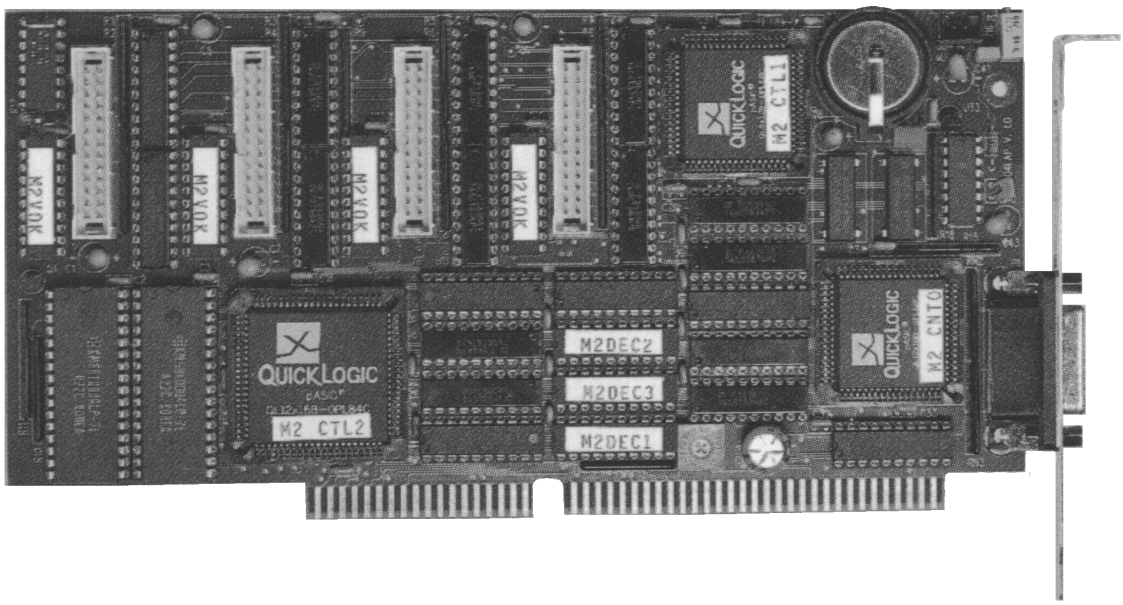
MCD4LAP

4 Input Multichannel Data Processor

User Manual

© copyright FAST ComTec GmbH
Grünwalder Weg 28a, D-82041 Oberhaching
Germany



Version 2.8, October 11, 2004

Software Warranty

FAST ComTec warrants proper operation of this software only when used with software and hardware supplied by FAST ComTec. FAST ComTec assumes no responsibility for modifications made to this software by third parties, or for the use or reliability of this software if used with hardware or software not supplied by FAST ComTec. FAST ComTec makes no other warranty, expressed or implied, as to the merchantability or fitness for an intended purpose of this software.

Software License

You have purchased the license to use this software, not the software itself. Since title to this software remains with FAST ComTec , you may not sell or transfer this software. This license allows you to use this software on only one compatible computer at a time. You must get FAST ComTec's written permission for any exception to this license.

Backup Copy

This software is protected by German Copyright Law and by International Copyright Treaties. You have FAST ComTec's express permission to make one archival copy of this software for backup protection. You may not otherwise copy this software or any part of it for any other purpose.

Copyright © 1994 - 2004 FAST ComTec Communication Technology GmbH,
D-82041 Oberhaching, Germany. All rights reserved.

This manual contains proprietary information; no part of it may be reproduced by any means without prior written permission of FAST ComTec, Grünwalder Weg 28a, D-82041 Oberhaching, Germany. Tel: +49 89 66518050, FAX: +49 89 66518040, http://www.fastcomtec.com

The information in this manual describes the hardware and the software as accurately as possible, but is subject to change without notice.

Table of Contents

Table of Figures

1. Introduction

The MCD4LAP is a complete highly sophisticated data acquisition system for IBM or compatible personal computers. It is based on a 2/3 length low profile PC-AT-card and therefore also ideal for most Laptop computers. For the use in battery powered systems the MCD4LAP is designed to consume very little energy - no HDD operations are required during data acquisition thus making the MCD4LAP ideally suited for operation in remote places. The high performance of the hardware is matched by the sophisticated operating and analysis software delivered with each system.

The MCD4LAP card not only provides four standard ADC input ports to the fast Multichannel Data Processor (MCD) but also two fast scalers/counters, an analog output and an 8 bit digital I/O port. The MCD section has a large 4 x 16k x 32 bit fast memory .The hardware data processing provides high count rates and low data transfer time. Each of the four ADC inputs has its own live and true time storage with 5 ms resolution. Every timer is presettable to stop data acquisition of the corresponding or all input channels. In case of several MCD4LAP cards being operated in one PC - all cards can be stopped synchronously. Each of the two 32 bit scalers/counters provides a 50 MHz maximum count rate capability. One of the scalers can be preset to stop data acquisition when the preset number of counts is reached. The scalers can be started and stopped synchronously to the ADC inputs. The second scaler can also be externally controlled or gated.

The battery backup of the MCD4LAP memory prevents the loss of data even in power down conditions. This is especially important in long time experiments or when the experimentation time is limited. With the MCD4LAP the user can continue his measurement after the power resumes without loosing any data.

The on-board Digital to Analog Converter (DAC) provides software control of external devices (e.g. High Voltage supplies) without requiring further hardware.

The on-board 8 bit Digital I/O port can be used to remote control data acquisition or to control external devices like sample changers, alerts etc.

In addition the highly sophisticated operating software allows the user to easily control all the features of the MCD4LAP data acquisition system. Since all the functions are fully under software control no jumpers or switches have to be set. This means once the MCD4LAP card is installed in your computer no further hardware changes are necessary. The program offers many features such as parameter setup, energy calibration, efficiency calibration, isotope library, nuclide library manager, peak search, spectrum analysis, calculation of activities, activity report, neutron activation analysis, NAA library manager, display of high resolution gamma spectra etc.

The applications range from nuclear-, alpha- and x-ray spectroscopy to portable laptop spectrum analysis, on-site data analysis, experiments that required analog mixer-routers, OEM applications, remote data acquisition etc.

2. Installation Procedure

2.1. Hard- and Software Requirements

The MCD4LAP requires an IBM AT or compatible computer with a 286, 386, 486 or higher processor and an available 16 bit slot. A maximum of 16 MCDLAP cards can be installed in your computer if you have enough available slots.

2.2. Hardware Installation

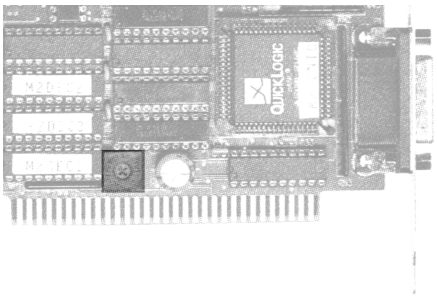


Figure 2.1: Address select rotary switch

First you have to find an unused address in the I/O address space of your computer. The MCD4LAP has a small rotary switch (Figure 2.1) that determines the base I/O address of the card. The MCD4LAP occupies 9 addresses starting at this base address. The supported addresses and corresponding switch settings are:

Switch	Base Address [hex]	Switch	Base Address [hex]
0	200	8	300
1	210	9	310
2	220	A	320
3	230	B	330
4	240	C	340
5	250	D	350
6	260	E	360
7	270	F	370

Figure 2.2: Table of the base addresses

The factory setting is 320hex - an address commonly not used by other devices.

2.3. Software Installation

To install the MCD4LAP software on your hard disk insert the MCD4LAP disk into drive A. Log to drive A: by typing from a DOS box or searching with the explorer

```
C:> A: <RETURN>
```

and start the installation batch file by typing (or double clicking)

```
A:> SETUP <RETURN>
```

A directory called C:\MCD4LAP is created on the hard disk and all MCD4LAP and MCDWIN files are transferred to this directory. Drive C: is taken as default drive and \MCD4LAP as default directory. It is not mandatory that the MCD4LAP operating software is located in this directory. You may specify another directory during the setup.

When using Windows NT 4.0, the driver from the NTDRIVER subdirectory on drive A can be installed by hand if not already done as follows, then Windows must be restarted:

```
A:> CD NTDRIVER <RETURN>
A:NTDRIVER> INSTALL <RETURN>
```

The Setup program has installed one shortcut on the desktop that starts the MCD4LAP server program. The server program will automatically call the MCDWIN.EXE program when it is executed. The MCD4LAP Server program controls the MCD4LAP board but provides no graphics display capability by itself. By using the MCDWIN program, the user has complete control of the MCD4LAP along with the MCDWIN display capabilities.

To run the MCD4LAP software, simply double click on the “ MCD4LAP“ icon. To close it, close the MCD4LAP server in the Taskbar.

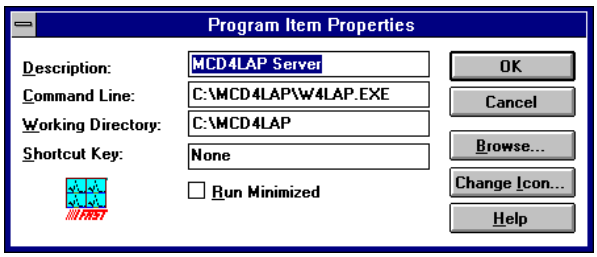


Figure 2.3: WINDOWS program item properties

3. Hardware Description

3.1. Overview

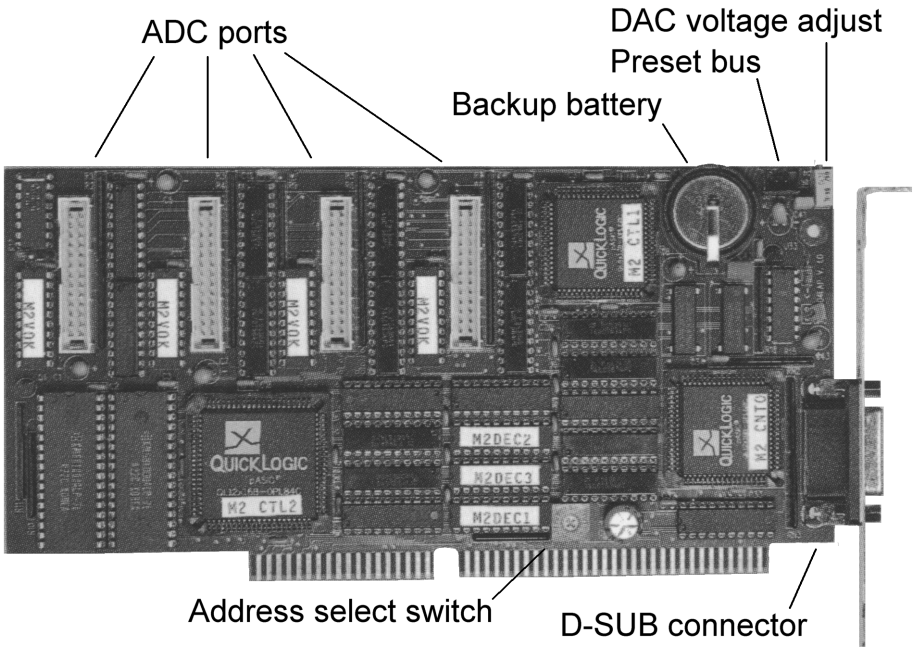


Figure 3.1: MCD4LAP PC card

The MCD4LAP PC card combines a four input Multichannel Data Processor (MCD) with 2 fast scalers, 8 digital I/O ports and an analog output. It also provides a backup battery for the 4 x 16k x 32bit MCD memory. The MCD4LAP supports full background operation thus freeing the computer for other work. All parameter setups are software controlled.

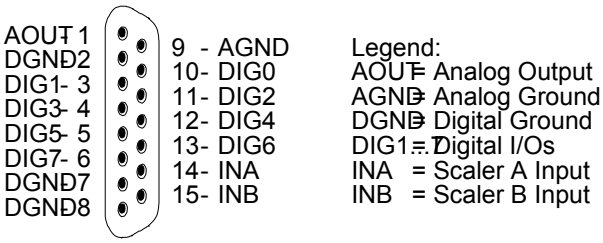


Figure 3.2: 15 pin female D-SUB connector

3.2. ADC Input Ports

Four standard ADC input ports allow operation with most pulse height analyzing ADCs. Also other devices like TOFs, Multiscalers, transient recorders etc. with a Data Ready / Data Accepted handshake interface can be connected.

The data bits (D0...D13) are active low each. The polarities of all other signals are software selectable.

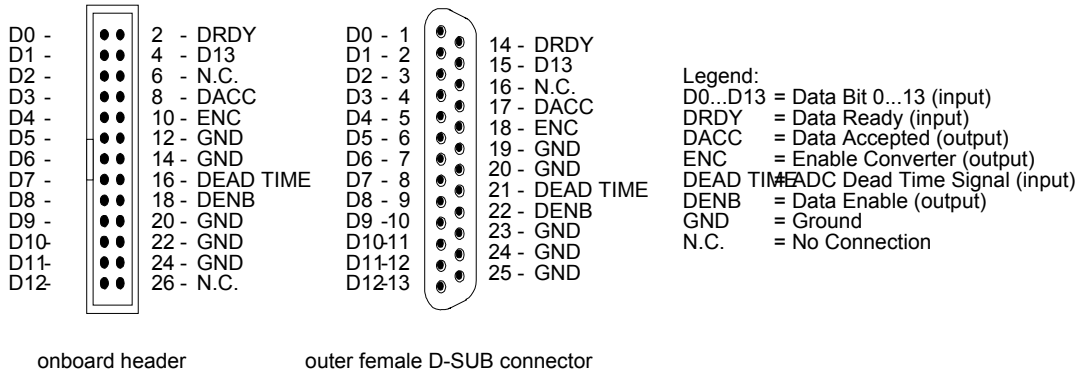


Figure 3.3: ADC port connectors

The ports are brought out of the PC through the PC card bracket via 4 ribbon cables. These have 25 pin female D-SUB connectors at the outer ends.

3.3. Scaler A

A 50 MHz maximum count rate 32 bit scaler / counter. It is synchronously started and stopped with ADC port 1. It is presettable to any value and allows for stopping all MCD4LAP data acquisition on overflow. The input is at the 15 pin D-SUB connector on the rear bracket.

3.4. Scaler B

A 50 MHz maximum count rate 32 bit scaler / counter. It can be synchronously started and stopped with ADC port 1, ADC port 2, digital input line DIG 7 or under software control. The input is at the 15 pin D-SUB connector on the rear bracket.

3.5. Analog Output

An 8 bit DAC provides an analog output voltage. The output voltage range is 0 ... +2.55V when the jumper is „ON“. With the jumper removed the maximum voltage is adjustable from +2.55 ... ≈+7V by a precision trimmer. The DAC output is accessible on the 15 pin D-SUB connector.

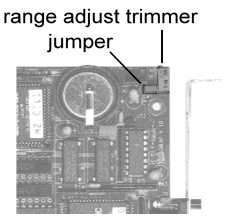


Figure 3.4: DAC range control

3.6. Backup Battery

A 3V Lithium battery provides multichannel data retention even on PC power failure.

Remark: The scaler contents are not backed up.

3.7. Digital I/O Ports

8 open drain digital I/O lines (DIG0...DIG7) at the 15 pin D-SUB connector. They have 4.7kΩ pull-ups and 100Ω series resistors. They are fully user accessible. DIG7 can also be used to control scaler B.

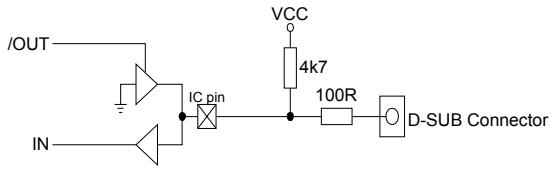


Figure 3.5: Digital I/O pins

3.8. Live and True Timer

Each ADC port has a live and true time storage with 5 ms resolution. The data is stored in the channels 0 and 1 of the corresponding MCD memory section. On overflow a *clear* either of the corresponding or of all ENCs can be generated. Thus timer presets either of live or true time are possible.

3.9. Preset Bus

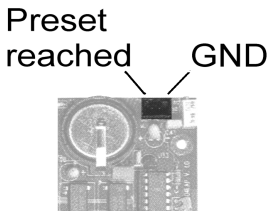


Figure 3.6: Preset bus header

The preset bus header provides a means to stop several MCD4LAP cards synchronously when any preset is reached.

All bus drivers are open drain outputs and wired-ANDed. If enabled any Enable Converter (ENC) of the MCD4LAP is removed when a „Low“ appears on the preset bus. Also scaler A has an output to this bus.

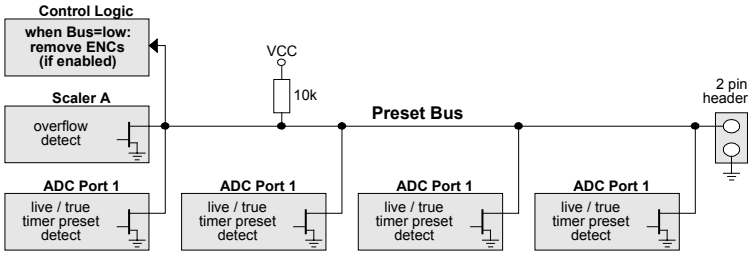


Figure 3.7: Preset bus logic

4. MCD4LAP Windows Server Program

The window of the MCD4LAP server program W4LAP.EXE is shown here. It allows to fully control the MCD4LAP, perform measurements and save data. This program has no own graphics, but it provides via a DLL („dynamic link library“) access to all functions, parameters and data. The server can be completely controlled from the MCDWIN software that provides all necessary graphic displays.

