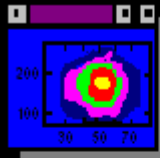


ROE n t D e k



Computer Based Online-offline Listmode Dataanalyser

CoboldPC

User Manual

Version 6.0.1.8 B77

This program is protected by international copyright laws as described in the About Box

Table of Content

Table of Content	2
1. General Introduction	5
1.1. Overview	6
1.2. List-Mode.....	6
1.3. Organisation	9
1.4. Windows95/98, Windows NT differences	10
1.5. Quick Startup	10
2. Installation.....	12
2.1. System requirements	13
2.2. Windows NT/Windows 2000	13
2.3. Windows 95/98	13
2.4. After installation requirements.....	14
3. The structure of CoboldPC	15
3.1. Coordinates	16
3.2. Spectra	16
3.3. Conditions.....	17
3.4. Parameters	17
4. The command level	19
5. Overview of commands	21
5.1. General and most frequently used commands.....	22
5.2. Data Acquisition Handling Commands	22
5.3. File Handling Commands.....	22
5.4. Definition and Delete Commands.....	22
5.5. Spectrum handling Commands.....	23
5.6. Display Commands.....	23
5.7. Mathematical Commands	23
6. Getting Started	24
7. Building your custom DAN.dll	26
7.1. Compiler Requirements.....	27
7.2. Sample for a User defined DAN.dll written in a MS/DEC-Fortran (.F90).....	27
8. Trouble Shooting	32
9. Also information available on.....	34
10. Commands.....	36
10.1. Definition and delete commands	37
10.1.1. Condition	37
10.1.2. Coordinate	37
10.1.3. Define1DimensionalSpectrum	38
10.1.4. Define2DimensionalSpectrum	38
10.1.5. DefineExperiment	39
10.1.6. Delete	40
10.1.7. Grid.....	40
10.1.8. Marker	41
10.1.9. Mode	41
10.1.10. Mode2D	41
10.1.11. Parameter	41
10.1.12. SetAxisText.....	42
10.1.13. SetPath	42
10.2. Display commands	43
10.2.1. Cursor	43
10.2.2. NextSpectrum.....	43
10.2.3. NoDisplay.....	43
10.2.4. OverlaySpectrum.....	43
10.2.5. PreviousSpectrum.....	43

10.2.6.	Show	43
10.2.7.	UpdateSpectrum	44
10.2.8.	ViewSpectrum	44
10.3.	Spectrum handling commands.....	45
10.3.1.	CalibrateSpectrum	45
10.3.2.	ClearSpectrum	45
10.3.3.	CopySpectrum.....	45
10.3.4.	CutOffNevativeValues	46
10.3.5.	ExpandSpectrum.....	46
10.3.6.	Remove	46
10.3.7.	SetChannelToValue.....	46
10.3.8.	ShiftSpectrum	47
10.3.9.	ZeroSpectrum	47
10.4.	Data acquisition commands	48
10.4.1.	NewAcquisition	48
10.4.2.	PauseAcquisition	50
10.4.3.	RewindListModeFile	50
10.4.4.	StartAcquisition.....	50
10.4.5.	StopAcquisition	50
10.4.6.	Wait.....	50
10.5.	File handling commands	52
10.5.1.	Restart command.....	52
10.5.2.	Open command	52
10.5.3.	ExecuteCommandFile	52
10.5.4.	Save command	52
10.5.5.	Save As command.....	52
10.5.6.	Print	53
10.5.7.	PrintPreview	53
10.5.8.	PrintSetup	53
10.5.9.	Preferences	53
10.5.10.	Exit	53
10.6.	Mathematical commands	54
10.6.1.	AddConstant	54
10.6.2.	AddSpectrum	54
10.6.3.	BinomialSmooth.....	54
10.6.4.	ContourSmooth	55
10.6.5.	DivideConstant	55
10.6.6.	DivideFunctionQ	55
10.6.7.	DivideSpectrum	56
10.6.8.	FitSpectrum	56
10.6.9.	GaussSmooth	56
10.6.10.	GP1Correct.....	57
10.6.11.	IntegrateSpectrum.....	57
10.6.12.	MeanSmooth.....	57
10.6.13.	MultiplyConstant	58
10.6.14.	MultiplyFunctionQ	58
10.6.15.	MultiplySpectrum.....	58
10.6.16.	ProjectSpectrum.....	59
10.6.17.	RotateSpectrum	59
10.6.18.	SubtractConstant	60
10.6.19.	SubtractFit	60
10.6.20.	SubtractSpectrum.....	61
A.	Appendix – List Mode Data Format.....	62
B.	Appendix – Table of Figures.....	64
Index	66	

1. General Introduction

1.1. Overview

CoboldPC is designed as a convenient data acquisition (DAQ) and online/offline data analysis (DAN) program for Windows NT (recommended) or Windows 95/98.

Due to its modular structure it can potentially address almost any hardware and allows very elaborate data treatments as the analysis plug-in can be modified by the user. The interface between the main program and the hardware is done by a dynamic link library named DAQ.dll while the analysis is performed by another link library named DAN.dll

With this program package you have received a DAQ.dll and at least one standard example analysis file DAN.dll suitable for the purchased hardware or as you have specified. You also received a start-up batch file (CoboldCommandFile, filename.ccf) which will enable you to easily start your respective CoboldPC session (e.g. DAQ-hardware read-out with simultaneous on-line control via the example DAN.dll) and already learn about the most frequently used CoboldPC commands.

If additional DAQ plug-ins are necessary due to change of hardware please contact **RoentDek** GmbH. You may change the analysis plug-ins in MS-Visual C++ 6, MS-Fortran Power Station 4 or DEC-Visual Fortran 6 code to adapt your DAN.dll data treatment procedure to your actual needs (see chapter 7).

The acquired data (events) received from the DAQ hardware can be stored on disc in list-mode format (ListModeFile, filename.lmf), i.e. the data are stored event by event (consisting of a row of coordinates simultaneously acquired) to re-run the analysis even after the experiment is finished as often as you like. The size of the ListModeFile is defined by the number of events required and the number of coordinates per event.

Simultaneously a previously defined DAN.dll allows on-line analysis and control of the acquired data (see chapter 7).

Your online sorting procedure using the DAN.dll and the CoboldCommandFile results in a number of one- and two-dimensional spectra showing acquired data in mutual relation. These spectra can also be stored to disc in the so called (DumpCoboldFile, filename.dcf), printed, exported to other programs (text-editors, display programs or other data analysis programs) or exported as plain ASCII via "Cut and Paste". You can also import ASCII-files in proper format to CoboldPC for data analysis in order to take advantage of the mathematical routines defined in CoboldPC.

All CoboldPC commands are listed and explained in the CoboldPC Help File. Just type "help" in the program command bar to start the help or press F1.

1.2. List-Mode

The program *CoboldPC* is a 'simple' C++ program to analyse List-Mode-Data. *CoboldPC* is standing for "Computer Based Online offline Listmode Dataanalyser"

List-Mode is a special technique used in collision-physics. In this mode all detected informations (named coordinates in *CoboldPC*) are stored event by event in a data-list (see Figure 1).

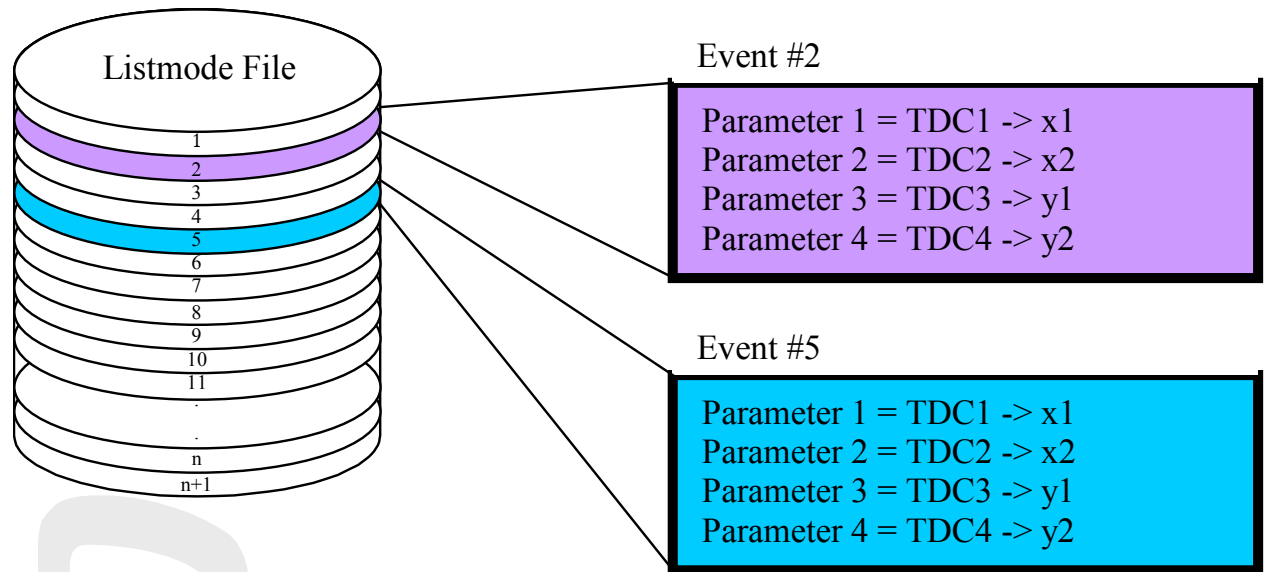


Figure 1: Schematic of a Listmode-File

Either during dataacquisition (on-line mode) or after the experiment (off-line mode) the list is presented to *CoboldPC* program. The program allows to sort and display the information. To combine informations of one event complex calculation and mathematics can be performed with the original event-data. During this process new co-ordinates are created and also sorted and displayed by the *CoboldPC* program.

Figure 2 shows the flow chart of the *CoboldPC* data taking and analysis.

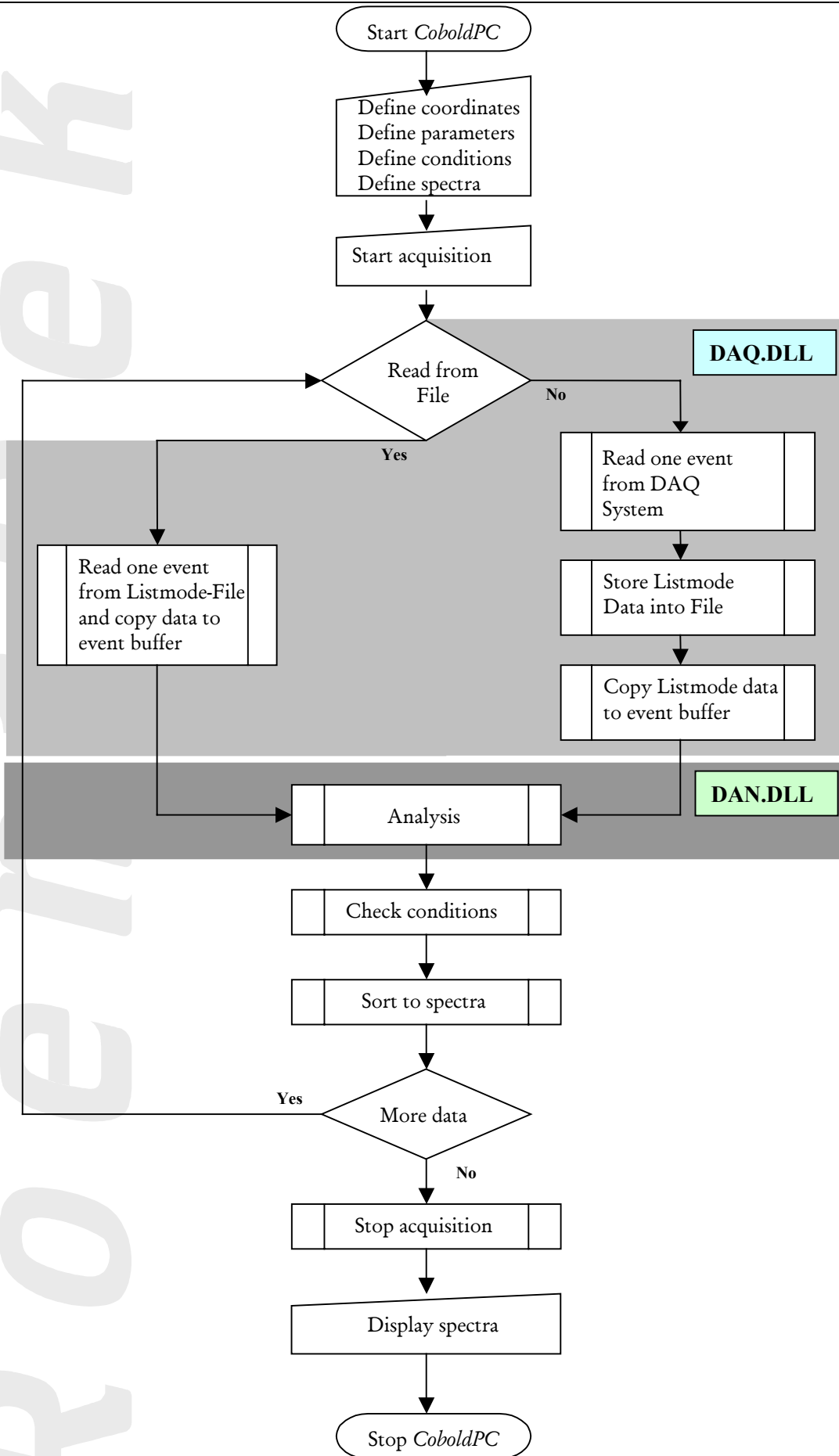


Figure 2: Flowchart of a CoboldPC session

1.3. *Organisation*

The CoboldPC program contains the main program that controls an included subset of acting-programs. For analysis and data acquisition this main program needs two additional program parts. DAN.DLL is responsible for recalculating the original List-Mode-Data to receive new co-ordinates. DAQ.DLL is handling all procedures for communicating with the DAQ hardware. It will also handle the writing of the List-Mode-Data to a file if selected by the User.

