td>Fourier transform in both dimensions</td>
</tr>
<tr>
<td>XFBP</td>
<td>Phase correction in both dimensions</td>
</tr>
<tr>
<td>XF1M</td>
<td>Magnitude calculation in F1 dimension</td>
</tr>
<tr>
<td>XF2M</td>
<td>Magnitude calculation in F2 dimension</td>
</tr>
<tr>
<td>XFBM</td>
<td>Magnitude calculation in both dimensions</td>
</tr>
<tr>
<td>XF1PS</td>
<td>Power spectrum in F1 dimension</td>
</tr>
<tr>
<td>XF2PS</td>
<td>Power spectrum in F2 dimension</td>
</tr>
<tr>
<td>XFBPS</td>
<td>Power spectrum in both dimensions</td>
</tr>
<tr>
<td>XHT1</td>
<td>Hilbert Transform in F1 dimension</td>
</tr>
<tr>
<td>XHT2</td>
<td>Hilbert transform in F2 dimension</td>
</tr>
<tr>
<td>XIF1</td>
<td>Inverse Fourier transform in F1 dimension</td>
</tr>
<tr>
<td>XIF2</td>
<td>Inverse Fourier transform in F2 dimension</td>
</tr>
<tr>
<td>XTRF</td>
<td>2D processing according to processing parameter flags (starts always on the raw data !)</td>
</tr>
</tbody>
</table>
### 2.14 Macros reading and writing projections etc.

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1SUM(i1,i2,pno)</td>
<td>Read sum of columns from i1 to i2 into the 1D processing number pno</td>
</tr>
<tr>
<td>F2SUM(i1,i2,pno)</td>
<td>Read sum of rows from i1 to i2 into the 1D processing number pno</td>
</tr>
<tr>
<td>F1DISCO(i1,i2,i3,pno)</td>
<td>Read disco projection between i1 and i2 columns with reference row i3 into the 1D processing number pno</td>
</tr>
<tr>
<td>F2DISCO(i1,i2,i3,pno)</td>
<td>Read disco projection between i1 and i2 rows with reference column i3 into the 1D processing number pno</td>
</tr>
<tr>
<td>F1PROJN(i1,i2,pno)</td>
<td>Read partial negative projection between columns i1 and i2 into the 1D processing number pno</td>
</tr>
<tr>
<td>F1PROJP(i1,i2,pno)</td>
<td>Read partial positive projection between columns i1 and i2 into the 1D processing number pno</td>
</tr>
<tr>
<td>F2PROJN(i1,i2,pno)</td>
<td>Read partial negative projection between rows i1 and i2 into the 1D processing number pno</td>
</tr>
<tr>
<td>Macro</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>F2PROJP(i1,i2,pno)</td>
<td>Read partial positive projection between rows i1 and i2 into the 1D processing number pno</td>
</tr>
<tr>
<td>PROJ</td>
<td>Calculate the projections of the 2D spectrum</td>
</tr>
<tr>
<td>RHNPP(pno)</td>
<td>Read horizontal (F2) negative projection into the 1D processing number pno</td>
</tr>
<tr>
<td>RHPPP(pno)</td>
<td>Read horizontal (F2) positive projection into the 1D processing number pno</td>
</tr>
<tr>
<td>RSC(i1,pno) *</td>
<td>Read column i1 of 2D into the 1D processing number pno</td>
</tr>
<tr>
<td>RSR(i1,pno) *</td>
<td>Read row i1 of 2D into the 1D processing number pno</td>
</tr>
<tr>
<td>RVNP(pno) *</td>
<td>Read vertical (F1) negative projection into the 1D processing number pno</td>
</tr>
<tr>
<td>RVPPP(pno) *</td>
<td>Read vertical (F1) positive projection into the 1D processing number pno</td>
</tr>
<tr>
<td>RSER(i1,eno,pno) *</td>
<td>Read row i1 of 2D raw data into the eno and pno</td>
</tr>
<tr>
<td>RSER2D(direc, i1,eno, pno) *</td>
<td>Read plane number i1 in direction direc of 3D raw data into the eno and pno</td>
</tr>
<tr>
<td>WSC(i1,pno,eno,nam,usr,dsk) *</td>
<td>Write a column back into position i1 of a 2D raw dataset defined by pno, eno, nam, usr and dsk</td>
</tr>
<tr>
<td>WSR(i1,pno,eno,nam,usr,dsk) *</td>
<td>Write a row back into position i1 of a 2D dataset defined by pno, eno, nam, usr and dsk.</td>
</tr>
<tr>
<td>WSER(i1,nam,eno,pno,dsk,usr) *</td>
<td>Write an FID back into position i1 of a 2D raw data defined by eno, pno, nam, dsk and usr.</td>
</tr>
<tr>
<td>WSERP(i1,nam,eno,pno,dsk,usr) *</td>
<td>Write a processed FID back into position i1 of a 2D raw data defined by eno, pno, nam, dsk and usr.</td>
</tr>
</tbody>
</table>
2.15 3D processing macros

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF3(flag,dsk)</td>
<td>Fourier transform in F3 dimension. The flag can be &quot;y&quot; or &quot;n&quot; and determines whether the imaginary parts are stored or not. The processed are stored on disk unit dsk.</td>
</tr>
<tr>
<td>TF2(flag)</td>
<td>Fourier transform in F2 dimension (flag as in TF3)</td>
</tr>
<tr>
<td>TF1(flag)</td>
<td>Fourier transform in F1 dimension (flag as in TF3)</td>
</tr>
<tr>
<td>TF3P(flag)</td>
<td>Phase correction in F3 dimension (flag as in TF3)</td>
</tr>
<tr>
<td>TF2P(flag)</td>
<td>Phase correction in F2 dimension (flag as in TF3)</td>
</tr>
<tr>
<td>TF1P(flag)</td>
<td>Phase correction in F1 dimension (flag as in TF3)</td>
</tr>
<tr>
<td>TABS3</td>
<td>Automatic baseline correction in F3 dimension</td>
</tr>
<tr>
<td>TABS2</td>
<td>Automatic baseline correction in F2 dimension</td>
</tr>
<tr>
<td>TABS1</td>
<td>Automatic baseline correction in F1 dimension</td>
</tr>
<tr>
<td>R12(i1,eno,flag,dsk)</td>
<td>Read F1-F2 plane into a new expno (with or without imaginary parts according to flag) on the unit dsk</td>
</tr>
<tr>
<td>R13(i1,eno,flag,dsk)</td>
<td>Read F1-F3 plane into a new expno (with or without imaginary parts according to flag) on the disk unit dsk</td>
</tr>
<tr>
<td>R23(i1,eno,flag,dsk)</td>
<td>Read F2-F3 plane into a new expno (with or without imaginary parts according to flag) on disk unit dsk</td>
</tr>
</tbody>
</table>

2.16 XWIN-NMR plotting macros

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLOT</td>
<td>Plot according to current plot parameters (edg)</td>
</tr>
<tr>
<td>PLOTX</td>
<td>Plot expansions of the integral ranges as defined with edgx</td>
</tr>
<tr>
<td>PLOTS</td>
<td>Suspend plot (put plot in queue)</td>
</tr>
<tr>
<td>PLOTW</td>
<td>Plot a white washed stack plot as defined with edgw</td>
</tr>
<tr>
<td>FLPLOT</td>
<td>Flush all suspended plots</td>
</tr>
</tbody>
</table>
LOCKPLOTS | Lock plot queue for current AU program. No other plot commands can now interfere with the suspended plots generated in the current AU program.
UNLOCKPLOTS | Unlock the plot queue of the current AU program.
RMPLLOT | Remove all suspended plots from queue
GETLIM | Get frequency of leftmost and rightmost peak from a 1D spectrum and adjust the sweep width of the 1D spectrum to the difference + 10%
GETLCOSY | Get frequency of leftmost and rightmost peak from a 1D spectrum and adjust the sweep width of a COSY spectrum to the difference + 10%
GETLXHCO | Get frequency of leftmost and rightmost peak from two 1D spectra and adjust the sweep width of an X-H correlation spectrum to the difference + 10%
GETLJRES | Get frequency of leftmost and rightmost peak from a 1D spectrum and adjust the sweep width of an J-RESolved spectrum to the difference + 10%
GETLINV | Get frequency of leftmost and rightmost peak from a 1D spectrum and adjust the sweep width of an INVerse spectrum to the difference + 10%

2.17 XWIN-PLOT related macros

XWP_LP * | Create a parameter listing for a plot with XWIN-PLOT
XWP_PP * | Create a peak picking listing for a plot with XWIN-PLOT
AUTOPLLOT * | Plot the current dataset according to the XWIN-PLOT layout defined by the edo parameter LAYOUT.
### AUTOPLT_TO_FILE(filename) *

as AUTOPLT except that the plot is not sent to the printer but store in the postscript file `filename`.

### DECLARE_PORTFOLIO *

Initialize the usage of other XWIN-PLT portfolio AU macros. Must be inserted at the beginning of the AU program.

### CREATE_PORTFOLIO(filename) *

Create the XWIN-PLT portfolio `filename`.

### ADD_TO_PORTFOLIO(disk, user, name, expno, procno) *

Add the dataset that is specified with the arguments to the portfolio created with CREATE_PORTFOLIO.

### ADD_CURDAT_TO_PORTFOLIO *

Add the current dataset to the portfolio created with CREATE_PORTFOLIO.

### CLOSE_PORTFOLIO *

Close the definition for the portfolio created with CREATE_PORTFOLIO.

### AUTOPLT_WITH_PORTFOLIO *

Plot the dataset(s) defined in the portfolio created with CREATE_PORTFOLIO according to the layout defined by the `edo` parameter LAYOUT.

### AUTOPLT_WITH_PORTFOLIO_TO_FILE(filename) *

as AUTOPLT_WITH_PORTFOLIO except that the plot is not sent to the printer but store in the postscript file `filename`.

### 2.18 Macros converting datasets from Aspect 2000/3000 and other vendors

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONV(....)</td>
<td>Convert Bruker Aspect 2000/3000 datasets to XWIN-NMR format</td>
</tr>
<tr>
<td>CONVCP(....)</td>
<td>Like CONV but with additional backup copy of the original data</td>
</tr>
<tr>
<td>FROMJDX(....)</td>
<td>Convert a JCAMP-DX file to XWIN-NMR data format</td>
</tr>
<tr>
<td>TOJDX(....)</td>
<td>Convert a dataset to JCAMP-DX format</td>
</tr>
</tbody>
</table>
2.19 Macros to execute other AU programs, XWIN-NMR macros or commands

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCONV(....) *</td>
<td>Convert a Jeol dataset to Bruker XWIN-NMR format</td>
</tr>
<tr>
<td>VCONV(....) *</td>
<td>Convert a Varian dataset to Bruker XWIN-NMR format</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPR_exec(....) *</td>
<td>C-function for executing special XWIN-NMR commands</td>
</tr>
<tr>
<td>WAIT_UNTIL(....) *</td>
<td>Hold the AU program until the specified date and time</td>
</tr>
<tr>
<td>XAUA</td>
<td>Execute the acquisition AU program stored in AUNM (eda). The next line in the AU program is executed after the AU program AUNM has finished.</td>
</tr>
<tr>
<td>XAUP</td>
<td>Execute the processing AU program stored in AUNMP (edp). The next line in the AU program is immediately executed after the AU program AUNMP has been started.</td>
</tr>
<tr>
<td>XAUPW</td>
<td>Execute the processing AU program stored in AUNMP (edp). Like XAUP, but now the next line in the AU program is executed after the AU program AUNMP has finished.</td>
</tr>
<tr>
<td>XAU(prog)</td>
<td>Execute the AU program prog with the wait option</td>
</tr>
<tr>
<td>XCMD(cmd) *</td>
<td>Execute the XWIN-NMR command for which no dedicated macro exists.</td>
</tr>
<tr>
<td>XMAC(mac)</td>
<td>Execute an XWIN-NMR macro mac.</td>
</tr>
</tbody>
</table>

2.20 Bruker library functions

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalcExpTime() *</td>
<td>Calculate the experiment time for the current experiment</td>
</tr>
<tr>
<td>PrintExpTime(....) *</td>
<td>Print the experiment time for the current experiment</td>
</tr>
<tr>
<td>check_pwd(usr) *</td>
<td>Prompt the user usr to enter a password</td>
</tr>
<tr>
<td>GetNmrSuperUser() *</td>
<td>Get the name of the current XWIN-NMR superuser</td>
</tr>
</tbody>
</table>
2.21 Macros to return from an AU program

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT</td>
<td>Abort the AU program or any of its subroutines with the return value of -1</td>
</tr>
<tr>
<td>ERRORABORT</td>
<td>Return from an AU program or any of its subroutines with the value of AUERR if it is less than 0</td>
</tr>
</tbody>
</table>
### Macro Description

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUIT</td>
<td>Return from an AU program with the value of AUERR. QUIT is usually the last statement of the AU program code.</td>
</tr>
<tr>
<td>QUITMSG(text)</td>
<td>Print the text message and then return from the AU program with the value of AUERR. This is an alternative to QUIT.</td>
</tr>
<tr>
<td>STOP</td>
<td>Stop the AU program with the return value of AUERR.</td>
</tr>
<tr>
<td>STOPMSG(&quot;text&quot;)</td>
<td>Stop the AU program with the return value of AUERR and display the message &quot;text&quot;</td>
</tr>
</tbody>
</table>
Chapter 3

General AU macros

This chapter contains a description of all general AU macros which can be used for various purposes.
CPR_exec

NAME
CPR_exec - C-function for executing special XWIN-NMR commands

SYNTAX
CPR_exec(char *command, int mode);

DESCRIPTION
CPR_exec is a C-library function which can be used for executing XWIN-NMR commands in AU. The first argument of CPR_exec is an XWIN-NMR command, the second argument must be one of the following values:

WAIT_TERM - wait for the command to finish, then start the next command
CONT_EX - start the command and immediately start the next command

Most commands are available as a dedicated macro, like ZG for zg and FT for ft. If you want to use XWIN-NMR commands for which no dedicated macro exist, e.g. editor commands or commands with special arguments, then you can use the general macro XCMD which takes only one argument, the XWIN-NMR command. Dedicated macros and XCMD internally call CPR_exec with WAIT_TERM. The only reason to use CPR_exec explicitly is to start a command with CONT_EX, i.e. to run commands simultaneously. In summary:

• Use dedicated AU macros whenever you can
• Use XCMD when no dedicated macro is available
• Use CPR_EXEC only when you want to use CONT_EX

NOTES
Dedicated macros and XCMD call SETCURDATA before they do their actual task. This ensures that they operate on the current AU dataset. If you use CPR_exec explicitly, it is recommended to precede it with SETCURDATA. This is not always necessary but hardly ever wrong (see SETCURDATA).

During sample changer automation, processing and plotting commands are started with CONT_EX to allow simultaneous acquisition on the next sample.
EXAMPLE
The following AU program gets the foreground dataset, runs an acquisition, starts the Fourier Transform and immediately continues an acquisition on the next experiment number:

GETCURDATA
TIMES(10)
ZG
CPR_exec("ft", CONT_EX);
IEXPNO
END
QUIT

SEE ALSO
XCMD - general macro to execute commands for which no dedicated macro exists
SETCURDATA - make the current AU dataset available for subsequent AU statements
**XCMD**

**NAME**

XCMD - execute a command for which no dedicated macro exists

**SYNTAX**

XCMD(char *command)

**DESCRIPTION**

XCMD is a general macro to execute XWIN-NMR commands for which no dedicated macro exists. For most XWIN-NMR commands a dedicated macro does exist and we strongly recommend to:

Use dedicated macros whenever available

**EXAMPLE**

The following AU program gets the foreground dataset, opens the acquisition parameter editor (**eda**) and runs an acquisition and Fourier transform:

```
GETCURDATA
XCMD("eda")
ZG
FT
QUIT
```

**SEE ALSO**

CPR_exec - C-function for executing special XWIN-NMR commands
**WAIT_UNTIL**

**NAME**

WAIT_UNTIL - hold the AU program until the specified date and time

**SYNTAX**

int WAIT_UNTIL(int hour, int minute, int day, int month)

**DESCRIPTION**

The function WAIT_UNTIL waits in an AU program until the specified date has been reached. The variables are internally converted to seconds. Every sixty seconds, the function checks whether the current date matches with the selected date. This function basically allows to program an event or command to start at a certain date rather than waiting for a certain time until something is executed.

**EXAMPLE**

Wait in the AU program until the 31st of October, 6 pm, and then continue:

WAIT_UNTIL(18,0,31,10)

**SEE ALSO**

ssleep - pause in an AU program for a certain number of seconds
Chapter 4

Macros changing the current AU dataset

This chapter contains a description of all AU macros which can be used to change the current AU dataset, i.e. the dataset on which subsequent AU statements operate.
GETCURDATA

NAME
GETCURDATA - the first AU program statement; get the foreground dataset

SYNTAX
GETCURDATA

DESCRIPTION
GETCURDATA should be the first macro in any AU program and should only be used at the beginning of an AU program. GETCURDATA gets the foreground dataset, i.e. the dataset which is visible when the AU program is started. This dataset becomes the current AU dataset. In fact, GETCURDATA sets the predefined data path variables disk, user, type, name, expno and procno which define the pathname of the current dataset. Subsequent macros, like ZG, FT etc. will then work on this dataset. The current AU dataset can be changed within the AU program with macros like DATASET, IEXPNO or IPROCNO.

GETCURDATA defines the current AU dataset but it takes another macro, SETCURDATA, to make this dataset available for subsequent AU statements. For most predefined macros, like ZG and FT, as well as XCMD etc., this is automatic because SETCURDATA is part of the macro definition.

EXAMPLE
The following AU program gets the foreground dataset and runs an acquisition on it:

GETCURDATA
ZG
QUIT

SEE ALSO
GETCURDATA - make the current AU dataset available for AU commands
DATASET - set the current AU dataset
GETDATASET - prompt the user to specify a new dataset
SETCURDATA

NAME
SETCURDATA - make the current AU dataset available for subsequent AU statements

SYNTAX
SETCURDATA

DESCRIPTION
SETCURDATA makes the current AU dataset, i.e. the dataset defined by the data path variables disk, user, type, name, expno and procno, available for subsequent AU commands. Normally, you do not need to enter SETCURDATA because it is automatically called by macros which operate on datasets before they perform their actual task. Furthermore, the macros DATASET and GETDATASET, which change the current AU dataset, automatically call SETCURDATA after they performed their actual task. In some cases, however, SETCURDATA must be specified explicitly in the AU program. For example, the macros IEXPNO and IPROCN0 change the current AU dataset, but do not call SETCURDATA. If they are followed by a CPR_exec or any C-statement which access the current AU dataset, then you must precede that statement with SETCURDATA.