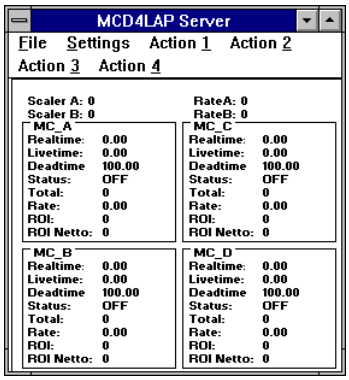


Figure 4.1: Status display of the MCD4LAP server program

At program start the configuration files W4LAP.INI (contains for example the port base address in a format base=320 and the ADC port handshake signal polarities; see Figure 4.2) and W4LAPA.INF, W4LAPB.INF, W4LAPC.INF and W4LAPD.INF are read. Instead of these four .INF files any other set of setup files can be used if its name without appendix ‘A.INF’ is used as command line parameter. The server program is normally shown as an icon. After a double click it is opened to show a status window.

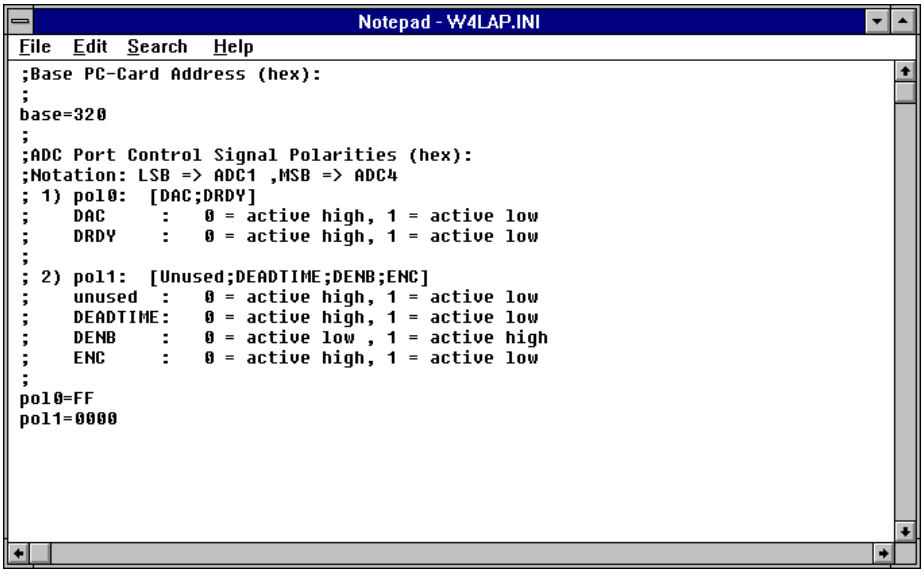


Figure 4.2: Sample W4LAP.INI file

In the following the several dialogs are described in detail:

Clicking in the File menu on the Data... item opens the Data Operations dialog box.

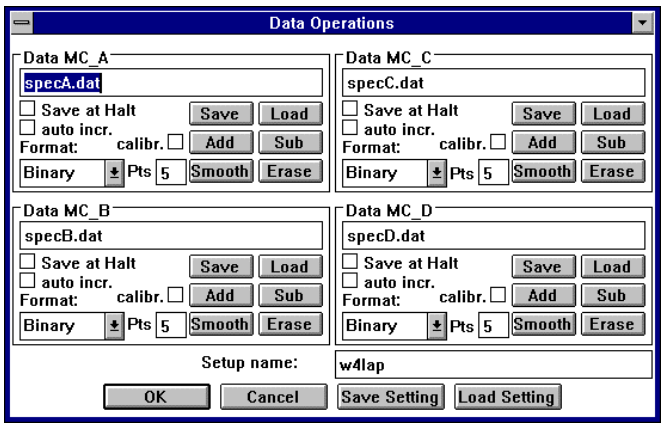


Figure 4.3: Data Operations dialog box

This dialog allows to edit the data settings of the four MCA's. Mark the checkbox „**Save at Halt**" to write a spectrum- and a configuration file at the end of a measurement. The filename can be entered. If the checkbox **auto incr.** is crossed, a 3-digit number is appended to the filename that is automatically incremented with each saving. The format of the datafile can be ASCII or binary (extension .ASC or .DAT). The buttons **Save**, **Load**, and **Erase** in each of the four sections perform the respective operation for each MCA. With **Add** and **Sub** a spectrum can be added or subtracted from the present data. The Checkbox **calibr.** is checked if a calibration is used and the data is then shifted according to the calibration. The **Smooth** button performs an n-point smoothing of the data. The number of points to average can be set with the **Pts** edit field between 3 and 21. The Data dialog of the optional 2*MCD4LAP+ server (W9LAP.EXE) in addition allows to erase or calculate the sum spectrum.

Clicking in the Settings menu on the MCD... item opens the MCD4LAP Settings dialog box. Here the presets and range parameters for the MCA's and Scaler A are set.

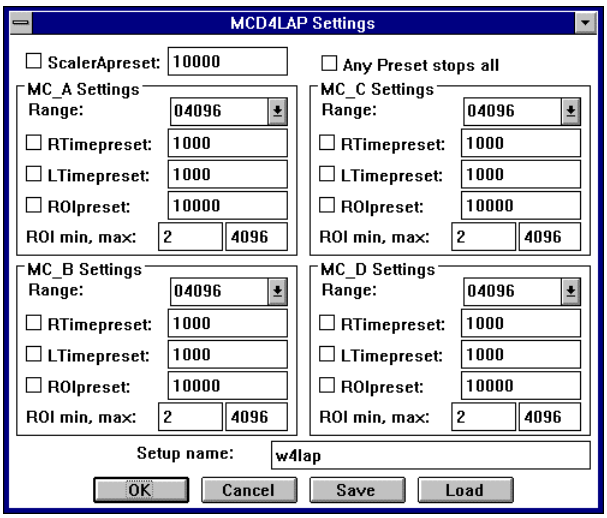


Figure 4.4: MCD4LAP Settings dialog box

In the edit field „**Range**" the size of the spectrum can be chosen between 256 and 16384. If the checkbox „**ROIpreset**" is marked, the measurement will be stopped after the events specified in the corresponding edit field have been reached. The events are counted only if they are within the „**ROI**" limits, i.e. >= the lower limit and < the upper limit. Another possibility is to acquire data for a given real time via the „**Rtimepreset**" or a given live time via the „**Ltimepreset**". A preset value for the presettable scaler A can be entered in the edit field „**ScalerApreset**". A measurement will be stopped if the corresponding checkbox is marked. An enabled Scaler preset automatically enables the checkbox „**Any preset stops all**". It is optional to set this flag in addtition to any of the other presets. In this case any preset reached will stop every active measurement.

The „System..." item in the settings menu opens the System Definition dialog box. Here the several MCA's and Scalers can be combined to form one or up to four separate systems that can

be started, stopped and erased by one command. In addition the use of the digital input / output and the analog output ports can be defined.

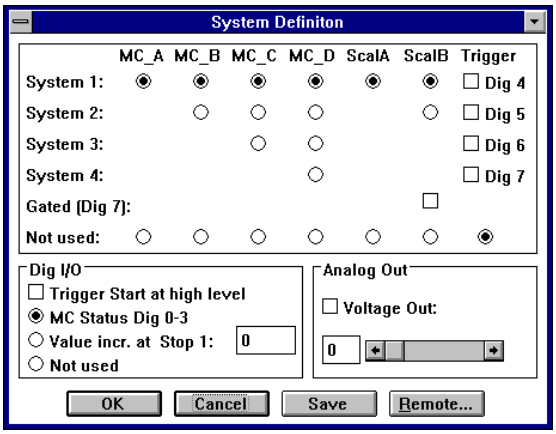


Figure 4.5: System Definition dialog box

In the shown setting two systems a defined. In system 1 the two multichannel analyzers MC_A and MC_C and Scaler A are combined, and in system 2 MC_B, MC_D and Scaler B. The system 1 can be started, stopped, erased, and continued with the respective commands in the Action 1 menu, and system 2 in the Action 2 menu.

The digital input / output port can be used either to show the status of the four MCA's in bits 0...3 (bit 0 set means MC_A ON and so on) if the DIG I/O radio button „**4 Status bits**" is checked. It can also be used for example with a sample changer by checking „**Value inc. at Stop 1**". Here, the 8 bit value entered in the edit field (a number between 0 and 255) is output at the Dig I/O port. This value will always be incremented by one if MC_A is stopped.

Together with using bits 0..3 as status signals for MC_A to MC_D, it is also possible to use the digital inputs 4-7 as an external trigger for starting the systems 1-4 (**„DESY control line"**). If the corresponding checkbox is marked, a start command for the respective system will not immediately start the system. After the start command, the digital input will be permanently checked for its logical level. If the level changes from high to low, the data for the system are cleared and it will immediately be started. It will stop if the level returns to high (or vice versa if „Trigger Start at high level" is marked). A stop command for the system will finish the digital input checking.

The DAC output can be enabled by checking the box labeled „**Voltage Out**". The voltage can be defined in 256 steps by setting the value in the edit field to a number between 0 and 255 or the corresponding scrollbar can be used.

OK accepts all settings. **Cancel** cancels all changes. Clicking „**Save Settings**" saves all settings in the files **W4LAPA.INF ... W4LAPC.INF** in a form:

```
wndwidth=36
wndheight=36
sysdef=1651
range=4096
dacena=0
outlevel=0
orpreset=0
scalprena=0
scalpreset=10000
digstsena=1
digsmplena=0
digsmplval=0
rtpreset=1000
rtprena=0
ltpreset=1000
ltprena=0
roipreset=10000
roiprena=0
```

```
roimin=2
roimax=4096
savedata=0
datname=SPECA.dat
fmt=dat
autoinc=0
smoothpts=5
caluse=0
calch0=0.00
calvl0=0.000000
calch1=100.00
calvl1=100.000000
caloff=0.000000
calfact=1.000000
calunit=keV
```

This file is automatically read at the start of the program and the parameters are set. Together with each data file a header file with extension .4LP is saved. This header also contains all settings and in addition some information like the date and time of the measurement, comments and calibration parameters entered in the MCDWIN program.

The „**Remote mode...**“ item in the settings menu or the „**Remote“** button in the System Definition dialog box opens the Remote Control dialog box. Here all settings can be made for the control of the MCD4LAP server program via a serial port.

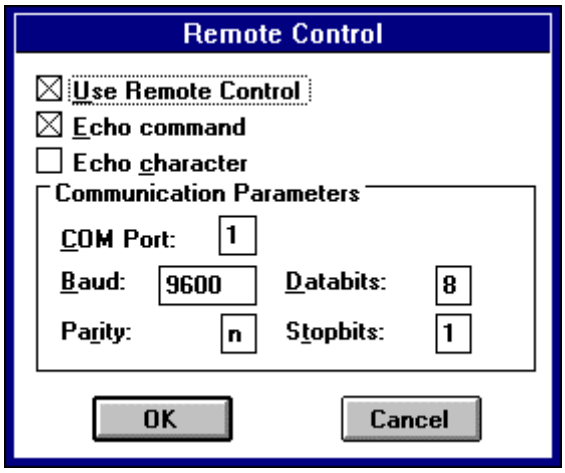


Figure 4.6: Remote Control dialog box

If the Checkbox „Use Remote Control“ is marked and the COMCTL.DLL is available, the specified COM port will be used for accepting commands. If „Echo command“ is marked, the input line will be echoed after the newline character was sent. „Echo character“, on the other hand, immediately echoes each character. The possible commands and their syntax are listed in the following section.

4.1. Control Language

A sequence of commands that are stored in a file with extension .CTL can be executed by the MCD4LAP server program or MCDWIN with the „Load “ command. A lot of these commands are already known as the configuration files W4LAPx.INF or the headerfiles with extension .4LP contain such commands to set the parameters. Each command starts at the begin of a new line with a typical keyword, the case is ignored. Any more characters in a line may contain a value or a comment.

The file W4LAPA.INF contains a complete list of commands for setting parameters; an example is:

```
wndwidth=284        ; Sets width of server window
wndheight=308       ; Sets height of server window
sysdef=1651         ; Sets system definition word (hexadecimal)
range=4096          ; Sets histogram length
dacena=0            ; DAC enable (1=enable, 0=disable)
outlevel=0          ; Sets DAC output level
orpreset=0          ; All presets OR enable
scalprena=0         ; Scaler A preset enable
scalpreset=10000    ; Scaler A preset value
digstsena=1         ; 4 Statusbits at Dig I/O enable
digsmplena=0        ; Value for samplechanger at Dig I/O enable
digsmplval=0        ; Samplechanger value
rtpreset=1000       ; Realtimepreset value (seconds)
rtprena=0           ; Realtimepreset enable
ltpreset=1000       ; Livetimepreset value (seconds)
ltprena=0           ; Livetimepreset enable
roipreset=10000     ; ROIpreset value
roiprena=0          ; ROIpreset
roimin=2            ; ROI lower limit (inclusive)
roimax=4096         ; ROI upper limit (exclusive)
savedata=0          ; Save at Halt
datname=SPECA.dat   ; Filename
fmt=dat             ; Format (ASCII: asc, Binary: dat, GANAAS: spe)
autoinc=0           ; Enable Auto increment of filename
smoothpts=5         ; Number of points to average for a smooth operation
caluse=0            ; Use calibration
calch0=0.00         ; First calibration point channel
calvl0=0.000000     ; First calibration point value
calch1=100.00       ; Second calibration point channel
calvl1=100.000000   ; Second calibration point value
caloff=0.000000     ; Calibration parameter: Offset
calfact=1.000000    ; Calibration parameter: Factor
```

calunit=keV ; Calibration unit

The following commands perform actions and therefore usually are not included in a W4LAPx.INF file:

start ; Clears the data and starts a new acquisition for system 1. Further ; execution of the .CTL file is suspended until any acquisition stops ; due to a preset.

start2 ; Clears and starts system 2. Further execution is NOT suspended.

start3 ; Clears and starts system 3. Further execution is NOT suspended.

start4 ; Clears and starts system 4. Further execution is NOT suspended.

halt ; Stops acquisition of system 1 if one is running.

halt2 ; Stops acquisition of system 2 if one is running.

halt3 ; Stops acquisition of system 3 if one is running.

halt4 ; Stops acquisition of system 4 if one is running.

cont ; Continues acquisition of system 1. If a time preset is already ; reached, the time preset is prolongated by the value which was valid ; when the „start“ command was executed. Further execution of the ;.CTL file is suspended (see start).

cont2 ; Continues acquisition of system 2 (see start2).

cont3 ; Continues acquisition of system 3 (see start3).

cont4 ; Continues acquisition of system 4 (see start4).

savecnf ; Writes the settings into W4LAPA.INF... W4LAPD.INF

MC_A ; Sets actual multichannel analyzer to MC_A for the rest of the ; controlfile.

MC_B ; Sets actual multichannel analyzer to MC_B for the rest of the ; controlfile.

MC_C ; Sets actual multichannel analyzer to MC_C for the rest of the ; controlfile.