1.4. *Windows95/98, Windows NT differences*

Due to memory management differences in Windows95/98 and Windows NT there are a minor restrictions in the functionality of CoboldPC running on Windows95/98.

- All redrawing of the CoboldPC window has to perform the complete display routine. In Windows NT there is a screen buffering so only the first call of a spectrum will perform a complete drawing.
- Cut of spectrum graphics is disabled in Windows 95/98.
- Cut of spectrum ASCII is only enabled when displaying a one-dimensional spectrum in Windows95/98.

We hope we can support full functionality also in Windows95/98 soon.

1.5. *Quick Startup*

Start the CoboldPC program out of the CoboldPC program group as shown in Figure 3.

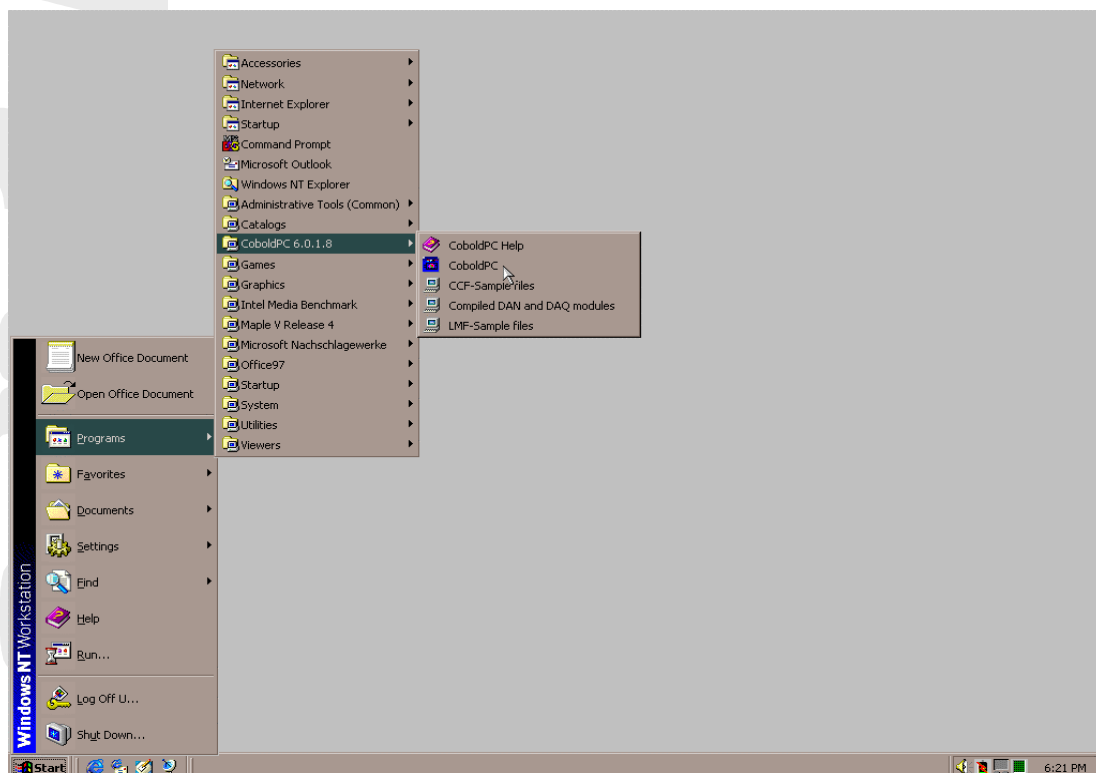


Figure 3: Start CoboldPC program from Start-button under Program-CoboldPC group

The opening window of CoboldPC is shown in Figure 4.

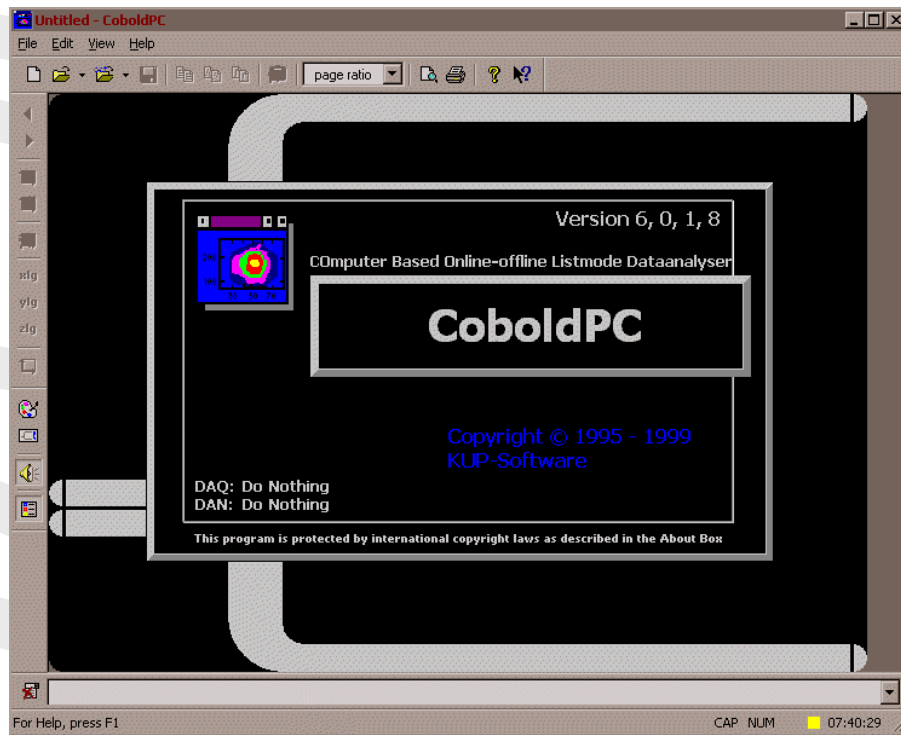


Figure 4: Startup window of CoboldPC

2. Installation

2.1. *System requirements*

The minimum system requirements to run Cobold are:

- Standard Intel (Pentium) based Computer
- 64MByte Memory
- Windows 95/98 or WindowsNT 4.0/Windows 2000
- Internet Explorer 3.0 or higher
- latest service releases for Windows95/98 or service packs for Windows NT

We recommend the following system

- Standard Intel Pentium II or Pentium III based Computer with > 300MHz
- > 128Mbyte memory
- Windows NT 4.0 with installed service pack 5 or higher or Windows 2000
- Internet Explorer 5

It is possible that CoboldPC will not run on Intel compatible PCs. (For example special japanese PCs)

2.2. *Windows NT/Windows 2000*

Be sure you logged on to an account related to the administration group (i.e. ADMINISTRATOR).

- switch to main directory on the CoboldPC CD.
- run the setup.exe program and follow the instructions on screen.
- during the setup of CoboldPC (not IE5) there will be a possible message that tells you to reboot. Do not reboot at this point. Continue the installation. After the installation if finished you should reboot the computer manually.

After the program is installed you can run CoboldPC from any standard user account.

If routine fails please do this and that, and if this fails contact **RoentDek**.

2.3. *Windows 95/98*

Please read comments in the Windows 95 or Windows 98NT in the help file for differences running CoboldPC in Windows95/98 or WindowsNT operation system, since not all functions are accessible in Windows 95/98 due to the internal memory management of these OS.

If you have not installed Internet Explorer 4.0 or newer

- switch to main directory on the CoboldPC CD.
- switch to Internet Explorer 5 directory
- run the ie.exe program and follow the instructions on screen.
- switch to main directory on the CoboldPC CD.
- switch to the CoboldPC 6.0.1.8 directory
- run the setup.exe program and follow the instructions on screen.

If Internet Explorer 4 or newer is installed

- switch to main directory on the CoboldPC CD.
- switch to the CoboldPC 6.0.1.8 directory
- run the setup.exe program and follow the instructions on screen.

If routine fails please do this and that, and if this fails contact **RoentDek**.

2.4. *After installation requirements*

Copy your specific DAN.dll-and DAQ.DLL version you want to use to the main CoboldPC-folder. You will find available precompiled versions of these DLLs in the **DAN and DAQ** directory that is located in your installation directory. If you have ordered special versions of these DLLs you will find the in the **DAN and DAQ\SPECIAL\...** directory.

Note that you have to copy the specific DLLs. After installation there are only dummy DLLs installed in the installation directory to allow immediate start of the CoboldPC program. These DLLs do nothing than to provide the CoboldPC main program with the required functions.

3. The structure of CoboldPC

The program package of CoboldPC consist of a main routine where the general data I/O infrastructure in Windows NT/98/95 are imbedded and a large number of data-treatment, storing and display routines are accessible. It contains exchangeable subroutines that can address different data I/O hardware (DAQ.dll) and allows the advanced user to compile custom DAN.dll-routines. When you start CoboldPC.exe it will automatically link the DAQ.dll and DAN.dll subroutines as well as some other .dll-files. Now your analysis session may begin. The main constituents of a CoboldPC analysis session are:

- Coordinates
- Spectra
- Conditions
- Parameters

These functions have to be defined now after starting the program by typing the proper commands in the command bar, which is located at the bottom of the CoboldPC window.

3.1. *Coordinates*

A coordinate attributes to the digital output of your hardware module like an ADC or TDC register. Thus you have to define at least as many coordinates as hardware channels are read out by the DAQ.dll and you can define additional coordinates as much as you like in addition to be used in your personal analysis routine DAN.dll. These additional coordinates can attribute for example to computations with hardware coordinates. In any case the number of hardware coordinates are always reserved in consecutive order for the hardware purpose. The syntax for defining the coordinates named e.g. x,y,z is:

```
coordinate x  
coordinate y  
coordinate z
```

or

```
coordinate x,y,z
```

3.2. *Spectra*

A spectrum is a 1- or 2-dimensional histogram memory allocated in the RAM of the PC which is used to visualise the acquired data, or better, a respective coordinate output. You need to define each spectrum with a set of command parameters (attributes for this spectrum) that define for example the size and the coordinate you want to visualise. You may define as many 1- or 2-dimensional spectra as you like (see chapter 6. In each hardware read-out cycle a number will be acquired from a the registers of your DAQ hardware, each register attributed to a coordinate. A 1-dimensional spectrum will thus to be filled with data resulting in a histogram for the specific coordinate. (like a multi-channel-analyser spectrum. In a 2-dimensional spectrum a bi-dimensional histogram of two coordinates (one at the x- and one at the y-axis) can be displayed, for example a 2d-detector image.

The syntax for a 1-dimensional spectrum is:

```
define1 0,1000,5,x,always,X-Spectrum
```

which defines a spectrum ranging from channel 0 to channel 1000 with a binsize of 5 sorting coordinate x without any condition (always) (see next subsection), with the Spectrum name „X-Spectrum“. The syntax for a 2-dimensional spectrum is:

```
define2 0,1000,5,x,0,1000,5,y,always,X-Y-Spectrum
```

sorting the x-coordinate on the x-axis and the y-coordinate on the y-axis, both ranging from 0 to 1000 with a binsize of 5 without condition.

