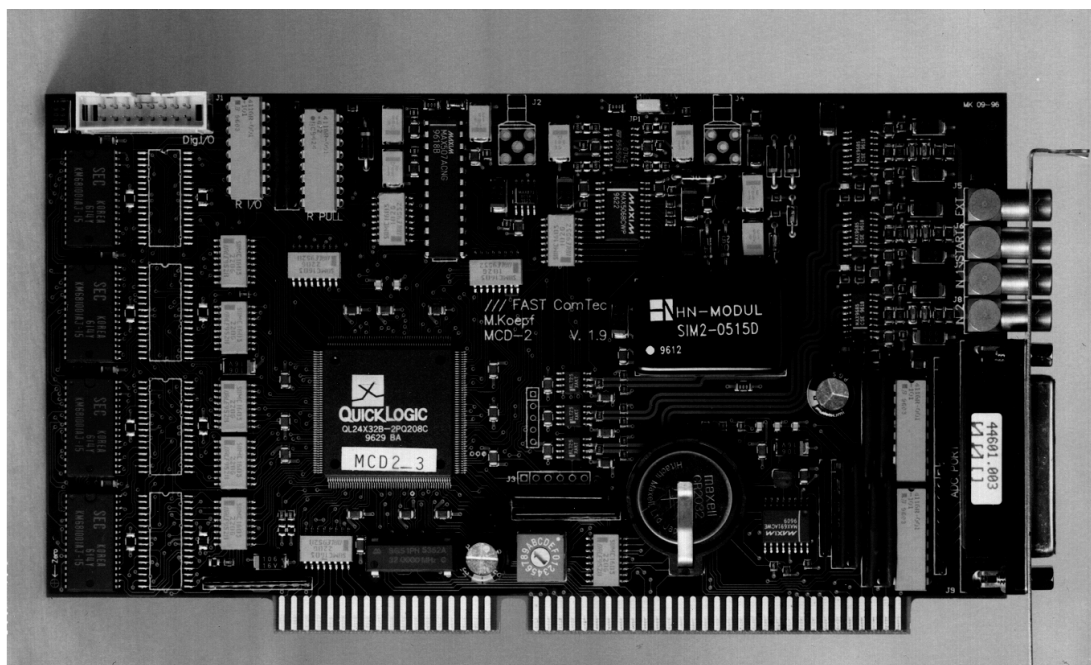


7882

Dual Input Multiscaler / TOF / Photon Counter

User Manual

© copyright FAST ComTec GmbH
Grünwalder Weg 28a, D-82041 Oberhaching
Tel ++49 (0)89 66518050; FAX ++49 (0)89 66518040
Germany



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 © 1995-2003 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 (0)89 66518050, FAX: ++49 89 66518040, e-mail: support@fastcomtec.com, <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

1.	Introduction	1-1
2.	Installation Procedure	2-1
2.1.	Hard- and Software Requirements	2-1
2.2.	Hardware Installation	2-1
2.3.	Software Installation	2-2
2.4.	Setup a basic MCS measurement	2-3
3.	Hardware Description	3-1
3.1.	Overview	3-1
3.2.	Multiscaler	3-1
3.2.1.	Trigger/Start, Count 1 & 2 Inputs	3-1
3.2.2.	Dwell Timer	3-1
3.2.3.	Sweep Counter	3-1
3.3.	Analog Outputs	3-1
3.3.1.	12 Bit / X-Ramp DAC	3-2
3.3.2.	8 Bit DAC	3-2
4.	Windows Server Program	4-1
4.1.	Control Language	4-6
4.2.	Controlling the 7882 Windows Server via DDE	4-12
4.2.1.	Open Conversation	4-12
4.2.2.	DDE Execute	4-12
4.2.3.	DDE Request	4-13
4.2.4.	Close Conversation	4-14
4.3.	Controlling the 7882 Windows Server via DLL	4-16
5.	MCDWIN Program	5-2
5.1.	File Menu	5-2
5.2.	Window Menu	5-4
5.3.	Region Menu	5-4
5.4.	Options Menu	5-7
5.5.	Action Menu	5-14
6.	7882 Programming	6-1
6.1.	Register Specification	6-1
6.2.	The Subroutines for controlling the 7882	6-7
7.	Appendix	7-2
7.1.	Absolute maximum ratings	7-2
7.2.	Connectors	7-2
7.2.1.	MCS Inputs	7-2
7.2.2.	Analog Output	7-3
7.3.	Performance	7-4
7.3.1.	General	7-4
7.3.2.	MCS Mode	7-4
7.4.	Power Requirements	7-4
7.5.	Physical	7-4