MC_D ; Sets actual multichannel analyzer to MC_D for the rest of the ; controlfile.

savedat ; Saves data of actual multichannel analyzer. An existing file is ; overwritten.

pushname ; pushes the actual filename on an internal stack that can hold 4 ; names.

popname ; pops the last filename from the internal stack.

load ; Loads data of actual multichannel analyzer; the filename must be ; specified before with a command datname=...

add ; Adds data to actual multichannel analyzer; the filename must be ; specified before with a command datname=...

sub ; Subtracts data from actual multichannel analyzer; the filename must be ; specified before with a command datname=...

smooth ; Smoothes the data in actual multichannel analyzer

eras ; Clears the data of system 1.

eras2 ; Clears the data of system 2.

eras3 ; Clears the data of system 3.

eras4 ; Clears the data of system 4.

exit ; Exits the W4LAP (and MCDWIN) programs

alert Message	; Displays a Messagebox containing Message and an OK button that ; must be pressed before execution can continue.
waitinfo 5000 Message	; Displays a Messagebox containing Message, an OK and an END ; button. After the specified time (5000 msec) the Messagebox ; vanishes and execution continues. OK continues immediately, END ; escapes execution.
delay 4000	; Waits specified time (4000 msec = 4 sec).
run controlfile	; Runs a sequence of commands stored in controlfile. This command ; cannot be nested, i.e. it is not possible to execute a run command ; from the controlfile called.
onstart command	; The command is executed always after a start action when the ; acquisition is already running. The command can be any valid ; command, also 'run controlfile' is possible.
onstart off	; Switches off the 'onstart' feature. Also a manual Stop command ; switches it off.
onstop command	; The command is executed always after a stop caused by a ; preset reached or trigger. This can be used to program measure ; cycles. For example the command 'onstop start' makes a ; loop of this kind.
onstop off	; Switches off the 'onstop' feature. Also a manual Stop command ; switches it off.
lastrun=5	; Defines the file count for the last run in a measure cycle. After a ; file with this count or greater was saved with autoinc on, instead ; of the 'onstop command' the 'onlast command' is executed.
numruns=5	; Defines the file count for the last run in a measure cycle. The ; last count is the present one plus the numruns number.After a ; file with this count was saved with autoinc on, instead of the ; 'onstop command' the 'onlast command' is executed.
onlast command	; The command is executed after a stop caused by a preset ; reached or trigger instead of the 'onstop command', when the ; last file count is reached with autoinc on. This can be used to ; finish programmed measure cycles.
onlast off	; Switches off the 'onlast' feature. Also a manual Stop command ; switches it off.
onstart2 command	; The command is executed always after a start2 action.
onstart3 command	; The command is executed always after a start3 action.
onstart4 command	; The command is executed always after a start4 action.
onstart2 off	; Switches off the 'onstart2' feature.
onstart3 off	; Switches off the 'onstart3' feature.
onstart4 off	; Switches off the 'onstart4' feature.
onstop2 command	; The command is executed always after a stop of system2 ; caused by a preset reached. This can be used to ; program measure cycles. For example the command ; 'onstop2 start2' makes a loop of this kind.
onstop3 command	; The command is executed always after a stop of system3 ; caused by a preset reached.
onstop4 command	; The command is executed always after a stop of system4 ; caused by a preset reached.
onstop2 off	; Switches off the 'onstop2' feature. Also a manual Stop2 ; command switches it off.

onstop3 off	; Switches off the 'onstop3' feature. Also a manual Stop3 ; command switches it off.
onstop4 off	; Switches off the 'onstop4' feature. Also a manual Stop4 ; command switches it off.
numruns2=5	; Defines the number of runs in a measure cycle with "onstop2".
numruns3=5	; Defines the number of runs in a measure cycle with "onstop3".
numruns4=5	; Defines the number of runs in a measure cycle with "onstop4".
onlast2 command	; The command is executed after a stop caused by a preset ; reached or trigger instead of the 'onstop2' command, when the ; last file count is reached with autoinc on. This can be used to ; finish programmed measure cycles.
onlast3 command	; See onlast2, for loops with 'onstop3'.
onlast4 command	; See onlast2, for loops with 'onstop4'.
onlast2 off	; Switches off the 'onlast2' feature. Also a manual Stop2 command ; switches it off.
onlast3 off	; Switches off the 'onlast3' feature. Also a manual Stop3 ommand ; switches it off.
onlast4 off	; Switches off the 'onlast4' feature. Also a manual Stop4 ommand ; switches it off.
exec program	; Executes a Windows program or .PIF file. ; Example: exec notepad test.ctl ; opens the notepad editor and loads test.ctl.
fitrois	; Makes a single peak Gaussian fit for all ROIs and dumps the result ; into a logfile. This is performed by the MCDWIN program and ; therefore can be made only if this application is running.
autocal	; Makes a single peak Gaussian fit for all ROIs in the active Display of ; MCDWIN, for which a peak value was entered and uses the result for ; a calibration. This is performed by the MCDWIN program and ; therefore can be made only if this application is running.

The following commands make sense only when using the serial line control:

MC_A?	; Sends the status of MC_A via the serial port and make MC_A actual.
MC_B?	; Sends the status of MC_B via the serial port and make MC_B actual.
MC_C?	; Sends the status of MC_C via the serial port and make MC_C actual.
MC_D?	; Sends the status of MC_D via the serial port and make MC_D actual.
?	; Send the status of the actual multichannel analyzer
RROI(0,1)	; Sends the sum, mean value and max positive and negative ; deviation from mean of rectangular ROI #1 in spectra #0
sendfile filename	; Sends the ASCII file with name ‘filename’ via the serial line.

The execution of a control file can be finished from the Server or MCDWIN with any Halt command.

4.2. Controlling the MCD4LAP Windows Server via DDE

The W4LAP program can be a server for DDE (Dynamic Data Exchange). Many Windows software packages can use the DDE standard protocols to communicate with other Windows programs, for example GRAMS, FAMOS or LABVIEW. In the following the DDE capabilities of the W4LAP program are described together with a demo VI („Virtual Instrument“) for LABVIEW. It is not recommended to use the DDE protocol for LabView, as also a DLL interface is available which is much faster. The following should be seen as a general description of the DDE conversation capabilities of the W4LAP program.

4.2.1. Open Conversation

application: W4LAP

topic: MCD4LAP

Any application that wants to be a client of a DDE server, must first open the conversation by specifying an application and a topic name. The application name is W4LAP and the topic is MCD4LAP.

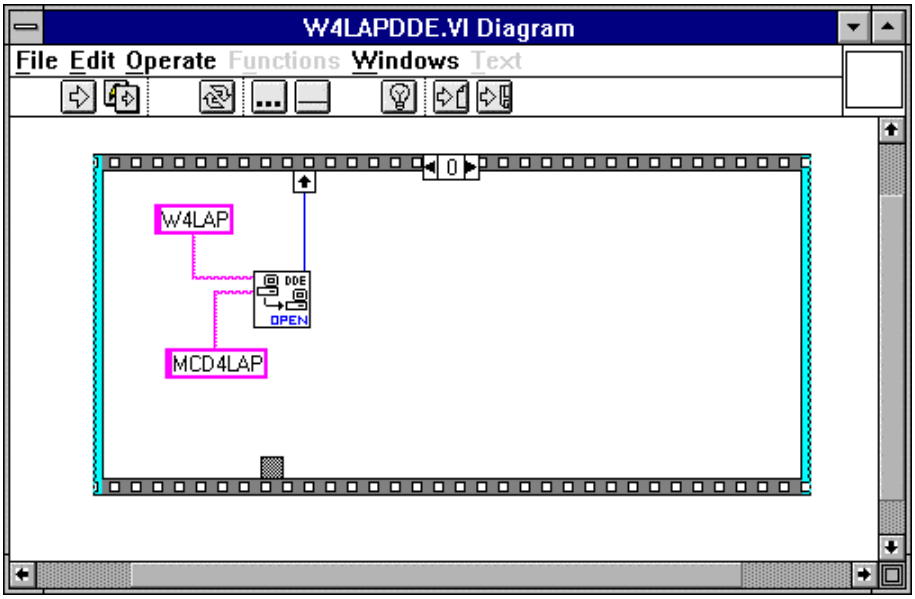


Figure 4.7: Opening the DDE conversation with the W4LAP server in LABVIEW

4.2.2. DDE Execute

The DDE Execute command can be used to perform any action of the W4LAP program. Any of the Control command lines described in section 4.1 can be used. For example a sequence of control commands saved in a file TEST.CTL can be executed by specifying the command:

```
RUN TEST.CTL
```

The W4LAP program then executes the command and, after finishing, it sends an Acknowledge message to the DDE client. This can be used for synchronizing the actions in both applications.

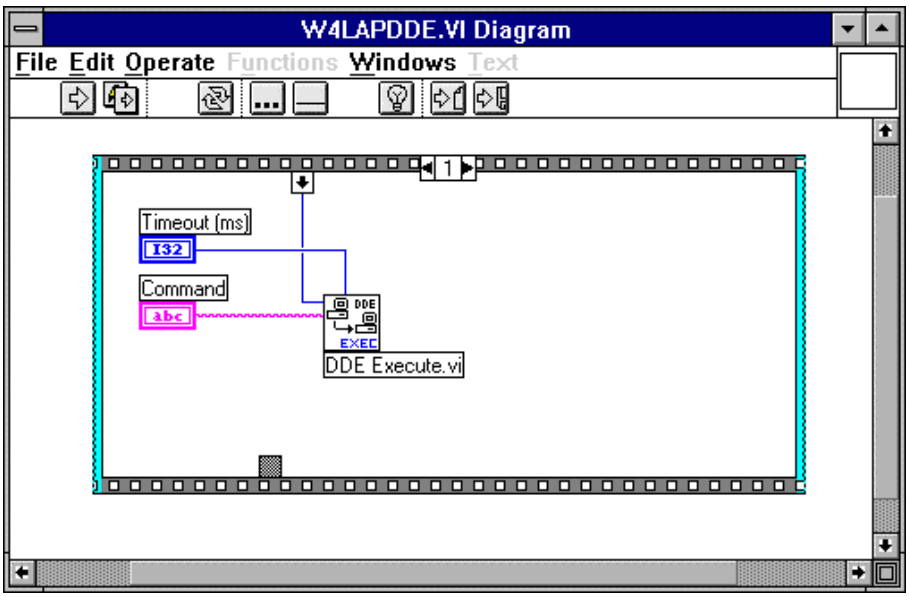


Figure 4.8: Executing a W4LAP command from a LABVIEW application

4.2.3. DDE Request

The DDE Request is a message exchange to obtain the value of a specified item. Only two items are defined for DDE request up to now: RANGE and DATA. The value is obtained as an ASCII string, i.e. it must be converted by the client to get the numbers. All other parameters concerning the W4LAP Setup can be obtained by the client application by reading and evaluating the configuration file.

RANGE

The RANGE item can be used to obtain the total number of data in the actual multichannel analyzer. The returned number is 2 less than the real range, the two time channels are omitted. The multichannel analyzer can be selected before by a command MC_A, ..., MC_D.

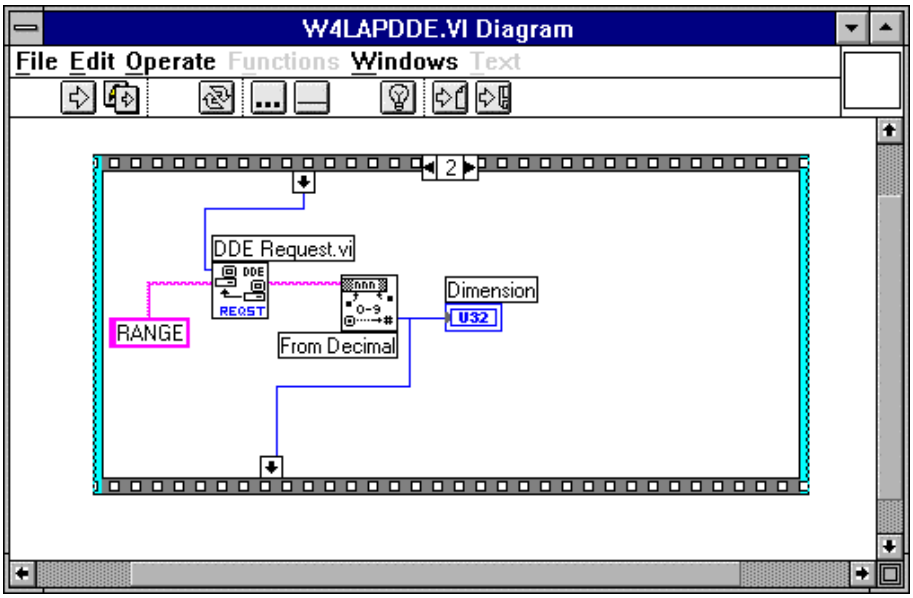


Figure 4.9: Getting the total number of data with LABVIEW

DATA

With the DATA item the data are obtained. The value of this item is a multiline string that contains in each line a decimal number as an ASCII string.

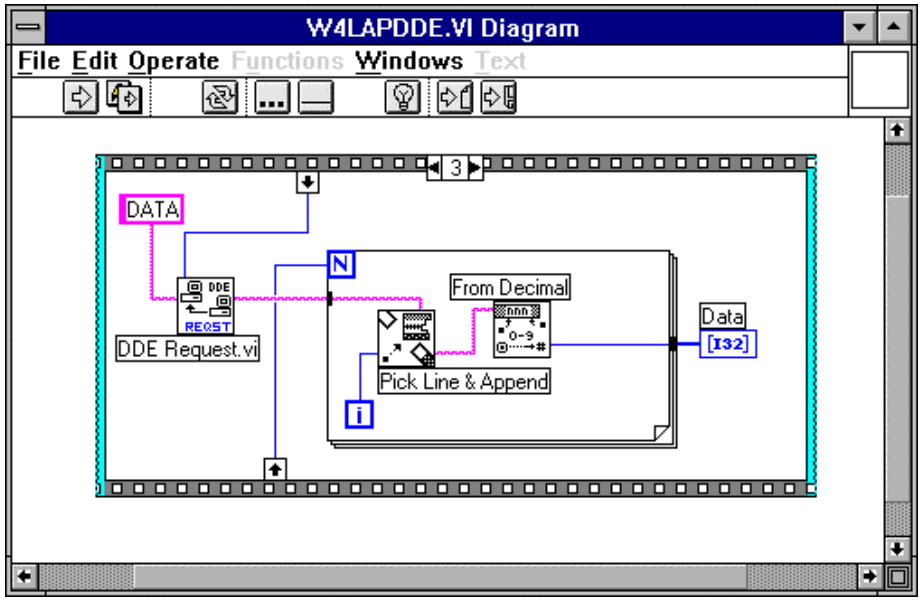


Figure 4.10: Getting the data with LABVIEW

4.2.4. Close Conversation

After finishing the DDE communication with the W4LAP program, it must be closed.

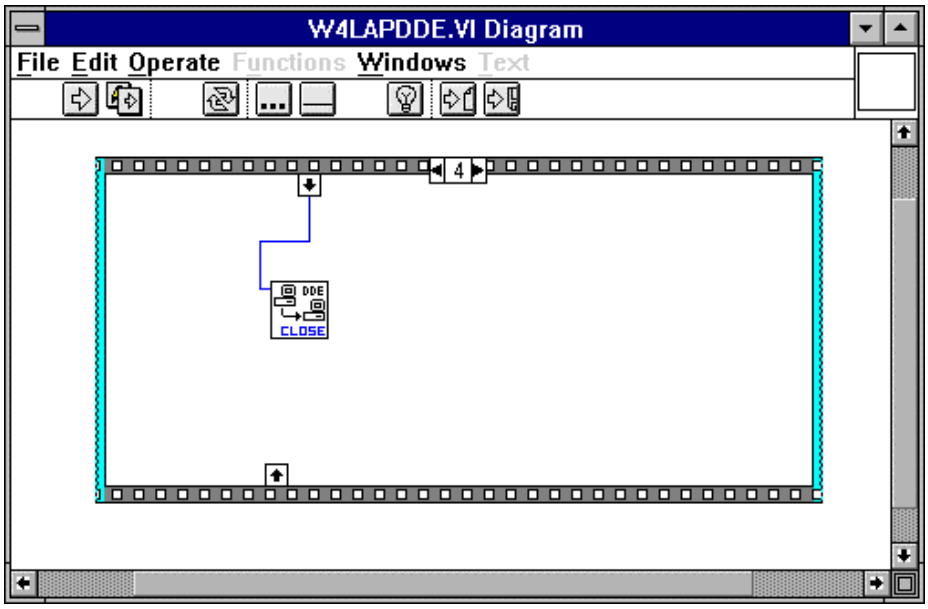


Figure 4.11: Closing the DDE communication in LABVIEW

The following figure shows the „Panel“ of the described VI for Labview.

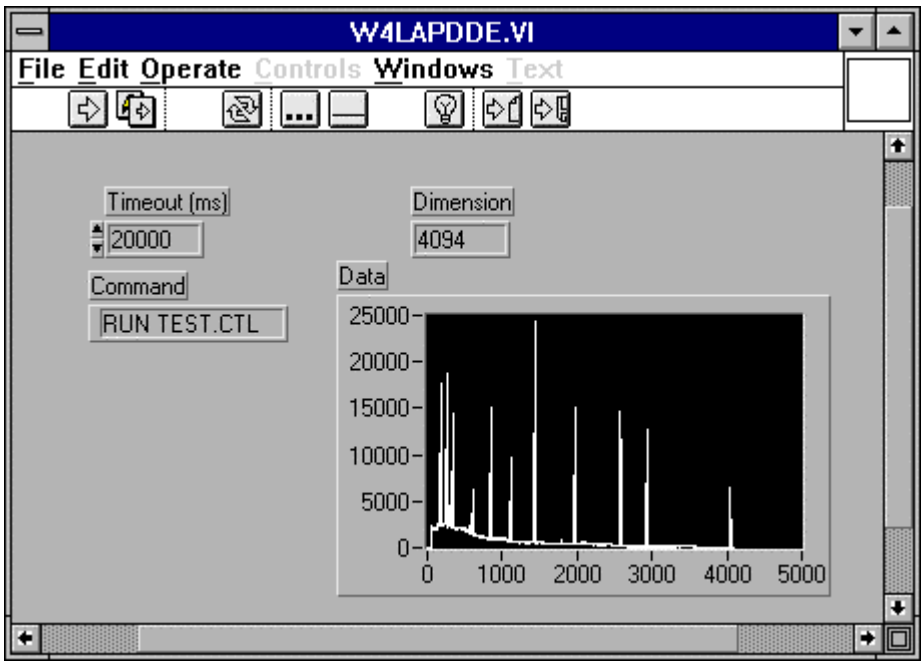


Figure 4.12: Control Panel of the demo VI for LABVIEW

4.3. Controlling the MCD4LAP Windows Server via DLL

The W4LAP server program provides via a DLL („dynamic link library“) access to all functions, parameters and data. So the server can be completely controled from the MCDWIN software that provides all necessary graphic displays.