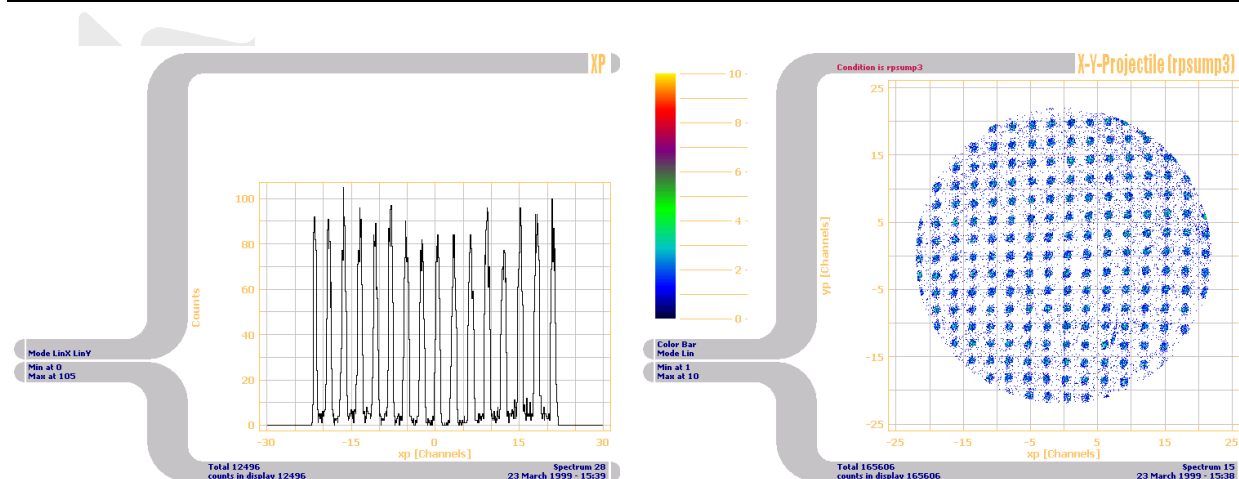


Figure 4: Example for a 1- and a 2-dimensional spectrum

3.3. Conditions

A condition is a one- or multidimensional window which is preventing data to be acquired into a certain spectrum (defined with that condition as command parameter) if the condition is not fulfilled. Conditions are either single windows on a specific coordinate or boolean combinations of two such windows and can be stacked as much as needed. The syntax is:

```
condition x,100,200,x-cond
```

for a window testing the coordinate “x” ranging from channel 100 to 200 with the condition name “x-cond”. If for an event the value of coordinate “x” is within that window, the event will be sorted to the spectra with the respective condition, otherwise it will be omitted.

Thus defining a spectrum:

```
define2 0,1000,5,y,0,1000,5,z,x-cond,Y-Z-Spectrum
```

will result in a 2 dimensional spectrum showing the y- and the z- coordinate, where only events with the x-coordinate between 100 and 200 are sorted in this respective spectrum.

The syntax for boolean combinations is:

```
condition x-cond,and,y-cond,xy-cond
```

for an “and” combination. Also “or”, “not” and “xor” are allowed combinations.

3.4. Parameters

A Parameter is a fixed number which can be set during running the analysis session and will be used within the DAN.dll (if also define there). This feature enables for example changing a calibration factor or something similar in the analysis without compiling it again. The syntax is

```
parameter 7,389.56
```

which gives the parameter 7 the value 389.56. Please start your custom parameter not before 100, since the first 99 parameter are usually reserved for the communication with the hardware (in the DAQ.dll, such as I/O addresses) and should not be used in the DAN.dll.

See the example.ccf-file for more examples.

The definition of the parameters and coordinates are interlinked with the DAQ- and DAN.dll-subroutines and must be consistent. See the example source code for more explanation.

The definition of spectra and conditions must address to coordinates that are defined and, in case of spectra with condition, the conditions must be defined.

For each Cobold session the appropriate set of coordinates, control parameters, (condition, and spectra) must be entered from the command level or by loading an early Cobold document from disc (.dcf-file). The recommended mode of entering the definition commands is to execute a batchfile (ASCII-format) which should have the extension .ccf from the command level with the “exe” command e.g., type ‘exe batchfile.ccf’ (batchfile.ccf must be in the main CoboldPC folder) in the command line or start from the menu.

To ease the startup and the operation of Cobold at least one CCF is predefined according to your specified requirements (startup.ccf) and you received an appropriate startup set of files.

In the LMF-Sample folder you find a CCF, LMF and two DLLs. You may start the COBOLDPC.prg in this directory and try your first CoboldPC session.

4. The command level

All commands in a CoboldPC session have to be entered via the command line or the menu. To ease up the input via command line one can use predefined batchfiles (ext. ccf) with the exe command (see chapter 10.5.1). If a started batchfile is finished a certain tone sequence will sound. All defined CoboldPC commands can be found in the help-file. To access the help menu press F1 or type "help".

It is not necessary to type the full command name. One needs only to type as many characters to make the beginning unique with respect to other CoboldPC commands. If too few characters are given and the command is ambiguous Cobold will execute the command which is first in alphabetical order.

Example: **shift** and **show**

batchfiles can also be addressed without the exe command just by typing the filename (without extension) in the command file. Note that there can be a name conflict with predefined commands and the above selection rules apply.

If a certain character combination is not found in the list of commands or as batchfile in the main folder a certain two-tone combination will sound.

Most commands require one or more command parameters, e.g. viewspectrum requires a spectrum number.

Note the difference of the term parameters as defined in chapter 3.4 (control parameters) and the command parameters as defined here.

A row of command parameters is separated from the command by a space and from each other by comma. The order of the command parameters is defined for each command (see helpfile). Some parameters are always required with the command (as the spectrum number after ViewSpectrum), some are optional as an input of xlow and xhigh after ViewSpectrum n,... (n is the spectrum number here).

If a required parameter is not supplied a Windows menu box will open and require the command parameter input. This is not the case when optional parameters are omitted, then the program uses default values or values given before.

Sometimes a command requires the setting of markers on the displayed spectrum with the mouse. Before the marks are not set the command line is disabled.

Note: in rare cases after mouse operation the command line will be disabled although there is no obvious reason. In this case click on the command line and the blinking command marker indicates that the command line is enabled again.

With arrow up/arrow down you may scroll to the previous command list, you also find the last typed commands in a drop down list (right button next to the command line).

When you use a commands in a batch file that requires for example mouse operation or starts a subroutine you have to follow this command by the wait command. Then the execution of the batch-file is paused until the previous command operation is finished properly. Examples:

Cursor

Marker

IntegrateSpectrum

5. Overview of commands

The following list of all CoboldPC commands shall only give hints what the CoboldPC program package is capable of from the command level and shall ease up the use of the help-menu.

Here you find a list of the most frequently used commands and some brief explanation of there function. For details consult the help-file by typing help or press F1

5.1. *General and most frequently used commands*

exe	calls a command file
new hardware	prepares for starting
start	starts the acquisition
pause	pauses the acquisition (for starting again use the start command)
stop	stops the acquisition (for starting again use the new hardware and the start command)
clear all	clears the contents of all spectra (instead of all, a certain spectrum number is also possible to clear only one spectrum contents)
restart	deletes all coordinates, parameters, conditions and spectra for a total reset
show status	shows the status report window
help	shows the help file
wait	pauses the batch file until the previous command operation is finished

5.2. *Data Acquisition Handling Commands*

NewAcquisition	Prepare Cobold for taking data.
PauseAcquisition	Pauses the data acquisition.
RewindListModeFile	Set the read file pointer to the first data in ListModeFile.
StartAcquisition	Starts (restarts) the data acquisition.
StopAcquisition	Stops data acquisition and closes ListModeFile.
Wait	Wait command to block inputs by mouse or keyboard.

5.3. *File Handling Commands*

Restart	Restarts with an empty Cobold document.
Open	Opens an existing document.
Execute	Execute a command file.
Save	Saves an opened document using the same file name.
Save As	Saves an opened document to a specified file name.
Print	Prints a document.
Print Preview	Displays the document on the screen as it would appear printed.
Print Setup	Selects a printer and printer connection.
Preferences	Define global CoboldPC settings
Exit	Exits Cobold.

5.4. *Definition and Delete Commands*

Condition	Defines a condition.
Coordinate	Defines coordinate(s).
Define1DimensionalSpectrum	Defines a 1 dimensional spectrum.
Define2DimensionalSpectrum	Defines a 2 dimensional spectrum.
DefineExperiment	Define experiment information.
Delete	Delete function.
Grid	Toggle grid on and off.
Marker	Defines a marker in a 1- or 2-dimensional spectrum.
Mode	Defines lin- or log-modes for the axes.
Mode2D	Defines the display type of a 2-dimensional spectrum.
Parameter	Defines the value of a specified parameter.
SetAxisText	Define the axis text.

SetPath Define default paths for file io.

5.5. *Spectrum handling Commands*

CalibrateSpectrum Calibrate a spectrum.
ClearSpectrum Clear one or more spectra.
CopySpectrum Copy a spectrum.
CutOffNegativeValues Set all negative values in spectrum to zero.
ExpandSpectrum Expand a spectrum.
Remove Remove additional data from a spectrum
SetChannelToValue Set a channel in a spectrum to a specific value.
ShiftSpectrum Shift the spectrum.
ZeroSpectrum Set a region in a spectrum to zero.

5.6. *Display Commands*

CopyGraphics Copy displayed spectrum graphic to clipboard.
CopyASCII Copy displayed spectrum channel values to clipboard.
CopyIntegrationData Copy displayed spectrum integration information to clipboard.
Cursor Display a cursor marker without storing data.
ExportASCII Export displayed spectrum channel values to a file.
ImportASCII Import channel values from a file to the displayed spectrum.
NextSpectrum Display the next available spectrum.
NoDisplay Removes any spectrum from the display area.
OverlaySpectrum Overlay a spectrum to the displayed one.
PasteASCII Copy channel values in ASCII format from the clipboard to the displayed spectrum.
PreviousSpectrum Display the previous available spectrum.
Show Show the specified lists.
UpdateSpectrum Update the display of the spectrum being viewed.
ViewSpectrum View the specified spectrum.

5.7. *Mathematical Commands*

AddConstant Add a constant value to a spectrum.
AddSpectrum Add two spectra.
BinomialSmooth Perform a binomial smooth.
ContourSmooth Smooth a contour plot.
DivideConstant Divide spectrum by a constant value.
DivideFunctionQ Multiply all channels by $X^{(1/Q)}$.
DivideSpectrum Divide two spectra.
FitSpectrum Fitting Spectrum Data
GaussSmooth Perform a smooth using a gaussian distribution.
GP1CORRECT Correct the differential nonlinearity of the GP1-TDC
IntegrateSpectrum Integrates a defined area in the displayed spectrum.
MeanSmooth Perform a mean smooth.
MultiplyConstant Multiply a spectrum by a constant value.
MultiplyFunctionQ Multiply all channels by x^Q .
MultiplySpectrum Multiply two spectra.
ProjectSpectrum Makes a projection.
RotateSpectrum Rotate a two dimensional spectrum.
SubtractConstant Subtract a constant value from a spectrum.
SubtractFit Subtract the last fit from the given spectrum.
SubtractSpectrum Subtract two spectra.

6. Getting Started

By now you should have understood the program structure and have an idea what command options are available.

You may now start a Cobold session either in the example folder or in the main CoboldPC folder for example with your own hardware (make sure the hardware is working properly and data are coming in). Check the .ccf-files and verify that you understand most of the commands. Some parameter settings may not be understandable for you at this point. Also some defined coordinates might not be obvious. These are depending on the DAQ.dll and DAN.dll you are using. In the .ccf-files names are attributed to the coordinates for further use.

A special coordinate is "n" which is attributed to the event number. A counter is incremented with each event analysed. The number and order of coordinates are defined by the DAN.dll and DAQ.dll routines. A spectrum that was displayed once will not automatically update as new data come in. You may update manually or use the update command to set an auto-update.

You may change, remove or add spectra or condition definitions and observe the effect of your changes of the ccf-file. If you change parameters make sure you don't supply parameter combinations that are not in accordance with the requirements of the DAN.dll or DAQ-routines.

The spectrum numbers will be attributed to the spectra in the order they have been defined.

Note that each spectrum definition will allocate a certain portion of the PC-RAM according to the number of channels and bin-size. Especially 2D spectra might absorb a large amount of RAM. As the RAM is limited you should choose spectra definitions with minimal size and tolerable bin-size (i.e. spectral resolution) for your respective information needs from this spectrum. When you want to save your CoboldPC session as a dump-file have in mind that it will occupy as much hard disc space as RAM was allocated. The save command will store all spectra information (definition and actual content) and also other definitions as control parameters.

You may store graphics or ASCII of single spectra by clicking with the right mouse button on the displayed spectra (follow pop-up menu instructions) likewise you may import ASCII in proper format to predefined spectra. You may also print a displayed spectrum, colour or gray mode are optional (see help file: **mode2D**)

Note again the difference between storing the acquired data in CoboldListMode format (.lmf) and CoboldDumpFile format (.dcf):

The .lmf will store the hardware data in a list if you have defined a filename as parameter in the **new** command. This file contains the basic information content of your hardware acquired data. To extract the information you run a CoboldPC session with defined spectra and condition, on-line (while acquiring and storing the data in .lmf) or off-line (when analyzing the data in the list-mode file afterwards). Then you eventually store the CoboldPC session with the spectra content, etc. to disc. This is a "snapshot" of the data treatment as performed so far. From this save you will be able to do fit routines or spectra calculus later, but there is no way to do changes of conditions for spectra or parameters for the analysis from this CoboldDumpFile.