EXAMPLE
This example shows the part of the library AU program multizg which calculates the total experiment time of all acquisitions performed by this AU program:

```c
int expTime;
static void PrintExpTime();
GETCURDATA
....
expTime = 0;
TIMES(i1)
    SETCURDATA;
    expTime += CalcExpTime() + 4;
```
Macros changing the current AU dataset

... IEXPNO;
END
DEXPNO;
....
QUIT

Note that IEXPNO is followed by SETCURDATA in the next cycle of the loop.

SEE ALSO
GETCURDATA - the first AU program statement; get the foreground dataset
DATASET - set the current AU dataset
IEXPNO - increase the experiment number by one
DATASET

NAME
DATASET - set the current AU dataset

SYNTAX
DATASET(char *name, int expno, int procno, char *disk, char *user)

DESCRIPTION
The macro DATASET sets the current AU dataset. All data path variables name, expno, procno, disk and user must be specified as arguments. Subsequent AU commands will operate on this dataset.

EXAMPLE
The following AU program first gets the foreground dataset, then selects a new dataset and runs an acquisition:

```au
char[20] newname;
GETCURDATA
strcpy(newname, "glycerine");
DATASET(newname, expno, 3, disk, "peter")
ZG
QUIT
```

The data path variables in this example are entered in the following way:

- expno and disk keep the values which were obtained with GETCURDATA
- name gets the value of newname, a variable defined in this AU program
- procno and user get the values 3 and peter, respectively, which are entered as constants

SEE ALSO
GETCURDATA - the first AU program statement; get the foreground dataset
GETDATASET - prompt the user to specify a new dataset
DATASET2 - set the second dataset
IEXPNO - increase the experiment number by one
DATASET2/DATASET3

NAME
DATASET2 - set the second AU dataset
DATASET3 - set the third AU dataset

SYNTAX
DATASET2(char *name, int expno, int procno, char *disk, char *user)
DATA /x/xw2.5

cd SET3(char *name, int expno, int procno, char *disk, char *user)

DESCRIPTION
The macro DATASET2 sets the second AU dataset. The current (first) AU dataset is not affected by this macro. DATASET2 is typically used in combination with algebra macros, like ADD or MUL, which operate on the second and third dataset.

EXAMPLE
The following AU program gets the foreground dataset, adds the spectra of the next processing number and the one after that and stores the result into the current dataset:

GETCURDATA
DATASET2(name, expno, procno+1, disk, user)
DATASET3(name, expno, procno+2, disk, user)
ADD
QUIT

SEE ALSO
GETCURDATA - the first AU program statement; get the foreground dataset
DATASET - set the current AU dataset
GETDATASET - prompt the user to specify a new dataset
GETDATASET

NAME

GETDATASET - prompt the user to specify a new dataset

SYNTAX

GETDATASET

DESCRIPTION

The macro GETDATASET prompts the user to specify a new dataset. A dialogue is opened and the user is requested to enter the data path variables name, expno, procno, user and disk. Subsequent AU commands will operate on this dataset. GETDATASET can be used anywhere in an AU program but, since it requires user input, should not be used in fully automated sequences.

NOTE

GETDATASET is not used very often. In AU programs, datasets are usually changed without user interaction, e.g. with the macros DATASET, IEXPNO etc.

EXAMPLE

The following AU program gets the foreground dataset, prompts the user to specify a new dataset and then processes this dataset:

   GETCURDATA
   GETDATASET
   EFP
   QUIT

Actually you could begin this AU program with GETDATASET. This is one of the very few examples where you may omit the macro GETCURDATA.

SEE ALSO

GETCURDATA - the first AU program statement; get the foreground dataset
DATASET - set the current AU dataset
IEXPNO - increase the experiment number by one
IPROCNO - increase the processing number by one
IEXPNO

NAME
IEXPNO - increase the experiment number by one

SYNTAX
IEXPNO

DESCRIPTION
The macro IEXPNO increases the experiment number of the current AU dataset by one. In fact, the value of the data path variable expno is incremented by one. Subsequent macros will operate on this new expno. IEXPNO is typically used in AU programs which run a series of acquisitions on datasets with the same name and successive expnos.

EXAMPLE
The following AU program gets the foreground dataset and runs acquisitions on eight successive expnos:

GETCURDATA
TIMES(8)
ZG
IEXPNO
END
QUIT

NOTE
IEXPNO must be followed by a SETCURDATA if the AU program continues with an explicit CPR_exec or C-statement (see SETCURDATA).

SEE ALSO
DEXPNO - decrease the experiment number by one
REXPNO - set the experiment number to the specified value
IPROCNO - increase the processing number by one
DATASET- set the current AU dataset
DEXPNO

NAME
DEXPNO - decrease the experiment number by one

SYNTAX
DEXPNO

DESCRIPTION
The macro DEXPNO decreases the experiment number of the current AU dataset by one. In fact, the value of the data path variable expno is decremented by one. Subsequent macros will operate on this new expno. DEXPNO is typically used after a loop which includes an IEXPNO at the end, to revert the effect of the last (unnecessary) IEXPNO.

EXAMPLE
The following AU program gets the foreground dataset, runs acquisitions on eight successive expnos and displays the data of the last expno:

GETCURDATA
TIMES(8)
ZG
IEXPNO
END
DEXPNO
VIEWDATA
QUIT

Note that DEXPNO must be followed by a SETCURDATA if the AU program continues with an explicit CPR_exec or C-statement (see SETCURDATA).

SEE ALSO
IEXPNO - increase the experiment number by one
REXPNO - set the experiment number to the specified value
DPROCNO - decrease the processing number by one
REXPNO

NAME
REXPNO - set the experiment number to the specified value

SYNTAX
REXPNO(int number)

DESCRIPTION
The macro REXPNO sets the experiment number of the current AU dataset to
the specified value. In fact, the value of the data path variable expno is set. Sub-
sequent macros will operate on this new expno.

EXAMPLE
The following AU program gets the foreground dataset, runs acquisitions on
eight successive expnos then sets the current AU dataset back to the first expno
and Fourier transforms it:

GETCURDATA
i1=expno;
TIMES(8)
  ZG
  IEXPNO
END
REXPNO(i1)
FT
QUIT

Note that REXPNO must be followed by a SETCURDATA if the AU program
continues with an explicit CPR_exec or C-statement (see SETCURDATA).

SEE ALSO
IEXPNO - increase the experiment number by one
DEXPNO - decrease the experiment number by one
RPROCNO - set the processing number to the specified value
IPROCNO

NAME
IPROCNO - increase the processing number by one

SYNTAX
IPROCNO

DESCRIPTION
The macro IPROCNO increases the processing number of the current AU dataset by one. In fact, the value of the data path variable procno is incremented by one. Subsequent macros will operate on this new procno. IPROCNO is typically used in an AU program which processes a series of datasets with same name and expno and successive procnos.

EXAMPLE
The following AU program runs Fourier transforms on eight successive procnos:

GETCURDATA
TIMES(8)
FT
IPROCNO
END
QUIT

Note that IPROCNO must be followed by a SETCURDATA if the AU program continues with an explicit CPR_exec or C-statement (see setcurdata).

SEE ALSO
DPROCNO - decrease the processing number by one
RPROCNO - set the processing number to the specified value
IEXPNO - increase the experiment number by one
DPROCNO

NAME
DPROCNO - decrease the processing number by one

SYNTAX
DPROCNO

DESCRIPTION
The macro DPROCNO decreases the processing number of the current AU dataset by one. In fact, the value of the data path variable procno is decremented by one. Subsequent macros will operate on this new procno. DPROCNO is typically used after a loop which includes an IPROCNO at the end, to revert the effect of the last (unnecessary) IPROCNO.

EXAMPLE
The following AU program gets the foreground dataset, runs a Fourier transform on eight successive procnos and displays the data of the last procno:

GETCURDATA
TIMES(8)
FT
IPROCNO
END
DPROCNO
VIEWDATA
QUIT

Note that DPROCNO must be followed by a SETCURDATA if the AU program continues with an explicit CPR_exec or C-statement (see setcurdata).

SEE ALSO
IPROCNO - decrease the experiment number by one
RPROCNO - set the processing number to specified value
DEXPNO - decrease the experiment number by one
RPROCNO

NAME
RPROCNO - set the processing number to the specified value

SYNTAX
RPROCNO(int number)

DESCRIPTION
The macro RPROCNO changes the current AU dataset by setting the processing number to the specified value. In fact, the value of the data path variable procno is set. Subsequent macros will then operate on this new procno.

EXAMPLE
The following AU program gets the foreground dataset and runs a Fourier transform on eight successive procnos. Then the current AU dataset is set back to the first procno which is then phase corrected:

GETCURDATA
i1=procno;
TIMES(8)
FT
IPROCNO
END
RPROCNO(i1)
APK
QUIT

Note that RPROCNO must be followed by a SETCURDATA if the AU program continues with an explicit CPR_exec or C-statement (see SETCURDATA).

SEE ALSO
IPROCNO - increase the processing number by one
DPROCNO - decrease the processing number by one
REXPNO - set the experiment number to the specified value
VIEWDATA

NAME

VIEWDATA - Show the current AU program dataset in XWIN-NMR

SYNTAX

VIEWDATA

DESCRIPTION

The macro VIEWDATA shows the current AU program dataset in XWIN-NMR. In other words, the current AU dataset becomes the foreground dataset. VIEWDATA is used whenever the current AU dataset is changed within the AU program, i.e. with DATASET, IEXPNO etc. and this dataset must be shown in XWIN-NMR.

EXAMPLE

The following AU program gets the foreground dataset, increases the processing number and performs a Fourier transform storing the spectrum in this processing number. The spectrum is then shown in XWIN-NMR:

    GETCURDATA
    IPROCNO
    FT
    VIEWDATA
    QUIT

SEE ALSO

GETCURDATA - the first AU program statement; get the foreground dataset
GETDATASET - prompt the user to specify a new dataset
DATASET - set the current AU dataset
IEXPNO - increase the experiment number by one
IPROCNO - increase the processing number by one
Chapter 5

Macros copying datasets

This chapter contains a description of all AU macros which can be used to copy the current AU dataset or parts of it to a new dataset.
**WRA**

**NAME**

WRA - copy the raw data to the specified experiment number

**SYNTAX**

WRA(int expno)

**DESCRIPTION**

The macro WRA copies the raw data, including the acquisition and processing parameters of the current AU dataset to a new experiment number. It does not copy the processed data.

**EXAMPLE**

The following AU program gets the foreground dataset and copies the raw data to eight successive experiment numbers, starting with *expno 11*:

```au
GETCURDATA
i1=11;
TIMES(8)
  WRA(i1)
  i1++;
END
QUIT
```

**SEE ALSO**

WRP - copy the processed data to the specified processing number
WRPA - copy the raw and processed data to the specified dataset
WRP

NAME
WRP - copy the processed data to the specified processing number

SYNTAX
WRP(int procno)

DESCRIPTION
The macro WRP copies the processed data, including the processing parameters of the current AU dataset, to the specified processing number.

EXAMPLE
The following AU program gets the foreground dataset and copies the processed data to eight successive processing numbers, starting with procno 11:

GETCURDATA
i1=11;
TIMES(8)
    WRP(i1)
    i1++;
END
QUIT

SEE ALSO
WRA - copy the raw data to the specified experiment number
WRPA - copy the raw and processed data to the specified dataset
WRPA

NAME
WRPA - copy the raw and processed data to the specified dataset

SYNTAX
WRPA(char *name, int expno, int procno, char *disk, char *user)

DESCRIPTION
The macro WRPA copies the raw and processed data of the current AU dataset to the specified dataset. WRPA takes 5 arguments, name, expno, procno, disk and user, i.e. the data path variables which define the dataset path. You can set one, several, or all of these variables to new values in order to define the destination dataset. You can, for instance, archive your data to an external medium by changing the value of the variable disk and leaving the other path variables the same.

EXAMPLE
The following AU program copies the current dataset to the jaz drive which is mounted as /jaz:

GETCURDATA
WRPA(name, expno, procno, "/jaz", user)
QUIT

SEE ALSO
WRA- copy the raw data to the specified experiment number
WRP- copy the processed data to the specified processing number
Chapter 6

Macros handling rows/columns

This chapter contains a description of all AU macros which can be used to read (write) rows or columns from (to) a 2D dataset and AU macros that can be used to read rows or planes from 3D raw data.
**RSR**

**NAME**
RSR - read a row from a 2D spectrum and store it as a 1D spectrum

**SYNTAX**
RSR(int row, int procno)

**DESCRIPTION**
The macro RSR reads a row from a 2D spectrum and stores it as a 1D spectrum. It can be used in the following ways:

- specified with procno > 0, executed on a 2D dataset
  the specified row is stored under the current dataname, the current expno and the specified procno.
- specified with procno = -1, executed on a 2D dataset
  the specified row is stored under dataset ~TEMP/1/pdata/1
- specified with procno > 0, executed on a 1D dataset
  the specified row is read from a 2D dataset that resides under the current dataname, the current expno and the specified procno.
- specified with procno = -1, executed on a 1D dataset
  the specified row is read from the 2D dataset from which the current 1D dataset was extracted (as defined in the file used_from).

**EXAMPLE**
The following AU program gets a 2D dataset and processes it. Then it reads row 16 and stores that under procno 999:

```
GETCURDATA
DATASET("my_2D_data", 1, 1, "/x", "guest")
XFB
RSR(16, 999)
QUIT
```

**SEE ALSO**
RSC - read a column from a 2D spectrum and store it as a 1D spectrum
RSC

NAME

RSC - read a column from a 2D spectrum and store it as a 1D spectrum

SYNTAX

RSC(int column, int procno)

DESCRIPTION

The macro RSC reads a column from a 2D spectrum and stores it as a 1D spectrum. It can be used in the following ways:

- specified with procno > 0, executed on a 2D dataset
  the specified column is stored under the current dataname, the current expno and the specified procno.
- specified with procno = -1, executed on a 2D dataset
  the specified column is stored under dataset ~TEMP/1/pdata/1
- specified with procno > 0, executed on a 1D dataset
  the specified column is read from a 2D dataset that resides under the current dataname, the current expno and the specified procno.
- specified with procno = -1, executed on a 1D dataset
  the specified column is read from the 2D dataset from which the current 1D dataset was extracted (as defined in the file used_from).

EXAMPLE

The following AU program gets a 2D dataset and processes it in the F2 dimension. Then it reads column 128 and processes the resulting 1D dataset:

GETCURDATA
DATASET("my_2D_data", 1, 1, "/x", "guest")
XF2
RSC(128, 10)
RPROCNO(10)
EF
QUIT
SEE ALSO

RSR - read a row from a 2D spectrum and store it as a 1D spectrum
WSC - replace a column of a 2D spectrum by a 1D spectrum
WSR

NAME
WSR - replace a row of a 2D spectrum by a 1D spectrum

SYNTAX
WSR(int row, int procno, int expno, char *name, char *user, char *disk)

DESCRIPTION
The macro WSR replaces a row of a 2D spectrum by a 1D spectrum. It can be used in the following ways:

- executed on a 1D dataset
  the specified row of the specified dataset (must 2D data) is replaced by the current 1D data.
- executed on a 2D dataset
  the specified row of the current 2D dataset is replaced by the specified dataset (must be 1D data)

Note that WSR exist in this form in XWIN-NMR 3.1 and newer. In XWIN-NMR 3.0 and older, WSR takes only 5 arguments, the expno cannot be specified.

EXAMPLE
The following AU program gets a 2D dataset, reads row 16, phase corrects this row and writes it back to the 2D data:

GETCURDATA
DATASET("my_2D_data", 1, 1, "/x", "guest")
XFB
RSR(16, 999)
RPROCNO(999)
APK
WSR(16, 1, expno, name, user, disk)
QUIT

SEE ALSO
WSC - replace a column of a 2D spectrum by a 1D spectrum
RSR - read a row from a 2D spectrum and store it as a 1D spectrum
WSC

NAME
WSC - replace a column of a 2D spectrum by a 1D spectrum

SYNTAX
WSC(int column, int procno, int expno, char *name, char *user, char *disk)

DESCRIPTION
The macro WSC replaces a column of a 2D spectrum by a 1D spectrum. It can be used in the following ways:

• executed on a 1D dataset
  the specified column of the specified dataset (must 2D data) is replaced by the current 1D data.

• executed on a 2D dataset
  the specified column of the current 2D dataset is replaced by the specified dataset (must be 1D data)

Note that WSC exist in this form in XWIN-NMR 3.1 and newer. In XWIN-NMR 3.0 and older, WSC takes only 5 arguments, the expno cannot be specified.

EXAMPLE
The following AU program gets a 2D dataset, reads column 16, phase corrects this column and writes it back to the 2D data:

GETCURDATA
DATASET("my_2D_data", 1, 1, "/x", "guest")
RSC(16, 999)
RPROCNO(999)
APK
WSC(16, 1, expno, name, user, disk)
QUIT

SEE ALSO
WSR - replace a row of a 2D spectrum by a 1D spectrum
RSC - read a column from a 2D spectrum and store it as a 1D spectrum
RSER

NAME
RSER - read a row from 2D or 3D raw data and store it as a 1D FID

SYNTAX
RSER(int row, int expno, int procno)

DESCRIPTION
The macro RSER reads a row from 2D or 3D raw data and stores it as a 1D fid. It can be used in the following ways:

- specified with expno > 0, executed on a 2D dataset
  the specified row is stored under the current dataname and the specified expno. Processing parameters are stored under procno 1.
- specified with expno = -1, executed on a 2D dataset
  the specified row is stored under dataset ~TEMP/1/pdata/1
- specified with expno > 0, executed on a 1D dataset
  the specified row is read from a 2D raw data that resides under the current dataname and the specified expno. Processing parameters are read from procno 1.
- specified with expno = -1, executed on a 1D dataset
  the specified row is read from the 2D dataset from which the current 1D dataset was extracted (as defined in the file used_from).

EXAMPLE
The following AU program splits 2D raw data into single fids that are stored in successive expnos:

```c
int td;
GETCURDATA
FETCHPAR1S("TD",&td)
i1=0;
TIMES(td)
  i1 ++;
  RSER(i1,i1+expno,1)
```
END
QUITMSG("--- splitser finished ---")

Note that this is the AU program **splitser** that is delivered with XWIN-NMR.