In the following some parts of the header and definition files of the D4LAP.DLL are listed, that may help an experienced programer to use the DLL for own written applications. Please note that the complete documented sourcecode of the DLL and server including fundamental VI's and an example VI for LABVIEW and example program in Visual Basic is available as an option.

```
typedef struct{
  int started;           // aquisition status: 1 if running, 0 else
  double realtime;       // real time in seconds
  double totalsum;       // total events
  double roisum;         // events within ROI
  double totalrate;      // acquired events per second
  double nettosum;       // ROI sum with background subtracted
  double livetime;       // Livetime in seconds
  double deadtime;       // Deadtime in percent
  unsigned long maxval; // Maximum value in spectrum
} ACQSTATUS;

typedef struct{
  long range;            // spectrum length
  int rtprena;           // 1 if time preset enabled, 0 else
  int roiprena;          // 1 if ROI event preset enabled, 0 else
  long roimin;           // lower ROI limit
  long roimax;           // upper limit: roimin <= channel < roimax
  double roipreset;      // ROI preset value
  double rtpreset;       // time preset value
  int savedata;          // 1 if auto save after stop
  int fmt;               // format type: 0 == ASCII, 1 == binary
  int autoinc;           // 1 if auto increment filename
  int orpreset;          // OrPreset flag (only Setting[0] evaluated)
  int ltprena;           // enable livetime preset
  int outlevel;          // DAC out only Setting[0]
  int dacena;            // DAC enable only Setting[0]
  int scalprena;         // Scaler A Preset enable only Setting[0]
  int digstsena;         // DIG I/O only Setting[0]
                         // bit0: Status
                         // bit1: Invert Trigger
                         // bit2: Trigger System 1
                         // bit3: Trigger System 2
                         // bit4: Trigger System 3
                         // bit5: Trigger System 4
  int digsmplena;        // DIG I/O Samplchanger enable
                         // only Setting[0]
  int digsmplval;        // DIG I/O Samplchanger value
                         // only Setting[0]
  double ltpreset;       // livetime preset value
  int nregions;          // number of regions
  int caluse;            // 1 if calibration used
  double scalpreset;     // ScalerA prest
  int active;            // 1 if active
  int calpoints;         // number of calibration points
} ACQSETTING;

typedef struct{
  unsigned long huge *s0;          // pointer to spectrum
  unsigned long far *region;       // pointer to regions
  unsigned char far *comment0;     // pointer to strings
  double far *cnt;                 // pointer to counters
} ACQDATA;

typedef struct {
  int nDevices;          // Number of devices: always 4
  int nDisplays;         // Number of displays (active MCA's): 0...4
  int nSystems;          // Number of systems
```

```
  int bRemote;            // 1 if server controled by MCDWIN
  int sys;                // System definition word
} ACQDEF;

/*** FUNCTION PROTOTYPES (do not change) ***/
int  FAR PASCAL LibMain(HANDLE, WORD, WORD, LPSTR);
VOID FAR PASCAL StoreSettingData(ACQSETTING FAR *Setting, int nDisplay);
                          // Stores Settings into the DLL
int FAR PASCAL GetSettingData(ACQSETTING FAR *Setting, int nDisplay);
                          // Get Settings stored in the DLL
VOID FAR PASCAL StoreStatusData(ACQSTATUS FAR *Status, int nDisplay);
                          // Store the Status into the DLL
int FAR PASCAL GetStatusData(ACQSTATUS FAR *Status, int nDisplay);
                          // Get the Status
VOID FAR PASCAL Start(int nSystem);             // Start
VOID FAR PASCAL Halt(int nSystem);              // Halt
VOID FAR PASCAL Continue(int nSystem);          // Continue
VOID FAR PASCAL NewSetting(int nDevice);
                         // Indicate new Settings to Server
UINT FAR PASCAL ServExec(HWND ClientWnd);
      // Execute the Server W4LAP.EXE
VOID FAR PASCAL StoreData(ACQDATA FAR *Data, int nDisplay);
                          // Stores Data pointers into the DLL
int FAR PASCAL GetData(ACQDATA FAR *Data, int nDisplay);
                          // Get Data pointers
long FAR PASCAL GetSpec(long i, int nDisplay);
                          // Get a spectrum value
VOID FAR PASCAL SaveSetting(void);              // Save Settings
int FAR PASCAL GetStatus(int nDevice);          // Request actual Status
from Server
VOID FAR PASCAL Erase(int nSystem);             // Erase spectrum
VOID FAR PASCAL SaveData(int nDevice);          // Saves data
VOID FAR PASCAL GetBlock(long FAR *hist, int start, int end, int step,
  int nDisplay);          // Get a block of spectrum data
VOID FAR PASCAL StoreDefData(ACQDEF FAR *Def);
                          // Store System Definition into DLL
int FAR PASCAL GetDefData(ACQDEF FAR *Def);
                          // Get System Definition
VOID FAR PASCAL LoadData(int nDevice);          // Loads data
VOID FAR PASCAL AddData(int nDevice);       // Adds data
VOID FAR PASCAL SubData(int nDevice);       // Subtracts data
VOID FAR PASCAL Smooth(int nDevice);        // Smooth data
VOID FAR PASCAL NewData(void);                  // Indicate new ROI or
                                                // string Data
VOID FAR PASCAL HardwareDlg(int item);          // Calls the Settings
                                                dialog box
VOID FAR PASCAL UnregisterClient(void);         // Clears remote mode
                                                from MCDWIN
VOID FAR PASCAL DestroyClient(void);            // Close MCDWIN
UINT FAR PASCAL ClientExec(HWND ServerWnd);
                              // Execute the Client MCDWIN.EXE
int FAR PASCAL LVGetDat(unsigned long huge *datp, int nDisplay);
                              // Copies the spectrum to an array
VOID FAR PASCAL RunCmd(int nDevice, LPSTR Cmd);
                                                // Executes command
int FAR PASCAL LVGetRoi(unsigned long far *roip, int nDevice);
                              // Copies the ROI boundaries to an array
int FAR PASCAL LVGetCnt(double far *cntp, int nDevice);
                              // Copies the Cnt numbers to an array
int FAR PASCAL LVGetStr(char far *strp, int nDevice);
                              // Copies the strings to an array


EXPORTS
WEP   @1 RESIDENTNAME
        StoreSettingData          @2
        GetSettingData            @3
        StoreStatusData           @4
        GetStatusData             @5
        Start                     @6
        Halt                      @7
```

```
Continue                @8
NewSetting              @9
ServExec                @10
StoreData               @11
GetData                 @12
GetSpec                 @13
SaveSetting             @14
GetStatus               @15
Erase                   @16
SaveData                @17
GetBlock                @18
StoreDefData            @19
GetDefData              @20
LoadData                @21
NewData                 @22
HardwareDlg             @23
UnregisterClient        @24
DestroyClient           @25
ClientExec              @26
LVGetDat                @27
RunCmd                  @28
AddData                 @29
LVGetRoi                @30
LVGetCnt                @31
LVGetStr                @32
SubData                 @33
Smooth                  @34
```

5. MCDWIN Program

The window of the MCDWIN program is shown here. It allows to fully control the MCD4LAP card via the server program, perform measurements and save data, and shows the data on-line in several windows.

The server program W4LAP.EXE automatically starts MCDWIN. If you try to start MCDWIN before the server is started, a message box warns that you should start first the server.

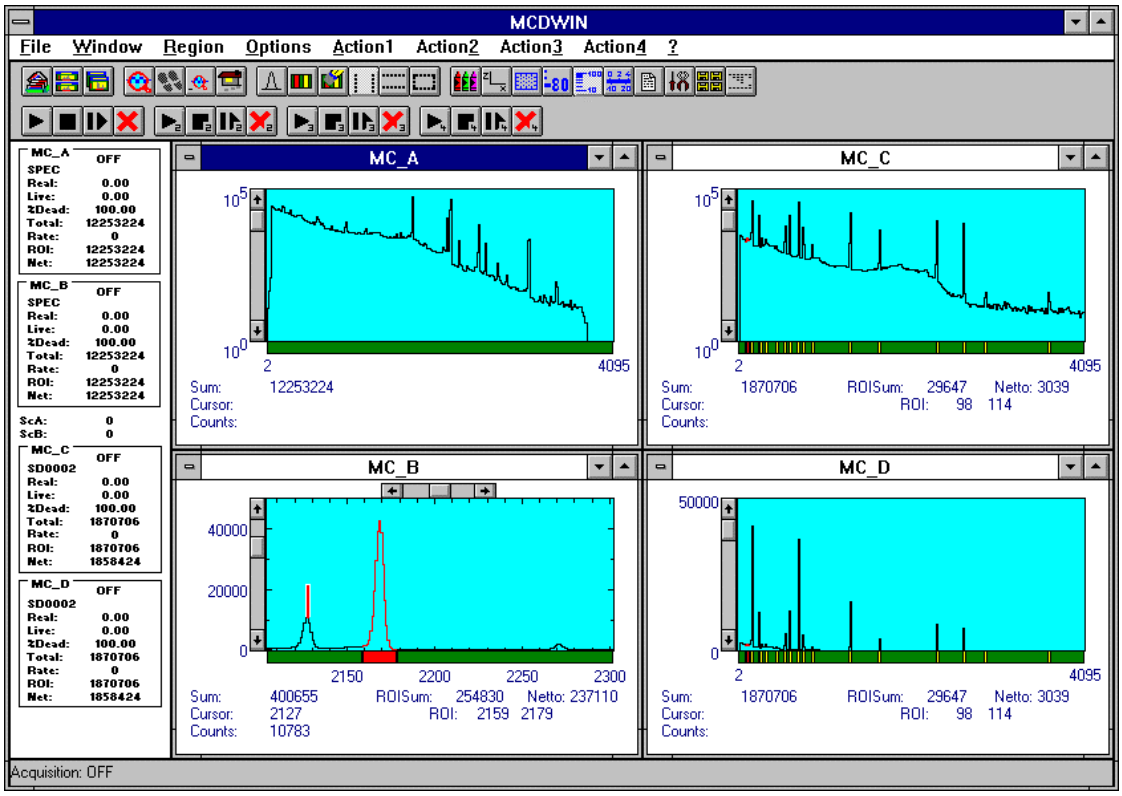


Figure 5.1: MCDWIN main window

A status window at the left side gives all information about the status of the MCD4LAP. A toolbar provides fast access to many used functions in the menu. A status bar at the bottom indicates the meaning of the toolbar icons. A cursor appears when clicking the left mouse button inside the graphic area. To clear the cursor, move the cursor outside the spectrum display and double click with the right mouse button. To define a region, press the right mouse button, and while keeping the button pressed, drag a rectangle. In the zoomed state a scrollbar appears that allows to scroll through the spectrum.

In the following the several menu functions are described together with the corresponding toolbar icons.

5.1. File Menu

Load...,Add..., Save, Save As...

These menu items provide the usual functions for loading and saving data into the MCA selected by the active window. When saving data, you have the choice between binary (.DAT) and ASCII (.ASC) format. When you load data, select a header file (extension .4LP). This file contains the information about the size and format of the data file, which is then automatically read. With „**Add**“ the data is added to the present data. The data read from a file is shifted according to the calibration, if it is available.

Open New...

With the Open New menu item or the corresponding icon a new Display window can be created and shown as the active window. In the „**Open New Display**“ dialog box the MCA for the new display can be selected.

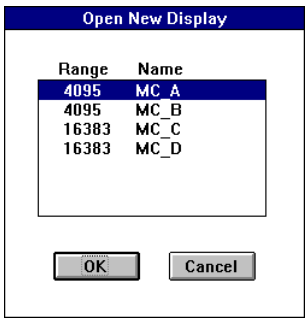


Figure 5.2: Open New Display dialog box

Open All

By selecting the Open All menu item, all available Displays are shown. The windows of the last opened Display becomes active.

Print...

The Print menu item prints a Display to the printer. Only the visible part of the spectra will be printed. The size and position of the graphic on paper can be adjusted in a dialog box.

If printing takes a long time and disk activity is high, please note the following: The picture for the printing is first built in the memory, but it may need quite a lot of memory if the printer resolution is high and therefore Windows makes intense virtual memory swapping to disk if for example only 8 Mb RAM are available. Therefore it is recommended: never use a 600 dpi printer driver for the printout of spectra. For example for an HP Laser 4, install the PCL driver and use 300 dpi. The PCL driver is also much more effective than a Postscript driver, printing is much faster. With 600 dpi, the maximum figure size is indeed limited to about 12 cm x 7 cm (Windows cannot handle on an easy way bitmaps larger than 16 Mb).

Setup Printer...

The Setup Printer menu item allows to configure the printer.

Exit

The Exit menu item exits the MCDWIN.

5.2. Window Menu

The Window menu allows to arrange the Display windows.

Tile

With the Tile menu item or clicking the corresponding icon, all opened and displayed MCDWIN Display windows are arranged over the full MCDWIN client area trying to allocate the same size for each window.

Cascade

The Cascade menu item or respective icon arranges all windows in a cascade display.

Arrange Icons

By the Arrange Icons menu item, the minimized MCDWIN Display windows are arranged in a series at the bottom of the MCDWIN client area.

Close All

By selecting the Close All menu item, all Display windows are closed.

Window list

At the end of the Window menu, all created Display windows are listed with their names, the current active window is checked. By selecting any of the names, this window becomes the active window and is displayed in front of all the others.

5.3. Region Menu

The Region menu contains commands for Regions and ROIs (Regions of Interest). A Region can be defined by marking it in a display, with the mouse using the right mouse button and dragging a rectangle over the area one is interested in. A ROI, i.e. an already defined region in a single spectrum can be shown zoomed by double-clicking with the left mouse button on the corresponding colored area in the bar at the bottom of the spectra display. A single mouse click with the left button on the corresponding colored area makes this to the active ROI and lets the counts contained in this ROI be displayed in the information lines of the respective window.

Zoom

The Zoom item or respective icon enlarges a Region to the maximum Spectrum Display size.

Back

The Back menu item or clicking the corresponding icon restores the last zoom view. Each time a Back command is clicked the view is stepped back one step.

Zoom Out

The Zoom Out menu item or clicking the corresponding icon enlarges the actual zoom view by a factor 2, if possible.

Home

Clicking the Home menu item or the corresponding icon restores a Display to the basic configuration.

Shape

Selecting the Shape menu item opens a submenu with the items Rectangle, X-Slice, Y-Slice, and Polygon to choose the ROI shape.

Rectangle

Sets the region shape to a rectangle with arbitrary dimensions. To enter the rectangular region, press the right mouse button, drag a rectangle, and release the button to define the region.

X-Slice

Sets the Region shape to the rectangle with maximal height.

Y-Slice

Sets the Region shape to the rectangle with maximal width.

Create

The Create menu item creates a new ROI from the current marked Region.

Delete

By selecting the Delete menu item or the respective icon, the current active ROI is deleted and the previously defined ROI is activated.

Edit...

With the Edit item, a dialog box is opened which allows to edit the ROI list, i.e. create a new or delete, change and activate an existing ROI. Also the peak values for an automatic calibration can be entered here. A ROI can be edited and added to the list. It can also be made to the „Active ROI“, that is the special ROI that is used by the server program to calculate the events within this ROI and look for an event preset. The ROI list can be cleared and it can be written into a file with extension .CTL, which can be directly loaded into the server to restore the ROI list.

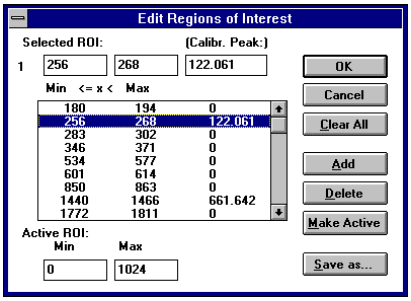


Figure 5.3: ROI Editing dialog box

Fit...