To re-run the data analysis with all options you need the .lmf-file as written during data acquisition.

During tests and start-up of an experiment (data read-out) it might be sufficient just to analyze the data as they come on-line without saving it as listmode-file.

The save of list mode-file and .dcf-file to disc can be automated by parameters (e.g. store a .lmf after 10000 events) also, you may purchase an additional I/O board to control outer hardware parameters via CoboldPC or to control CoboldPC via external I/O commands.

For details about this strongly application dependent option contact **RoentDek**

7. Building your custom DAN.dll

For most applications the DAN.dll subroutine(s) you purchased with the program are sufficient for your data acquisition and analysis needs.

However for special and more flexible DAN.dll we enable the user to adjust the DAN.dll according by compiling its own analysis.dll if the appropriate compiler is available.

The main aim changing the code is usually to define additional coordinates that are computed from other coordinates (for example to make an R/Phi representation of an acquired detector image). You may define additional control parameters that can be addressed from the command level later. Make sure that you adjust your .ccf according to newly defined control parameters or coordinates. The newly defined coordinates will be treated as the other predefined coordinates, i.e. they can be visualized in spectra in on-line and off-line analysis.

Note that in the .lmf only those coordinates defined by the hardware are stored (those containing the complete hardware information). So you may address a given .lmf-file with different DAN.dll, as long as your DAN.dll complies with the DAQ (the control parameters and coordinate definition required by the hardware must be consistent). To use your self-compiled you must copy your DAN.dll to the main CoboldPC folder.

7.1. *Compiler Requirements*

To be able to build your own DAN.dll you need to fulfill the following software requirements

- MS Visual C++ 6.0 or
- MS Fortran Power Station 4.0 or
- DEC Visual Fortran 6.0

To build your own DAQ.dll you need to fulfill the following software requirements

- MS Visual C++ 6.0

For these programming languages you will find ready to use project files in the source folder located in your CoboldPC installation directory. (only available if selected during setup)

If you have problem to build your own DAN.dll contact **RoentDek** Note that the compiler is not part of the program package and the support from **RoentDek** in building your own DAN.dll can only be directive. There are well documented program examples included in the program package, so a medium skilled programmer will be able to compile its own .dll-file if the hardware and software requirements are met. Usually the DAN.dll you purchased are written in C++ language.

7.2. *Sample for a User defined DAN.dll written in a MS/DEC-Fortran (.F90)*

Please do not change the following structure, just include your code in the marked areas (Don't even remove parts which look like a comment, they might not be comments!). All this is necessary for the communication between this Fortran subroutine and the CoboldPC program which is written in C++ (Be careful with word wrap)

The programm code is the same for the DEC compiler. You should only replace *MS!* with *DEC!* in the following listing.

Here is the code for MS-Fortran.

```
!      DoUserStuff called from cobold main program

!      this is the VC++ declaration of the function
!      DoUserStuff(CDoubleArray *pEventData,CDoubleArray *pParameters)

integer*4 function AnalysisGetInformationString()
!MS$ATTRIBUTES C, alias:'_AnalysisGetInformationString'::AnalysisGetInformationString
!MS$ATTRIBUTES DLLEXPORT::AnalysisGetInformationString

integer StringEnd
character*256 AnalysisInformationString
common /AnalysisInformation/ AnalysisInformationString

      AnalysisInformationString = 'Sample for Wedge-and-Strip-detector readout'
      AnalysisGetInformationString = loc(AnalysisInformationString)
      StringEnd = index(AnalysisInformationString,' ')
```

```

        AnalysisInformationString(StringEnd:StringEnd) = char(0)
end

integer*4 function AnalysisInitialize(CEvents,CParameters)
!MS$ATTRIBUTES C, alias:'_AnalysisInitialize'::AnalysisInitialize
!MS$ATTRIBUTES DLLEXPORT::AnalysisInitialize
!MS$ATTRIBUTES REFERENCE::CEvents
!MS$ATTRIBUTES REFERENCE::CParameters

implicit none

type CDoubleArray
    integer*4    CObject
    integer*4    aData
    integer*4    nVersionNumber
    integer*4    nSize
end type CDoubleArray

type (CDoubleArray)::CEvents
type (CDoubleArray)::CParameters

! creating pointer to the double array
real*8 EventData(0)
real*8 Parameters(0)
!////////////////////////////////////
! user definitions please insert here
! This part is for the user code executed with each "new"-command
!////////////////////////////////////

!////////////////////////////////////
! end of user definitions
!////////////////////////////////////

! assigning pointers to type of source
POINTER (p1,EventData)
POINTER (p2,Parameters)
! the address of the array is in aData
p1=CEvents%aData
p2=CParameters%aData

!////////////////////////////////////
! user code insert here
! This part will be executed with each "new"-command
!////////////////////////////////////

!////////////////////////////////////
! end of user code
!////////////////////////////////////

AnalysisInitialize = 1

end

subroutine AnalysisProcessEvent(CEvents,CParameters)
!MS$ATTRIBUTES C, alias:'_AnalysisProcessEvent'::AnalysisProcessEvent
!MS$ATTRIBUTES DLLEXPORT::AnalysisProcessEvent
!MS$ATTRIBUTES REFERENCE::CEvents
!MS$ATTRIBUTES REFERENCE::CParameters

implicit none

type CDoubleArray
    integer*4    CObject
    integer*4    aData
    integer*4    nVersionNumber
    integer*4    nSize
end type CDoubleArray

type (CDoubleArray)::CEvents
type (CDoubleArray)::CParameters

! creating pointer to the double array
real*8 EventData(0)
real*8 Parameters(0)

```

```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! user definitions please insert here
! This part is for the user code which will be executed with each single event
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

real*8 meander,wedge,strip
real*8 sum,x,y,r
real*8 amp_m,amp_w,amp_s,off_m,off_w,off_s,mw,ms,ws
real*8 switch,inv_x,inv_y,x0,y0,cal_x,cal_y,excxy,rot

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! end of user definitions
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! assigning pointers to type of source
POINTER (p1,EventData)
POINTER (p2,Parameters)
! the address of the array is in aData
p1=CEvents%aData
p2=CParameters%aData

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! user code insert here
! This part will be executed with each single event
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

meander      = EventData(1)
wedge       = EventData(2)
strip       = EventData(3)

!-----
!      Extract parameters
!-----

switch = Parameters(10)
amp_m  = Parameters(11)
amp_w  = Parameters(12)
amp_s  = Parameters(13)
off_m  = Parameters(21)
off_w  = Parameters(22)
off_s  = Parameters(23)
mw     = Parameters(31)
ms     = Parameters(32)
ws     = Parameters(33)
excxy  = Parameters(41)
inv_x  = Parameters(42)
inv_y  = Parameters(43)
rot    = Parameters(44)
x0     = Parameters(45)
y0     = Parameters(46)
cal_x  = Parameters(47)
cal_y  = Parameters(48)

!-----
!      detector 1
!-----
call wedge_and_strip(switch,meander,wedge,strip,amp_m,amp_w,amp_s,off_m,off_w,off_s,
mw,ms,ws,x0,y0,cal_x,cal_y,excxy,inv_x,inv_y,rot,sum,x,y)

      r = sqrt(x**2 + y**2)

!-----
!      redefinition of EventData
!-----

EventData(11) = sum
EventData(12) = x
EventData(13) = y
EventData(14) = r

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! end of user code
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

```

```

end

!////////////////////////////////////
! subroutine wedge_and_strip
!////////////////////////////////////

SUBROUTINE wedge_and_strip(switch,meander,wedge,strip,amp_m,amp_w,amp_s,off_m,off_w,
off_s,mw,ms,ws,x0,y0,cal_x,cal_y,exc_xy,inv_x,inv_y,rotate,sum,x,y)

implicit none
real*8 switch,cal_x,cal_y,x0,y0,exc_xy,inv_x,inv_y
real*8 amp_m,amp_w,amp_s,off_m,off_w,off_s,mw,ms,ws
real*8 meander,wedge,strip,truemeander,truewedge,truestrip,sum
real*8 x,y,tempx,tempy,a,b,determinante,xnorm,rotate
real*8 irange
!-----
!                                     start
!-----
! irange = 2048.0                                !Full ADC range
!-----
! checks
!-----
! if (switch.eq.0.) then                                !All corrections off
!   amp_m = 1.
!   amp_w = 1.
!   amp_s = 1.
!   off_m = 0.
!   off_w = 0.
!   off_s = 0.
!   mw      = 0.
!   ms      = 0.
!   ws      = 0.
!   x0      = 0.
!   y0      = 0.
!   cal_x   = 1.
!   cal_y   = 1.
!   inv_x   = 0.
!   inv_y   = 0.
!   exc_xy  = 0.
! else
!   if (amp_m.eq.0.) amp_m = 1.
!   if (amp_w.eq.0.) amp_w = 1.
!   if (amp_s.eq.0.) amp_s = 1.
!   if (cal_x.eq.0.) cal_x = 1.
!   if (cal_y.eq.0.) cal_y = 1.
! endif
!-----
! offset subtraction and amplification
!-----
! meander = (meander - off_m) * amp_m
! wedge   = (wedge   - off_w) * amp_w
! strip   = (strip   - off_s) * amp_s
!-----
! capacitive linear crosstalk
!-----
! if(mw.eq.0.and.ms.eq.0.and.ws.eq.0)then
!   truemeander = meander
!   truewedge   = wedge
!   truestrip   = strip
! else
!   a = mw + ms + ws
!   b = (mw*ms) + (mw*ws) + (ms*ws)
!   determinante=1.-(2.*a)+(3.*b)
!   if(determinante.ne.0)xnorm=1./determinante
!   truemeander=xnorm*(meander*(1.-ws-a+b) + wedge *(b-mw) + strip *(b-ms))
!   truewedge=xnorm*(wedge *(1.-ms-a+b) + strip *(b-ws) + meander*(b-mw))
!   truestrip=xnorm*(strip *(1.-mw-a+b) + meander*(b-ms) + wedge *(b-ws))
! endif
!-----
! sum and x,y
!-----
! sum = truemeander + truewedge + truestrip

```

```
      if (sum.ne.0.) then
        x = cal_x * (truewedge / sum * irange - x0)
        y = cal_y * (truestrip / sum * irange - y0)
      else
        x = 1.0E9
        y = 1.0E9
      endif
      sum = sum / 3.0
!-----
!   Exchange
!-----
      if (exc_xy.eq.1.) then
        tempx = x
        x = y
        y = tempx
      endif
!-----
!   Invert
!-----
      if (inv_x.eq.1.) x = irange/2.0 - x
      if (inv_y.eq.1.) y = irange/2.0 - y
!-----
! rotating
!-----
      if (rotate .ne. 0.0) then
        if (x0.eq.0..and.y0.eq.0.) then
          tempx=x - irange/4.0
          tempy=y - irange/4.0
          x = tempx*cosd(rotate) - tempy*sind(rotate) + irange/4.0
          y = tempx*sind(rotate) + tempy*cosd(rotate) + irange/4.0
        else
          tempx=x
          tempy=y
          x = tempx*cosd(rotate) - tempy*sind(rotate)
          y = tempx*sind(rotate) + tempy*cosd(rotate)
        endif
      endif
    endif
  return
end
```

8. Trouble Shooting

If you have problems during the installation or using our software package (source codes for DAN and DAQ DLLs) visit our [web site](#) or contact RoentDek.

WEB: <http://www.roentdek.com>

Fax: +49(69)798 21601

e-Mail software@roentdek.com

9. Also information available on

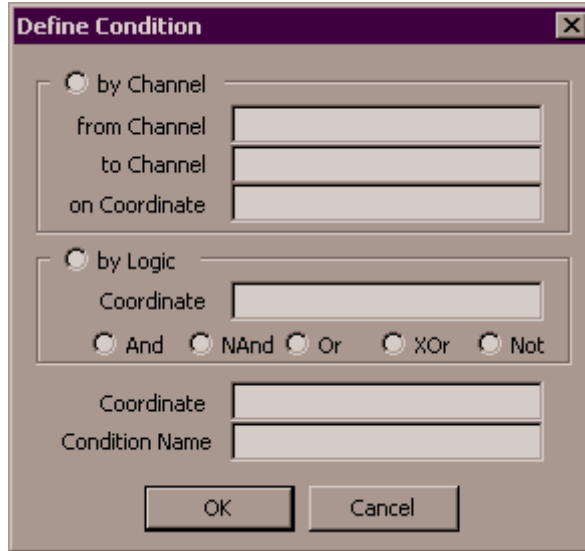
- MCP-detectors with UHV-compatible Wedge-and-Strip anodes.
- MCP-detectors with UHV-compatible one-dimensional resistive anodes and one-dimensional delay-lines
- Residual gas beam profile monitor for UHV based on one- and two-dimensional MCP-detectors.
- Specially adapted electronic modules for position sensitive detectors.
- 2D-position sensitive proportional counters for X-rays and neutrons
- Supersonic gas jet targets
- Multi fragment imaging systems (Cold Target Recoil Ion Momentum Spectroscopy COLTRIMS).