**SEE ALSO**
- WSER - replace a row of 2D raw data by 1D raw data
- RSER2D - read a plane from 3D raw data and store it as 2D raw data
- RSR - read a row from a 2D spectrum and store it as a 1D spectrum
WSER

NAME

WSER - replace a row of 2D raw data by 1D raw data

SYNTAX

WSER(int row, char *name, int expno, int procno, char *disk, char *user)

DESCRIPTION

The macro WSER replaces a row of 2D raw data by 1D raw data. It can be used in the following ways:

- executed on a 1D dataset
  the specified row of the specified dataset (must be 2D data) is replaced by the current 1D data.

- executed on a 2D dataset
  the specified row of the current 2D dataset is replaced by the specified dataset (must be 1D data)

EXAMPLE

The following AU program writes a number of 1D fids that are stored under the same data name and incremental expnos to 2D raw data.: 

```au
int ne, exp1, proc1;
char nm1[20];

GETCURDATA
ne=1; exp1=1; proc1=1;
strcpy(nm1, name);
GETSTRING("Enter name of 1D series: ", nm1)
GETINT("Enter starting EXPNO: ", exp1)
GETINT("Enter starting PROCNO: ", proc1)
GETINT("Enter # of Fids: ", ne)
USECURPARS
TIMES(ne)
  WSER(loopcount1+1, nm1, exp1, proc1, disk, user)
  exp1++;
```
Note that this is the AU program `fidtoser` that is delivered with XWIN-NMR.

**SEE ALSO**

- **RSER** - read a row from 2D or 3D raw data and store it as a 1D FID
- **WSR** - replace a row of a 2D spectrum by a 1D spectrum
- **WSC** - replace a column of a 2D spectrum by a 1D spectrum
RSER2D

NAME
RSER2D - read a plane from 3D raw data and store it as 2D pseudo raw data

SYNTAX
RSER2D(char *direction, int plane, int expno, int procno)

DESCRIPTION
The macro RSER2D reads a plane from 3D raw data and stores it as 2D pseudo raw data. The first argument, the plane direction can be "s23" or "s13" for the F2-F3 or F1-F3 direction, respectively. The specified plane is stored under the current data name, the specified expno and the specified procno.

RSER2D only exists in XWIN-NMR 3.1 an newer.

EXAMPLE
The following AU program gets a 3D dataset, reads the F2-F3-plane 64 and stores that under expno 11. It then switches to the output 2D dataset and processes it.

GETCURDATA
DATASET("my_3D_data", 1, 1, "/x", "guest")
RSER2D("s23", 64, 11, procno)
REXPNO(11)
XFB
END
QUIT

SEE ALSO
RSER - read a row from 2D or 3D raw data and store it as a 1D FID
WSER - replace a row of 2D raw data by 1D raw data
Chapter 7

Macros converting datasets

This chapter contains a description of all AU macros which can be used to convert XWIN-NMR data. This includes the conversion of Bruker Aspect 2000/3000 data, Varian data and Jeol data to XWIN-NMR data format as well as the conversion of XWIN-NMR data to JCAMP-DX.
TOJDX

NAME

TOJDX - convert a dataset to JCAMP-DX format

SYNTAX

TOJDX(char *path, int type, int mode, char *title, char *origin, char *owner)

DESCRIPTION

The macro TOJDX converts the current AU data to standard JCAMP-DX format. It takes 6 arguments:

1. the pathname of the output file, e.g. /tmp/data1.dx
2. the output type: enter 0, 1 or 2
   0=FID (default), 1=real spectrum, 2=complex spectrum.
3. the compression mode: enter 0, 1, 2 or 3
   0=FIX, 1=PACKED, 2=SQUEEZED, 3=DIFF/DUP (default)
4. the title as it appears in the output file: enter a character-string
5. the origin as it appears in the output file: enter a character-string
6. the owner as it appears in the output file: enter a character-string

If "*" is entered as an argument, then the default value is used.

In XWIN-NMR 3.1 and newer, TOJDX can convert 1D and 2D data. In older versions, it can only convert 1D data.

EXAMPLE

The following AU program gets the foreground dataset and performs a conversion to JCAMP on 5 successive experiment numbers. The name of the JCAMP file contains the name and expno of the corresponding XWIN-NMR dataset.

GETCURDATA
TIMES(5)
    sprintf(text,"/tmp/%s_%d.dx", name, expno);
    TOJDX(text, 0, 3, "", "", "")
IEXPNO
END
QUIT

SEE ALSO
FROMJDIX - convert a JCAMP-DX file to XWIN-NMR data format
FROMJDX

NAME
FROMJDX - convert a JCAMP-DX file to XWIN-NMR data format

SYNTAX
FROMJDX(char *input-file, char *overwrite)

DESCRIPTION
The macro FROMJDX converts a JCAMP-DX file to XWIN-NMR data format. It takes 2 arguments:
1. the pathname of the input file, e.g. /tmp/data1.dx
2. the overwrite flag which is either "o" or "n"

In XWIN-NMR 3.1 and newer, FROMJDX can convert 1D and 2D data. In older versions, it can only convert 1D data.

EXAMPLE
The following AU program converts all files with the extension .dx in the directory /tmp to an XWIN-NMR dataset:

    char **listfile;
    GETCURDATA
    i1 = getdir("/tmp", &listfile, ".dx");
    TIMES(i1)
        sprintf(text, "/tmp/%s", listfile[loopcount1]);
        FROMJDX(text, "o")
    END
    QUIT

SEE ALSO
TOJDX - convert a dataset to JCAMP-DX format
getdir - get all file names and/or directory names within a directory
CONV/CONVCP

NAME

CONV - convert Bruker Aspect 2000/3000 datasets to XWIN-NMR format
CONVCP - like CONV but with additional backup copy of the original data

SYNTAX

CONV(char *instrument, char *filename)
CONVCP(char *instrument, char *filename, char *targetdir)

DESCRIPTION

The macro CONV converts Bruker Aspect 2000/3000 datasets to XWIN-NMR data format. Input datasets must be stored in the directory:

/<var>/bruknet/<instrument>/<user>

where <var> can be one of the following:

- <du> the disk unit of the current XWIN-NMR dataset specified in the file /usr/local/lib/destination (UNIX only)
- specified in the file /usr/local/lib/destination (UNIX only)

CONV takes 2 parameters:

1. the instrument where the dataset was acquired, e.g. ac250
2. the name of the dataset. If "*" is specified the next available dataset in the storage directory will be converted.

The macro CONVCP is identical to CONV except that it makes a copy of the original data. Therefore, it takes a third argument; the target directory this copy.

EXAMPLE

This example shows the content of the library AU program remproc which continuously converts all datasets which are sent from an A3000 computer named asp. After conversion it processes each dataset by calling a dataset specific processing AU program with the macro XAUPW:

```c
#define STATION "asp"

GETCURDATA
CPR_exec("setdef ackn no",CONT_EX);
```
TIMESINFINITE
   CONV(STATION,"*")
   GETCURDATA
   XAUPW
END
QUIT
VCONV

NAME

VCONV - convert a Varian dataset to Bruker XWIN-NMR format

SYNTAX

VCONV(char *v-name, char *x-name, int expno, char *disk, char *user)

DESCRIPTION

The macro VCONV converts a Varian dataset to XWIN-NMR data format. It takes 5 parameters:

1. the name of the input Varian dataset
2. the name of the output XWIN-NMR dataset
3. the experiment number of the output XWIN-NMR dataset
4. the disk unit of the output XWIN-NMR dataset
5. the user of the output XWIN-NMR dataset

EXAMPLE

The following AU program converts a Varian dataset to XWIN-NMR format:

GETCURDATA
VCONV("pinen_h.fid", "pinen_h", 1, "u", "joe")
QUIT

SEE ALSO

JCONV - convert a Jeol dataset to Bruker XWIN-NMR format
JCONV

NAME
JCONV - convert a Jeol dataset to Bruker XWIN-NMR format

SYNTAX
JCONV(char *j-name, char *x-name, int expno, char *disk, char *user)

DESCRIPTION
The macro JCONV converts a Jeol dataset to XWIN-NMR data format. It takes 5 parameters:
1. the name of the input Jeol dataset
2. the name of the output XWIN-NMR dataset
3. the experiment number of the output XWIN-NMR dataset
4. the disk unit of the output XWIN-NMR dataset
5. the user of the output XWIN-NMR dataset

EXAMPLE
The following AU program converts a Jeol dataset to XWIN-NMR format:

GETCURDATA
    JCONV("gx400h.gxd", "gx400h", 1, "u", "joe")
QUIT

SEE ALSO
VCONV - convert a Varian dataset to Bruker XWIN-NMR format
Chapter 8

Macros handling XWIN-NMR parameters

This chapter contains a description of AU macros which can be used to get and store XWIN-NMR parameters. Parameters are subdivided in acquisition, processing, output and plot parameters. Furthermore, they exist in two different forms; as foreground and status parameters. Finally, multi-dimensional datasets have parameter sets for each dimension. Different AU macros are available for getting and storing parameters of all categories, forms or dimensions.
**FETCHPAR**

**NAME**

FETCHPAR - get an acquisition, processing or output parameter

**SYNTAX**

FETCHPAR(parm, &val)

**DESCRIPTION**

The macro FETCHPAR gets the value of a foreground parameter and stores it into an AU variable. This AU variable can then be used in subsequent AU statements. FETCHPAR allows to get acquisition parameters (eda), processing parameters (edp) and output parameters (edo). It is typically used to check or modify a parameter prior to an acquisition or processing statement.

The macro FETCHPAR takes two arguments:

1. the name of the parameter
2. the AU variable into which the parameter value will be stored

There are two important things to be considered:

1. The type of the AU variable must be the same as the type of the parameter (see Chapter 13).
2. The second argument must be specified as the variable’s address, i.e. it must be prepended with the ‘&’ character. This, however, does not count for a text variable since a text variable is already an address.

FETCHPAR works on 1D, 2D or 3D datasets and always gets a parameter of the first dimension (F2 for 1D, F2 for 2D and F3 for 3D).

The handling of the macros FETCHPAR1, FETCHPAR3, FETCHPARM, FETCHPLPAR, FETCHPLWPAR, FETCHPLXPAR, FETCHT1PAR and FETCHDOSYPAR is equivalent to the handling of FETCHPAR.

**EXAMPLES**

The following AU program gets the value of the processing parameter SI and processes the data 4 times, each time doubling the spectrum size and storing the data in successive processing numbers:
GETCURDATA
FETCHPAR("SI", &i1)
TIMES(4)
  EFP
  IPROCNO
  i1 = i1*2;
  STOREPAR("SI", i1)
END
QUIT

The following AU statements get the values of the acquisition parameter DW and the processing parameter STSI and stores them in the predefined variables \( f1 \) and \( i1 \), respectively. Then it gets value of the parameter ABSF1 and stores it in the user defined variable \( \text{leftlimit} \). Finally, it gets the value of the output parameter CURPLOT and stores it in the predefined variable \( \text{text} \).

float leftlimit;
...
FETCHPAR("DW", &f1)
FETCHPAR("STSI", &i1)
FETCHPAR("ABSF1", &leftlimit)
FETCHPAR("CURPLOT", text)

SEE ALSO

FETCHPARS - get a status parameter (acquisition and processing)
STOREPAR - store an acquisition, processing or output parameter
FETCPARS

NAME
FETCPARS - get a status parameter (acquisition and processing)

SYNTAX
FETCPARS(parm, &val)

DESCRIPTION
The macro FETCPARS gets the value of a status parameter and stores it into an AU variable. This AU variable can then be used in subsequent AU statements. Acquisition status parameters are set by acquisition commands and describe the status of the dataset after acquisition. Note that the status parameters (dpa) describe what really happened and that this is sometimes different from what was set up before the acquisition as acquisition parameters (eda). For example, the status NS is smaller than originally specified when an acquisition was halted prematurely. Any AU program statement which follows an acquisition command and evaluates acquisition parameters must read status parameters. Therefore, FETCPARS is typically used after acquisition or processing statements, for example for error or abort conditions (see example below).

The macro FETCPARS takes two arguments:

1. the name of the parameter
2. the AU variable into which the value is value will be stored

There are two important things to be considered:

1. The type of the AU variable must be the same as the type of the parameter (see Chapter 13).
2. The second argument must be specified as the variable’s address, i.e. it must be prepended with the ‘&’ character. This, however, does not count for a text variable since a text variable is already an address.

The handling of the macros FETCPARS1 and FETCPARS3 is equivalent to the handling of FETCPARS.

EXAMPLE
The following AU program performs a series of acquisitions on the same dataset
until a minimum signal/noise is reached. In a loop 8 scans are acquired, Fourier
transformed and phase corrected. Then the signal/noise of the spectrum is calcu-
lated and compared with the minimum value. If the minimum signal/noise was
not reached yet, 8 more scans are accumulated etc. A maximum of 8000 scans
is acquired. After the acquisition has been stopped, the total number of actually
acquired scans is displayed.

GETCURDATA
STOREPAR("NS", 8)
GETFLOAT("Please enter the minimum signal/noise", f1)
ZG
TIMES(1000)
    FT
    APK
    SINO
    FETCHPARS("SINO", f2)
    if (f1 >= f2)
        break;
    GO
END
FETCHPARS("NS", i1)
Proc_err (DEF_ERR_OPT,"Acquisition stopped after %d scans", i1);
QUIT

SEE ALSO

FETCHPAR - get an acquisition, processing or output parameter
STOREPARS - store a status parameter (acquisition and processing)
STOREPAR

NAME
STOREPAR - store an acquisition, processing or output parameter

SYNTAX
STOREPAR(parm, val)

DESCRIPTION
The macro STOREPAR stores the value of an AU variable into a parameter. This AU variable can then be used in subsequent AU statements. STOREPAR can be used for acquisition parameters (eda), processing parameters (edp) and output parameters (edo). It is typically used to set parameters prior to an acquisition or processing statement. STOREPAR takes two arguments:

1. the name of the parameter
2. the value to be stored which can specified in two different forms:
   • as a constant
   • as the name of an AU variable

Important: the type of the parameter must be the same as the type of the constant or variable. (see Chapter 13).

NOTES
STOREPAR works on 1D, 2D or 3D datasets and always stores a parameter of the first dimension (F2 for 1D, F2 for 2D and F3 for 3D).

The handling of the macros STOREPAR1, STOREPAR3, STOREPARAM, STOREPLPAR, STOREPLWPAR, STOREPLXPAR, STORET1PAR and STOREDOSYPAR is equivalent to the handling of STOREPAR.

EXAMPLE
The following AU program reads a standard parameter set, sets the pulse program and power level, asks the user for the number of scans and sets the plotter. Then a dataset is acquired, processed and plotted according to these parameters.

char curplotter[20];
GETCURDATA
RPAR("PROTON", "all")
STOREPAR("PULPROG", "zg30")
STOREPAR("PL 1", 10.0)
GETINT("Please enter the number of scans:", i1)
STOREPAR("NS", i1)
(void) strcpy(curplotter, "hplj5l");
STOREPAR("CURPLOT", curplotter)
ZG
EFP
PLOT
QUIT

SEE ALSO
STOREPARS - store a status parameter (acquisition and processing)
FETCHPAR - get an acquisition, processing or output parameter
STOREPARS

NAME
STOREPARS - store a status parameter (acquisition and processing)

SYNTAX
STOREPARS(parm, val)

DESCRIPTION
The macro STOREPARS stores the value of an AU variable into a status parameter. This AU variable can then be used in subsequent AU statements. Status parameters are set by an acquisition or processing command and describe the status of the dataset after this acquisition or processing command. If the data are now manipulated by AU statements which do not update the status parameters, these must be set explicitly with STOREPARS. For example, if you add two fid's with addfid, the total number of scans of the resulting dataset is not automatically updated. This must be done explicitly with STOREPARS.

The handling of the macros STOREPARS1S and STOREPARS3S is equivalent to the handling of STOREPARS.

EXAMPLE
The following AU program reads the foreground dataset, adds the fid of the next experiment number and the experiment number after that and stores the result in the foreground dataset. The number of scans of the original fid’s are added and stored in the status parameter NS of the resulting dataset.

```au
int expno_save;
GETCURDATA
DATASET2(name, expno+1, procno, disk, user)
DATASET3(name, expno+2, procno, disk, user)
expno_save=expno;
IEXPNO
FETCHPARGS("NS", &i1)
IEXPNO
FETCHPARGS("NS", &i2)
```
REXPNO(expno_save)
ADDFID
STOREPARS("NS", i1+i2)
QUIT

SEE ALSO
FETCHPARS - get a status parameter (acquisition and processing)
STOREPAR - store an acquisition, processing or output parameter
RPAR

NAME
RPAR - read a parameter set to the current AU dataset

SYNTAX
RPAR(char *parset, char *typ)

DESCRIPTION
The macro RPAR reads a parameter set to the current AU dataset. This can be a standard Bruker parameter set or a user defined parameter set which was stored with WPAR. RPAR takes two arguments:

1. the name of the parameter set
2. the type of parameters which are to be read

The second argument can be:

- *acqu* for acquisition parameters (*eda*)
- *proc* for processing parameters (*edp*)
- *plot* for plot parameters (*edg*)
- *outd* for output parameters (*edo*)
- *all* for acquisition, processing, plot and output parameters

EXAMPLE
The following AU program reads the standard Bruker parameter set PROTON, sets the number of scans to 1024 and runs an acquisition:

```
GETCURDATA
RPAR("PROTON", "all")
STOREPAR("NS", 1024)
ZG
QUIT
```

SEE ALSO
WPAR - write the current dataset parameters to a parameter set
WPAR

NAME
WPAR - write the current dataset parameters to a parameter set

SYNTAX
WPAR(char *parset, char *typ)

DESCRIPTION
The macro WPAR writes the parameters of the current AU dataset to a parameter set. You can only write to user defined parameter sets. Bruker standard parameters sets cannot be overwritten. WPAR is typically used in AU programs to store a temporary parameter set. It takes two arguments:

1. the name of the parameter set
2. the type of parameters which are to be stored

The second argument can be:

- *acqu* for acquisition parameters (*eda*)
- *proc* for processing parameters (*edp*)
- *plot* for plot parameters (*edg*)
- *outdir* for output parameters (*edo*)
- *all* for acquisition, processing, plot and output parameters

EXAMPLE
The following AU program reads the acquisition parameters of the Bruker standard parameter set PROTON, sets the number of scans, the frequency offset and time domain data size and writes the acquisition parameters to a temporary parameter set. It then performs 8 successive acquisitions with these parameters.