By selecting the Fit... menu item or the respective icon, A single Gaussian peak fit with linear background is performed for the currently marked region. The fitted curve is displayed and a dialog box shows the the results:

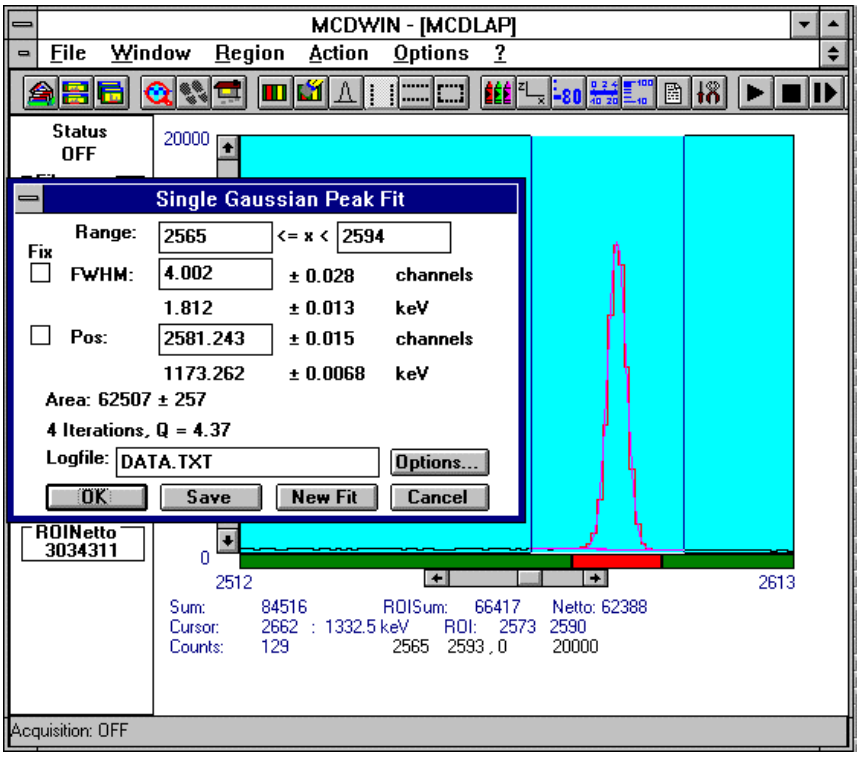


Figure 5.4: Single Gaussian Peak Fit

The full width at half maximum FWHM and Position of the Gaussian can be changed and a New Fit can be performed, they even can be fixed to the entered value by marking the respective checkbox. The Position and FWHM are displayed in channels and also in calibrated units, if a calibration is available. The area of the Gaussian is also shown. For all values also the standard deviations are given. The value of Q is the normalized chi**2. To take into account the systematic error of the lineshape, you may multiply the errors with the squareroot of Q. Click on Save to append a line containing the results to a Logfile with the specified name. OK closes the dialog and lets the fitted function in the display also if it is refreshed, whereas after Cancel the curve no longer will be shown in a refrehsed display. Options... opens a new dialog box to define the information in the logfile:

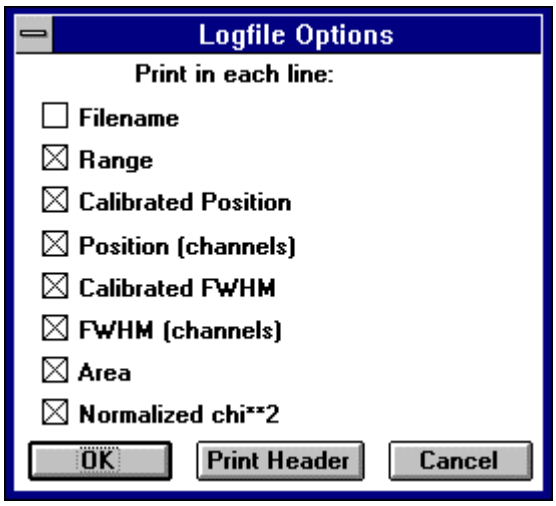


Figure 5.5: Logfile Options for the Single Gaussian Peak Fit

The several quantities are written in standard text format with Tabs as seperators and a Newline character at the end of each line, so the file can be read with standard calculation programs like EXCEL. Click on Print Header to write a header line.

Fit ROIs

With the Fit ROIs item, for all ROIs a Single Gaussian Peak Fit is performed and the results are dumped into the logfile.

Auto Calib

Makes a Gauss fit for all ROIs in the active Display for which a peak value was entered, and performs a calibration using the fit results.

5.4. Options Menu

The Options Menu contains commands for changing display properties like scale, colors etc., hardware settings, calibration and comments.

Colors...

The Colors menu item or respective icon opens the Colors dialog box. It changes the palette or Display element color depending on which mode is chosen. The current color and palette set-up may be saved or a new one can be loaded.

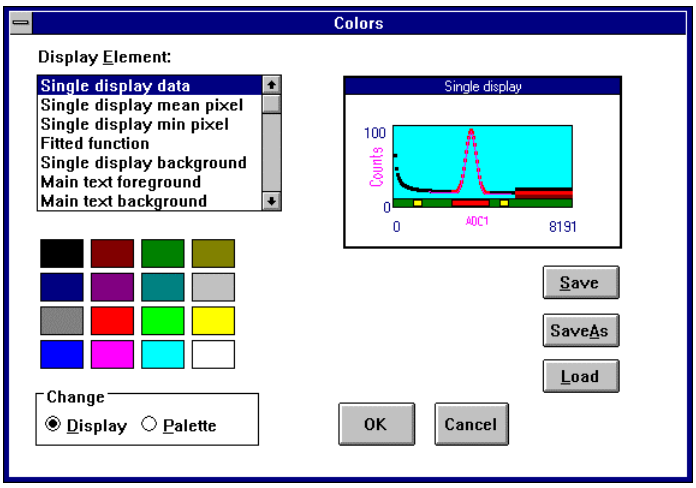


Figure 5.6: Colors dialog box

Display...

The Display menu item or the corresponding icon opens the Display Options dialog box.

Here the graphic display mode of single spectra can be chosen. The 'type' combo box gives a choice between **dot**, **histogram**, **spline I** and **line**.

'Dot' means that each spectra point is shown as a small rectangle, the size of this rectangle can be adjusted with the **size** combo box. 'Histogram' is the usual display with horizontal and vertical lines, 'spline I' means linear interpolation between the points, and 'line' means vertical lines from the ground to each spectra point.

If the displayed spectra range contains more channels as pixel columns are available in the video graphic display, usually only the maximum value of the channels falling into that pixel columns is displayed. But it can also explicitly specified by marking the checkboxes „**Max Pixel**“, „**Mean Pixel**“ or „**Min Pixel**“ which value will be displayed. It is also possible to display all three possible values in different colors that can be chosen in the colors dialog. For the „Mean Pixel“ a Threshold value can be entered; channel contents that are below this value are then not taken into account for the mean value calculation.

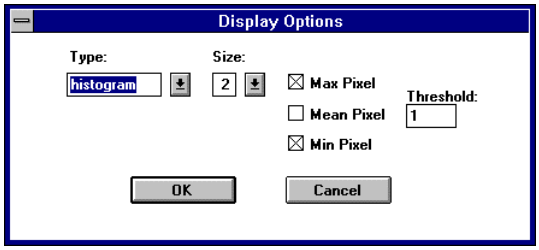


Figure 5.7: Display Options dialog box

Axis...

By the Axis... menu item or the respective icon, the Axis Parameters dialog box is opened.

It provides many choices for the axis of a display. The frame can be rectangular or L-shape, the frame thickness can be adjusted (xWidth, yWidth). A grid for x and y can be enabled, the style can be chosen between Solid, Dash, DashDot and DashDotDot. Ticks on each of the four frame borders can be enabled, the tick length and thickness can be chosen. The style of the axis

labeling depends on enabled ticks at the bottom respective left side: If no ticks are enabled there, only the lowest and highest values are displayed at the axis, otherwise the ticks are labeled.

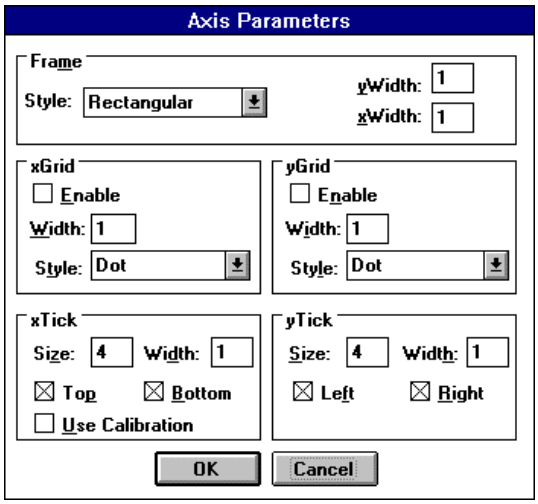


Figure 5.8: Axis Parameter dialog box

Scaling...

The Scaling menu item or the corresponding icon opens the Scale Parameters dialog box.

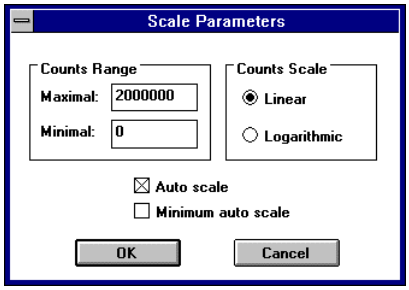


Figure 5.9: :Scale Parameters dialog box

It allows to change the ranges and attributes of a Spectrum axis. By setting the Auto scaling mode, the MCDWIN will automatically recalculate the maximum y axes of the visible Spectrum region only. To keep the same height of the visible region for a longer time, set the Auto scaling mode off. Then with the scroll bar thumb one can quickly change the visible region scale, otherwise the scale will be changed automatically. The Minimum auto scale mode helps to display weak structures on a large background.

Lin / Log scale

For a Lin scale all data intervals have the same size. With Log scale the intervals will be small for small y values and large for large y values. All options have effect only on the active Display.

Calibration...

Using the Calibration menu item or the corresponding icon opens the Calibration dialog box.

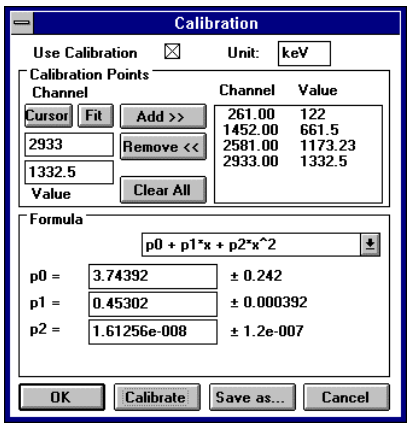


Figure 5.10: Calibration dialog box

Make a choice of several calibration formulas. Enter some cursor positions and the corresponding values, click on Add, then on Calibrate. The obtained coefficients can be inspected together with the statistical error, or they can be changed and entered by hand. If 'use calibration' is on, the calibrated values are displayed together with the channel position of the cursor.

Comments...

Up to eleven comment lines with each 60 characters can be entered using the Comments dialog box. The content of these lines is saved in the data header file. The first line contains automatically the time and date when a measurement was started. The titles of each line can be changed by editing the file COMMENT.TXT.

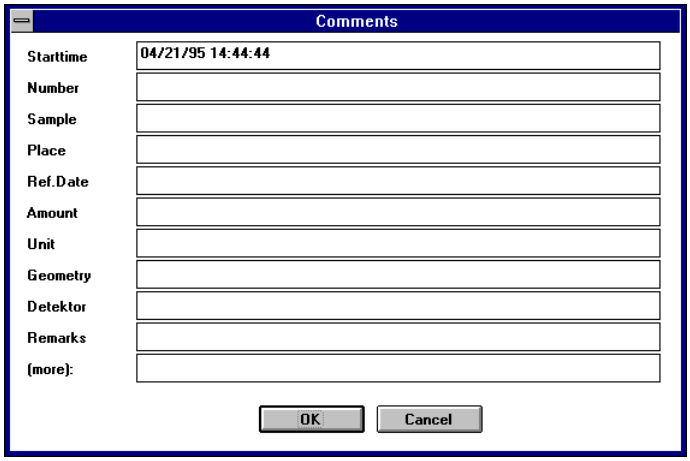


Figure 5.11: Comments dialog box

Range, Preset...

The Range, Preset dialog box allows to make all the respective MCD4LAP settings (See MCD4LAP Server documentation).

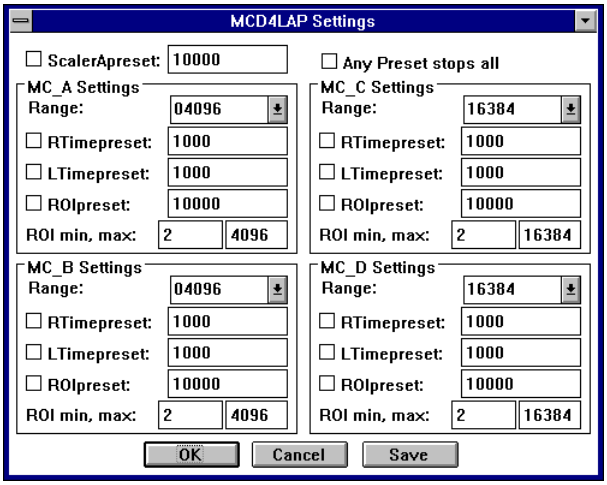


Figure 5.12: Settings dialog box

Data...

The Data dialog box allows to make all the respective MCD4LAP settings (See MCD4LAP Server documentation).

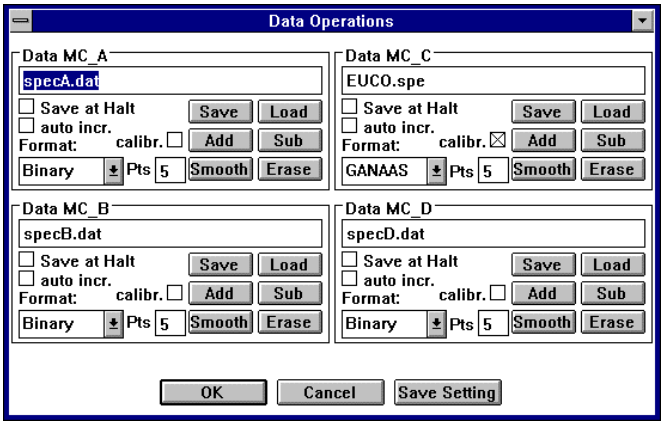


Figure 5.13: Data Operations dialog box

System...

The System Definition dialog box allows to make all the respective MCD4LAP settings (See MCD4LAP Server documentation).

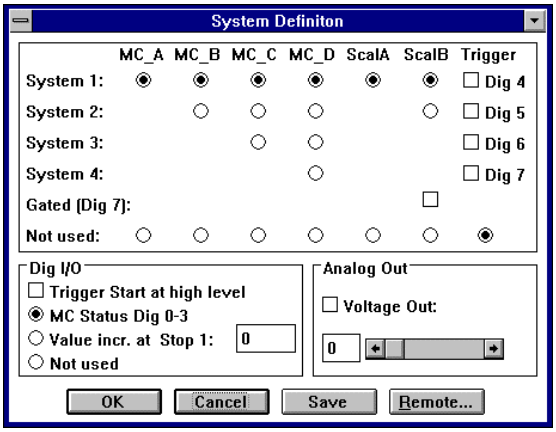
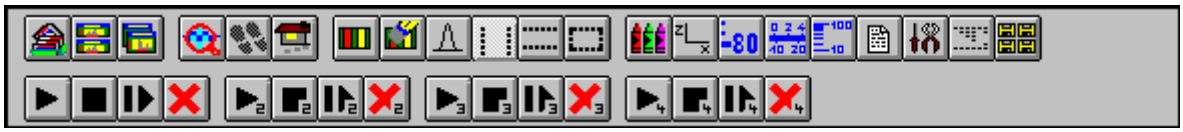


Figure 5.14: System Definition dialog box

Tool Bar...



Selecting the Tool Bar Menu item opens the Tool Bar Dialog Box. It allows to arrange the icons in the Tool Bar.

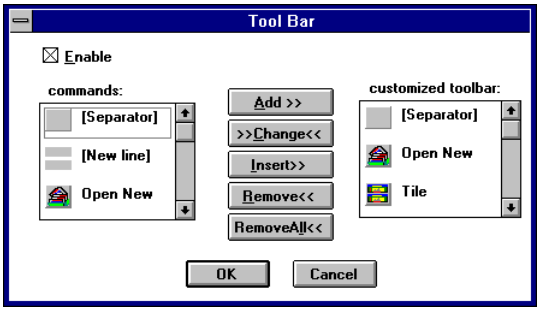


Figure 5.15: Tool Bar dialog box

If it is enabled, an array of icons in the MCDWIN Menu is shown. Clicking the left mouse button with the cursor positioned on an icon, the user can perform a corresponding MCDWIN Menu command very quick.

Status bar

With this menu item the Status bar at the bottom of the MCDWIN main window can be switched on or off. A corresponding checkmark shows if it is active or not. The Status bar usually shows if an acquisition is active. When the left mouse button is pressed while the mouse cursor is within a toolbar icon, it displays a short help message what the meaning of the toolbar icon is.

Status window

The same way it is possible to hide or show the status window at the left side of the MCDWIN main window.

Save

Saves all parameters defined in the Options menu to the MCDWIN.CNF config file.

Save As...

Saves all parameters defined in the Options menu to a user defined config file.

Retrieve...

Loads a new configuration.

5.5. Action Menu

The Action Menu or corresponding toolbar icons contain the commands to start, stop, continue and erase a measurement. If more than one systems are formed, also more actions menus are available, otherwise they are grayed.

Start

The Start toolbar button erases the data and starts a new measurement.

Halt

The Halt toolbar button stops a measurement.

Continue

The Continue toolbar button continues a measurement.

Erase

The Erase toolbar button erases the data.

6. Two MCD4LAP cards in one computer

6.1. The W9LAP Windows Server program

Two MCD4LAP cards can be used in a computer. In this case they must be set to different port addresses. Also the Preset Bus of both cards should be connected (ref. Hardware Description). The W9LAP server program must then be used instead of W4LAP. It handles two MCD4LAP cards and is in addition able to calculate online a sumspectrum of the several MCA spectra. The calibration of the individual spectra can be used to shift the spectra accordingly before summing them up. Via the D9LAP.DLL it communicates with MCDWIN.