10. Commands

10.1. Definition and delete commands

10.1.1. Condition

Defines or changes a condition.



Parameters:

Coordinate,MinimumValue,MaximumValue,NewConditionName
 Condition1,Operation,Condition2,NewConditionName
 NOperation,Condition,NewConditionName

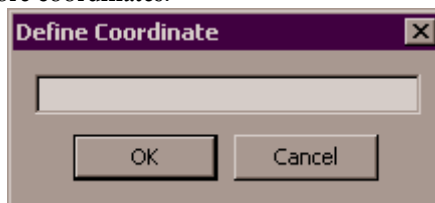
<i>Coordinate</i>	the coordinate that will be checked.
<i>MinimumValue, MaximumValue</i>	the checked range.
<i>NewConditionName</i>	the name of the defined condition.
<i>Operation</i>	Can have the following values.
And	both of the conditions are true.
Or	either condition is true.
Xor	one condition is true, the other false.
Nand	none of the conditions are true.
<i>NOperation</i>	
Not	the condition is false.

Example:

Condition coordinate1,100,350,condition1;	test condition1 between 100 and 350
condition coordinate2,500,600,condition2;	test condition2 between 500 and 600
condition condition1,and,condition2,condition3;	require both condition1 and condition2 to be true
condition not,condition1,condition4;	not condition1

10.1.2. Coordinate

This command will define one or more coordinates.



Parameters:

Coordinate1[,Coordinate2,Coordinate3...]

Examples:

coordinate x

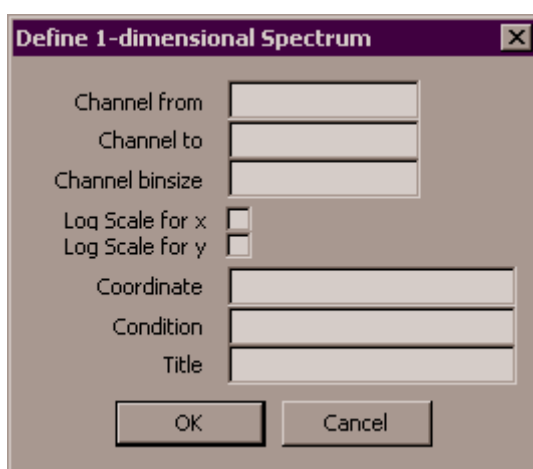
define a coordinate with the name x

coordinate sum1,sum2

define coordinates with the names sum1 and sum2

10.1.3. Define1DimensionalSpectrum

Defines a 1-dimensional spectrum. If the leading character of **Minimum**, **Maximum** or **Binsize** is a P or p then the following number will be interpreted as a parameter. The content of the parameter will be used for the specified definition.

**Parameters:**

Minimum,Maximum,Binsize,Coordinate,Condition,Name[,LogXFlag,LogYFlag]

Minimum

Lower boundary of the spectrum.

Maximum

Upper boundary of the spectrum.

Binsize

Binning of the spectrum.

*Coordinate*Coordinate for this spectrum or **none**.*Condition*Condition for this spectrum or **always**.*Name*

Title of the spectrum.

*LogXFlag***true** to set logx mode in spectrum*LogYFlag***true** to set logy mode in spectrum**Examples:**

define1dimensionalspectrum 0,500,5,x,csum,X-Projection

define1dimensionalspectrum 0,500,5,none,always,X-Projection

Note:

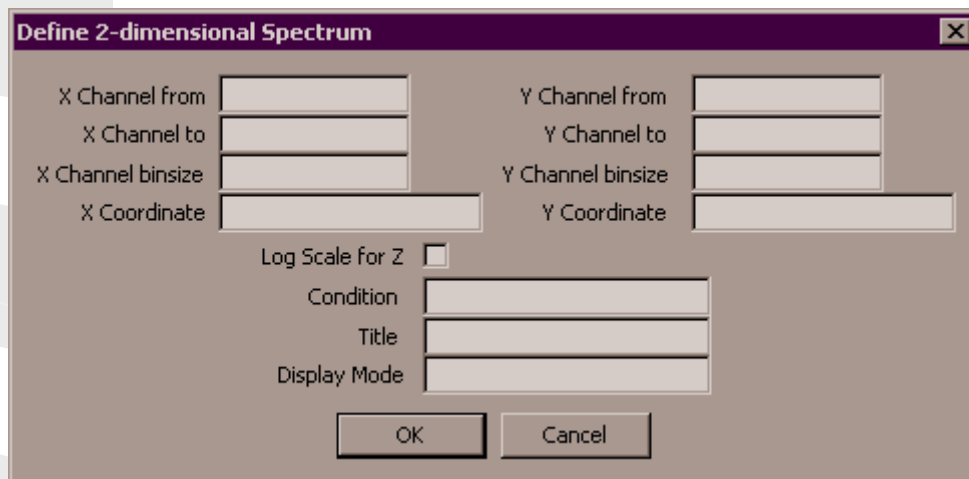
Due to the floating point problematic CoboldPC will accept values for coordinates even if they are $10e-9$ * Minimum below or $10e-9$ * Maximum above the range definition.

The channel acceptance (except definition borders) is from Channel-Binsize/2 to Channel+Binsize/2.

The right value is not included.

10.1.4. Define2DimensionalSpectrum

Defines a 2-dimensional spectrum. If the leading character of **XMinimum**, **XMaximum**, **XBinsize**, **YMinimum**, **YMaximum** and **YBinsize** is a P or p then the following number will be interpreted as a parameter. The content of the parameter will be used for the specified definition.

**Parameters:**

XMinimum,XMaximum,XBinsize,XCoordinate,YMinimum,YMaximum,YBinsize,YCoordinate,Condition,Name[,Mode2d,LogZFlag]

<i>XMinimum</i>	Lower x-boundary of the spectrum.
<i>XMaximum</i>	Upper x-boundary of the spectrum.
<i>XBinsize</i>	Binning for the x-axis.
<i>XCoordinate</i>	Coordinate for x-axis or none .
<i>YMinimum</i>	Lower y-boundary of the spectrum.
<i>YMaximum</i>	Upper y-boundary of the spectrum.
<i>YBinsize</i>	Binning for y-axis.
<i>YCoordinate</i>	Coordinate for y-axis or none .
<i>Condition</i>	Condition for this spectrum or always .
<i>Name</i>	Title of this spectrum.
<i>Mode2d</i>	Define viewing mode of the displayed spectrum. (see Mode2d) valid values are, color , gray , scatter , density , contourcolor , contourgray .
<i>LogZFlag</i>	true to set logz mode in spectrum

Examples:

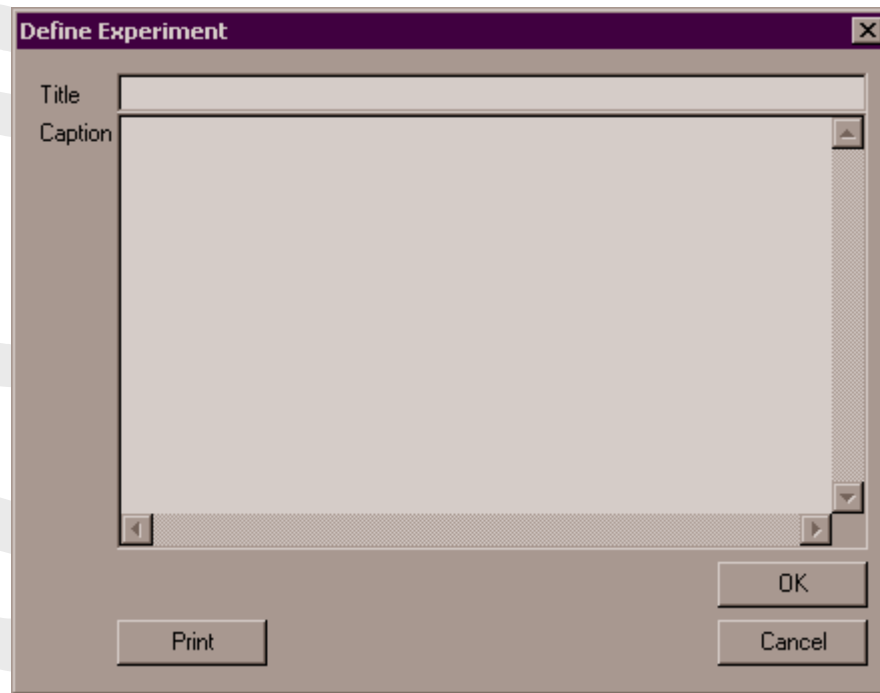
```
define2dimensionalspectrum 0,500,5,x,200,700,8,y,csum,Position Picture
define2dimensionalspectrum 0,500,5,x,200,700,8,y,none,always,Dummy 2D Spectrum,gray
```

Note:

Due to the floating point problematic CoboldPC will accept values for coordinates even if they are $10e-9$ * Minimum below or $10e-9$ * Maximum above the range definition.
The channel acceptance (except definition borders) is from $\text{Channel}-\text{Binsize}/2$ to $\text{Channel}+\text{Binsize}/2$.
The right value is not included.

10.1.5. DefineExperiment

Define the relevant information about the experiment and this document.



10.1.6. Delete

Function to delete parts of a Cobold document.

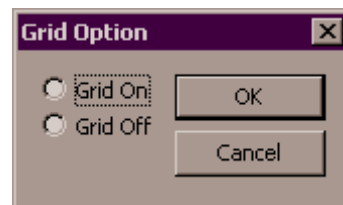
Parameters:

Type,Number1[, Number2,Number3...]

<i>Type</i>	Spectrum	Color-scale plot.
<i>Number#</i>		Color-scaled contour plot.

10.1.7. Grid

Turn grid on or off in a spectrum.



Parameter:

Flag

<i>Flag</i>	On or Off	Specify on to turn grid on Specify off to turn grid off
-------------	------------------	--

Note:

For dialog box support leave parameters empty.

10.1.8. Marker

Defines and displays a dashed marker in a 1- or 2-dimensional spectrum. While the crosshair cursor is displayed the mouse cursor is disabled.

Parameters:

Value1[,Value2]

Value1

x position of the marker (1- or 2-dimensional spectrum)

Value2

y position of the marker (2-dimensional mode only)

10.1.9. Mode

Defines the linear or logarithm mode for the axis of the displayed spectrum. For a 1-dimensional spectrum, the x- and/or the y-axis can be set to linear or logarithm mode. For a 2-dimensional spectrum, only the z-axis can be set to linear or logarithm mode.

Parameter:

Axismode

Axismode

lin or **log** for a 2-dimensional spectrum.

linlin both axis in linear mode.

linlog x-axis linear-, y-axis log-mode.

loglin x-axis log-, y-axis linear-mode

loglog both axis in log-mode.

10.1.10. Mode2D

Define viewing mode of the displayed spectrum. This function is only available for 2-dimensional spectra. For a contour plot, the smoothing function will not affect the original data.

Parameters:

Viewmode

Contourmode[,# of lines,# of iterations]

Viewmode

Color

Color-scale plot.

Gray

Gray-scale plot.

Grey

Grey-scale plot.

Scatter

Scatter plot.

Density

Density plot.

Contourmode

Contourcolor

Color-scaled contour plot.

Contourgray

Gray-scaled contour plot.

Contourgrey

Grey-scaled contour plot.

of lines

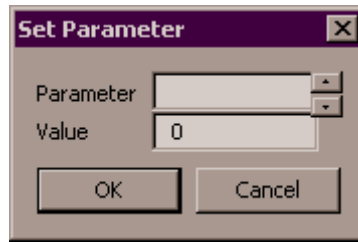
Number of lines in the contour plot. (default is 10)

of iterations

Number of iterations of smoothing before displaying the contour plot. (default is 10)

10.1.11. Parameter

Defining or modifying a parameter.

**Parameters:**

Number, Value

Number
Value

The number of the parameter.

The content of the specified parameter.

Note:

To define a hexadecimal number use 0x before the number or **h** after the number.

10.1.12. SetAxisText

Define the axis text. This text will replace the coordinate text. To remove selected text leave entry for axis text empty.

**Parameters:**

Spectrum, XText, YText

Spectrum
Xtext
Ytext

Define spectrum number.

Text to replace x coordinate.

Text to replace y coordinate.

10.1.13. SetPath

Define the initial default path for Cobold Documents, Command Files, List Mode Files, Import- and Export Files.