GETCURDATA
RPAR("PROTON", "all")
STOREPAR("NS", 16)
STOREPAR("O1", 766.23)
STOREPAR("TD", 8192)
WPAR("partemp", "acqu")
TIMES(8)
  ZG
  IEXPNO
  RPAR("partemp", "acqu")
END
QUIT

SEE ALSO

RPAR - read a parameter set to the current AU dataset
Chapter 9

Macros for XWIN-PLOT/autoplot

This chapter contains a description of AU macros which can be used to plot data using XWIN-PLOT portfolios and layouts. These include macros for the creation and definition of portfolios and for plotting to the printer, to a postscript file or enhanced metafile.
AUTOPLOT

NAME
AUTOPLOT - plot the current dataset according to an XWIN-PLOT layout

SYNTAX
AUTOPLOT

DESCRIPTION
The macro AUTOPLOT plots the current dataset according to the XWIN-PLOT layout that is defined by the \textit{edo} parameter LAYOUT.

The XWIN-PLOT layout can be:
\begin{itemize}
  \item a standard layout that was delivered with XWIN-NMR
  \item a user defined layout that was setup and stored from XWIN-PLOT
\end{itemize}

In XWIN-NMR 3.1 and newer, standard processing AU programs, like \texttt{proc\_1d} contain the AUTOPLOT macro for plotting (see the example below) whereas in older versions, the PLOT macro was used. However, AU programs that contain PLOTX macros; are still used in the original form. Furthermore, the old AU programs, using PLOT macro, are still available under the names \texttt{p\_*}.

EXAMPLE
This AU program processes the current 1D dataset and plots it according to the XWIN-PLOT layout specified in \texttt{edo}:

\begin{verbatim}
GETCURDATA
EF
APK
SREF
ABS
AUTOPLOT
QUIT
\end{verbatim}

SEE ALSO
AUTOPLOT\_TO\_FILE, AUTOPLOT\_WITH\_PORTFOLIO, AUTOPLOT\_WITH\_PORTFOLIO\_TO\_FILE
AUTOPLOT_TO_FILE

NAME

AUTOPLOT_TO_FILE - as AUTOPLOT but store the output into a file

SYNTAX

AUTOPLOT_TO_FILE(filename)

DESCRIPTION

The macro AUTOPLOT_TO_FILE plots the current dataset according to the XWIN-plot layout defined by the `edo` parameter LAYOUT. The output is not sent to the printer but stored in the file that is specified as an argument. The argument is normally a full pathname. If it is not, the file is stored in the XWIN-NMR home directory.

If the filename has the extension `.ps`, the output is stored as a postscript file. If (under Windows) it has the extension `.emf`, as in the example below, the output will be stored as an enhanced metafile.

AUTOPLOT_TO_FILE is actually a composite macro that consists of several PORTFOLIO*/AUTOPLOT* macros. This, however, is transparent to the user.

EXAMPLE

This AU program processes the current 1D dataset and plots it according to the XWIN-plot layout specified in `edo`. The result is stored in an enhanced metafile.

```
GETCURDATA

EF
APK
SREF
ABS
AUTOPLOT_TO_FILE("/users/guest/mydata.emf")
QUIT
```

SEE ALSO

AUTOPLOT, AUTOPLOT_WITH_PORTFOLIO
DECLARE_PORTFOLIO

NAME
DECLARE_PORTFOLIO - initialize the use of XWIN- PLOT portfolio macros

SYNTAX
DECLARE_PORTFOLIO

DESCRIPTION
The macro DECLARE_PORTFOLIO initializes the use of other XWIN- PLOT portfolio AU macros. It must be inserted at the beginning of the AU program (before GETCURDATA). Note that this macro initializes portfolio use but does not create one.

EXAMPLE
This AU program plots the current dataset according to the XWIN- PLOT layout specified in edo. It just is a simple demonstration of the use of PORTFOLIO macros.

DECLARE_PORTFOLIO
GETCURDATA
CREATE_PORTFOLIO(“/temp/myPortfolio.por”)
ADD_CURDAT_TO_PORTFOLIO
CLOSE_PORTFOLIO
AUTOPLOT_WITH_PORTFOLIO
QUIT

Note that this AU program does the same as the command autoplot.

SEE ALSO
CREATE_PORTFOLIO, ADD_TO_PORTFOLIO, CLOSE_PORTFOLIO
CREATE_PORTFOLIO

NAME
CREATE_PORTFOLIO - create a XWIN-PLOT portfolio

SYNTAX
CREATE_PORTFOLIO(name)

DESCRIPTION
The macro CREATE_PORTFOLIO creates the XWIN-PLOT portfolio that is specified as an argument. It takes one argument; the filename of the portfolio.

The argument is normally specified as a full pathname. If it is not, the portfolio is stored under the XWIN-NMR home directory. If the specified file already exists, it is overwritten. Note that CREATE_PORTFOLIO creates the portfolio but does not insert any dataset specifications.

EXAMPLE
This AU program plots the current dataset according to the XWIN-PLOT layout specified in edo. It is just a simple demonstration of the use of PORTFOLIO macros.

DECLARE_PORTFOLIO
GETCURDATA
CREATE_PORTFOLIO("/temp/myPortfolio.por")
ADD_CURDAT_TO_PORTFOLIO
CLOSE_PORTFOLIO
AUTOPLOT_WITH_PORTFOLIO
QUIT

Note that this AU program does the same as the command autoplot.

SEE ALSO
DECLARE_PORTFOLIO, ADD_TO_PORTFOLIO, CLOSE_PORTFOLIO
ADD_CURDAT_TO_PORTFOLIO

NAME
ADD_CURDAT_TO_PORTFOLIO - add the current dataset to the portfolio

SYNTAX
ADD_CURDAT_TO_PORTFOLIO

DESCRIPTION
The macro ADD_CURDAT_TO_PORTFOLIO adds the current dataset to the XWIN- PLOT portfolio that was previously create with CREATE_PORTFOLIO.

EXAMPLE
This AU program plots two datasets, the current and next processing number of the current data name, according to the XWIN- PLOT layout specified in edo.

DECLARE_PORTFOLIO
GETCURDATA
CREATE_PORTFOLIO("/temp/myPortfolio.por")
ADD_CURDAT_TO_PORTFOLIO
IPROCNO
ADD_CURDAT_TO_PORTFOLIO
CLOSE_PORTFOLIO
AUTOPlot_WiTH_PORTFOLIO
QUIT

SEE ALSO
DECLARE_PORTFOLIO, CREATE_PORTFOLIO, CLOSEPORTFOLIO
ADD_TO_PORTFOLIO

NAME
ADD_TO_PORTFOLIO - add the specified dataset to the portfolio

SYNTAX
ADD_TO_PORTFOLIO(disk, user, name, expno, procno)

DESCRIPTION
The macro ADD_TO_PORTFOLIO adds a dataset to the portfolio that was previously created with CREATE_PORTFOLIO. The dataset to be added is completely specified by the five arguments of ADD_TO_PORTFOLIO. Note that these arguments can be constants (values) or variables.

EXAMPLE
This AU program plot two datasets according to the XWIN- PLOT layout specified in edo. Note that the first dataset to be plotted is the so called second dataset (edc2), specified by the predefined dedicated variables disk2, user2 etc.

DECLARE_PORTFOLIO
GETCURDATA
CREATE_PORTFOLIO("/temp/myPortfolio.por")
ADD_TO_PORTFOLIO(disk2, user2, name2, expno2, procno2)
ADD_TO_PORTFOLIO("C:/xw", "guest", "mydata", 1, 3)
CLOSE_PORTFOLIO
AUTOPLOT_WITH_PORTFOLIO
QUIT

SEE ALSO
ADD_CURDAT_TO_PORTFOLIO
CLOSE_PORTFOLIO

NAME
CLOSE_PORTFOLIO - closes the portfolio definition

SYNTAX
CLOSE_PORTFOLIO

DESCRIPTION
The macro CLOSE_PORTFOLIO closes the definition of the portfolio that was previously created with CREATE_PORTFOLIO. It must be used after the last ADD_CURDAT_TO_PORTFOLIO or ADD_TO_PORTFOLIO macro and before the first AUTOPLOT* macro.

EXAMPLE
This AU program plots the current dataset according to the XWIN-PLOT layout specified in edo. It is just a simple demonstration of the use of PORTFOLIO macros.

DECLARE_PORTFOLIO
GETCURDATA
CREATE_PORTFOLIO(“/temp/myPortfolio.por”)
ADD_CURDAT_TO_PORTFOLIO
CLOSE_PORTFOLIO
AUTOPLOT_WITH_PORTFOLIO
QUIT

Note that this AU program does the same as the command autoplot.

SEE ALSO
DECLARE_PORTFOLIO, CREATE_PORTFOLIO, ADD_TO_PORTFOLIO
AUTOPLOT_WITH_PORTFOLIO

NAME
AUTOPLOT_WITH_PORTFOLIO - plot the dataset(s) of the current portfolio

SYNTAX
AUTOPLOT_WITH_PORTFOLIO

DESCRIPTION
The macro AUTOPLOT_WITH_PORTFOLIO plots the dataset(s) defined in the portfolio created with CREATE_PORTFOLIO according to the XWIN- PLOT layout defined by the edo parameter LAYOUT.

EXAMPLE
This AU program plots the current dataset according to the XWIN- PLOT layout specified in edo. It is just a simple demonstration of the use of PORTFOLIO macros.

DECLARE_PORTFOLIO
GETCURDATA
CREATE_PORTFOLIO(“/temp/myPortfolio.por”) ADD_CURDAT_TO_PORTFOLIO CLOSE_PORTFOLIO AUTOPLOT_WITH_PORTFOLIO QUIT

Note that this AU program does the same as the command autoplot.

SEE ALSO
AUTOPLOT, AUTOPLOT_WITH_PORTFOLIO_TO_FILE
AUTOPLOT_WITH_PORTFOLIO_TO_FILE

NAME

AUTOPLOT_WITH_PORTFOLIO_TO_FILE - plot the dataset(s) of the current portfolio and store the output into a file

SYNTAX

AUTOPLOT_WITH_PORTFOLIO_TO_FILE(filename)

DESCRIPTION

The macro AUTOPLOT_WITH_PORTFOLIO_TO_FILE plots the dataset(s) defined in the XWIN-PLOT portfolio that was previously created with CREATE_PORTFOLIO. The plot is made according to the layout defined by the edo parameter LAYOUT. The output is stored in the file that is specified as an argument to the macro. The argument is normally a full pathname. If it is not, the portfolio is stored under the XWIN-NMR home directory.

If the filename has the extension .ps, as in the example below, the output will be stored as a postscript file. If (under Windows) it has the extension .emf, the output is stored as an enhanced metafile.

EXAMPLE

This AU program plots the current dataset according to the XWIN-PLOT layout specified in edo and stores the result into a postscript file.

DECLARE_PORTFOLIO
GETCURDATA
CREATE_PORTFOLIO("/temp/myPortfolio.por")
ADD_CURDAT_TO_PORTFOLIO
CLOSE_PORTFOLIO
AUTOPLOT_WITH_PORTFOLIO_TO_FILE("/users/guest/mydata.ps")
QUIT

SEE ALSO AUTO

PLOT_WITH_PORTFOLIO, AUTOPLOT_TO_FILE
Chapter 10

Macros prompting the user for input.

This chapter contains a description of all AU macros which can be used to prompt the user for input and store the input into an AU variable. Different macros are available for requesting integer, float, double or text values.
GETINT

NAME
GETINT - prompt the user to enter an integer value

SYNTAX
GETINT("Please enter an integer value : ", i1)

DESCRIPTION
The macro GETINT prompts the user to enter an integer value and stores this value into an integer variable. It can be used for various purposes, for example to set the value of an XWIN-NMR (integer) parameter or to specify the number of cycles in an AU program loop. GETINT takes two arguments:

1. a text string which prompts the user to enter an integer value
2. an integer variable into which the value is stored

EXAMPLE
The following AU program prompts the user for the number of scans per acquisition and the number of experiments to be done:

GETCURDATA
GETINT("Please enter the number of scans: ", i1)
GETINT("Please enter the number of experiments: ", i2)
TIMES(i2)
STOREPAR("NS", i1)
ZG
IEXPNO
END
QUIT

SEE ALSO
GETFLOAT - prompt the user to enter a float value
GETDOUBLE - prompt the user to enter a double value
GETSTRING - prompt the user to enter a text string
GETFLOAT/GETDOUBLE

NAME
GETFLOAT - prompt the user to enter a float value
GETDOUBLE - prompt the user to enter a double value

SYNTAX
GETFLOAT(text, f1)
GETDOUBLE(text, d1)

DESCRIPTION
The macro GETFLOAT prompts the user to enter a float value and stores this value into a float AU variable. It is used to get the value for an XWIN-NMR (float) parameter which can then be stored with STOREPAR. GETFLOAT takes 2 arguments:

1. a text string which prompts the user to enter an float value
2. the float variable into which the value is store

The description for GETDOUBLE is equivalent, except that it is used for XWIN-NMR (double) parameters.

EXAMPLE
The following AU program prompts the user for the frequency offset and Gaussian broadening, stores these values into the parameters O1 and GB respectively and then runs an acquisition, Gaussian multiplication and Fourier transform:

GETCURDATA
GETDOUBLE("Please enter the frequency offset:", d1)
STOREPAR("o1", d1);
GETFLOAT("Please enter the Gaussian broadening:", f1)
STOREPAR("GB", f1)
ZG
GM
FT
QUIT
SEE ALSO

GETINT - prompt the user to enter an integer value
GETSTRING - prompt the user to enter a text string
GETSTRING

NAME
GETSTRING - prompt the user to enter a text string

SYNTAX
GETSTRING(text, cval)

DESCRIPTION
The macro GETSTRING prompts the user to enter a text string which is then stored into an AU variable. It can be used for various purposes, for example to ask the user a question or prompt the user to enter a name. GETINT takes two arguments:

1. a text string which prompts the user to enter a text string
2. the character-string variable into which the value is stored

EXAMPLE
The following AU program asks the user if an integration must be done and, if yes, which intrng file must be used. Then the integrals are calculated and listed:

    char answer[8];
    GETCURDATA
    GETSTRING("Do you want to do an integration (yes/no)?", answer)
    if ( !strcmp(answer,"yes") )
    {
        GETSTRING("Which intrng file must be used?", text)
        RMISC("intrng", text)
        LI
    }
    QUIT

SEE ALSO
GETINT - prompt the user to enter an integer value
GETFLOAT - prompt the user to enter a float value
GETDOUBLE - prompt the user to enter a double value
Macros prompting the user for input.
Chapter 11

Bruker library functions

This chapter contains a description of various C functions which are available as part of Bruker libraries. Several of them concern the handling of dataset lists or directory lists. You can, for instance, get a list of filenames, display it, select a file from the list and then copy the file to a different directory. The functions described in this chapter are particularly useful for files located in the directories /<xwhome>/conf and /<xwhome>/exp. For copying datasets, we recommend to use the macros described in Chapter 4.
CalcExpTime/PrintExpTime

NAME

CalcExpTime - calculate the experiment time for the current experiment
PrintExpTime - print the experiment time for the current experiment

SYNTAX

static void PrintExpTime();
int CalcExpTime();
void PrintExpTime (int exptime, int expno);
#include<inc/exptutil>

DESCRIPTION

The function CalcExpTime calculates the experiment time for the current experiment. The return value is the experiment time in seconds. The function PrintExpTime can be used to print the experiment time in the form "days hours minutes seconds".

EXAMPLE

The following AU program calculates and prints the experiment time of a sequence of 10 experiments starting with the foreground dataset.

    static void PrintExpTime();
    GETCURDATA
    TIMES(10)
        PrintExpTime(CalcExpTime(),loopcount1);
    IEXPNO
    END
    QUIT
    #include<inc/exptutil>

Note that the declaration of PrintExpTime must appear at the beginning of the AU program and the #include statement at the end of the AU program.

SEE ALSO

multiexpt - a standard Bruker library AU program
check_pwd, GetNmrSuperUser

NAME

check_pwd - prompt the user to enter a password
GetNmrSuperUser - get the name of the XWIN-NMR NMR superuser

SYNTAX

int check_pwd (char *username);
char *GetNmrSuperUser();

DESCRIPTION

The function check_pwd prompts the user for the password of the specified user. The return value is 0, if the correct password was entered. In all other cases, the return value is -1. This function can be combined with the function GetNmrSuperUser which returns the name of the XWIN-NMR NMR superuser.

EXAMPLE

GETCURDATA
if ( check_pwd (GetNmrSuperUser()) != 0)
{
    Proc_err(DEF_ERR_OPT,"Sorry, you are not privileged");
    ABORT
}
else
{
    Proc_err(DEF_ERR_OPT,"OK, you may proceed");
}
QUIT
getdir

NAME

getdir - get all file names and/or directory names within a directory

SYNTAX

int getdir (char *directory, char ***filelist, char *match-code);

DESCRIPTION

The function getdir opens a directory and gets all file and directory names in that directory. This list is stored in a 2 dimensional character-string variable which can be evaluated by subsequent AU statements. The list can be limited by specifying a match-code; only names matching this string are entered into the list. getdir takes three arguments:

1. the source directory
2. the variable into which the list of names is stored
3. the match-code; an arbitrary string of characters

The third argument, can also be "/files" to get all files but not directories, or "/dir" to get all directories but not files.

The return value of getdir is the number of successfully matched file names and/or directory names.

The function getdir is frequently used in connection with theuxselect function which is also described in this chapter. getdir internally allocates memory for the list of names. Officially, you must free this memory with the Bruker library function freedir. In practice, however, you can omit freedir because all memory allocated by the AU program is automatically freed when the AU program finishes.

EXAMPLES

The following AU statements will create a list of experiment directories from an XWIN-NMR dataset. All entries are returned because no match-code was specified.

    char sourcedir[200], **listfile;
    GETCURDATA
The following AU statements will create a list of shim files starting with the letters a to p from the bsms directory.

```c
char sourcedir[200], **listfile;
GETCURDATA
sprintf (sourcedir, "%s/lists/bsms/", getstan(0,0));
i1 = getdir (sourcedir,&listfile,"[a-p]*");
```

The following AU statement will create a list of all files but not directories from the users home directory.

```c
i1 = getdir (getenv("HOME"),&listfile,"/files");
```

The following AU statement will return a list of all directory names from the users home directory.

```c
i1 = getdir (getenv("HOME"),&listfile,"/dir");
```

**SEE ALSO**

uxselect - display a list from which an entry can be selected by mouse-click  
freedir - free memory allocated by getdir
freedir

NAME

freedir - free memory allocated by getdir

SYNTAX

void freedir (char **listfile);

DESCRIPTION

The function freedir frees the memory that was allocated by a getdir function call.

EXAMPLE

See the example under the function uxselect.