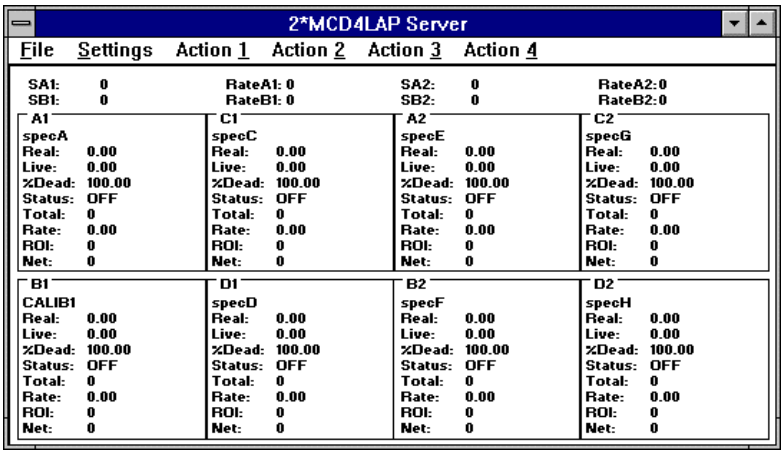


Figure 6.1: Status display of the W9LAP Server Program

The configuration file W9LAP.INI contains the port base addresses and the ADC port handshake signal polarities, see Figure 6.2. The other settings are stored in W9LAPA1.INF, ... W9LAPD2.INF. The server program is normally shown as an icon. After a double click it is opened to show a status window. It can be resized and arranged either horizontally or vertically.

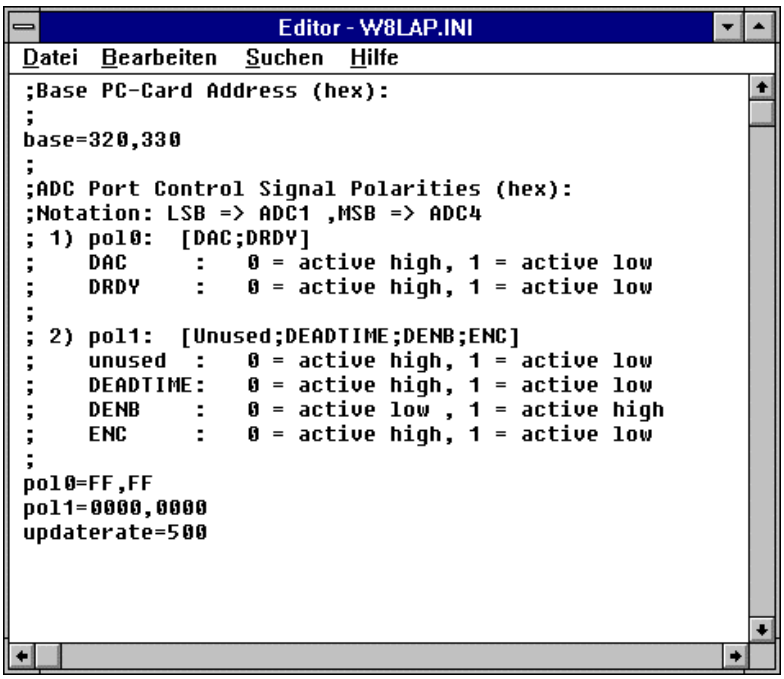


Figure 6.2: Sample W9LAP.INI file

Clicking in the File Menu on the Data... item opens the Data Operations dialog box.

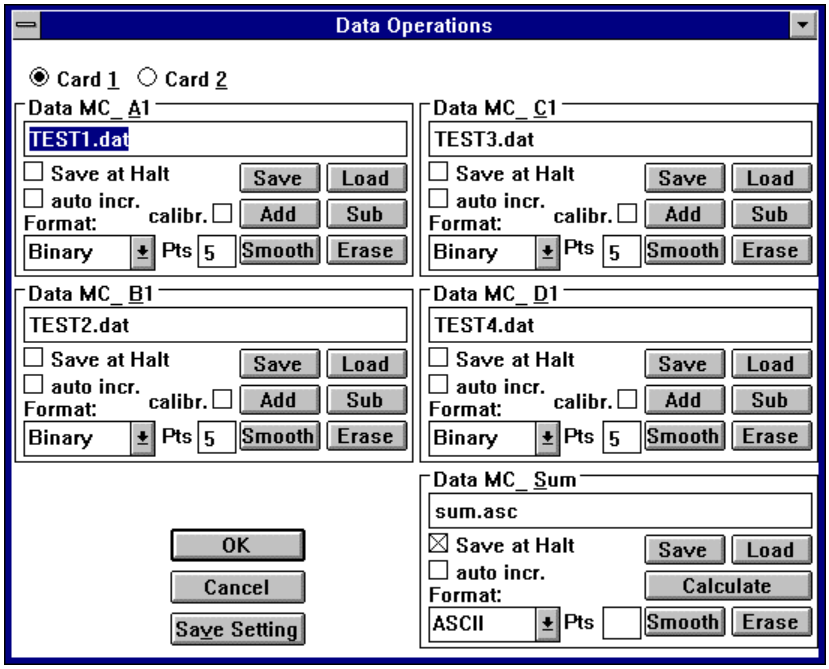


Figure 6.3: Data Operations dialog box

This dialog allows to edit the data settings of the eight MCA's. Two Radio buttons labeled **„Card1“** and **„Card2“** switch between both cards. It allows to save, calculate and erase the sum spectrum. The other dialog fields are analogue to the W4LAP server.

Clicking in the Settings menu on the MCD... item opens the MCD4LAP Settings dialog box. Here the presets and range parameters for the MCA's and Scaler A are set. Also this dialog is the same as the respective W4LAP dialog but expanded with the two radio buttons.

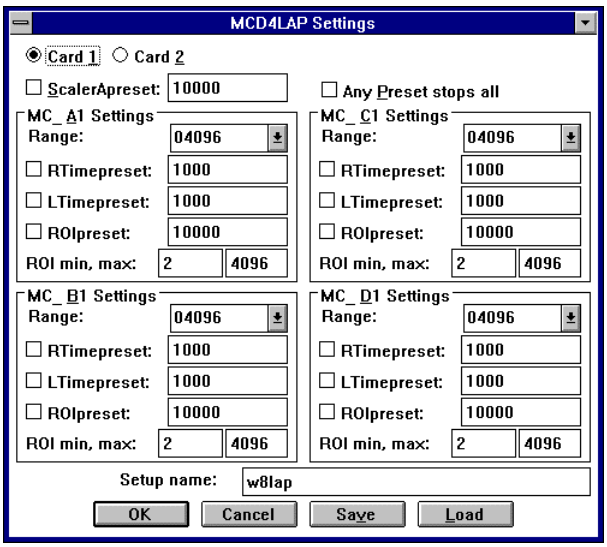


Figure 6.4: 2*MCD4LAP Settings dialog box

The „System...“ item in the settings menu opens the System Definition dialog box. Here the several MCA's and Scalers can be combined to form one or up to four separate systems that can be started, stopped and erased by one command. In addition the use of the digital input / output and the analog output ports can be defined.

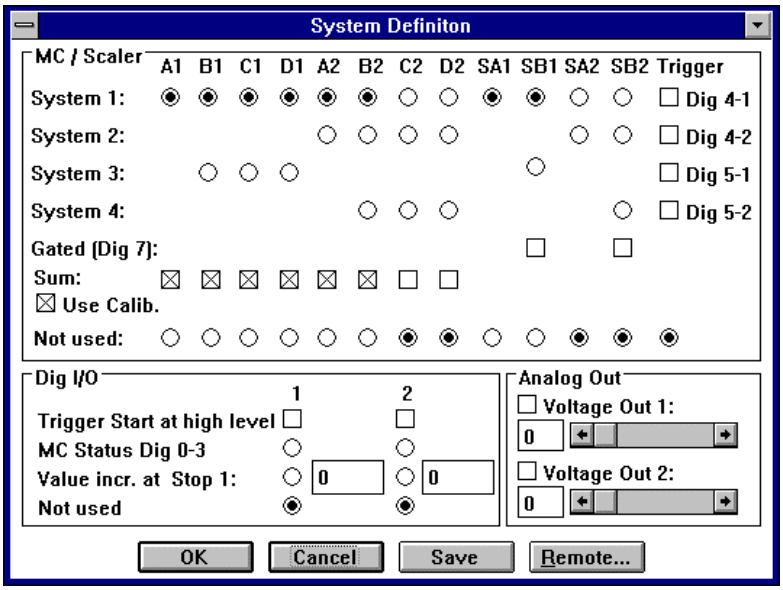


Figure 6.5: System Definition dialog box

In the shown setting one system is defined combining all MCA's of the first card and two of the second, i.e. total six multichannel analyzers are simultaneously started and stopped. The System Definition dialog box contains checkboxes to define which spectra are to be summed up and whether the calibrations are to be used. If 'Use Calibration' is not checked or the first spectrum is not calibrated, simply the channels are added. Otherwise the first spectrum in the set of spectra to be summed up defines the calibration, the other spectra are then shifted to fit to that calibration. If any of the spectra is not calibrated, it is nevertheless summed up and a calibration of energy = channel assumed. It is therefore necessary to check if all spectra are reasonably calibrated.

6.2. MCDWIN program with 2 MCD4LAP cards

The same MCDWIN program communicates also with the W9LAP server via a DLL named D9LAP.DLL. During a measurement the sum spectrum is updated online. The following figure shows MCDWIN with 6 spectra and the sum spectrum. Six detectors are in this case combined to form a „super-detector“ of six-fold size.

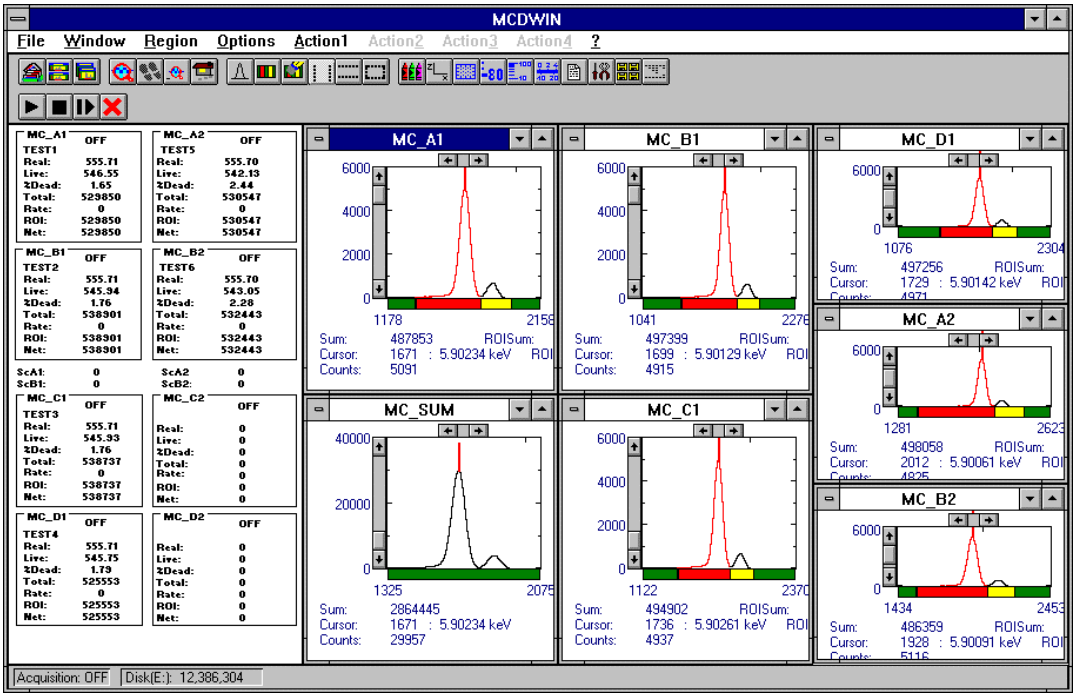


Figure 6.6: MCDWIN with 2 MCD4LAP cards and sum spectrum

7. MCD4LAP Programming

7.1. Register Specification

The MCD4LAP is controlled via input and output to some I/O port registers. The base address is defined by the rotary switch setting (ref. chapter Hardware Installation; standard: 320h).

Address BASE +	Width [bits]	Write Operation	Read Operation
0	16	CONTROL	CONTROL
1	8	POLARITY 0	not used
2	16	POLARITY 1	RAM DATA HIGHER WORD
3	8	RAM ADDRESS [9..16]	not used
4	16	RAM DATA	RAM DATA LOWER WORD
5	8	DAC DATA	not used
6	16	SCALER & I/O CONTROL	SCALER & I/O CONTROL
7	8	not used	not used
8	16	SCALER A PRESET	SCALER COUNT

Figure 7.1: Register Overview

Control register
Base + 0

The control register is accessed by a 16 bit output or input to the base address + 0. The bits in the control register are defined and used as follows:

Bit	Name (Write)	Name (Read)	Meaning
0	ARES	ARES	Reset. By reading a 1 in this bit a state after power on is indicated. The card and scalers must then be reset by writing a 1, and afterwards a zero must be written to return to normal operation.
1	WMOD	WMODRDY	Write Mode.
2	- - -	DAV	RAM Output Data Valid.
3	PREOR	PREOR	Preset Wired OR enabled.
4	ENC 0	ENC 0	Enable Converter MC_A
5	ENC 1	ENC 1	Enable Converter MC_B
6	ENC 2	ENC 2	Enable Converter MC_C
7	ENC 3	ENC 3	Enable Converter MC_D
8	PREREAL 0	PREREAL 0	Preset Real-time 0
9	PREREAL 1	PREREAL 1	Preset Real-time 1
10	PREREAL 2	PREREAL 2	Preset Real-time 2
11	PREREAL 3	PREREAL 3	Preset Real-time 3
12	PRELIVE 0	PRELIVE 0	Preset Live Time 0
13	PRELIVE 1	PRELIVE 1	Preset Live Time 1
14	PRELIVE 2	PRELIVE 2	Preset Live Time 2
15	PRELIVE 3	PRELIVE 3	Preset Live Time 3

Figure 7.2: Control register

Polarity 0 register
Base + 1

The Polarity 0 register is accessed by an 8 bit output to the base address + 1. The bits in the Polarity 0 register are defined and used as follows:

Bit	Name (Write)	Meaning	
0	PDRY 0	Polarity DRDY 0:	0 = act. high; 1 = act. low
1	PDRY 1	Polarity DRDY 1:	0 = act. high; 1 = act. low
2	PDRY 2	Polarity DRDY 2:	0 = act. high; 1 = act. low
3	PDRY 3	Polarity DRDY 3:	0 = act. high; 1 = act. low
4	PDAC 0	Polarity DAC 0:	0 = act. high; 1 = act. low
5	PDAC 1	Polarity DAC 1:	0 = act. high; 1 = act. low
6	PDAC 2	Polarity DAC 2:	0 = act. high; 1 = act. low
7	PDAC 3	Polarity DAC 3:	0 = act. high; 1 = act. low

Figure 7.3: Polarity 0 register

Polarity 1 register
Base + 2

The Polarity 1 register is accessed by a 16 bit output to the base address + 2. The bits in the Polarity 1 register are defined and used as follows:

Bit	Name (Write)	Meaning	
0	PENC 0	Polarity ENC 0:	0 = act. high; 1 = act. low
1	PENC 1	Polarity ENC 1:	0 = act. high; 1 = act. low
2	PENC 2	Polarity ENC 2:	0 = act. high; 1 = act. low
3	PENC 3	Polarity ENC 3:	0 = act. high; 1 = act. low
4	PENB 0	Polarity ENB 0:	0 = act. high; 1 = act. low
5	PENB 1	Polarity ENB 1:	0 = act. high; 1 = act. low
6	PENB 2	Polarity ENB 2:	0 = act. high; 1 = act. low
7	PENB 3	Polarity ENB 3:	0 = act. high; 1 = act. low
8	PDED 0	Polarity DEAD 0:	0 = act. high; 1 = act. low
9	PDED 1	Polarity DEAD 1:	0 = act. high; 1 = act. low
10	PDED 2	Polarity DEAD 2:	0 = act. high; 1 = act. low
11	PDED 3	Polarity DEAD 3:	0 = act. high; 1 = act. low

Figure 7.4: Polarity 1 register

READ RAM Data Higher Word
Base + 2

RAM data are read from this port address; first the higher, then the lower word must be read (from base + 4)

RAM Address [9..16]
Base + 3

The RAM address for a read or write operation is specified by an 8 bit output of bits 9..16 of the RAM address here.

RAM Data
Base + 4

RAM data are read from this port address; first the higher word must be read from base+2, then the lower word must be read here.

RAM data can be written to this port address; first the lower, then the higher word has to be written.

DAC Data
Base + 5

The DAC voltage is set by an 8 bit output. Bit 13 in the Scaler and I/O Control register must be set to enable the DAC.