Parameters:

Path

Path

New initial path for File IO.

APPLICATIONPATH will set the path to the directory where CoboldPC is located, **WORKPATH** sets the default working directory.

10.2. *Display commands*

10.2.1. **Cursor**

Display a cursor marker. On exiting cursor mode the last cursor values are not stored into the spectrum. While the crosshair cursor is displayed the mouse cursor is disabled.

10.2.2. **NextSpectrum**

View the next available spectrum.

10.2.3. **NoDisplay**

This function removes any spectrum from the display area and prevents its redrawing.

10.2.4. **OverlaySpectrum**

Overlays a specified spectrum with a specified color.

Parameters:

Spectrum,Color[,Spectrum,Color,...]

Spectrum
Color

Spectrum number to be overlaid
R G B value of the color.

Note:

Samples for R G B values:
0xff 0xff 0xff
255 0 0xc0

10.2.5. **PreviousSpectrum**

Display the previous available spectrum.

10.2.6. **Show**

Display the specified list or status. For show status the display is updated every second.

Parameters:

Type[, Text]

Type

Type one of these words to specify the list. **Coordinates**, **Conditions**, **IntegrationData**, **ListmodeFileHistory**, **Marker**, **Message**, **Overlay**, **Parameters**, **Spectra**, **Status**, **Rate** (Event Rate).

Text

Text to be displayed in a message box if Type is **Message**. A newline command can be given by the \n character sequence.

Note:

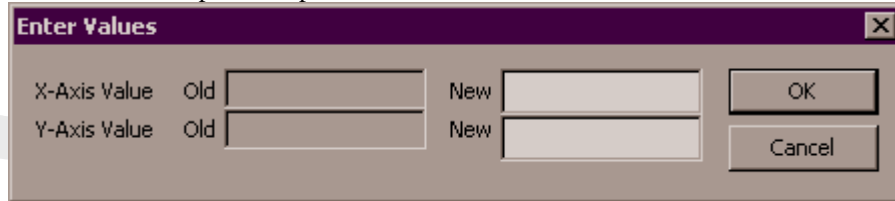
In the **IntegrationData** or **Marker** list you can copy the list to the clipboard.

In the Spectra List you can activate the selected spectrum by pressing the Go To button or by double clicking the line of the spectrum that should be displayed.

10.3. Spectrum handling commands

10.3.1. CalibrateSpectrum

This function will calibrate the specified spectrum.



Parameters:

Type, XMinimum, NewXMinimum, XMaximum, NewXMaximum, YMinimum, NewYMinimum, YMaximum, NewYMaximum

Type

LIN performs a linear calibration.

XMinimum

Old x-minimum value

NewXMinimum

new x-minimum value

XMaximum

Old x-maximum value

NewXMaximum

new x-maximum value

YMinimum

Old y-minimum value

NewYMinimum

new y-minimum value

YMaximum

Old y-maximum value

NewYMaximum

new y-maximum value

Note:

If no boundaries are entered a cursor-dialog based procedure will be used for boundary input.

10.3.2. ClearSpectrum

This function will clear the specified spectra.

Parameters:

Spectrum1[,Spectrum2,Spectrum3,...]

Word

Spectrum1..n

Word

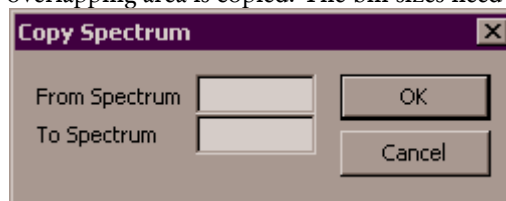
all

Spectrum number(s) to be cleared.

will clear all spectra.

10.3.3. CopySpectrum

Copies one spectrum to another one. The source and the destination spectra must have the same dimension. If the spectrum borders differ, only the overlapping area is copied. The bin-sizes need not to be the same.



Parameters:

Spectrum1,Spectrum2

Spectrum1

The source spectrum.

Spectrum2

The destination spectrum.

10.3.4. CutOffNevativeValues

This command sets all negative values of the displayed spectrum to zero.

Parameters:

Spectrum

Spectrum

The spectrum number.

Note:

Function will use the displayed spectrum if no spectrum number is given.

10.3.5. ExpandSpectrum

Expands either a 1- or a 2-dimensional spectrum using a cursor to determine the expansion area.

10.3.6. Remove

Remove special data from a spectrum.

Parameters:

FIT

FIT,Spectrum-Number(s)

MARKER,ALL

MARKER,InSpectrum, Marker-Number(s)

MARKER,InSpectrum,ALL

CALIBRATION,Spectrum-Number(s)

INTEGRATIONDATA

INTEGRATIONDATA, Spectrum-Number(s)

OVERLAY,ALL

OVERLAY,InSpectrum,ALL

OVERLAY,InSpectrum,Spectrum-Number(s)

CONDITION

CONDITION,Spectrum-Number(s)

UPDATE

InSpectrum

Spectrum-Number(s)

Marker-Number(s)

Function work in this spectrum.

spectrum number(s).

marker-number(s) that should be removed. **ALL** will remove all marker information of the selected spectrum. If no spectrum is selected the action is performed on the displayed spectrum.

Note:

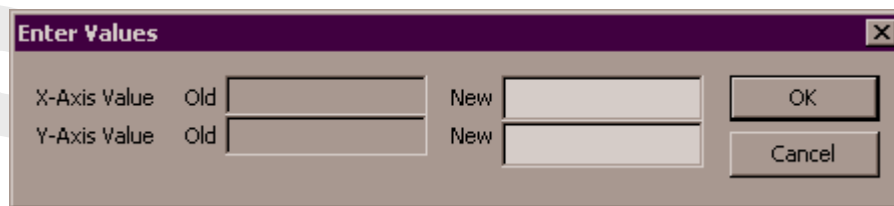
For **IntegrationData** the integration list is also updated!

For reactivate display of **CONDITION** information use **SHOW** command.

If no inspectrum is defined then the function will operate the displayed spectrum.

10.3.7. SetChannelToValue

Sets the specified channel to a given value.

**Parameters:**

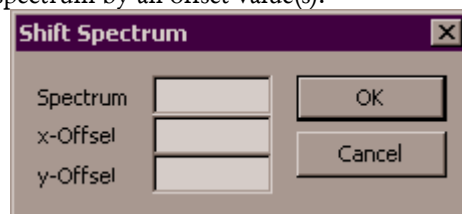
XChannel, Value
 XChannel, YChannel, value

XChannel
YChannel
Value

Channel of the x-axis.
 Channel of the y-axis (only for 2-dimensional spectra).
 New value.

10.3.8. ShiftSpectrum

This command shifts the specified spectrum by an offset value(s).

**Parameters:**

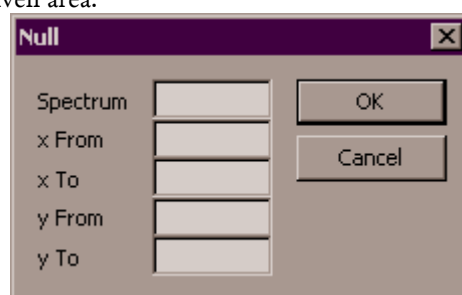
Spectrum, XOffset[, YOffset]

Spectrum
XOffset
YOffset

Spectrum number.
 Offset for the x-axis.
 Offset for the y-axis (only for 2-dimensional spectra).

10.3.9. ZeroSpectrum

Zeros a specified spectrum in the given area.

**Parameters:**


Spectrum, XMinimum, XMaximum[, YMinimum, YMaximum]

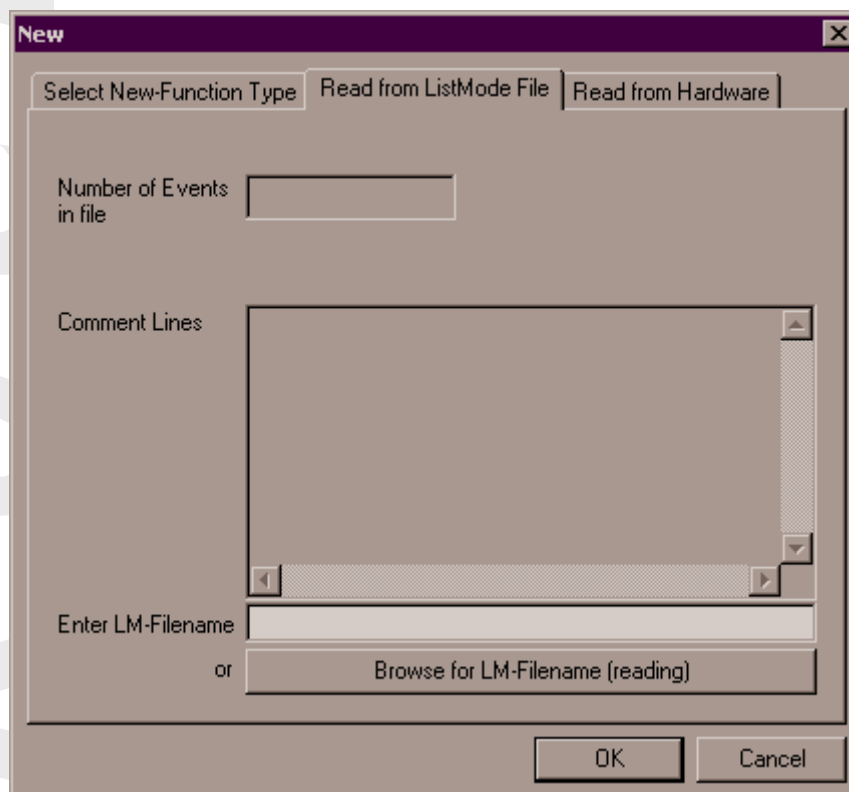
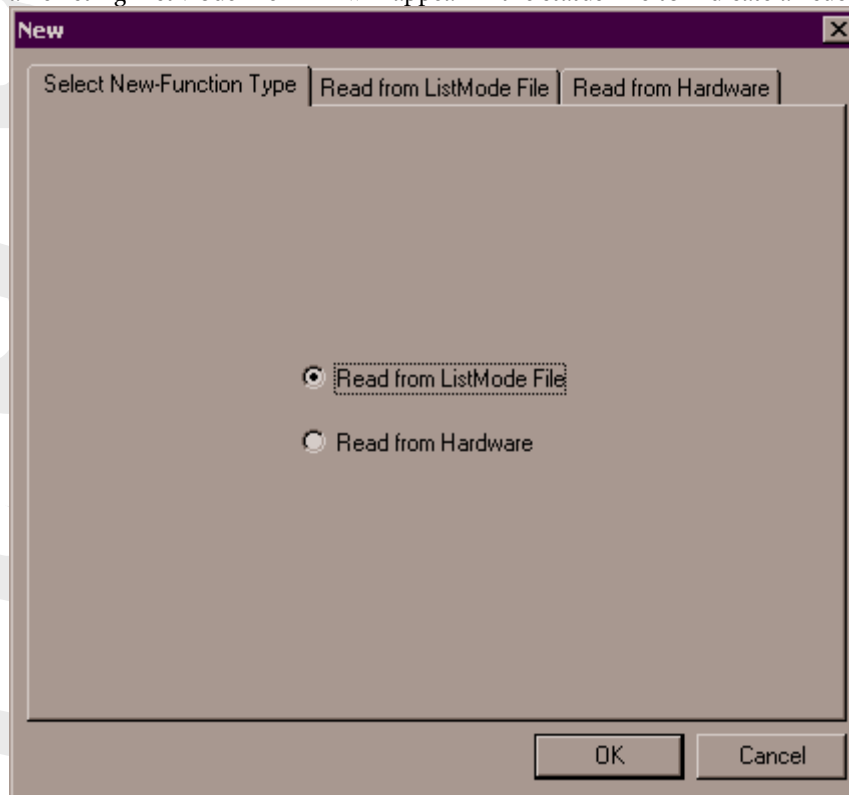
Spectrum
XMinimum, XMaximum
YMinimum, YMaximum

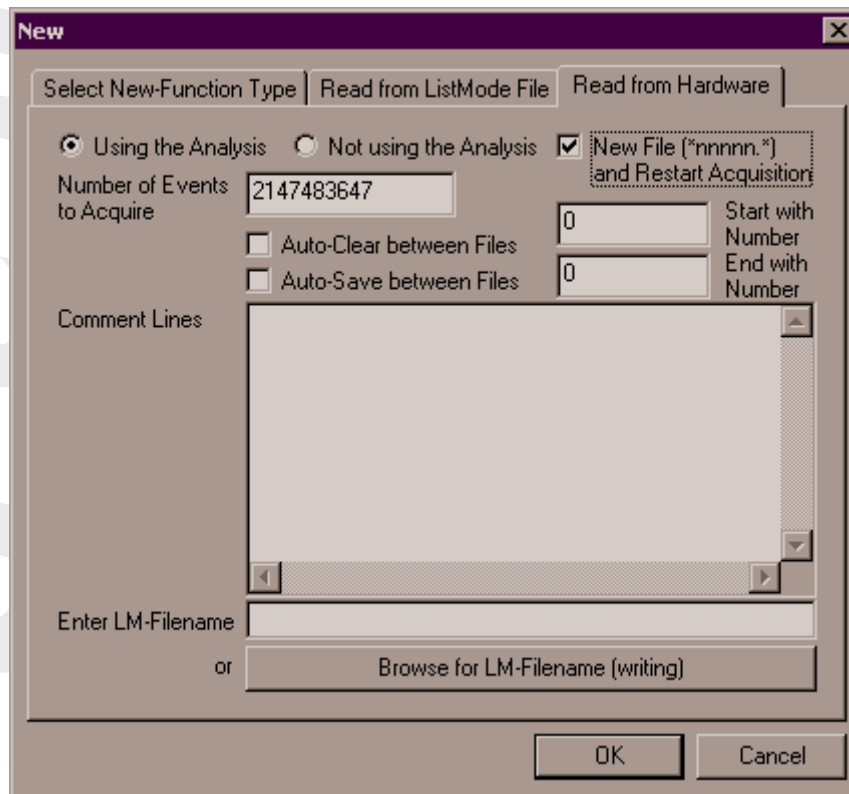
Spectrum number.
 Area in x-direction that will be zeroed.
 Area in y-direction that will be zeroed (only for 2-dimensional spectra).