SEE ALSO

gendir - get all file names and/or directory names within a directory
uxselect - display a list from which an entry can be selected by mouse-click
uxselect

NAME

uxselect - display a list from which an entry can be selected by mouse-click

SYNTAX

char *uxselect (char *sourcedir, char **listfile, char *headerline, int mode);

DESCRIPTION

The function uxselect displays a list of names, like file names or directory names, and allows the user to select one of these names. Depending on the mode with which uxselect was called the entries in the list can be viewed, renamed, deleted, new entries can be added etc. Note that uxselect does not create the list of names. Before you use uxselect in an AU program, you must first create such a list with another AU statement, e.g. with the function getdir.

The function uxselect takes four arguments:

1. the directory in which the listed files/directories reside
2. the variable containing the list of names
3. the header line which appears at the top of the selection window
4. the mode which determines the layout and functionality of the selection window

The first argument must be an actual directory name when the fourth argument (mode) is SEL_DIR_SUB, SEL_WR_CONF, SEL_RENAME or SEL_DELETE. Only in those cases, the uxselect function needs to know the source directory because it will access it. For all other modes, uxselect will simply use the list of names as it was created by a previous AU statement (like getdir) and the first argument of can be NULL.

The return value of uxselect is a character-string, usually the selected file or the directory name. The return value can be assigned to a variable and used in subsequent AU statements. If, however, you close the selection window by clicking the Cancel button the return value of uxselect is NULL.

The following modes are available.

- SEL_READ 0
A list of names is displayed in `select` mode. If you select an entry it becomes the return value of `uxselect` and can be used in subsequent AU statements.

- **SEL_WRITE** 1

A list of names is displayed in `select` mode. Furthermore, a new name can be entered at the bottom of the screen. If you select an entry or enter a new name, this will be the return value of `uxselect`. Note that entering a new name does not automatically create a file or directory with this name. You can use the return value in subsequent AU statements.

- **SEL_DIR ONLY** 2

A list of names is displayed in `view only` mode. No entry can be selected, but the whole listing can be printed out.

- **SEL_DIR SUB** 3

A list of directory names is displayed in `view only` mode. This mode can only be used for lists of directory names, not file names. All entries in the directories (files and subdirectories) are also displayed.

- **SEL_RD_WIDE** 4

A list of names is displayed in `select` mode in an extra wide window. If the user selects an entry, this becomes the return value of `uxselect`. This mode is similar to SEL_READ.

- **SEL_WR_CONF** 5

A list of names is displayed in `select` mode. Furthermore, a new name can be entered at the bottom of the screen. If you select an entry or enter a new name, the overwrite permissions of the corresponding file or directory are checked. If it is writable, the name will be the return value of `uxselect`. This mode is used in the XWIN-NMR command `wpar`. There are hardly any applications for this mode in AU programs.

- **SEL_RENAME** 6

A list of names is displayed in `write` mode. Selecting an entry allows to rename it.

- **SEL_ED_WIDE** 7

The list of names is displayed in `write` mode. An empty entry is added in which a new name can be entered. Every entry in the list can be renamed. This mode is used in the XWIN-NMR command `edhead`. There are hardly
any applications for this mode in AU programs.

- **SEL_ALL** 8

  A list of names is displayed in select mode. You can select a single entry or click the button **Select all** to select all entries.

- **SEL_DELETE** 9

  A list of names is displayed in select mode. The selection window contains the buttons **execute** and **select all**. When you select an entry and click on **execute** the corresponding file or directory is deleted. When you click **select all** and then **execute**, all files/directories in the list are deleted.

We strongly recommend to specify the mode with its symbolic name rather than with its number. The reason is that the numbers might change in future releases of XWIN-NMR, but the symbolic names will not.

**EXAMPLE**

The following AU program will get a listing of all shim files with the extension .mike and will display this list in a selection window. If an entry is selected, then the corresponding shim file is read with the macro RSH. If no entries were found or selected, the AU program aborts.

```c
char sourcedir[200], **listfile, *answer;

GETCURDATA
sprintf (sourcedir, "%s/lists/bsms/", getstan(0,0));
if ( (i1 = getdir (sourcedir,&listfile,"*.mike")) <= 0 )
{
    Proc_err (DEF_ERR_OPT,"No shim files with extension .mike found");
    ABORT
} else
{
    if ( (answer = uxselect(NULL,listfile,"shim files",SEL_READ)) != 0 )
    {
        RSH(answer)
    }
}
freedir (listfile);
QUIT
```
SEE ALSO

getdir - get all file names and/or directory names within a directory
getstan - return the pathname to the user’s current experiment directory
**dircp**

**NAME**

dircp - copy a file
dircp_err - return the error message that corresponds to the error return value of a dircp function call

**SYNTAX**

dircp (char *sourcefile, char *targetfile);
char *dircp_err (int return-value);

**DESCRIPTION**

The function dircp copies the sourcefile into the targetfile. If the targetfile exists, it will be overwritten. A negative number is returned if copying was not possible. The function dircp_err will return the corresponding C error message. A return value of 0 indicates successful execution.

**EXAMPLE**

The following AU program copies the title file of the foreground dataset to the users home directory.

```c
char sourcefile[200], targetfile[200];

GETCURDATA
sprintf (sourcefile, "%s/data/%s/%s/%s%d/pdata/%d/title", disk,user,type,name,expno,procno);

sprintf (targetfile, "%s/title",getenv("HOME"));

if ( (i1 = dircp (sourcefile,targetfile)) < 0 )
  Proc_err (DEF_ERR_OPT, dircp_err(i1));
QUIT
```
fetchstorpl

NAME

fetchstorpl - read or store one or several plot parameters

SYNTAX

int fetchstorpl (char *directory, int mode, int where, varargs);

DESCRIPTION

The function fetchstorpl can read or write one or several plot parameters in one function call. This routine is primarily used when several plot parameters are to be stored. fetchstorpl is also used when plot parameters are to be stored in a parameter set rather than an XWIN-NMR dataset. If you simply want to get or store one plot parameter, you can use the FETCHPLPAR and STOREPLPAR macros, respectively.

fetchstorpl takes four arguments:

1. directory
   Must be curdat if the third argument (where) is 0
   Must be the pathname to a parameter set if the third argument (where) is 1

2. mode
   Must be 0 if the parameters are to be stored.
   Must be 1 if the parameters are to be read.

3. where
   Must be 0 if the parameters are to be stored in a dataset.
   Must be 1 if the parameters are to be stored in a parameter set.

4. varargs consists of two parts
   - A list of plot parameters separated by white spaces. This list must be double quoted with " " which makes it a character-string.
   - A list of values or variables, all separated by commas. Each value or variable must match the type of parameter from the above mentioned list.

EXAMPLE

- This example stores CX and CY in the parameter set PROTON.AB:
  (void) sprintf (text,"%s/par/PROTON.AB",getstan(0,0));
i1 = fetchstorpl(text,0,1,"CX CY",20.0,12.0);
Note that this will only work, if the parameter set is writable for the user
who runs the AU program.

- This example stores CX and CY in the current AU program dataset:
  i1 = fetchstorpl(curdat,0,0,"CX CY",20.0,12.0);
  Note that the variable curdat has a special meaning. In any AU program,
curdat always and automatically refers to the current AU program dataset.

- This example reads SXLLEFT, SHEI, DHEI and CY from the current AU
  program dataset:
  float sxleft, shei, dhei, cy;
  i1 = fetchstorpl(curdat,1,0,"SXLLEFT SHEI DHEI CY",
                  &sxleft,&shei,&dhei,&cy);

SEE ALSO
FETCHPLPAR - get a plot parameter
STOREPLPAR - store a plot parameter
stack1d - generate a stacked plot of 1D spectra from a series of experiments
**gethighest**

**NAME**

gethighest - return the next highest unused experiment number of a dataset

**SYNTAX**

```c
int gethighest (char *directory);
#include <inc/sysutil>
```

**DESCRIPTION**

The function `gethighest` scans a directory for all subdirectories whose name is a number and returns then returns the next highest unused number. `gethighest` is typically used to scan a dataset name directory of an XWIN-NMR dataset. In that case, it returns the highest unused experiment number. If, for example, the highest used experiment number is 56, the function will return the value 57. The function can also be used to return the highest unused processing number of a dataset.

**EXAMPLE**

The following AU program will copy the current XWIN-NMR experiment into the next highest unused experiment dataset.

```c
GETCURDATA
(void) sprintf (text,"%s/data/%s/nmr/%s",disk,user,name);
i1 = gethighest (text);
WRA(i1)
QUIT
#include <inc/sysutil>
```

Note that the `#include` statement must be included at the end of the AU program.
**getstan**

**NAME**

*getstan* - return the pathname to the user’s current experiment directory

**SYNTAX**

```
char *getstan (char *pathname, const char *subdir);
```

**DESCRIPTION**

The function *getstan* returns the pathname to the user’s current experiment directory. The returned pathname can be concatenated with a known subdirectory pathname as a part of the same *getstan* function call.

**EXAMPLE**

The following AU program will get the pulse program of the current AU dataset. It will then prompt the user to confirm the name of the pulse program or to enter a new name. Finally, the pulse program will be shown in an XWIN-NMR window.

```
char pulprog[80];
GETCURDATA
FETCHPAR("PULPROG",pulprog)
GETSTRING ("Enter the name of the pulse program: ",pulprog);
(void) sprintf (text,"%s/%s",getstan (NULL,"lists/pp"),pulprog);
showfile (text);
QUIT
```

**NOTE**

In the above example, the function call *getstan* (NULL,"lists/pp") returns the pathname /<xwhome>/exp/stan/nmr/lists/pp. The function call *getstan*(NULL,NULL) returns /<xwhome>/exp/stan/nmr/.

**SEE ALSO**

PathXWinNMR - a class of functions which return pathnames to certain XWIN-NMR directories

*multi_integ* - an AU program for multiple integrations in AI mode.
getxwinvers

NAME
  getxwinvers - return the current version and patchlevel of XWIN-NMR

SYNTAX
  int getxwinvers (char *curversion);
  #include <inc/sysutil>

DESCRIPTION
  The function getxwinvers returns the version and patchlevel of the currently
  running XWIN-NMR program into the variable curversion. This variable can
  then be printed out.

EXAMPLE
  The following AU program prints the current version and patchlevel in the status
  line of XWIN-NMR.

  char curversion[80];
  GETCURDATA
  i1 = getxwinvers(curversion);
  show_status (curversion);
  QUIT
  #include <inc/sysutil>

  Note that the #include statement must be included at the end of the AU program.
**mkudir**

**NAME**

*mkudir* - create a complete directory path

**SYNTAX**

```c
int mkudir (char *directory);
```

**DESCRIPTION**

The function *mkudir* scans the specified directory for the last `/`. Then it checks recursively for the existence of all components of the directory path and creates them if necessary. The function returns -1 if an error occurs, otherwise 0.

If the full pathname is to be created, then the directory must end with a `/` (see the example below).

**EXAMPLE**

The following AU program will create a dataset which has an experiment number one higher than the current foreground dataset.

```c
GETCURDATA
(void) sprintf (text,"%s/data/%s/nmr/%s/d/pdata/%d",
    disk,user,name,expno+1,procno);
if (mkudir(text) < 0)
    Proc_err (DEF_ERR_OPT, "could not create :\n%s",text);
QUIT
```
PathXWinNMR

NAME

PathXWinNMR - a class of functions which return pathnames to certain XWIN-NMR directories

SYNTAX

char *PathXWinNMRCConf ();
char *PathXWinNMRCurDir ();
char *PathXWinNMRDotXWinNMR ();
char *PathXWinNMRExp ();
char *PathXWinNMRPlot ();
char *PathXWinNMRProg ();

DESCRIPTION

The above functions return pathnames to certain XWIN-NMR mostly subdirectories of the XWIN-NMR directory <xwhome>. For a standard installation, <xwhome> is:
- on UNIX systems: /u
- on Windows systems: C:\Bruker

For a user-defined installation, <xwhome> can be any directory. The following table lists the directory pathnames returned by the above functions. For examples, please check the Bruker AU program library.

char *PathXWinNMRCConf : returns /xwhome/conf
char *PathXWinNMRCurDir : returns /xwhome/prog/curdir
char *PathXWinNMRDotXWinNMR : returns $HOME/xwinmr-hostname
char *PathXWinNMRExp : returns /xwhome/exp
char *PathXWinNMRPlot : returns /xwhome/plot
char *PathXWinNMRProg : returns /xwhome/prog

SEE ALSO

getstan - return the pathname to the user’s current experiment directory
**pow_next**

**NAME**

`pow_next` - round to the next larger power of two

**SYNTAX**

```c
int pow_next (int i1);
#include <inc/sysutil>
```

**DESCRIPTION**

The function `pow_next` takes `i1` and rounds it to the next larger integer value which is a power of two. The return value of the function is this power of two value. The function has no error handling. If `i1` is smaller than 1, then the function will return 1.

**EXAMPLE**

The following AU program will return 8192 in `i2` because this is the next larger number (compared to `i1`) which is a power of two.

```au
GETCURDATA
i1 = 7000;
i2 = pow_next(i1);
QUIT
#include <inc/sysutil>
```

Note that the `#include` statement must be included at the end of the AU program.
Proc_err

NAME
Proc_err - show an error message in a window on the XWIN-NMR screen

SYNTAX
int Proc_err (int flag, char *format);
int Proc_err (int flag, char *format, varargs);

DESCRIPTION
The function Proc_err can be used to construct an error message which will be displayed in a window on the XWIN-NMR screen. The function takes two or three arguments:

1. a flag which determines the type and the number (2 or 3) of buttons in the error window.
2. the error message to be displayed. If this argument contains %d, %f, or %s statements, then Proc_err needs a third argument which provides the corresponding variables.
3. variables who’s values replace the corresponding %d, %f, or %s statements of the second argument.

The first argument (flag) can have the following values:

- **DEF_ERR_OPT**
  The error window has one button (OK). The AU programs holds until the user clicks **OK**.

- **INFO_OPT**
  The error window has one button (Seen). The AU program continues but the error window remains on the screen until it is cleared by another error window or the user clicks **Seen**.

- **QUESTION_OPT**
  The error window has two buttons, OK and CANCEL. Proc_err returns ERR_OK (0) if the **OK** button is clicked and ERRCANCEL (-1) if the **CANCEL** button is clicked. The return value is normally used by subsequent control statements to decide whether or not to continue the AU program.
Note that the message in the `Proc_err` window is constructed in the same way as the C function `sprintf` constructs its strings.

**EXAMPLE**

The following examples show several possibilities of constructing error messages for the `Proc_err` function call.

- **Example for DEF_ERR_OPT:**
  
  ```c
  (void) sprintf (text,"%s/data/%s/nmr/%s/d/pdata/%d/",
    disk,user,name,expno+1,procno);
  Proc_err (DEF_ERR_OPT, "could not create :\n%s",text);
  ```

- **Example for QUESTION_OPT:**
  
  ```c
  i1 = Proc_err(QUESTION_OPT,"Continue with the AU program ?\n  Click OK to continue, click cancel stop");
  if ( i1 == ERR_OK )
  {
    /* Further AU statements */
  }
  if ( i1 == ERR_CANCEL )
  {
    ABORT
  }
  ```

- **Example for INFO_OPT:**
  
  ```c
  i1 = 7;
  i2 = 5;
  Proc_err(INFO_OPT,"%d is bigger than %d",i1,i2);
  ```

**SEE ALSO**

- `Show_status` - show a string in the status line of XWIN-NMR

All AU programs from the Bruker AU program library which use `Proc_err`
Show_status

NAME

Show_status - show a string in the XWIN-NMR status line

SYNTAX

void Show_status (char *text);

DESCRIPTION

The function Show_status displays the specified text in the XWIN-NMR status line. This function can be used as an alternative to the Proc_err function. One difference to Proc_err is that there is no window that needs to be acknowledged.

EXAMPLE

The following AU program will display the line "The AU program test has started" in the status line of XWIN-NMR:

    GETCURDATA
    (void) strcpy(text,"The AU program test has started");
    Show_status (text);
    QUIT

SEE ALSO

Proc_err - show a message in a window on the XWIN-NMR screen
showfile

NAME

showfile - show the contents of a file in an XWIN-NMR window

SYNTAX

int showfile (char *file);

DESCRIPTION

The function showfile reads the specified file and displays it in an XWIN-NMR window. This display is a read-only display, so the file cannot be changed.

EXAMPLE

The following AU program will show the title file in an XWIN-NMR window.

GETCURDATA
(void) sprintf (text, "%s/data/%s/nmr/%s/d/pdata/%d/title",
disk, user, name, expno, procno);
i1 = showfile (text);
QUIT
ssleep

NAME
ssleep - pause in an AU program for a certain number of seconds

SYNTAX
int ssleep (int seconds);

DESCRIPTION
The function ssleep will cause the AU program to wait with the execution of the next statement until the specified number of seconds has elapsed.

EXAMPLE
The following AU program will wait for 3 minutes before it resumes execution.

    GETCURDATA
    i1 = ssleep (180);
    EFP
    QUIT

SEE ALSO
WAIT_UNTIL - hold the AU program until the specified date and time
**unlinkpr**

**NAME**

unlinkpr - delete all processed data files (1r, 1i, 2rr, 2ii etc.) of a dataset

**SYNTAX**

```c
int unlinkpr(char *directory);
#include <inc/sysutil>
```

**DESCRIPTION**

The function `unlinkpr` deletes all processed data files (1r, 1i, 2rr, 2ii, 2ri, 2ir, dsp, dsp.hdr, dsp_low) in the specified dataset directory. There is no error check whether the files could be deleted; the return value of the function is always 0 and can be ignored.