Scaler and I/O Control
Base + 6

The Scaler and I/O control register is accessed by a 16 bit output or input to the base address + 6. The bits in this register are defined and used as follows:

Bit	Name (Write)	Name (Read)	Meaning	
0	DIO 0	DIO 0	Digital I/O Port 0	Default value is 1
1	DIO 1	DIO 1	Digital I/O Port 1	Default value is 1
2	DIO 2	DIO 2	Digital I/O Port 2	Default value is 1
3	DIO 3	DIO 3	Digital I/O Port 3	Default value is 1
4	DIO 4	DIO 4	Digital I/O Port 4	Default value is 1
5	DIO 5	DIO 5	Digital I/O Port 5	Default value is 1.
6	DIO 6	DIO 6	Digital I/O Port 6	Default value is 1.
7	DIO 7	DIO 7	Digital I/O Port 7	Default value is 1.
8	READ	READ	Read Scaler Count into Register.	
9	LOAD	LOAD	Load Preset Data into Scaler A.	
10	PRESET SC_A	PRESET SC_A	Enable Preset from Scaler A. Bit 3 in the Control register must also be set.	
11	ENBA	ENBA	Enable Scaler A.	
12	ENBB	ENBB	Enable Scaler B.	
13	ENDAC	ENDAC	Enable DAC.	
14	SCB14	SCB14	Bits 14 and 15 define the use of Scaler B:	
15	SCB15	SCB15		

Figure 7.5: Scaler & I/O Control register

Bit 15	Bit 14	Scaler B
0	0	ENCA counts with MC_A.
0	1	ENCB counts with MC_B.
1	0	gated from Dig Input 7.
1	1	Reset.

Write Scaler A Preset
Read Scaler Data
Base + 8

The Scaler A (preset-) value can be set by a 16 bit output to this port address; first the lower, then the higher word must be written.

The data of Scalers A and B can be read: first the higher, then the lower word.

The procedure for writing data to RAM is as follows:

1) Set WMOD = 1
2) Wait for WMODRDY==1
3) Set RAM Address: Write bits 9...16 of the RAM address to base+3.
4) Write RAM data: first lower word, then higher word to base+4. 256 channels can be written without setting a new address.
5) Eventually continue at step 3)
6) Set WMOD = 0

The procedure for reading data from RAM is as follows:

1) Verify that WMOD == 0.
2) Set RAM Address: Write bits 9...16 of the RAM address to base+3.
3) Wait for DAV == 1.
4) Read RAM data: first higher word from base+2, then lower word from base+4. 256 channels can be read without setting a new address.
5) Eventually continue at step 2)

The procedure for reading Scaler Data is as follows:

1) Set READ = 1 (Scaler Control register bit 8)
2) Set READ = 0
3) Read Scaler A higher word, read Scaler A lower word, read Scaler B higher word, read Scaler B lower word.

The procedure for loading a value into scaler A is as follows. If the Preset is enabled, the measurement is stopped when the scaler would overflow.

1) write lower and then higher word to base+8.
2) Set LOAD = 1 (Scaler Control register bit 9)
3) Set LOAD = 0.

7.2. The Subroutines for controlling the MCD4LAP

In the following the low level subroutines used by the MCD4LAP server program (W4LAP.EXE) for controlling the MCD4LAP are listed and commented. The program language used is C

(Microsoft C; for use with Turbo-C replace:

```
outp   with    outportb
outpw  with    outport
inp    with    inportb
inpw   with    inport).
```

Variables and constants, if not clear from the context, are discussed when they appear first time.

cardinit

The cardinit routine defines the base address, initializes the MCD4LAP and performs a basic test if it is present and OK. It returns TRUE if a MCDLAP is present and working, otherwise FALSE.

```
int cardinit()
{
  int i,k,rn,rn0;
  int ret=1;
  int nDev;
  int running=0;
  FILE *f;
  char buf[80];

  if (demomod) {
    for (i=0; i<4; i++) Status[i].started = OFF;
    return FALSE;
  }
  creg = inpw(base);
  if (creg & 0x01)                  // Power On Zustand!
    cardreset();
  if (!readmem(0,0)) { ret=0; goto out; }
  creg = inpw(base);
  jminDev = 4;
  jmaxDev = 0;
  Status[0].started = (creg & 0x010) ? 1 : 0;
  if (Status[0].started) {
    running=1;
    jminDev=0;
    jmaxDev=1;
    Setting[0].rtprena = (creg & 0x0100) ? 1 : 0;
    Setting[0].ltprena = (creg & 0x1000) ? 1 : 0;
  }
  Status[1].started = (creg & 0x20) ? 1 : 0;
  if (Status[1].started) {
    running=1;
    if (jminDev==4) jminDev=1;
    jmaxDev=2;
    Setting[1].rtprena = (creg & 0x0200) ? 1 : 0;
    Setting[1].ltprena = (creg & 0x2000) ? 1 : 0;
  }
  Status[2].started = (creg & 0x40) ? 1 : 0;
  if (Status[2].started) {
    running=1;
    if (jminDev==4) jminDev=2;
    jmaxDev=3;
    Setting[2].rtprena = (creg & 0x0400) ? 1 : 0;
    Setting[2].ltprena = (creg & 0x4000) ? 1 : 0;
  }
  Status[3].started = (creg & 0x80) ? 1 : 0;
  if (Status[3].started) {
    running=1;
    if (jminDev==4) jminDev=3;
    jmaxDev=4;
    Setting[3].rtprena = (creg & 0x0800) ? 1 : 0;
    Setting[3].ltprena = (creg & 0x8000) ? 1 : 0;
  }
  if (running) {
    if (f = fopen("W4LAP.STS", "r+t")) {
      freadstr(f,buf);
      for (i=0; i<4; i++) {
        fscanf(f,"%lu %lu", rtimoff+i, ltimoff+i);
        if (Status[i].started)
          lstrcpy((LPSTR)Data[0].comment0, (LPSTR)buf);
      }
      fclose(f);
    }
  }
  for (nDev=0; nDev<4; nDev++) {
    for (i=0; i<16384; i+=256)
        readmem(nDev,i);
  }
```

```
out:
  return ret;
}
```

Variable	Meaning
base	An unsigned integer variable which specifies the i/o port base address.
base1	base+1
... base8	... base+8
creg	control register
Status	An array of 4 structures describing the Status of each MCA:

```
typedef struct{
  int started;
  double realtime;
  double totalsum;
  double roisum;
  double totalrate;

  double nettosum;
  double livetime;
  double deadtime;
  unsigned long maxval;
} ACQSTATUS;
```

Setting An array of 4 structures describing the settings of each MCA:

```
typedef struct{
  long range;
  int rtprena;
  int roiprena;
  long roimin;
  long roimax;
  double roipreset;
  double rtpreset;
  int savedata;
  int fmt;
  int autoinc;
  int orpreset;      //OrPreset flag (only
                     //Setting[0] evaluated)
  int ltprena;
  int outlevel;      //DAC out only Setting[0]
  int dacena;        //DAC enable only
                     //Setting[0]
  int scalprena;     //Scaler A Preset enable
                     //only Setting[0]
  int digstsena;     //DIG I/O Status/Trigger
                     //enable only Setting[0]
  int digsmplena;    //DIG I/O Samplchanger
                     //enable only Setting[0]
  int digsmplval;    //DIG I/O Samplchanger
                     //value only Setting[0]
  double ltpreset;
  int nregions;
  int caluse;
  double scalpreset;
  int active;
} ACQSETTING;
```

cardreset

The cardreset routine resets the MCD4LAP.

```
int cardreset()
{
  creg = inpw(base);
  creg |= 0x01;
```

```
  outpw(base,creg);          // ARES=1  Reset Scaler...
  creg = inpw(base);
  if (!creg & 0x01) return FALSE; // ARES must be ON now
  creg &= 0xFFFE;
  outpw(base,creg);     // ARES=0  Back to Regular state
  creg = inpw(base);
  if (creg & 0x01) return FALSE;  // must be OFF now
  cardset();
  return (TRUE);
}
```

cardset

The cardset routine initializes the MCD4LAP parameters.

```
void cardset()
{
  outp(base1,(unsigned char)pol0);
  outpw(base2,pol1);
  creg = inpw(base);
  if (Setting[0].orpreset) creg |= 0x0008;
  else creg &= 0xFFF7;
  if (Setting[0].rtprena) creg |= 0x0100;
  else creg &= 0xFEFF;
  if (Setting[1].rtprena) creg |= 0x0200;
  else creg &= 0xFDFF;
  if (Setting[2].rtprena) creg |= 0x0400;
  else creg &= 0xFBFF;
  if (Setting[3].rtprena) creg |= 0x0800;
  else creg &= 0xF7FF;
  if (Setting[0].ltprena) creg |= 0x1000;
  else creg &= 0xEFFF;
  if (Setting[1].ltprena) creg |= 0x2000;
  else creg &= 0xDFFF;
  if (Setting[2].ltprena) creg |= 0x4000;
  else creg &= 0xBFFF;
  if (Setting[3].ltprena) creg |= 0x8000;
  else creg &= 0x7FFF;
  outpw(base,creg);               // Set Preset Bits
}

int freadstr(FILE *stream, char *buff)
{
  int i=0,ic;

  while ((ic=getc(stream)) != 10) {
    if (ic == EOF) {
      buff[i]='\0';
      return 1;
    }
    buff[i]=(char)ic;
    i++;
  }
  buff[i]='\0';
  return 0;
}
```

readmem

The readmem routine reads a block of 256 channels.

```
int readmem(int nDev, unsigned int adr)
{
  int i,j;
  unsigned char badr;
  unsigned *pt;
  unsigned char *p;
  unsigned long val;
  unsigned *pw;
```

```
  if (demomod) return FALSE;
  pw = (unsigned *)(&val);
  creg = inpw(base);
  if (creg & 0x02) {              /* WMODRDY != 0 ? */
    creg = (creg & 0xfffd);       /* WMOD = 0       */
    outpw(base, creg);
    for (i=0; i<MAXLOOP; i++) {
     creg = inpw(base);
     if (!(creg & 0x02)) break;  /* WMODRDY == 0 ? */
    }
    if (i==MAXLOOP) return FALSE;
  }
  adr &= 0xFF00;
  p = (unsigned char *)(&adr);
  badr = p[1] | ((unsigned char)nDev << 6);
  outp(base3, badr);

  for (j=0; j<256; j++) {
    if (!(creg & 0x04)) {           /* wait for DAV */
      for (i=0; i<MAXLOOP; i++) {
        creg = inpw(base);
         if (creg & 0x04) break;    /* DAV == 1 ?   */
      }
      if (i==MAXLOOP) return FALSE;
    }
    pw[1] = inpw(base2);
    pw[0] = inpw(base4);
    Data[nDev].s0[adr++] = val;
  }
  return TRUE;
}
```

Variable	Meaning
Data	A array of 4 structures containing the pointers to the data of each MCA.

```
typedef struct{
  unsigned long huge *s0;       // The spectrum
  unsigned long far *region;          // The ROI boundaries
  unsigned char far *comment0;  // The comment strings
  double far *cnt;              // Scalers, rates
} ACQDATA;
```

writemem

The writemem routine writes a block of 256 channels.

```
int writemem(int nDev, unsigned int adr)
{
  int i,j;
  unsigned char badr;
  unsigned int hival, loval;
  unsigned int huge *pt;
  unsigned char *p;

  if (demomod) return 0;
  creg = inpw(base);
  creg = (creg | 0x02);           /* WMOD = 1 */
  outpw(base, creg);
  creg = inpw(base);
  if (!(creg & 0x02)) {           /* WMODRDY != 1 ? */
    creg = (creg | 0x02);         /* WMOD = 1 */
    outpw(base, creg);
    for (i=0; i<MAXLOOP; i++) {
     creg = inpw(base);
     if (creg & 0x02) break;      /* WMODRDY == 1 ? */
    }
```

```
    if (i==MAXLOOP) return FALSE;
  }
  adr &= 0xFF00;
  p = (unsigned char *)&adr;
  badr = p[1] | ((unsigned char)nDev << 6);
  outp(base3, badr);
  pt = (unsigned huge *)(&Data[nDev].s0[adr]);

  for (i=0; i<256; i++) {
    outpw(base4, *pt++);
    outpw(base4, *pt++);
  }
  creg = (creg & 0xfffd);      /* WMOD = 0 */
  outpw(base, creg);
  return TRUE;
}
```

read_scaler

The read_scaler routine reads the scaler contents.

```
unsigned long read_scaler()
{
  unsigned int ioreg, bval;
  unsigned long val, valA;
  unsigned int *pv;
  long newclock;
  double lastsc, rate, dt;

  pv =  (unsigned *)&val;
  if (demomod) return 0L;
  ioreg = inpw(base6);
  if (Setting[0].digsmplena) {
    bval = Setting[0].digsmplval & 0x00FF;
    ioreg &= 0xFF00;
    ioreg |= bval;
  }
  else if (Setting[0].digstsena)
    ioreg |= 0x0070;
  else
    ioreg |= 0x007F;

  ioreg |= 0x100;          // READ = 1
  outpw(base6, ioreg);
  ioreg &= 0xFEFF;         // READ = 0
  outpw(base6, ioreg);
  pv[1] = inpw(base8);     // Scaler A high word
  pv[0] = inpw(base8);     // Scaler A low word
  newclock = clock();
  if (val < oldscA) scAovl++;
  oldscA = val;
  valA = val;

  if (Setting[0].scalprena) valA += (unsigned
long)Setting[0].scalpreset+1;
  lastsc = Data[0].cnt[CN_SCALERA];
  Data[0].cnt[CN_SCALERA] = valA;
  if (scAovl) Data[0].cnt[CN_SCALERA] += scAovl * 4294967296.;
  dt = (double)(newclock - oldclock);
  if (dt > 0.) {
    rate = (Data[0].cnt[CN_SCALERA] - lastsc) / dt;
    Data[0].cnt[CN_SCARATE] = rate * CLK_TCK;
  }

  pv[1] = inpw(base8);     // Scaler B high word
  pv[0] = inpw(base8);     // Scaler B low word
  if (val < oldscB) scBovl++;
  oldscB = val;
```

```
  lastsc = Data[0].cnt[CN_SCALERB];
  Data[0].cnt[CN_SCALERB]=val;
  if (scBovl) Data[0].cnt[CN_SCALERB] += scBovl * 4294967296.;
  if (dt > 0.) {
    rate = (Data[0].cnt[CN_SCALERB] - lastsc) / dt;
    Data[0].cnt[CN_SCBRATE] = rate * CLK_TCK;
  }
  oldclock = newclock;
  return valA;
}
```

set_scalerA

The set_scalerA routine sets the value of scaler A.

```
int set_scalerA(unsigned long val)
{
  unsigned int ioreg;
  unsigned int *pv;
  pv = (unsigned int *)&val;
  if (demomod) return 0;
  ioreg = inpw(base6);
  ioreg &= 0xF7FF;          // ENBA = 0
  outpw(base6, ioreg);
  outpw(base8, pv[0]);      // Set low word
  outpw(base8, pv[1]);      // Set high word
  ioreg |= 0x0200;          // LOAD = 1
  outpw(base6, ioreg);
  ioreg &= 0xFDFF;          // LOAD = 0
  outpw(base6, ioreg);
  if (Def.sys & 0x10) ioreg |= 0x800; // eventually ENBA
            //  = 1
  if (Setting[0].scalprena) ioreg |= 0x0400;
  else ioreg &= 0xFBFF;
  outpw(base6, ioreg);
  oldscA = val;
  read_scaler();
  return 1;
}
```

reset_scalerB

The reset_scalerB routine resets scaler B.

```
int reset_scalerB()
{
  unsigned int ioreg;
  if (demomod) return 0;
  ioreg = inpw(base6);
  ioreg &= 0xEFFF;           // ENBB = 0
  outpw(base6, ioreg);
  ioreg |= 0xC000;           // Reset Scaler B
  outpw(base6, ioreg);
  ioreg &= 0x3FFF;           // ENCA default
  if (Def.sys & 0x20) {      // ScalB in System1
    if ((Def.sys & 0x02) && !(Def.sys & 0x01))
                             // MC_B ON AND MC_A OFF
      ioreg |= 0x4000;       // ENCB
  }
  else if (Def.sys & 0x200)  // ScalB in System 2
    ioreg |= 0x4000;         // ENCB
  else if (Def.sys & 0x2000) // ScalB extern gated
    ioreg |= 0x8000;
  outpw(base6, ioreg);
  if (Def.sys & 0x2220) {
    ioreg |= 0x1000;           // ENBB = 1
    outpw(base6, ioreg);
  }
  scBovl = 0L;
```

```
  oldscB = 0L;
  read_scaler();
  return 1;
}
```

Variable	Meaning
Def	A structure describing the system settings.

```
typedef struct {
  int nDevices;            // always 4 for MCD4LAP
  int nDisplays;           // 0..4 active MCA's
  int nSystems;            // number of systems
  int bRemote;       // remote controled from MCDWIN
  int sys;                 // system definition
} ACQDEF;
```

The meaning of each bit in the system definition word Def.sys is indicated in the Figure 7.6 showing the System Definition dialog box:

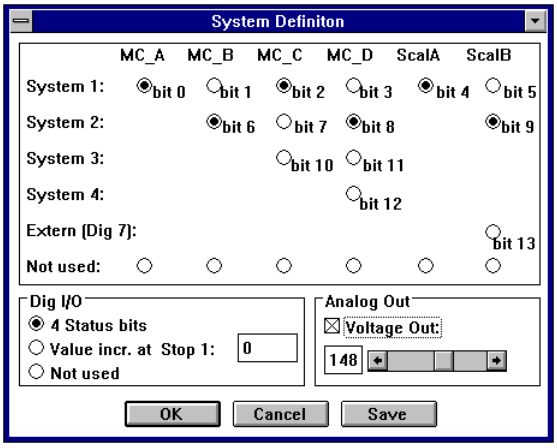


Figure 7.6: System Definition dialog box

setscaler

The setscaler routine sets the logic for enabling the scalers.

```
int setscaler()
{
  unsigned int ioreg;
  if (demomod) return 0;
  ioreg = inpw(base6);
  ioreg &= 0xEFFF;            // ENBB = 0
  outpw(base6, ioreg);
  ioreg &= 0x3FFF;            // ENCA default
  if (Def.sys & 0x20) {       // ScalB in System1
    if ((Def.sys & 0x02) && !(Def.sys & 0x01))
                              // MC_B ON AND MC_A OFF
      ioreg |= 0x4000;        // ENCB
  }
  else if (Def.sys & 0x200)   // ScalB in System 2
    ioreg |= 0x4000;          // ENCB
  else if (Def.sys & 0x2000) // ScalB extern gated
    ioreg |= 0x8000;
  outpw(base6, ioreg);
  if (Def.sys & 0x2220) {
    ioreg |= 0x1000;            // ENBB = 1
    outpw(base6, ioreg);
  }
  ioreg = inpw(base6);
  ioreg &= 0xF7FF;           // ENBA = 0
```

```
  if (Setting[0].scalprena) ioreg |= 0x0400;
  else ioreg &= 0xFBFF;
  if (Def.sys & 0x10) ioreg |= 0x800; // eventually ENBA
                      // = 1
  outpw(base6, ioreg);
  oldclock = clock();
  return 1;
}
```

setioreg

The setioreg routine sets the I/O Control register.

```
int setioreg()
{
  unsigned int ioreg;
  unsigned int bval;
  unsigned char cval=0;
  unsigned int enc;
  if (demomod) return 0;
  ioreg = inpw(base6);
  if (Def.sys & 0x2000) {                // ScalB extern gated
    ioreg |= 0x8000;
    creg = inpw(base);
    enc = (creg & 0x00F0) >> 4;
    if (Def.sys & 0x20) {                // ScalB in System 1
      if (!(enc &
            (Def.sys & 0x0F))) {         // System 1 OFF
        ioreg &= 0xEF7F;                 // ENBB = 0, Dig7 = 0
        outpw(base6, ioreg);
      }
    }
    else if (Def.sys & 0x200) {          // ScalB in System 2
      if (!((enc & 0x0E) &
            ((Def.sys & 0x1C0) >> 5))) { // System 2 OFF
        ioreg &= 0xEF7F;                 // ENBB = 0, Dig7 = 0
        outpw(base6, ioreg);
      }
    }
  }
  ioreg |= 0x007F;
  if (Setting[0].digsmplena) {
    bval = Setting[0].digsmplval & 0x00FF;
    ioreg &= 0xFF00;
    ioreg |= bval;
  }
  else if (Setting[0].digstsena) {
    if (Status[0].started)
      ioreg |= 0x1;
    else
      ioreg &= 0xFFFE;
    if (Status[1].started)
      ioreg |= 0x2;
    else
      ioreg &= 0xFFFD;
    if (Status[2].started)
      ioreg |= 0x4;
    else
      ioreg &= 0xFFFB;
    if (Status[3].started)
      ioreg |= 0x8;
    else
      ioreg &= 0xFFF7;
  }
  if (Setting[0].dacena) {
    if (!(ioreg & 0x2000)) {   // not enabled!
      outp(base5, cval);       // erase first
    }
    ioreg |= 0x2000;           // ENDAC = 1, enabled
```

```
    cval = (unsigned char)Setting[0].outlevel;
    outp(base5, cval);          // set DAC
  }
  else {
    ioreg &= 0xDFFF;            // ENDAC = 0, DAC disabled
  }
  outpw(base6, ioreg);
  return 1;
}
```

clearspec

The clearspec routine clears an MCA.

```
void clearspec(int nDev)
{
  long i;
  int j;
  int oldstate;

  oldstate = Status[nDev].started;

  if (nDev==0) {
    if (Def.sys & 0x10) {
      scAovl=0L;
      if (!Setting[0].scalprena)
        set_scalerA(0L);
            // ScalA ENCA
      else
        set_scalerA(-(long)(Setting[0].scalpreset+1));
    }
    if ((Def.sys & 0x20) &&
            // ScalB System1
        (Def.sys & 0x01)) reset_scalerB();
            //       ENCA
  }
  if (nDev==1) {
    if (((Def.sys & 0x02) &&
            // ScalB System1
         !(Def.sys & 0x01) &&
            // MC_A not active
         (Def.sys && 0x02)) ||
            // but MC_B active
        (Def.sys & 0x200))
            // ScalB System2
      reset_scalerB();
  }
  for (i=0L; i<Setting[nDev].range; i++)       Data[nDev].s0[i]=0L;
  Status[nDev].totalsum=0.;
  Status[nDev].totalrate=0.;
  Status[nDev].roisum=0.;
  Status[nDev].nettosum=0.;
  oldevents[nDev]=0.;
  Status[nDev].realtime=0.;
  Status[nDev].livetime=0.;
  Data[nDev].cnt[CN_TOTALSUM]=0.;
  Data[nDev].cnt[CN_TOTALRATE]=0.;
  Data[nDev].cnt[CN_ROISUM]=0.;
  Data[nDev].cnt[CN_NETTOSUM]=0.;
  Data[nDev].cnt[CN_REALTIME]=0.;
  Data[nDev].cnt[CN_LIVETIME]=0.;

  if (demomod == OFF) {
    for (j=0; j<Setting[nDev].range; j+=256) {
      writemem(nDev,j);
    }
  }
  ShowStatus(nDev);
```

```
}
```

MCA_Newtime

The MCA_Newtime routine sets the timers.

```
void MCA_Newtime(int nDev, unsigned long rtim, unsigned long ltim)
{
  if (demomod == ON)
    return;

  readmem(nDev,0);
  Data[nDev].s0[0] = rtim; // Realtime
  Data[nDev].s0[1] = ltim; // Livetime
  writemem(nDev,0);
}
```

setpreset

The setpreset routine sets real- and livetime presets.

```
int NEAR setpreset(int i)
{
  long rtim, ltim;
  int ret;
    readmem(i,0);
    rtim=Data[i].s0[0];
    ltim=Data[i].s0[1];
    if (Setting[i].rtprena) {
      if(rtim < Setting[i].rtpreset * TIMERTICKS) {
        rtimoff[i] = (long)(Setting[i].rtpreset *
      TIMERTICKS);
        Data[i].s0[0] = rtim - rtimoff[i];
      }
      else {
        ret = MessageBox(hwndLap,
          "Realtime Preset reached! \nNew measurement without clear?",
          szAppName, MB_YESNO);
        if (ret==IDNO) return 1;
        rtimoff[i] = rtim + (long)(Setting[i].rtpreset *
      TIMERTICKS);
        Data[i].s0[0] = - (long)(Setting[i].rtpreset *
      TIMERTICKS);
      }
    }
    else rtimoff[i] = 0L;
    if (Setting[i].ltprena) {
      if(ltim < Setting[i].ltpreset * TIMERTICKS) {
        ltimoff[i] = (long)(Setting[i].ltpreset *
      TIMERTICKS);
        Data[i].s0[1] = ltim - ltimoff[i];
      }
      else {
        ret = MessageBox(hwndLap,
          "Livetime Preset reached! \nNew measurement without clear?",
          szAppName, MB_YESNO);
        if (ret==IDNO) return 1;
        ltimoff[i] = ltim + (long)(Setting[i].ltpreset *
      TIMERTICKS);
        Data[i].s0[1] = - (long)(Setting[i].ltpreset *
      TIMERTICKS);
      }
    }
    else ltimoff[i] = 0L;
    writemem(i,0);
    return 0;
}
```

MCA_Start

The MCA_Start routine starts an acquisition.

```
void MCA_Start(unsigned int enc)
{
  time_t    tnow;
  struct tm *tmnow;
  char buf[80];
  FILE *f;
  int i;

  if (demomod == ON)
    goto demo;
  cardinit();
  if (enc & ENC0) { if (setpreset(0)) return; }
  if (enc & ENC1) { if (setpreset(1)) return; }
  if (enc & ENC2) { if (setpreset(2)) return; }
  if (enc & ENC3) { if (setpreset(3)) return; }
  cardset();
  setscaler();
  creg |= enc;
  outpw(base, creg);
demo:
  time(&tnow);
  tmnow = localtime(&tnow);
  wsprintf(buf, "%.2d/%.2d/%.2d %.2d:%.2d:%.2d ",
      tmnow->tm_mon+1, tmnow->tm_mday, tmnow->tm_year, tmnow->tm_hour,
      tmnow->tm_min, tmnow->tm_sec);
  f = fopen("W4LAP.STS", "w+t");
  fprintf(f,"%s\n", buf);
  for (i=0; i<4; i++)
  fprintf(f,"%lu %lu\n",rtimoff[i], ltimoff[i]);
  fclose(f);
  if (enc & ENC0) {
    Status[0].started=ON;
    lstrcpy((LPSTR)Data[0].comment0, (LPSTR)buf);
  }
  if (enc & ENC1) {
    Status[1].started=ON;
    lstrcpy((LPSTR)Data[1].comment0, (LPSTR)buf);
  }
  if (enc & ENC2) {
    Status[2].started=ON;
    lstrcpy((LPSTR)Data[2].comment0, (LPSTR)buf);
  }
  if (enc & ENC3) {
    Status[3].started=ON;
    lstrcpy((LPSTR)Data[3].comment0, (LPSTR)buf);
  }
  if (Setting[0].digstsena) setioreg();
  jminDev = 4;
  jmaxDev = 0;
  if (Status[0].started) {
    jminDev=0;
    jmaxDev=1;
  }
  if (Status[1].started) {
    if (jminDev==4) jminDev=1;
    jmaxDev=2;
  }
  if (Status[2].started) {
    if (jminDev==4) jminDev=2;
    jmaxDev=3;
  }
  if (Status[3].started) {
    if (jminDev==4) jminDev=3;
    jmaxDev=4;
  }
  StoreStatus();
}
```

MCA_Stop

The MCA_Stop finishes an acquisition.

```
void MCA_Stop(unsigned int enc)
{
  if (!demomod) {
    creg = inpw(base);
    creg &= ~(enc);
    outpw(base, creg);
  }
  if (enc & ENC0) Status[0].started=OFF;
  if (enc & ENC1) Status[1].started=OFF;
  if (enc & ENC2) Status[2].started=OFF;
  if (enc & ENC3) Status[3].started=OFF;
  if (Setting[0].digstsena) setioreg();
  if (enc & ENC0) {
    if (!demomod) {
      readmem(0,0);
      if(Setting[0].rtprena) Data[0].s0[0] +=              rtimoff[0];
      if(Setting[0].ltprena) Data[0].s0[1] +=              ltimoff[0];
      writemem(0,0);
      rtimoff[0]=0L;
      ltimoff[0]=0L;
    }
    totalupdate(0);
    ShowStatus(0);
    if (Setting[0].savedata) dumpdata(0);
  }
  if (enc & ENC1) {
    if (!demomod) {
      readmem(1,0);
      if(Setting[1].rtprena) Data[1].s0[0] +=              rtimoff[1];
      if(Setting[1].ltprena) Data[1].s0[1] +=              ltimoff[1];
      writemem(1,0);
      rtimoff[1]=0L;
      ltimoff[1]=0L;
    }
    totalupdate(1);
    ShowStatus(1);
    if (Setting[1].savedata) dumpdata(1);
  }
  if (enc & ENC2) {
    if (!demomod) {
      readmem(2,0);
      if(Setting[2].rtprena) Data[2].s0[0] +=              rtimoff[2];
      if(Setting[2].ltprena) Data[2].s0[1] +=              ltimoff[2];
      writemem(2,0);
      rtimoff[2]=0L;
      ltimoff[2]=0L;
    }
    totalupdate(2);
    ShowStatus(2);
    if (Setting[2].savedata) dumpdata(2);
  }
  if (enc & ENC3) {
    if (!demomod) {
      readmem(3,0);
      if(Setting[3].rtprena) Data[3].s0[0] +=              rtimoff[3];
      if(Setting[3].ltprena) Data[3].s0[1] +=              ltimoff[3];
      writemem(3,0);
      rtimoff[3]=0L;
      ltimoff[3]=0L;
    }
    totalupdate(3);
    ShowStatus(3);
    if (Setting[3].savedata) dumpdata(3);
  }
}
```

8. Appendix

8.1. Technical Data

8.1.1. Inputs

All inputs accept TTL level signals; low amplitude ≤0.5V; high amplitude ≥4.5V; -0.5 to +5.5V max.

ADC ports

D0...D13 - Active low data inputs from the ADCs, TOFs etc. Input impedance is 1k pull-up.

DRDY - Data Ready signal input indicating that valid data are present at the corresponding ADC port. The polarity is software selectable. Input impedance is a 1k pull-up resistor.

DEAD TIME - Dead-time signal. Input impedance is a 4.7k pull-up resistor.

I/O port

INA - Count input to scaler A. The positive going edge counts. Input impedance is a 1k pull-down resistor.

INB - Count input to scaler B. The positive going edge counts. Input impedance is a 1k pull-down resistor.

8.1.2. Outputs

ADC ports

DENB - TTL level output to enable tri-state data output drivers at the ADCs, TOFs etc. The polarity is software selectable. It is held active during whole operation.

DACC - TTL level handshake signal indicating that ADC input data have been successfully processed. The polarity is software selectable.

I/O port

AOUT - Analog voltage output from the DAC. Minimum is 0V; maximum output voltage is adjustable from 2.55V to ≈7V. Maximum output current is 5 mA.

8.1.3. Bidirectionals

DIG0...DIG7 - Digital I/O port. Open drain outputs with 4.7k pull-ups and 100R series resistors (see **Figure 3.5**). DIG7 can also be used to control or gate counter B.

Preset Bus - Open drain TTL level signal. 10k internal pull-up resistor.

8.1.4. Controls

A rotary switch at the bottom of the card selects the base PC I/O port address (ref. **Figure 2.1**).

A trimmer in the upper right corner of the card allows to adjust the maximum analog output voltage if the jumper beside it is removed. With the jumper ON the maximum analog voltage is 2.55V (ref. **Figure 3.4**).

8.1.5. Connectors

ADC ports

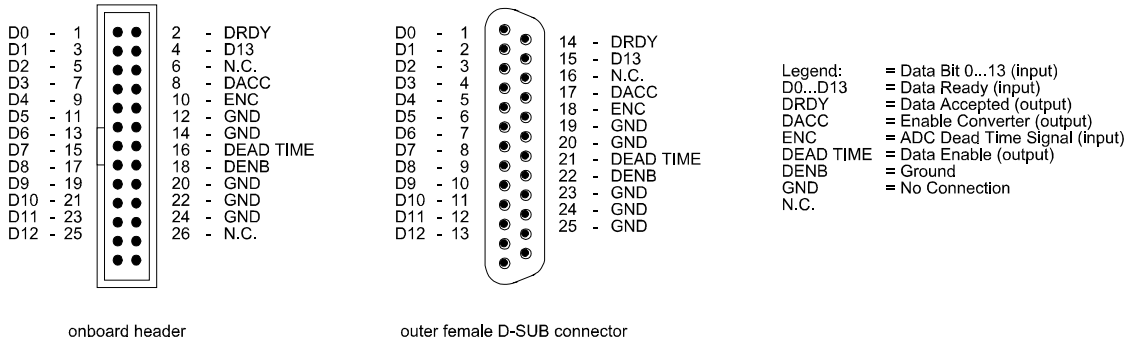


Figure 8.1: ADC port connectors

I/O port

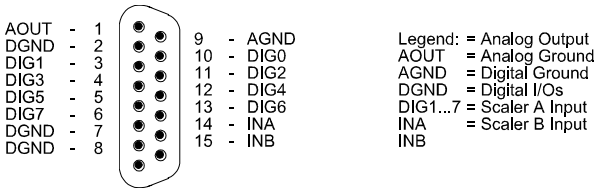


Figure 8.2: I/O port connector

Preset bus

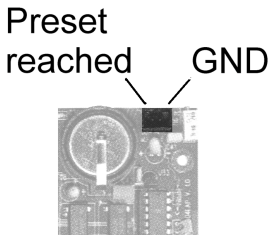


Figure 8.3: Preset bus connector

8.1.6. Performance

MCD

Memory: ..... 256 kByte

Organization: ..... 4 x 16k x 32 bit

Average storage time exclusive arbitration time: ..... ≈ 500 ns

Input data rate (1 input active): ..... > 0.8 MHz

Input data rate (4 inputs active): ..... > 1.5 MHz

Scaler / Counter

Maximum count rate: ..... 50 MHz

Counter width: ..... 32 bit each

DAC

Resolution: ..... 8 bit

Voltage range fixed: ..... 0 ... + 2.55 V

Voltage range adjustable: ..... 0 ... +7 V

Maximum output current: ..... 5 mA

8.1.7. Power Requirements

Computer power supply: ..... +5V, 700 mA, 3.5W typ.

..... 1000 mA, 5W max.

Battery: ..... 3V Lithium type

8.1.8. Physical

Size: 2/3 length low profile PC card, ≈ 22 x 11 cm