10.4. Data acquisition commands

10.4.1. NewAcquisition

This command will prepare Cobold for taking data. The data source can be either hardware data acquisition or offline analysis of an existing ListModeFile.  will appear in the status line to indicate a successful execution.



**Parameters:**

FileName

FileName

Hardware,FileName,Analysis,# of events,Comment,Start,Stop,ClearFlag,SaveFlag

Hardware

FileName

Analysis

of events

Comment

Start

Stop

ClearFlag

SaveFlag

is the name of the ListModeFile that will be analyzed.

type **HARDWARE** to select the hardware as data source for the acquisition.

Name of the ListModeFile that will be written during acquisition. If no name is given no ListModeFile will be written.

Type **ANALYSIS** if you want to run through the user dependent part of the data analysis or **NOANALYSIS** if you don't.

The number of events you want to acquire.

Comment lines written into the header of the ListModeFile for easier identification of the data in the file. For line separator use **\n**.

If defined this is the start number for the autofile support.

If defined this is the stop number for the autofile support. (must be specified if **Start** is specified).

Yes to clear all spectra between loops in autofile mode, otherwise **No**

Yes to save the document between loops in autofile mode, otherwise **No**

This command is only valid if an acquisition has not been started.


Note:

AutoFile-Mode:

In this mode CoboldPC will loop several DAQ-cycles automatically. One DAQ-Loop is finished if the selected numbers of events are reached. A number that ranges from **Start** to **Stop** will be appended to the original filename. The extension **.Imf** is appended automatically. If you

specify **Yes** for the **Saveflag** than CoboldPC will also store the actual document under the same filename as the listmode-file except the extension is now **.dcf**.



10.4.2. PauseAcquisition

This command will pause the data acquisition. The acquisition can be restarted by the StartAcquisition command. This command is only valid if an acquisition has been started. After this command in the status line the  will appear on the status line of CoboldPC


10.4.3. RewindListModeFile

This command will set the file pointer to the first data in a ListModeFile. The command is only valid in PAUSE mode and the data source is a ListModeFile.

10.4.4. StartAcquisition

This command starts the data acquisition. If a ListModeFile was selected, the file will be opened and the header of the ListModeFile will be filled with special information (e.g. start time of the acquisition).  in the status line indicates a running offline dataanalysis (read from a listmode file) while  indicates a running online data acquisition (taking data from the hardware).

10.4.5. StopAcquisition

This command stops the data acquisition. If a ListModeFile was selected, the file is closed and the header of the ListModeFile is updated (stop time of the acquisition and number of events in this file). This command is only valid if an acquisition has been started. The status indicator will show .

10.4.6. Wait

Wait command to block inputs by mouse or keyboard.

Parameters:

Time
Counts

Time

Time dependent input for Wait (see examples)

Time format can be seconds or

day-month-year:hours:minutes:seconds for absolute time or
+day:hours:minutes:seconds.

Count dependent input for Wait (see examples)

Counts

Examples:

Wait 10s

Block input for 10 seconds

Wait 14-04-1999:8:15:00

Block input for till Clock reaches 14th April 1998 8:15am

Wait +1:20:14:06

Block input for the next day 20hours 14minutes and 6seconds.

Wait

Blocks input until DAQ read status is stopped (automatically at EOF).



Wait 100000

Blocks input until 100000 counts are processed (EventCounter = = 100000) or EOF is reached.

Wait +100000

Blocks input until the next 100000 counts are processed or EOF is reached.

Note:

Count input or no input for the wait command is only valid if DAQ status is running ( or  indicator in status bar).

10.5. File handling commands

10.5.1. Restart command

Use this command to restart with a new document in Cobold.

You can open an existing document with the Open command.

Shortcuts

Toolbar: 
Keys: CTRL+R

10.5.2. Open command

Use this command to open an existing PC or ATARI document. If the filename extension is *.dmp* CoboldPC tries to load the old ATARI Dump file. In this case the dump file version must be 2.3 otherwise load fails. For a CoboldPC document file the version number must be newer than 4.1 otherwise load fails.

You can create new documents with the Restart command.

Shortcuts

Toolbar: 
Keys: CTRL+O

10.5.3. ExecuteCommandFile

Execute a command-file. This file can contain a summary of often used commands. A command file can call another command file.

Parameters:

CommandFileName

CommandFileName The name of the command file including drive and path.

Shortcuts

Toolbar: 

10.5.4. Save command

Use this command to save the active document to its current name and directory. When you save a document for the first time, Cobold displays the Save As dialog box so you can name your document. If you want to change the name and directory of an existing document before you save it, choose the Save As command.

Shortcuts

Toolbar: 
Keys: CTRL+S

10.5.5. Save As command

Use this command to save and name the active document. Cobold displays the Save As dialog box so you can name your document.

To save a document with its existing name and directory, use the Save command.

10.5.6. Print

Use this command to print a document. This command presents a Print dialog box, where you may specify the range of pages to be printed, the number of copies, the destination printer, and other printer setup options.

Shortcuts

Toolbar: 
Keys: CTRL+P

10.5.7. PrintPreview

Use this command to display the active document as it would appear when printed. When you choose this command, the main window will be replaced with a print preview window in which one or two pages will be displayed in their printed format. The print preview toolbar offers you options to view either one or two pages at a time; move back and forth through the document; zoom in and out of pages; and initiate a print job.

10.5.8. PrintSetup

Use this command to select a printer and a printer connection. This command presents a Print Setup dialog box, where you specify the printer and its connection.

10.5.9. Preferences

Use this command to define global Cobold settings such as print and display colors.

10.5.10. Exit

Use this command to end your Cobold session. You can also use the Close command on the application Control menu. Cobold prompts you to save documents with unsaved changes.

Shortcuts

Mouse: Double-click the application's Control menu button.

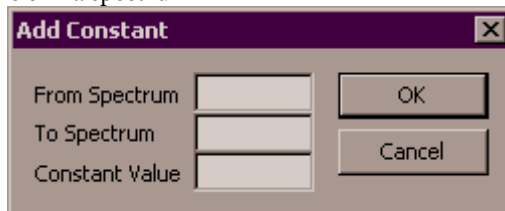


Keys: ALT+F4 or CTRL+Q

10.6. Mathematical commands

10.6.1. AddConstant

Adds a constant value to all channels in a spectrum.



Parameters:

Spectrum1, Value, Spectrum2

Spectrum1

Source spectrum.

Value

constant value (floating point value).

Spectrum2

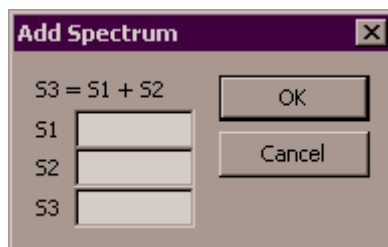
Destination spectrum.

Note:

$Spectrum2 = Spectrum1 + Value$

10.6.2. AddSpectrum

Adds the contents of two spectra, stores result in third. The two spectra must have the same dimensional definition but need not have the same boundaries. If boundaries are not equal, only the overlapping region will be operated on.



Parameters:

Spectrum1, Spectrum2, Spectrum3

Spectrum1

Source spectrum 1.

Spectrum2

Source spectrum 2.

Spectrum3

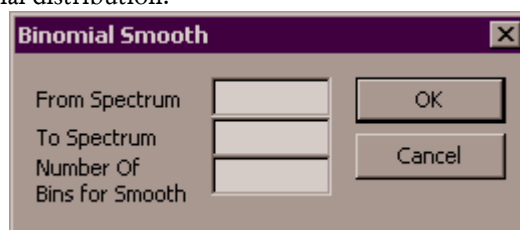
Destination spectrum.

Note:

$Spectrum3 = Spectrum1 + Spectrum2$

10.6.3. BinomialSmooth

Smooth spectrum using a binomial distribution.



Parameters:

Spectrum1,N,Spectrum2

Spectrum1

Source spectrum.

N

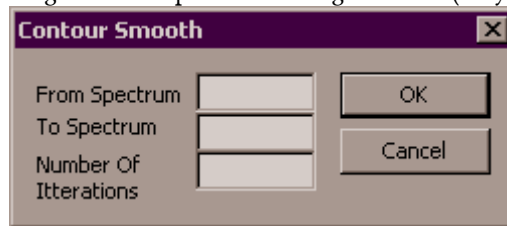
Number of bins for smooth. (uneven integer number ≥ 3)

Spectrum2

Destination spectrum.

10.6.4. ContourSmooth

Smooth 2-dimensional spectrum using a contour plot smoothing function (only for 2-dimensional spectra).

**Parameters:**

Spectrum1,N,Spectrum2

Spectrum1

Source spectrum.

N

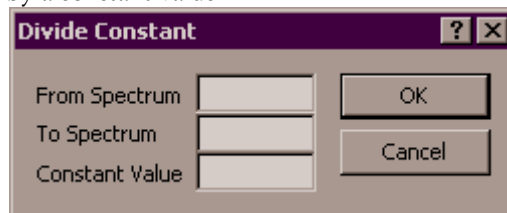
Number of iterations.

Spectrum2

Destination spectrum.

10.6.5. DivideConstant

Divide all channels in a spectrum by a constant value.

**Parameters:**

Spectrum1,Value,Spectrum2

Spectrum1

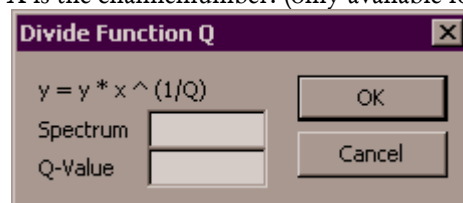
Source spectrum.

Value

constant value (floating point value).

Spectrum2

Destination spectrum.

Note: $Spectrum2 = Spectrum1 / Value$ **10.6.6. DivideFunctionQ**Multiplies all channels by $x^{(1/Q)}$. X is the channelnumber. (only available for 1-dimensional spectra).**Parameters:**

Spectrum1,Spectrum2,Q

*Spectrum1**Spectrum2*

Q

Source spectrum.

Destination spectrum.

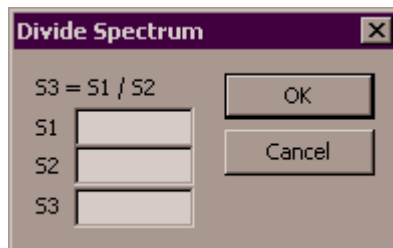
Q-Value (floating point value).

Note:

$$\text{Spectrum2} = \text{Spectrum1} * x^{(1/Q)}$$

10.6.7. DivideSpectrum

Divides the contents of two spectra, stores result in third. The two spectra must have the same dimensional definition but need not have the same boundaries. If boundaries are not equal, only the overlapping region will be operated on.

**Parameters:**

Spectrum1,Spectrum2,Spectrum3

*Spectrum1**Spectrum2**Spectrum3*

Source spectrum 1.

Source spectrum 2.

Destination spectrum.

Note:

$$\text{Spectrum3} = \text{Spectrum1} / \text{Spectrum2}$$

10.6.8. FitSpectrum

Fitting spectrum data with the given function. The boundaries are inclusive.

Parameters:

Type,XMinimum,Xmaximum

Type,Order,Xminimum,XMaximum

Type

Fit function type.