**EXAMPLE**

The following AU program will delete the processed data files of the foreground dataset.

```c
GETCURDATA
(void) sprintf (text, "%s/data/%s/nmr/%s/d/pdata/%d", 
  disk,user,name,expno,procno);
i1 = unlinkpr (text);
QUIT
#include <inc/sysutil>
```

Note that the `#include` statement must be included at the end of the AU program.
Chapter 12

List of Bruker AU programs

12.1 Short description of all Bruker AU programs

This chapter contains a list with the names and short-descriptions of all Bruker library AU programs. This list was made for XWIN-NMR 3.1. Some AU programs are not available for older versions of XWIN-NMR.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z-spoil</td>
<td>Set a Z-spoil value within the SCM.</td>
</tr>
<tr>
<td>abs2.water</td>
<td>Performs an F2 baseline correction on a 2D dataset left and right of the water peak.</td>
</tr>
<tr>
<td>abs2D</td>
<td>Performs a baseline correction on a 2D dataset in both dimensions.</td>
</tr>
<tr>
<td>acqu_fid_ser</td>
<td>Acquires a single FID of the current 2D experiment and replaces the old fid in the ser file.</td>
</tr>
<tr>
<td>acquulist</td>
<td>Set up and start acquisitions using f1, f2, f3, vt, vc, vd, vp lists.</td>
</tr>
<tr>
<td>amplsttab</td>
<td>Calculates the amplitude stability based on a peaklist file.</td>
</tr>
<tr>
<td>angle</td>
<td>Perform multiple acquisitions and ft’s. This program is particularly interesting when you want to adjust the magic angle for MAS type experiments.</td>
</tr>
<tr>
<td>ascllev</td>
<td>Converts the level file in the current processed data directory to ASCII and writes it to the file</td>
</tr>
<tr>
<td>au_get1d</td>
<td>Acquire sweep width optimized 1D spectra.</td>
</tr>
<tr>
<td>au_getlcosy</td>
<td>Acquire sweep width optimized COSY spectra.</td>
</tr>
<tr>
<td>au_getlinv</td>
<td>Acquire sweep width optimized 2D inverse spectra.</td>
</tr>
<tr>
<td>au_getlxhco</td>
<td>Acquire sweep width optimized XH correlated spectra.</td>
</tr>
<tr>
<td>au_mult</td>
<td>AU program for C13 multiplicity analysis.</td>
</tr>
<tr>
<td>au_noediff</td>
<td>noe difference spectroscopy using different expnos.</td>
</tr>
<tr>
<td>au_noemult</td>
<td>noe difference spectroscopy with multiple irradiation points for each multiplet using different expnos.</td>
</tr>
<tr>
<td>au_water</td>
<td>Acquire water-suppression spectra for use in foreground (xau,xaua).</td>
</tr>
<tr>
<td>au_watersc</td>
<td>Acquire water-suppression spectra for use in automation, e.g., with sample changer.</td>
</tr>
<tr>
<td>au_zg</td>
<td>General AU program for data acquisition.</td>
</tr>
<tr>
<td>au_zg135</td>
<td>Acquire DEPT135 type spectra.</td>
</tr>
<tr>
<td>au_zgcosy</td>
<td>Acquire COSY type spectra.</td>
</tr>
<tr>
<td>Program</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>au_zgglp</td>
<td>Automatic data evaluation according to GLP standards. This AU program takes O1, SW and O2 as arguments and then works like au_zg.</td>
</tr>
<tr>
<td>au_zgnr</td>
<td>Acquisition with rotation switched off.</td>
</tr>
<tr>
<td>au_zgonly</td>
<td>General AU program for data acquisition.</td>
</tr>
<tr>
<td>au_zgsino</td>
<td>Acquisition with signal to noise break up.</td>
</tr>
<tr>
<td>au_zgte</td>
<td>Acquisition with temperature setting.</td>
</tr>
<tr>
<td>aunmp_tojdx</td>
<td>Used in LIMS automation to process data. First, AUNMP is executed, then, if specified, the command given on the command line.</td>
</tr>
<tr>
<td>autoflist</td>
<td>Automatic generation of a frequency list for the peaks in the plot region of the spectrum.</td>
</tr>
<tr>
<td>autot1</td>
<td>Automatic processing of a 2D T1/T2 experiment with subsequent T1/T2 calculation.</td>
</tr>
<tr>
<td>bintoasc</td>
<td>Converts experiments from /&lt;xwhome&gt;/exp/stan/nmr/par containing old binary metafiles to an experiment which then contains the same plot parameters but in ASCII format.</td>
</tr>
<tr>
<td>bsms_exam</td>
<td>Example AU program which shows how to use low level functions to read or write BSMS parameters.</td>
</tr>
<tr>
<td>bsms_vtu_exam</td>
<td>Example AU program which shows how to use low level functions to read or write BSMS or VTU (BVT1000/BDTC) parameters.</td>
</tr>
<tr>
<td>butselau</td>
<td>Acquisition with butselnmr (buttonmr for selective experiments) from within XWIN-NMR.</td>
</tr>
<tr>
<td>buttonau</td>
<td>Acquisition with buttonnmr from within XWIN-NMR.</td>
</tr>
<tr>
<td>calcphhomo</td>
<td>Phase calculation program for noesytp, cosygsmftp, roesytp, mlevtp. Phase in F2 by reading and phasing ser file according to pulse program Phases in F1 are calculated.</td>
</tr>
<tr>
<td>calcphinv</td>
<td>Calculate the phase correction for the F1 dimension in HMQC/HSQC type experiments.</td>
</tr>
<tr>
<td>calcplen</td>
<td>Calculate the pulse length according to the power level.</td>
</tr>
<tr>
<td>calcpowlev</td>
<td>Calculate the power level according to the pulse length.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>calctemp</td>
<td>Calculate the temperature in the probe using the chemical shift difference between the aliphatic and OH protons.</td>
</tr>
<tr>
<td>calfun</td>
<td>Calculates an FID from an arbitrary function. This AU program is especially useful when you want to create a user defined window function for the 'uwm' command.</td>
</tr>
<tr>
<td>check-vtu</td>
<td>Updates TE variable from the actual temperature and store it in a separate file (edte) in the dataset.</td>
</tr>
<tr>
<td>cmdpanaux</td>
<td>Controls the start and function of command panels (see cpan command for more details).</td>
</tr>
<tr>
<td>coilttemp</td>
<td>Read the Shim Coil Temperature.</td>
</tr>
<tr>
<td>col_to_black</td>
<td>Set all colours to black that are defined for the current plot (NT only).</td>
</tr>
<tr>
<td>convto1d</td>
<td>Converts a 2D spectrum to 1D format.</td>
</tr>
<tr>
<td>decon_t1</td>
<td>Automatic deconvolution of a 2D T1/T2 experiment.</td>
</tr>
<tr>
<td>deptcyc</td>
<td>Creates 3 DEPT experiments from 13C experiment with CPD and then performs multiple cycles of NS scans (times 2 for DEPT90).</td>
</tr>
<tr>
<td>depthalt</td>
<td>Halt &quot;deptcyc&quot; AU program.</td>
</tr>
<tr>
<td>diffe</td>
<td>Calculate the difference spectra between expnos.</td>
</tr>
<tr>
<td>diffp</td>
<td>Calculate the difference spectra between procnos.</td>
</tr>
<tr>
<td>dosy</td>
<td>Setup for diffusion/DOSY experiments linear gradient amplitude ramp.</td>
</tr>
<tr>
<td>elim_ints</td>
<td>Eliminates regions from the intrng file that contain the solvent and/or reference signals. The result will be an intrng file where the integral trails have a more reasonable scaling and smaller integrals are better resolved.</td>
</tr>
<tr>
<td>f1ref</td>
<td>Corrects the referencing in F1 for inverse type experiments.</td>
</tr>
<tr>
<td>fidadd</td>
<td>Add up FID’s in incremented expno’s.</td>
</tr>
<tr>
<td>fidtoser</td>
<td>Writes a number of fids that are stored under the same NAME and incremental EXPNOs to a ser file.</td>
</tr>
<tr>
<td>getphsum</td>
<td>Reads the total phase values from the status parameters and stores them back to the actual parameters.</td>
</tr>
<tr>
<td>Program</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>getti</td>
<td></td>
</tr>
<tr>
<td>goalternate</td>
<td>Acquire alternated X/Y measurements. N averages are acquired alternatingly in two experiments.</td>
</tr>
<tr>
<td>graderror</td>
<td></td>
</tr>
<tr>
<td>gontp</td>
<td>Starts an ntp test program.</td>
</tr>
<tr>
<td>gradpreemp</td>
<td>Preemphasis adjustment for gradient spectroscopy on AMX/ARX/ASX with gradient waveform memory.</td>
</tr>
<tr>
<td>gradprog</td>
<td>Set the relevant parameters and generate the necessary files for the execution of pulse programs containing up to ten shaped gradient pulses in three orthogonal directions. For AMX/ARX/ASX with gradient waveform memory.</td>
</tr>
<tr>
<td>gradratio</td>
<td>Calculates gradient ratios for common inverse gradient pulse programs.</td>
</tr>
<tr>
<td>gradratiogs</td>
<td>Calculates gradient ratios for common inverse gradient pulse programs.</td>
</tr>
<tr>
<td>gradshapes</td>
<td>Calculate various gradient shapes.</td>
</tr>
<tr>
<td>gradshimau</td>
<td>Start gradshim gradient shimming procedure.</td>
</tr>
<tr>
<td>gssel_setup</td>
<td>AU program to determine the transmitter offset for 1H selective gradient shimming using 1H as observe nucleus.</td>
</tr>
<tr>
<td>guide</td>
<td>Perform a remote request server for the 'NMR Guide and Encyclopedia' client.</td>
</tr>
<tr>
<td>gy</td>
<td>Returns the currently running xwinnmr version and the directory it is stored in.</td>
</tr>
<tr>
<td>humpcal</td>
<td>Performs the 'hump test'. Measures the width of a peak at 0.55% and 0.11% of its signal height.</td>
</tr>
<tr>
<td>hwcal</td>
<td>Calculate the width of a peak at half height.</td>
</tr>
<tr>
<td>iexpno</td>
<td>Program to change to a new experiment number.</td>
</tr>
<tr>
<td>ilhalt</td>
<td>Stop an interleaved acquisition which was started with the AU program interleave.</td>
</tr>
<tr>
<td>interleave</td>
<td>Perform interleaved acquisitions</td>
</tr>
<tr>
<td>jconv_aufx</td>
<td>Converts Jeol FX data in a loop. The data must be stored with increasing extensions like proton.1, proton.2, ... etc.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>list_pp</td>
<td>Scans all experiment numbers of the current data set for the pulse program name and the first 30 characters of the title. All experiments that are found are listed on the screen. If an experiment is selected, then it is made the foreground data-set.</td>
</tr>
<tr>
<td>listall_au</td>
<td>Scans all AU programs and extracts the name and the short description. This information is then copied into the file listall in your home directory. This list corresponds to the list you are currently reading.</td>
</tr>
<tr>
<td>loadshimZ</td>
<td>Reads the on-axis shim values from disk and loads them to the BSMS.</td>
</tr>
<tr>
<td>lock_off</td>
<td>Switch off the lock to start data acquisition on the lock channel.</td>
</tr>
<tr>
<td>lock_on</td>
<td>Switch on the lock if it has been disabled.</td>
</tr>
<tr>
<td>loopadj</td>
<td>Parameter optimization au program which calculates the lock parameters loop filter, loop gain and loop time for optimal long-time stability after adjusting lock phase and lock gain to optimal.</td>
</tr>
<tr>
<td>make2d</td>
<td>Create a new 2D dataset from the current 1D dataset. Can be used for 2D spectroscopy and relaxation experiments. F2 parameters are copied from the 1D data, F1 parameters are set to reasonable values.</td>
</tr>
<tr>
<td>mkflist</td>
<td>Automatically generates a frequency list file.</td>
</tr>
<tr>
<td>mulabel</td>
<td>Processing AU program for determination of 13C multiplicity.</td>
</tr>
<tr>
<td>multanal</td>
<td>Interactive AU program for determination of 13C multiplicity.</td>
</tr>
<tr>
<td>multext</td>
<td>Processing AU program for determination of 13C multiplicity.</td>
</tr>
<tr>
<td>multi_decon</td>
<td>Automatic deconvolution of a series of 1D spectra with AI calibration.</td>
</tr>
<tr>
<td>multi_integ</td>
<td>Automatic integration of a series of 1D spectra with AI calibration.</td>
</tr>
<tr>
<td><strong>multi_integ2</strong></td>
<td>Automatic integration of a series of 1D spectra with calibration of the integral values.</td>
</tr>
<tr>
<td><strong>multi_integ3</strong></td>
<td>Automatic integration of a series of 1D spectra with AI calibration. The output is written in a format suitable for import in excel or similar desktop publishing programs.</td>
</tr>
<tr>
<td><strong>multi_zgvd</strong></td>
<td>Performs multiple acquisitions on increasing expnos with delays that are read from a vlist file. Alternatively, a fixed delay can be entered.</td>
</tr>
<tr>
<td><strong>multi_zgvt</strong></td>
<td>Performs multiple acquisitions on increasing expnos with temperatures that are read from a vtlist file.</td>
</tr>
<tr>
<td><strong>multicom</strong></td>
<td>Executes an XWIN-NMR command in increasing expnos.</td>
</tr>
<tr>
<td><strong>multicyc</strong></td>
<td>Cycles through a series of acquisitions of increasing expnos.</td>
</tr>
<tr>
<td><strong>multiefp</strong></td>
<td>Performs multiple &quot;efp&quot; on increasing expnos.</td>
</tr>
<tr>
<td><strong>multiexpt</strong></td>
<td>Calculates experimental time for multizg.</td>
</tr>
<tr>
<td><strong>multifp</strong></td>
<td>Performs multiple &quot;fp&quot; on increasing expnos.</td>
</tr>
<tr>
<td><strong>multifapk</strong></td>
<td>Performs multiple &quot;ft;apk&quot; on increasing expnos.</td>
</tr>
<tr>
<td><strong>multihalt</strong></td>
<td>Halt &quot;multicyc&quot; AU program.</td>
</tr>
<tr>
<td><strong>multimas</strong></td>
<td>Performs multiple MAS experiments on increasing expnos.</td>
</tr>
<tr>
<td><strong>multipcom</strong></td>
<td>Executes an XWIN-NMR command in increasing procnos.</td>
</tr>
<tr>
<td><strong>multiwinpro</strong></td>
<td>Performs multiple processing on increasing expnos. The program asks for the window function and its parameters.</td>
</tr>
<tr>
<td><strong>multixfb</strong></td>
<td>Performs multiple &quot;xfb&quot; on increasing expnos.</td>
</tr>
<tr>
<td><strong>multizg</strong></td>
<td>Performs multiple acquisitions on increasing expnos.</td>
</tr>
<tr>
<td><strong>noediff</strong></td>
<td>noe difference spectroscopy using different expnos.</td>
</tr>
<tr>
<td><strong>noeflist</strong></td>
<td>Automatic generation of a frequency list with the peaks from the current plot region for noe.</td>
</tr>
<tr>
<td><strong>noemult</strong></td>
<td>noe difference spectroscopy with multiple irradiation points for each multiplet using different expnos.</td>
</tr>
<tr>
<td><strong>o1f1</strong></td>
<td>Correct calibration of F1 axes in 2D MQ experiments of odd half integer nuclei.</td>
</tr>
<tr>
<td><strong>paropt</strong></td>
<td>Parameter optimization au program.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>pararray</td>
<td>Parameter optimization au program using parameter arrays. Derived from 'paropt', but several parameters may now be changed per experiment. In addition, parameters are not changed via constant increments. Instead, the values are taken from an array.</td>
</tr>
<tr>
<td>pass2d</td>
<td>Perform a PASS experiment with 5 Pi-pulses and 16 increments (samples up to 16 spinning side bands).</td>
</tr>
<tr>
<td>pecosy</td>
<td>Program to pre-process P.E.COSY raw data before 2D-FT.</td>
</tr>
<tr>
<td>phtran</td>
<td>Transfer phase correction parameters PHC0 and PHC1 into acquisition parameters PH_ref and DE.</td>
</tr>
<tr>
<td>plintfac</td>
<td>Plot integrals with different scaling factors.</td>
</tr>
<tr>
<td>plot_3d</td>
<td>Plot planes of a 3D dataset.</td>
</tr>
<tr>
<td>plot_sino</td>
<td>Plot spectrum, scaling depends on Signal/Noise.</td>
</tr>
<tr>
<td>plot_to_file</td>
<td>Creates a postscript file of the desired plot.</td>
</tr>
<tr>
<td>poptau</td>
<td>Parameter optimization au program using parameter arrays. Derived from 'paropt' but several parameters can be optimized. The parameters are changed according to the parameter arrays. The AU program will be started from user interface 'popt' (parameter editor).</td>
</tr>
<tr>
<td>popthalt</td>
<td>Halt &quot;popt&quot; AU program.</td>
</tr>
<tr>
<td>pp2d</td>
<td>Performs a 2D peak picking with the help of the XWIN-NMR command &quot;pp&quot; for 1D spectra.</td>
</tr>
<tr>
<td>pp2dmi</td>
<td>Sets the peak picking parameter MI according to the parameter S_DEV and performs a 2D peak picking with the AU program &quot;pp2d&quot;.</td>
</tr>
<tr>
<td>proc_1H</td>
<td>Processes and plots 1D spectra. Does not perform baseline correction.</td>
</tr>
<tr>
<td>proc_1d</td>
<td>Processes and plots 1D spectra. Uses 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_1dapks</td>
<td>Processes and plot 1D spectra. Uses 'apks' for phase correction and 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_1dconlf</td>
<td>Processes and plots 1D spectra. Uses 'autoplot' for plotting. Plots an additional spectrum on the same plot if there are peaks with a chemical shift &gt; 11.</td>
</tr>
<tr>
<td>proc_1dconlf_pr</td>
<td>Processes and plots 1D spectra. Uses 'autoplot' for plotting. Plots an additional wq1spectrum on the same plot if there are peaks in the lowfield range outside the plot limits.</td>
</tr>
<tr>
<td>proc_1dexp</td>
<td>Processing AU program for 1D spectra with automatic expansions.</td>
</tr>
<tr>
<td>proc_1dglp</td>
<td>Processing AU program with automatic data evaluation according to GLP standards. This AU program takes CY as an argument and then works like proc_1d.</td>