Linear Regression for these function types

**LINEARREGRESSION, LOGARITHMREGRESSION,
POWERREGRESSION, EXPONENTIALREGRESSION,
GAUSS, POLYNOM**

XMinimum,XMaximum

Boundaries for x-axis

Order

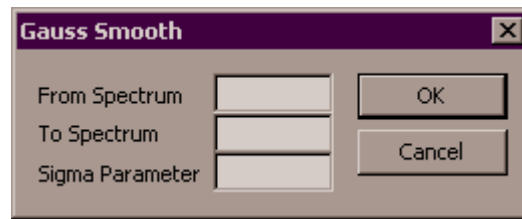
Order of the polynom ($2 \leq \text{order} \leq 9$)**Note:**

For cursor input of the boundaries do not enter data for the boundaries fields.

For gauss fits the peak should lie inside the boundaries to get accurate and correct fit results.

10.6.9. GaussSmooth

Smooth a spectrum using a gauss distribution.

**Parameters:**

Spectrum1, Sigma, Spectrum2

Spectrum1

Source spectrum.

Sigma

Sigma value in the gaussian formula.

Spectrum2

Destination spectrum.

10.6.10. GP1Correct

The function corrects the differential nonlinearity of the GP1-TDC by multiplying the counts of every second channel by a given number. After multiplication is done the spectrum is normalized so the total counts of the spectrum will be the same as before this operation.

Parameters:

CorrX[, CorrY]

CorrX

Correction value for x axis

CorrY

Correction value for y axis (in 2 dimension spectrum only)

Note:

To automatically adjust the values vor CorrX and CorrY do not specify these values.

10.6.11. IntegrateSpectrum

Integrates a defined area in a one- or two-dimensional spectrum. The integration data is stored temporary in the spectrum but it is not saved with the dump file. A new integration will override the previous integration data if it is present. Each spectrum can hold one integration data! To remove the information use the remove function. The boundaries are inclusive.

Parameters:

BFlag

XMinimum, XMaximum[, BFlag]

XMinimum, XMaximum, YMinimum, YMaximum[, BFlag]

XMinimum, XMaximum

Boundaries for x-axis

YMinimum, YMaximum

Boundaries for y-axis in a two-dimensional spectrum

BFlag

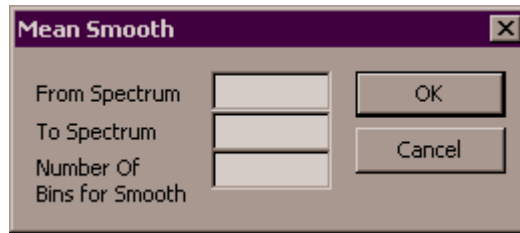
TRUE includes all boundaries for integration**FALSE** will include only left (and in 2-dim spectrum lower) boundaries into integration**Note:**

For cursor input of the boundaries do not enter data for the boundaries fields.

If no BFlag is given it is set automatically to **TRUE**.

10.6.12. MeanSmooth

Smooth a spectrum using a mean value smoothing function.

**Parameters:**

Spectrum1,N,Spectrum2

Spectrum1

N

Spectrum2

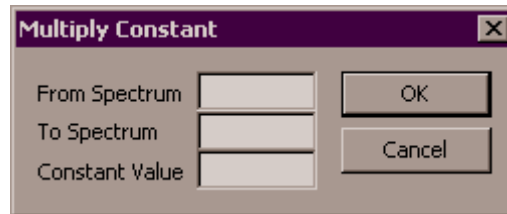
Source spectrum.

Number of bins for smooth. (uneven integer number ≥ 3)

Destination spectrum.

10.6.13. MultiplyConstant

Multiplies all channels by a constant value.

**Parameters:**

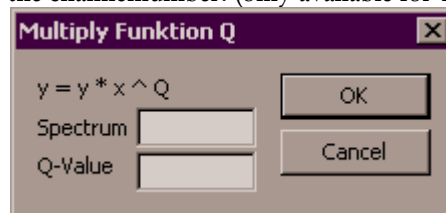
Spectrum1,Value,Spectrum2

*Spectrum1**Value**Spectrum2*

Source spectrum.

constant value (floating point value).

Destination spectrum.

Note: $Spectrum2 = Spectrum1 * Value$ **10.6.14. MultiplyFunctionQ**Multiplies all channels by x^Q . X is the channelnumber. (only available for 1-dimensional spectra).**Parameters:**

Spectrum1,Spectrum2,Q

*Spectrum1**Spectrum2*

Q

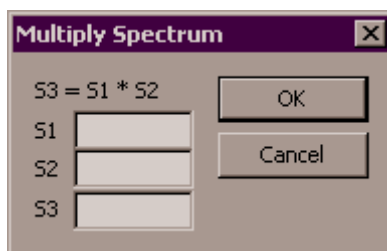
Source spectrum.

Destination spectrum.

Q-Value (floating point value).

Note: $Spectrum2 = Spectrum1 * x^Q$ **10.6.15. MultiplySpectrum**

Multiplies the contents of two spectra, stores result in third. The two spectra have to have the same dimensional definition but need not to have the same boundaries. If boundaries are not equal, only the overlapping region will be operated on.



Parameters:

Spectrum1,Spectrum2,Spectrum3

Spectrum1

Source spectrum 1.

Spectrum2

Source spectrum 2.

Spectrum3

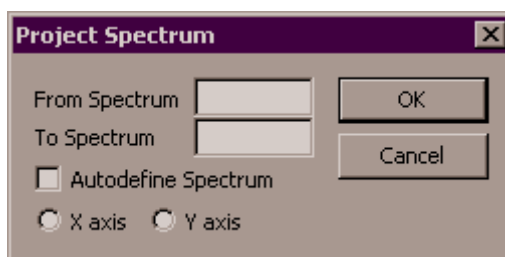
Destination spectrum.

Note:

spectrum3 = spectrum1 * spectrum2

10.6.16. ProjectSpectrum

The function projects a two-dimensional spectrum on x or y axis in a given range to a one-dimensional spectrum. The boundaries are inclusive.



Parameters:

Spectrum1,Spectrum2,Axis,From,To

Spectrum1

Source spectrum.

Spectrum2

Destination spectrum or

Axis

auto to let CoboldPC automatically define the spectrum x or y to specify the axis to project to.

From

Lower boundary.

To

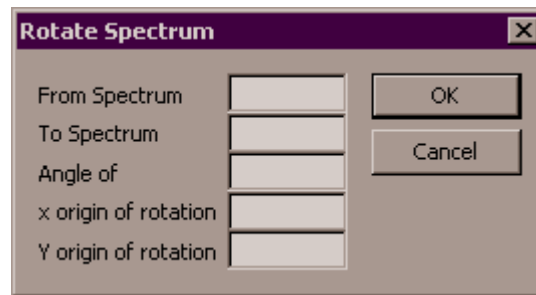
Upper boundary.

Note:

For cursor input of the boundaries do not enter data for the boundaries fields.
For Dialog-Box input do not enter parameters for the function.

10.6.17. RotateSpectrum

The function rotates a two dimensional spectrum by a given angle. The origin of rotation can be specified.

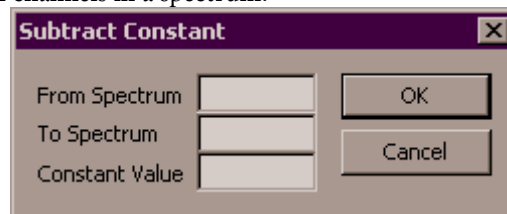
**Parameters:**

From, Angle, XOrigin, YOrigin, To

From	Source spectrum.
Angle	Angle of rotation.
XOrigin	X value of origin of the rotation point.
YOrigin	Y value of origin of the rotation point.
To	Destination spectrum.

10.6.18. SubtractConstant

Subtract a constant value from all channels in a spectrum.

**Parameters:**

Spectrum1, Value, Spectrum2

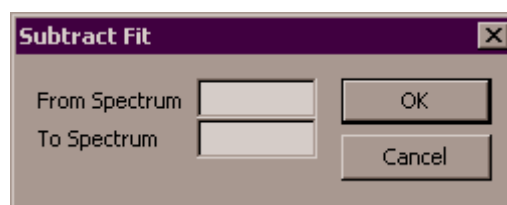
<i>Spectrum1</i>	Source spectrum.
<i>Value</i>	constant value (floating point value).
<i>Spectrum2</i>	Destination spectrum.

Note:

$Spectrum2 = Spectrum1 - Value$

10.6.19. SubtractFit

Subtract the last fit from the given spectrum.

**Parameters:**

[From, To]

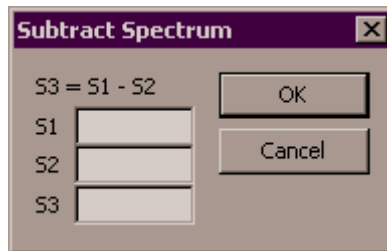
<i>From</i>	Source spectrum.
<i>To</i>	Destination spectrum.

Note:

If no **From** and **To** is given this command will use the displayed spectrum for **From** and **To**.
If **To** is not given, **From** will be set to **To** value.

10.6.20. SubtractSpectrum

Subtract the contents of two spectra, store results in third spectrum. The two spectra must have the same dimensional definition but need not have the same boundaries. If boundaries are not equal, only the overlapping region will be operated on.



Parameters:

Spectrum1,Spectrum2,Spectrum3

Spectrum1

Source spectrum 1.

Spectrum2

Source spectrum 2.

Spectrum3

Destination spectrum.

Note:

Spectrum3 = Spectrum1 - Spectrum2

A. Appendix - List Mode Data Format

The List-Mode data file consists of 2 sections. The first section is the data-header. The second is the data-area. The main header is constructed as followed.

```

unsigned int    iArchiveFlag = true;
unsigned int    LMDataFormat;
unsigned int    LMNumberOfCoordinates;
unsigned int    LMHeaderSize;
unsigned int    LMUserDefinedHeaderSize;
unsigned int    LMNumberOfEvents;
CTime          LMStartTime;
CTime          LMStopTime;
CString        LMVersionString;
CString        LMFilePathName;
CString        LMComment;

```

Unsigned int is a 32bit integer Number.

CTime and CString are classes of Microsoft Foundation Class 4.x of the Visual C++ Compiler of Microsoft.

LMDataFormat can have the following values:

```

// definitions for main program event data format
#define LM_BYTE          1      // 8bit
#define LM_SHORT        2      // 16bit
#define LM_LONG         3      // 32bit
#define LM_FLOAT        4      // 32bit
#define LM_DOUBLE       5      // 64bit
#define LM_CAMAC        6      // 24bit
#define LM_USERDEF      -1     // user will handle the
                               // reading
                               // of file and filling of
                               // eventData Array
                               // except reading the main
                               // header file

```

If the LMUserDefinedHeaderSize is not equal to zero then a user defined header will follow. The size of this datablock is stored in the LMUserDefineHeaderSize variable of the main header block.

After this header block information the evnet data follows as described in DAQ.DLL source code.

B. Appendix - Table of Figures

ROE n t d e k

Figure 1: Schematic of a Listmode-File7
Figure 2: Flowchart of a CoboldPC session8
Figure 3: Start CoboldPC program from Start-button under Program-CoboldPC group..... 10
Figure 4: Startup window of CoboldPC..... 11

ROENTGEN

Index

ROE NT d e k

A			
AddConstant.....	54	Mode.....	41
AddSpectrum.....	54	Mode2D.....	41
B		MultiplyConstant.....	58
BinomialSmooth.....	54	MultiplyFunctionQ.....	58
C		MultiplySpectrum.....	58
CalibrateSpectrum.....	45	N	
ClearSpectrum.....	45	NewAcquisition.....	48
Condition.....	37	NextSpectrum.....	43
ContourSmooth.....	55	NoDisplay.....	43
Coordinate.....	37	O	
CopySpectrum.....	45	Open command.....	52
Cursor.....	43	P	
CutOffNevativeValues.....	46	Parameter.....	41
D		PauseAcquisition.....	50
Define1DimensionalSpectrum.....	38	PreviousSpectrum.....	43
Define2DimensionalSpectrum.....	38	ProjectSpectrum.....	59
Delete.....	40	R	
DivideConstant.....	55	Remove.....	46
DivideSpectrum.....	56	Restart command.....	52
E		RewindListModeFile.....	50
ExecuteCommandFile.....	52	S	
ExpandSpectrum.....	46	Save As command.....	52
F		Save command.....	52
FitSpectrum.....	56	SetChannelToValue.....	46
Format.....	62	ShiftSpectrum.....	47
G		Show.....	43
GaussSmooth.....	56	StartAcquisition.....	50
Grid.....	40	StopAcquisition.....	50
I		SubtractConstant.....	60
IntegrateSpectrum.....	57	SubtractSpectrum.....	61
L		U	
List-Mode.....	63	UpdateSpectrum.....	44
M		V	
Marker.....	41	ViewSpectrum.....	44
MeanSmooth.....	57	Z	
		ZeroSpectrum.....	47