</tr>
<tr>
<td>proc_1dinfo</td>
<td>Processes and plots 1D spectra. Uses 'autoplot' for plotting. Prints the info file ('edinfo') on the plot.</td>
</tr>
<tr>
<td>proc_1dlf</td>
<td>Processes and plot 1D spectra. Uses 'autoplot' for plotting. Plots an additional lowfield plot.</td>
</tr>
<tr>
<td>proc_1dlfexp</td>
<td>Processing AU program for 1D spectra with additional low field plot and automatic expansions.</td>
</tr>
<tr>
<td>proc_1dppti</td>
<td>Processes and plots 1D spectra. Creates a special peaklist file (frequency (Hz) and half width) and prints this on the plot. Uses 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_1dppti</td>
<td>Processes and plots 1D spectra. Creates a peak picking list and prints this on the plot. Uses 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_2d</td>
<td>Processing AU program for 2D spectra without plotting.</td>
</tr>
<tr>
<td>proc_2dhom</td>
<td>Processes and plots 2D homonuclear type spectra. Uses 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_2dhom_2pp</td>
<td>Processes and plots 2D homonuclear type spectra with two positive projections. Uses 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_2dinv</td>
<td>Processes and plots 2D inverse type spectra. Uses 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_2dinv_2p</td>
<td>Processes and plots 2D inverse type spectra. Plots two projections. Uses 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_2dpl</td>
<td>Processes and plots 2D type spectra. Uses 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_2dsym</td>
<td>Processes and symmetrizes 2D type spectra. Uses 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_2dt1</td>
<td>Automatic processing of a 2D T1/T2 experiment with subsequent T1/T2 calculation.</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>proc_cpd135</td>
<td>Processes and plots 13C CPD and DEPT135 spectra that were acquired with the AU program au_zg135. Uses 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_glp</td>
<td>Automatic GLP data evaluation.</td>
</tr>
<tr>
<td>proc_intrng</td>
<td>Processes and plots 1D spectra. Uses the predefined integral range file 'testrng' for integration and 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_no</td>
<td>AU program which does no processing.</td>
</tr>
<tr>
<td>proc_noe</td>
<td>Processes and plots noediff spectra. Uses 'autoplot' for plotting.</td>
</tr>
<tr>
<td>proc_t1</td>
<td>Automatic processing of a 2D T1/T2 experiment with subsequent T1/T2 calculation.</td>
</tr>
<tr>
<td>proc_tecalib</td>
<td>Evaluation of previous temperature calibration experiments.</td>
</tr>
<tr>
<td>psys180f1t1</td>
<td>Processing AU program for the 180 degree pulse calibration tests.</td>
</tr>
<tr>
<td>psysamp1s39</td>
<td>Processing AU program for the amplitude stability tests - with shaped pulse - with 30 degree pulse - with 90 degree pulse - after gradient echo (5msec, 30 G/cm) - after gradient echo (5msec, 10 G/cm) - after gradient pulse (1msec, 10G/cm).</td>
</tr>
<tr>
<td>psysb1hom</td>
<td>Processing AU program for the B1 homogeneity test.</td>
</tr>
<tr>
<td>psysb2hom</td>
<td>Processing AU program for the B2 homogeneity test.</td>
</tr>
<tr>
<td>psyscancel</td>
<td>Processing AU program for the - phase cycling cancellation test - phase cycling cancellation test after gradient pulse.</td>
</tr>
<tr>
<td>psysdante1</td>
<td>Processing AU program for the dante type turn on test.</td>
</tr>
<tr>
<td>psysdecpro1</td>
<td>Processing AU program for the decoupler profile test.</td>
</tr>
<tr>
<td>psysexpro1</td>
<td>Processing AU program for the - excitation profile (16 usec gauss shape) test - excitation profile (6 msec gauss shape) test.</td>
</tr>
<tr>
<td>psyssglitch</td>
<td>Processing AU program for the glitch test.</td>
</tr>
<tr>
<td>psyssgrreco1</td>
<td>Processing AU program for the gradient recovery test.</td>
</tr>
<tr>
<td>AU Program</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>psygrzpro</td>
<td>Processing AU program for the z-gradient profile.</td>
</tr>
<tr>
<td>psymsmodl1</td>
<td>Processing AU program for the - modulator linearity test - shaped pulse modulator linearity test.</td>
</tr>
<tr>
<td>psymsmultl1</td>
<td>Processing AU program for the amplitude linearity test (1dB power level steps).</td>
</tr>
<tr>
<td>psyphaslst</td>
<td>Processing AU program for the - phase stability test (&quot;13 degree test&quot;) - shaped pulse phase stability test (16 usec gaussian shape, &quot;13 degree test&quot;).</td>
</tr>
<tr>
<td>psyphasf1</td>
<td>Processing AU program for the - phase propagation test - phase shifting test.</td>
</tr>
<tr>
<td>psysspullin1</td>
<td>Processing AU program for the - amplitude linearity test - shaped pulse amplitude linearity test (pulse length *2, power level +6).</td>
</tr>
<tr>
<td>psyssquadim</td>
<td>Processing AU program for the quad image suppression test.</td>
</tr>
<tr>
<td>psyrrgtest</td>
<td>Processing AU program for the receiver gain test (analog and digital).</td>
</tr>
<tr>
<td>psyssoftp1</td>
<td>Processing AU program for the shaped pulse comparison (rectangular, gaussian, eburp1).</td>
</tr>
<tr>
<td>psyssstestab</td>
<td>Automatic processing of a 2D Temperature stability experiment, evaluation of temperature and statistic of temperature stability. Can be used to process data obtained with the AU program systestab.</td>
</tr>
<tr>
<td>psysssturnon</td>
<td>Processing AU program for the turn on test.</td>
</tr>
<tr>
<td>pulse</td>
<td>Program to calculate attenuation value for given pulse length or nutation frequency, or vice versa.</td>
</tr>
<tr>
<td>pulsecalib</td>
<td>Offset/Pulsecalibration H2O/D2O and Offset/Pulsecalibration 13C, 15N probehead.</td>
</tr>
<tr>
<td>qnpset</td>
<td>Define the QNP parameter according to the currently defined probehead.</td>
</tr>
<tr>
<td>quadplot</td>
<td>First plots a 2D overview spectrum and then the 4 quadrants of the 2D spectrum.</td>
</tr>
<tr>
<td>queue</td>
<td>Queue data acquisition.</td>
</tr>
<tr>
<td>queue_init</td>
<td>Initialise data acquisition with the AU program queue.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>queuega</td>
<td>Queue data acquisition.</td>
</tr>
<tr>
<td>r23mplot</td>
<td>Read 2D slices from a 3D data set and plot them.</td>
</tr>
<tr>
<td>r23mult</td>
<td>Repeatedly reads slices from a 3D data set (3rrr) into successive experiment numbers.</td>
</tr>
<tr>
<td>rampXY</td>
<td>3D gradient shimming with the BSMS RCB board.</td>
</tr>
<tr>
<td>remproc</td>
<td>Automatic conversion and processing of data sets transferred via BRUKNET, LIGHTNET, NMR-LINK or TCP-LINK.</td>
</tr>
<tr>
<td>repeat</td>
<td>Repeat an acquisition with exactly the same parameters, pulse program and other lists.</td>
</tr>
<tr>
<td>secplot</td>
<td>Generate a section plot. The overview spectrum is plotted together with a vertical expansion of a smaller part of the spectrum on top of it.</td>
</tr>
<tr>
<td>selget</td>
<td>Import a shaped pulse into the current dataset.</td>
</tr>
<tr>
<td>selput</td>
<td>Export the current FID into a wave form file.</td>
</tr>
<tr>
<td>set2hdecgp</td>
<td>Setup AU program for standard 3D parameter sets.</td>
</tr>
<tr>
<td>setccnh3dgp</td>
<td>Setup AU program for standard 3D parameter sets.</td>
</tr>
<tr>
<td>setdiffparm</td>
<td>Extracts diffusion sequence parameters and stores parameters for &quot;vargrad&quot; simfit fitting (T1/T2) or DOSY processing.</td>
</tr>
<tr>
<td>seteditedgp</td>
<td>Setup AU program for standard 3D parameter sets.</td>
</tr>
<tr>
<td>sethccc3dgp</td>
<td>Setup AU program for standard 3D parameter sets.</td>
</tr>
<tr>
<td>setpar3dgp</td>
<td>Setup AU program for standard 3D parameter sets.</td>
</tr>
<tr>
<td>setproj</td>
<td>Sets the edg projection parameters to appropriate values.</td>
</tr>
<tr>
<td>set_sreglist</td>
<td>Set SREGLST parameter from NUC1 and SOLVENT.</td>
</tr>
<tr>
<td>shear</td>
<td>Program for 2D MQ experiments on nuclei with odd half integer spin for shearing the 2D spectrum after 2D FT.</td>
</tr>
<tr>
<td>showpp</td>
<td>Start NMR-SIM pulse program display. This program is used by the NMR guide&amp;encyclopedia.</td>
</tr>
<tr>
<td>simplex</td>
<td>AU program for autoshimming. It is suitable for adjustment of strongly coupled shim groups which may be far from the optimum position.</td>
</tr>
<tr>
<td>Program</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>simtoseq</td>
<td>Converts data which have been recorded in digital and qsim mode to data which appear to be acquired in qseq mode.</td>
</tr>
<tr>
<td>sinocal</td>
<td>Calculates the signal to noise ratio.</td>
</tr>
<tr>
<td>split2D</td>
<td>Splits a processed 2D file into single 1D spectra.</td>
</tr>
<tr>
<td>splitinvnoe</td>
<td>Separate NOE and NONOE data obtained with a pulse program like invinoef3gpsi.</td>
</tr>
<tr>
<td>splitser</td>
<td>Splits a ser file into single fids, starting with the expno which follows the ser file.</td>
</tr>
<tr>
<td>splitxf</td>
<td>Separate and combine double half filtered data.</td>
</tr>
<tr>
<td>stack1d</td>
<td>Generates a stacked plot of 1D spectra from increasing or decreasing EXPNOs or PROCNOs.</td>
</tr>
<tr>
<td>stack2d</td>
<td>Generate a 2D stack plot.</td>
</tr>
<tr>
<td>stackp1d</td>
<td>Generates a stacked plot of 3 to 12 1D spectra from increasing or decreasing EXPNOs or PROCNOs.</td>
</tr>
<tr>
<td>sti</td>
<td>Displays the title of the current dataset in a window without opening an editor (FAST!)</td>
</tr>
<tr>
<td>suppical</td>
<td>Calculates the width of the Water peak at 100% and 50% of the DSS signal height. The result is referred to as the 'water suppression test'.</td>
</tr>
<tr>
<td>sys180f1t1</td>
<td>Acquisition AU program for the 180 degree pulse calibration test with different phases.</td>
</tr>
<tr>
<td>sys180f1t2</td>
<td>Acquisition AU program for the 180 degree pulse calibration test with different flip angles.</td>
</tr>
<tr>
<td>sysamp1sp9</td>
<td>Acquisition AU program for the shaped pulse amplitude stability test.</td>
</tr>
<tr>
<td>sysamp1st</td>
<td>Acquisition AU program for the amplitude stability tests - with 30 degree pulse - with 90 degree pulse.</td>
</tr>
<tr>
<td>sysb1hom</td>
<td>Acquisition AU program for the B1 homogeneity test.</td>
</tr>
<tr>
<td>sysb2hom</td>
<td>Acquisition AU program for the B2 homogeneity test.</td>
</tr>
<tr>
<td>syscancel</td>
<td>Acquisition AU program for the phase cycling cancellation test.</td>
</tr>
<tr>
<td>sysdante1</td>
<td>Acquisition AU program for the dante type turn on test.</td>
</tr>
<tr>
<td>Program Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sysdecpro1</td>
<td>Acquisition AU program for the decoupler profile test.</td>
</tr>
<tr>
<td>sysexpro1</td>
<td>Acquisition AU program for the - excitation profile (16 usec gauss shape) test - excitation profile (6 msec gauss shape) test.</td>
</tr>
<tr>
<td>sysgenpar</td>
<td>Preparation AU program for all HWT test programs.</td>
</tr>
<tr>
<td>sysglitch</td>
<td>Acquisition AU program for the glitch test.</td>
</tr>
<tr>
<td>sysgrcan</td>
<td>Acquisition AU program for the phase cycling cancellation test after gradient pulse.</td>
</tr>
<tr>
<td>sysgrecho</td>
<td>Acquisition AU program for the amplitude stability test after gradient echo (5msec, 30 G/cm and 5msec, 10 G/cm).</td>
</tr>
<tr>
<td>sysgrreco1</td>
<td>Acquisition AU program for the gradient recovery test.</td>
</tr>
<tr>
<td>sysgrstab</td>
<td>Acquisition AU program for the amplitude stability test after gradient pulse (1msec, 10G/cm).</td>
</tr>
<tr>
<td>sysgrzpro</td>
<td>Acquisition AU program for the z-gradient profile.</td>
</tr>
<tr>
<td>sysmodl1</td>
<td>Acquisition AU program for the modulator linearity test.</td>
</tr>
<tr>
<td>sysmodls1</td>
<td>Acquisition AU program for the shaped pulse modulator linearity test.</td>
</tr>
<tr>
<td>symsmultl1</td>
<td>Acquisition AU program for the amplitude linearity test (1dB power level steps).</td>
</tr>
<tr>
<td>sysphas1sp</td>
<td>Acquisition AU program for the shaped pulse phase stability test (16 usec gaussian shape, &quot;13 degree test&quot;).</td>
</tr>
<tr>
<td>sysphas1st</td>
<td>Acquisition AU program for the phase stability test (&quot;13 degree test&quot;).</td>
</tr>
<tr>
<td>sysphasf1</td>
<td>Acquisition AU program for the - phase propagation test - phase shifting test.</td>
</tr>
<tr>
<td>syspullin1</td>
<td>Acquisition AU program for the amplitude linearity test (pulse length *2, power level +6).</td>
</tr>
<tr>
<td>sysquadim</td>
<td>Acquisition AU program for the quad image suppression test.</td>
</tr>
<tr>
<td>sysrgtest</td>
<td>Acquisition AU program for the receiver gain test (analog and digital).</td>
</tr>
<tr>
<td>Program</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>syssoftp1</td>
<td>Acquisition AU program for the shaped pulse comparison (rectangular, gaussian, eburp1).</td>
</tr>
<tr>
<td>syssplin1</td>
<td>Acquisition AU program for the shaped pulse amplitude linearity test (pulse length *2, power level +6).</td>
</tr>
<tr>
<td>systestab</td>
<td>AU program for a temperature stability experiment performed as pseudo 2D experiment including evaluation of temperature and statistics of temperature stability.</td>
</tr>
<tr>
<td>systurnon</td>
<td>Acquisition AU program for the turn on test.</td>
</tr>
<tr>
<td>tecalib</td>
<td>AU program to determine the temperature calibration curve.</td>
</tr>
<tr>
<td>testsuite</td>
<td>Test the general functionality of an XWIN-NMR release version. Basic functionality is given if this program is completed without error messages.</td>
</tr>
<tr>
<td>tmscal</td>
<td>Performs a peak picking around the TMS signal. If the two satellites from the 29Si - 1H coupling can be detected, the resolution is OK.</td>
</tr>
<tr>
<td>tune</td>
<td>Tune a probehead.</td>
</tr>
<tr>
<td>update_layout</td>
<td>Sets the parameter 'LAYOUT' in all parameter sets.</td>
</tr>
<tr>
<td>update_aunmp</td>
<td>Sets the parameter AUNMP in all parameter sets.</td>
</tr>
<tr>
<td>vtu_exam</td>
<td>Example AU program which shows how to use low level functions to read or write VTU (BVT1000/BDTC) parameters.</td>
</tr>
<tr>
<td>writeshimZ</td>
<td>Reads the on-axis shim values and writes a pseudo shim file.</td>
</tr>
<tr>
<td>xfshear</td>
<td>Program for shearing of 2D MQMAS spectra of odd half integer quadrupolar nuclei. Data need to be acquired in States Mode.</td>
</tr>
<tr>
<td>xwp_p1dlf</td>
<td>Processing AU program for 1D spectra with additional low field plot. AUTOPLOT (the automatic version of XWIN-PLOT) is used instead of XWIN-NMR's plot.</td>
</tr>
<tr>
<td>xwp_p2dpl</td>
<td>Processing AU program for 2D type spectra which don't need a symmetrization. AUTOPLOT (the automatic version *of XWIN-PLOT) is used instead of XWIN-NMR's plot.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>xwp_pcpd135</td>
<td>Processing AU program for 13C CPD and DEPT135 spectra which were acquired</td>
</tr>
<tr>
<td></td>
<td>with the AU program au_zg135. AUTO PLOT (the automatic version of XWIN-</td>
</tr>
<tr>
<td></td>
<td>PLOT) is used instead of XWIN-NMR's plot.</td>
</tr>
<tr>
<td>zeroim</td>
<td>Zeroes the imaginary data of a 1D or 2D data set.</td>
</tr>
<tr>
<td>zg_2Hoffon</td>
<td>General AU program for data acquisition. The lock is switched off before</td>
</tr>
<tr>
<td></td>
<td>the acquisition is started.</td>
</tr>
<tr>
<td>zgchkte</td>
<td>Starts acquisition with zg and monitors the temperature. The experiment is</td>
</tr>
<tr>
<td></td>
<td>halted if the current temperature differs too much from the target</td>
</tr>
<tr>
<td></td>
<td>temperature.</td>
</tr>
<tr>
<td>zg_dfs</td>
<td>Calculates shape file for double frequency sweep and subsequent data-</td>
</tr>
<tr>
<td></td>
<td>acquisition.</td>
</tr>
<tr>
<td>2df1shift</td>
<td>Shift a 2D spectrum along the F1 axis.</td>
</tr>
<tr>
<td>2dgetref</td>
<td>Gets parameters for a 2D spectrum from the 1D reference spectra: Nucleus,</td>
</tr>
<tr>
<td></td>
<td>Frequencies, Spectral Width, and reference plot data set names. The F2</td>
</tr>
<tr>
<td></td>
<td>reference is taken from the second dataset. The F1 reference is taken from</td>
</tr>
<tr>
<td></td>
<td>the third dataset.</td>
</tr>
<tr>
<td>2dshift</td>
<td>Shift 2D time domain data left or right over NSP points.</td>
</tr>
<tr>
<td>2nde</td>
<td>Set 2nd data set to new expno and 3rd data set equal to foreground</td>
</tr>
<tr>
<td></td>
<td>data set.</td>
</tr>
<tr>
<td>2ndn</td>
<td>Set 2nd data set to new name and 3rd data set equal to foreground</td>
</tr>
<tr>
<td></td>
<td>data set.</td>
</tr>
</tbody>
</table>
Chapter 13

XWIN-NMR parameter types

This chapter contains a list of all XWIN-NMR parameters grouped by their type. The type of a parameter can be integer, float, double or character-string. Several AU macros read XWIN-NMR parameters into AU variables or store the value of AU variables into XWIN-NMR parameters. In both cases it is important that the type of the AU variable is the same as the parameter type.
### 13.1 Integer parameters

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13.1</td>
<td>Integer parameters</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following XWIN-NMR parameters are of the type integer:
13.2 Float parameters

The following XWIN-NMR parameters are of the type float:

<table>
<thead>
<tr>
<th>Parameter1</th>
<th>Parameter2</th>
<th>Parameter3</th>
<th>Parameter4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSF1</td>
<td>ABSF2</td>
<td>ABSL</td>
<td>ALPHA</td>
</tr>
<tr>
<td>ASSFAC</td>
<td>ASSFAC1</td>
<td>ASSFACX</td>
<td>ASSWID</td>
</tr>
<tr>
<td>AZFE</td>
<td>AZFW</td>
<td>BCFW</td>
<td>CNST[32]</td>
</tr>
<tr>
<td>FOV</td>
<td>FW</td>
<td>GAMMA</td>
<td>GB</td>
</tr>
<tr>
<td>ISEN</td>
<td>LB</td>
<td>LEV0</td>
<td>LOCPHAS</td>
</tr>
<tr>
<td>MASR</td>
<td>MAXI</td>
<td>MI</td>
<td>NOISF1</td>
</tr>
<tr>
<td>NOISF2</td>
<td>OFFSET</td>
<td>PC</td>
<td>PCPD[10]</td>
</tr>
<tr>
<td>PHC0</td>
<td>PHC1</td>
<td>PHCOR[32]</td>
<td>PH_ref</td>
</tr>
<tr>
<td>SIGF1</td>
<td>SIGF2</td>
<td>SINO</td>
<td>SPOAL[32]</td>
</tr>
<tr>
<td>TE</td>
<td>TE2</td>
<td>TL[8]</td>
<td>TM1</td>
</tr>
<tr>
<td>TM2</td>
<td>TOLEVEL</td>
<td>TPOAL[8]</td>
<td>TPOFFS[8]</td>
</tr>
<tr>
<td>TP[8]</td>
<td>V9</td>
<td>VD</td>
<td>ZL1</td>
</tr>
<tr>
<td>ZL2</td>
<td>ZL3</td>
<td>ZL4</td>
<td></td>
</tr>
</tbody>
</table>
13.3 Double parameters

The following XWIN-NMR parameters are of the type double:

<table>
<thead>
<tr>
<th>BF1</th>
<th>BF2</th>
<th>BF3</th>
<th>BF4</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF5</td>
<td>BF6</td>
<td>BF7</td>
<td>BF8</td>
</tr>
<tr>
<td>COROFFS</td>
<td>CY</td>
<td>F1P</td>
<td>F2P</td>
</tr>
<tr>
<td>LOCKPOW</td>
<td>LTIME</td>
<td>O1</td>
<td>O2</td>
</tr>
<tr>
<td>O3</td>
<td>O4</td>
<td>O5</td>
<td>O6</td>
</tr>
<tr>
<td>O7</td>
<td>O8</td>
<td>SF</td>
<td>SFO1</td>
</tr>
<tr>
<td>SFO2</td>
<td>SFO3</td>
<td>SFO4</td>
<td>SFO5</td>
</tr>
<tr>
<td>SFO6</td>
<td>SFO7</td>
<td>SFO8</td>
<td>SW</td>
</tr>
<tr>
<td>WBSW</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 13.4 Character-string parameters

The following XWIN-NMR are of the type character-string:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F2LIST[16]</td>
<td>F3LIST[16]</td>
<td>FQ1LIST[16]</td>
<td>GPNAM0[64]</td>
</tr>
<tr>
<td>SOLVENT[32]</td>
<td>SPNAM0[64]</td>
<td>SREGLST[40]</td>
<td>TI[72]</td>
</tr>
<tr>
<td>TPNAME0[16]</td>
<td>TYPE[16]</td>
<td>USER[64]</td>
<td>USERA1[80]</td>
</tr>
<tr>
<td>VTLIST[16]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Index

A
ABORT 43
ABS 31
ABS1 35
ABS2 35
ABSD 31
ABSD1 35
ABSD2 35
ABSF 31
ABSF1 31
ABSF2 31
ABSO1 35
ABSO2 35
ABST 35
ABST1 35
ABST2 35
ADD 33, 56
ADD_CURDAT_TO_PORTFOLIO 41, 106
ADD_TO_PORTFOLIO 41, 107
ADD2D 35
ADDC 33
addfid command 96
AND 33
APK 31, 63
APK0 31
APK1 31
APK 31
APKS 31
Aspect 2000/3000 dataset 85
AT 33
AU macro 5
aucmd.h 16, 18
Automatic baseline correction 39
AUTOPLOT 40, 102
autoplot command 104
AUTOPLOT_TO_FILE 41, 103
AUTOPLOT_WITH_PORTFOLIO 41, 109
AUTOPLOT_WITH_PORTFOLIO_TO_FILE 41, 110
autoshimming 29

B
base_info file 33
baslpts file 33
BAYED 34
BAYEDX 34
Bayesian calculation 34
BAYX 34
BC 31
BCM1 35
BCM2 35
brukdef.h 18
Bruker library functions 9
BSMS 29

C
CalcExpTime function 53, 118
cc compiler 16
C-code 16
character string parameters 163
check_pwd function 119
CLOSE_PORTFOLIO 41, 108
CMPL 33
column of a 2D spectrum 71, 75
compileall command 7
compiling AU programs 7
constants 16
control statements 15
CONV 41, 85
CONVCP 41, 85
CONVDTA 31
cplbruk command 7
cpluser command 7
CPR_exec 15, 42, 46, 47, 53
CREATE_PORTFOLIO 41, 105

D
DAT1 34
I-166

DAT2 34
DATASET 25, 52, 53, 55, 64
DATASET2 25, 56
DATASET3 25, 56
DDATASSETLIST 25
DECLARE_PORTFOLIO 41, 104
define statements 13, 16
DEG90 28
DELPAR 27
DEXPN0 25, 54, 59
dircp function 127
dircp_err function 127
disk unit 85
DIV 33
dpa command 92
DPARSETLIST 27
DPROCNO 25, 62
DPULPROGLIST 28
DT 33
DU 25
DVTLIST 30

E
eda command 42, 90, 92, 98, 99
edau command 7, 8, 10
edc2 command 25, 107
eddosy command 27
edg command 39, 98, 99
edgw command 39
edgx command 39
edhead command 124
edit mode 8
edlock command 29
edmac command 9
edmisc command 34
edo command 90, 98, 99
edp command 42, 90
EF 31
EFP 31, 57
EJ 29
EM 31
enhanced metafile 103, 110
erropt.h 18
ERRORABORT 43
Executing AU programs 8
expinstall command 6, 7

F
F1DISCO 37
F1PROJN 37
F1PROJP 37
F1SUM 37
F2DISCO 37
F2PROJN 37
F2PROJP 38
F2SUM 37
fcntl.h 18
FETCHDOSYPAR 27
FETCHHPAR 26, 90
FETCHHPAR1 26, 90
FETCHHPAR1S 26
FETCHHPAR3 26
FETCHHPAR3S 26
FETCHHPARM 27, 90
FETCHHPARS 26, 92
FETCHHPARS1 92
FETCHHPARS3 92
FETCHPLPAR 27, 90
FETCHPLWPAR 27, 90
FETCHPLXPAR 27, 90
fetchstorpl function 128
FETCHT1PAR 27, 90
fidtoser AU program 79
FILT 33
first order phase correction 31
float parameters 161
FLPLOT 39
FMC 31
FP 32
freedir function 122
FROMJDX 41, 84
FT 32, 63
ft command 9

G
Gaussian deconvolution 34
Gaussian window multiplication 32
gcc compiler 16
GDATASSETLIST 25
GDCON 34
I-167

GENFID 32
GENSER 37
GETCURDATA 19, 24, 52
GETCURDATA2 25
GETCURDATA3 25
GETDATASET 24, 57
getdir function 120
GETDOUBLE 26, 113
GETFLOAT 26, 93, 113
gethighest function 130
GETHPCU 31
GETINT 26, 112
GETLCOSY 40
GETLIM 40
GETLINV 40
GETLJRES 40
GETLXHCO 40
GetNmrSuperUser function 119
getstan function 131
GETSTRING 26, 115
getxwinvers function 132
GF 32
GFP 32
GLIST 25
GM 32
GO 28
GPARSETLIST 27
GPULPROGLIST 28
GVTLIST 30

H
header files 18
Hilbert Transform 32, 36
HPCU parameters 31
HT 32

I
IDATASETLIST 20, 25
IEXPNO 25, 47, 52, 54, 58, 64
IFEOSDATASETLIST 25
IFEOPARSETLIST 28
IFEOPULPROGLIST 28
IFT 32
II 28
IJ 29
ILOOPCOUNTLIST 20
inc directory 16
include statements 13, 14, 16
integer parameters 160
intrng file 33, 115
Inverse Fourier Transform 32
2D 36
INVSF 35
IPARSETLIST 20, 27
IPROCNO 25, 52, 61
IPULPROGLIST 20, 28
IVTLIST 20, 30

J
jaz drive 68
JCAMP-DX file 41, 84
JCAMP-DX format 33, 82
JCONV 42
Jeol dataset 42, 88

L
lastparflag variable 11
LDCON 34
LEVCALC 35
LFILTER 29
LG 29
LGAIN 29
LI 33, 115
LIBAY 34
libcb.h 18
limits.h 18
LIPP 33
LIPPF 33
listall_au AU program 8
LO 29
LOCK 29
lock power 29
LOCKPLOTS 40
LOCNUC 29
loop gain 29
loop statements 15
loop structures 13
loop time 29
loopcount1 variable 11
loopcount2 variable 11
<table>
<thead>
<tr>
<th>Patent</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lopo</td>
<td>29</td>
</tr>
<tr>
<td>LS</td>
<td>33</td>
</tr>
<tr>
<td>LTIme</td>
<td>29</td>
</tr>
<tr>
<td>M</td>
<td>Magnitude calculation 32</td>
</tr>
<tr>
<td>MAKE_ZERO_FID</td>
<td>28</td>
</tr>
<tr>
<td>makeau file</td>
<td>16</td>
</tr>
<tr>
<td>MAS unit</td>
<td>30</td>
</tr>
<tr>
<td>MASE</td>
<td>30</td>
</tr>
<tr>
<td>MASG</td>
<td>31</td>
</tr>
<tr>
<td>MASH</td>
<td>31</td>
</tr>
<tr>
<td>MASI</td>
<td>30</td>
</tr>
<tr>
<td>MASR</td>
<td>30</td>
</tr>
<tr>
<td>MASRGET</td>
<td>31</td>
</tr>
<tr>
<td>math.h</td>
<td>18</td>
</tr>
<tr>
<td>MC</td>
<td>32</td>
</tr>
<tr>
<td>MDCON</td>
<td>34</td>
</tr>
<tr>
<td>mkudir function</td>
<td>133</td>
</tr>
<tr>
<td>MUL</td>
<td>33, 56</td>
</tr>
<tr>
<td>MULC</td>
<td>33</td>
</tr>
<tr>
<td>multi_integ AU program</td>
<td>131</td>
</tr>
<tr>
<td>multie xpt AU program</td>
<td>118</td>
</tr>
<tr>
<td>multizg AU program</td>
<td>53</td>
</tr>
<tr>
<td>N</td>
<td>NM</td>
</tr>
<tr>
<td>NMRQUANT</td>
<td>33</td>
</tr>
<tr>
<td>NZP</td>
<td>34</td>
</tr>
<tr>
<td>p_1d AU program</td>
<td>6</td>
</tr>
<tr>
<td>parameter type</td>
<td>159</td>
</tr>
<tr>
<td>PARSETYP</td>
<td>19, 27</td>
</tr>
<tr>
<td>PathXWinNMR function</td>
<td>131, 134</td>
</tr>
<tr>
<td>PD</td>
<td>34</td>
</tr>
<tr>
<td>PD0</td>
<td>34</td>
</tr>
<tr>
<td>peaklist file</td>
<td>33</td>
</tr>
<tr>
<td>PF</td>
<td>34</td>
</tr>
<tr>
<td>PFT2</td>
<td>34</td>
</tr>
<tr>
<td>phase correction first order</td>
<td>31</td>
</tr>
<tr>
<td>phase correction zero order</td>
<td>31</td>
</tr>
<tr>
<td>PHC0</td>
<td>31</td>
</tr>
<tr>
<td>PHC1</td>
<td>31</td>
</tr>
<tr>
<td>PK</td>
<td>32</td>
</tr>
<tr>
<td>Plane from 3D raw data</td>
<td>80</td>
</tr>
<tr>
<td>PLOT</td>
<td>39</td>
</tr>
<tr>
<td>plot_to_file AU program</td>
<td>6</td>
</tr>
<tr>
<td>PLOTS</td>
<td>39</td>
</tr>
<tr>
<td>PLOTW</td>
<td>39</td>
</tr>
<tr>
<td>PLOTX</td>
<td>39</td>
</tr>
<tr>
<td>PLTo of XWIN-PLOT</td>
<td>104, 105, 106, 107, 108, 109, 110</td>
</tr>
<tr>
<td>postscript file</td>
<td>6, 103, 110</td>
</tr>
<tr>
<td>pow_next function</td>
<td>135</td>
</tr>
<tr>
<td>Power spectrum</td>
<td>36</td>
</tr>
<tr>
<td>POWMOD</td>
<td>31</td>
</tr>
<tr>
<td>PP</td>
<td>32</td>
</tr>
<tr>
<td>PPH</td>
<td>32</td>
</tr>
<tr>
<td>PPP</td>
<td>32</td>
</tr>
<tr>
<td>ppp command</td>
<td>34</td>
</tr>
<tr>
<td>predefined dedicated variables</td>
<td>10</td>
</tr>
<tr>
<td>predefined general variables</td>
<td>10</td>
</tr>
<tr>
<td>PrintExpTime function</td>
<td>53, 118</td>
</tr>
<tr>
<td>proc_1d AU program</td>
<td>5, 102</td>
</tr>
<tr>
<td>Proc_err function</td>
<td>13, 136</td>
</tr>
<tr>
<td>processed data</td>
<td>66, 67, 68</td>
</tr>
<tr>
<td>PROJ</td>
<td>38</td>
</tr>
<tr>
<td>PS</td>
<td>32</td>
</tr>
<tr>
<td>PTILT</td>
<td>35</td>
</tr>
<tr>
<td>PTILT1</td>
<td>35</td>
</tr>
<tr>
<td>Q</td>
<td>QSIN</td>
</tr>
<tr>
<td>quick reference</td>
<td>6</td>
</tr>
<tr>
<td>QUIT</td>
<td>44</td>
</tr>
<tr>
<td>QUITMSG</td>
<td>44</td>
</tr>
<tr>
<td>R</td>
<td>R12</td>
</tr>
<tr>
<td>R13</td>
<td>39</td>
</tr>
<tr>
<td>R23</td>
<td>39</td>
</tr>
<tr>
<td>RACKPOW</td>
<td>31</td>
</tr>
<tr>
<td>raw data</td>
<td>66, 68</td>
</tr>
<tr>
<td>RDATASETLIST</td>
<td>25</td>
</tr>
<tr>
<td>reg file</td>
<td>33</td>
</tr>
<tr>
<td>relaxation analysis</td>
<td>34</td>
</tr>
<tr>
<td>remproc AU program</td>
<td>85</td>
</tr>
<tr>
<td>REV1</td>
<td>35</td>
</tr>
<tr>
<td>REV2</td>
<td>35</td>
</tr>
</tbody>
</table>
REXPNO 25, 60
RGA 28
RHNP 38
RHPP 38
RLUT 33
RMISC 33, 115
RMPLOT 40
ROT 29
rotation 29
ROTOFF 29
row of 2D raw data 76, 78
row of a 2D spectrum 70, 73
RPAR 27, 98
RPARSETLIST 27
RPROCNO 25, 63
RPULPROGLIST 20, 28
RS 34
RSC 38, 71
RSER 38, 76
RSER2D 80
rser2d command 6
RSH 29
RSR 38, 70
RV 34
RVNP 38
RVPP 38
RVTLIST 20, 30
S
SAB 32
sample.h 18
second AU dataset 56
SETCURDATA 24, 52, 53, 54
SETDATASET 25
SETHPCU 31
SETPARSER 19, 27, 28
SETPULPROG 28
SETSH 29
SETUSER 25
Show_status function 138
showfile function 139
Sine window multiplication 32
SINM 32
SINO 32
SOLVENT 29
Spline baseline correction 32
splitser AU program 77
SREF 32
ssleep function 140
stack plot 39
stdio.h 17, 18
stdlib.h 17, 18
STOP 44
STOPMSG 44
STOREDOSYPAR 27
STOREPAR 26, 94
STOREPARI 26, 94
STOREPARIS 26, 96
STOREPAR3 26, 94
STOREPAR3S 27, 96
STOREPARM 27, 94
STOREPARS 26, 96
STOREPLPAR 27, 94
STOREPLWPAR 27, 94
STOREPLXPAR 27, 94
STORET1PAR 27, 94
strcpy C-function 55
string.h 18
SUB1 36
SUB1D1 36
SUB1D2 36
SUB2 36
subroutines 10, 11
Suspend plot 39
SYM 36
SYMA 36
SYMJ 36
T
T1 value 34
T2 value 34
TABS1 39
TABS2 39
TABS3 39
Tcl/Tk scripts 5
TE2GET 30
TE2READY 30
TE2SET 30
TEGET 30
temperature unit 30
TEPAR 30
TEREADY 30
TESET 30
TF1 39
TF1P 39
TF2 39
TF2P 39
TF3 39
TF3P 39
third AU dataset 56
TILT 36
TIMES2 19
TIMES3 19
TIMESLIST 20
TM 32
TOJDX 41, 82
Trapezoidal baseline correction 35
Trapezoidal window multiplication 32
TRF 32
TUNE 29
TUNESX 29

U
uni.h 18
unistd.h 18
unlinkpr function 141
UNLOCKPLOTS 40
USECURPARS 19
USELASTPARS 19
user defined variables 10, 11
util.h 18
UWM 32
uxselect function 123

V
variable assignments 13
variable declarations 13
Varian dataset 42, 87
VCONV 42, 87
VIBAY 34
view mode 8
VIEWDATA 26, 59, 62, 64
viewing AU programs 8
vorspann file 16
VT 30

W
WAIT_UNTIL 42, 49
white washed stack plot 39
WMISC 33
WPAR 27, 99
WRA 25, 66
WRP 25, 67
WRPA 26, 68
WSC 38, 75
wsc command 6
WSER 38, 78
WSERP 38
WSH 29
WSR 73
wsr command 6

X
XAU 42
xau command 7, 8
XAUA 42
XAUP 42
XAUPW 42, 86
XCMD 42, 48
XFI 36
XF1M 36
XF1P 36
XF1PS 36
XF2 36
XF2M 36
XF2P 36
XF2PS 36
XFB 36
XFBM 36
XFBP 36
XFBPS 36
XHT1 36
XHT2 36
XIF1 36
XIF2 36
XMACH 42
xmac command 9
XTRF 36
XTRF2 37
XTRFP 37
XTRFP1 37
XTRFP2 37
XWP_LP 40
XWP_PP 40

Z
zero order phase correction 31
ZERT1 37
ZERT2 37
ZF 34
ZG 28
zg command 9
ZP 34
ZSPOIL 29