Table of Figures

Figure 2.1: 7882 PC card	2-1
Figure 2.2: Table of the base I/O addresses	2-2
Figure 2.3: WINDOWS program item properties	2-2
Figure 2.4: Setup for a basic measurement	2-3
Figure 2.5: Basic spectrum from a 100kHz TTL generator	2-3
Figure 2.6: Manually triggered spectrum	2-4
Figure 3.1: 25pin D-SUB connector	3-1
Figure 4.1: Status display in dual MCS mode (left) or MCA mode (right)	4-1
Figure 4.2: Sample 7882.INI file	4-1
Figure 4.3: Data Operations dialog box	4-2
Figure 4.4: 7882 Settings dialog box	4-2
Figure 4.5: Extended Settings dialog box	4-3
Figure 4.6: System Definition dialog box for a single 7882 card	4-4
Figure 4.7: System Definition dialog box, two 7882 cards	4-4
Figure 4.8: Remote Control dialog box	4-5
Figure 4.9: Opening the DDE conversation with the 7882 server in LabVIEW	4-12
Figure 4.10: Executing a control command from a LabVIEW application	4-13
Figure 4.11: Getting the total number of data with LabVIEW	4-13
Figure 4.12: Getting the data with LabVIEW	4-14
Figure 4.13: Closing the DDE communication in LabVIEW	4-14
Figure 4.14: Control Panel of the demo VI for LabVIEW	4-15
Figure 5.1: MCDWIN main window	5-2
Figure 5.2: File New Display dialog box	5-3
Figure 5.3: ROI Editing dialog box	5-6
Figure 5.4: Single Gaussian Peak Fit	5-6
Figure 5.5: Log file Options for the Single Gaussian Peak Fit	5-7
Figure 5.6: Colors dialog box	5-8
Figure 5.7: Display Options dialog box	5-8
Figure 5.8: Axis Parameter dialog box	5-9
Figure 5.9: :Scale Parameters dialog box	5-9
Figure 5.10: Calibration dialog box	5-10
Figure 5.11: Comments dialog box	5-10
Figure 5.12: Settings dialog box	5-11
Figure 5.13: Extended Settings dialog box	5-11
Figure 5.14: Data Operations dialog box	5-12
Figure 5.15: System Definition dialog box	5-12
Figure 5.16: Tool Bar dialog box	5-13
Figure 6.1: Register Overview	6-1
Figure 6.2: Control Register	6-2
Figure 6.3: ADC PORT CONTROL Register	6-3
Figure 6.4: SPEC CONTROL Register	6-3
Figure 6.5: OFFSET Register	6-4
Figure 6.6: OUT CONTROL Register	6-4
Figure 6.7: DAC CONTROL Register	6-5
Figure 7.1: Analog Output Connector	7-3

1. Introduction

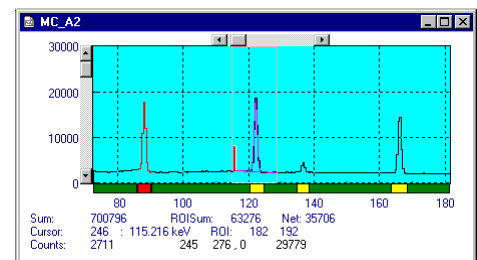
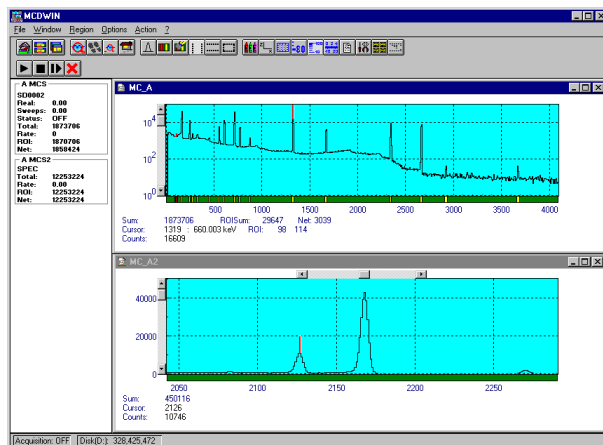
The 7882 is an advanced Dual Input Multiscaler on a $\frac{2}{3}$ length PC-card. The design offers unmatched performance in its price range.

Multiscaling in the 7882 is far advanced and offers improved precision by signal averaging. The two counting inputs allow to acquire - for example - two Mössbauer spectra at a time from a single Mössbauer drive. This is a very economical solution requiring just one Mössbauer system. The very short dwell times starting at 125 ns make the 7882 the ideal choice for many types of spectrometers and LIDAR applications where high count rates of more than 150 MHz are also required.

The large 128k x 32 bit onboard data memory and the high integrated logic enable to accumulate data 100% in the background thus freeing the computer for other applications. Even a hardware reset of the computer does not affect data acquisition.

In addition to the high performance of the hardware the sophisticated WINDOWS based control and analysis software MCDWIN ensures quick learning and easy usage.

Some of MCDWIN's features are high resolution graphics displays with zoom, linear and logarithmic (auto)scaling, grids, ROIs¹, Gaussian fit, calibration using diverse formulas. and FWHM² calculations. Macro generation using the powerful command language allows task oriented batch processing and selfrunning experiments. An IAEA compatible software interface allows to directly use such analysis packages as GANAAS, QXAS, POSFIT or others.



¹ ROI: Region of Interest

² FWHM: Full Width at Half Maximum

2. Installation Procedure

WARNING !!!

The Trigger, Count 1 and Count 2 inputs are very high speed inputs. To obtain this very high speed no ESD protection circuitry is provided on these inputs. Be very careful handling the 7882 card to avoid high voltage on these inputs.

2.1. Hard- and Software Requirements

The 7882 requires an IBM AT or compatible computer with a 386, 486, Pentium or higher processor and an available 16 bit slot. Several 7882 cards might be installed in your computer if you have enough available slots.

A PC with Microsoft Windows 3.1 or Windows 95 installed is required for use of the supplied control and analysis software MCDWIN.

2.2. Hardware Installation

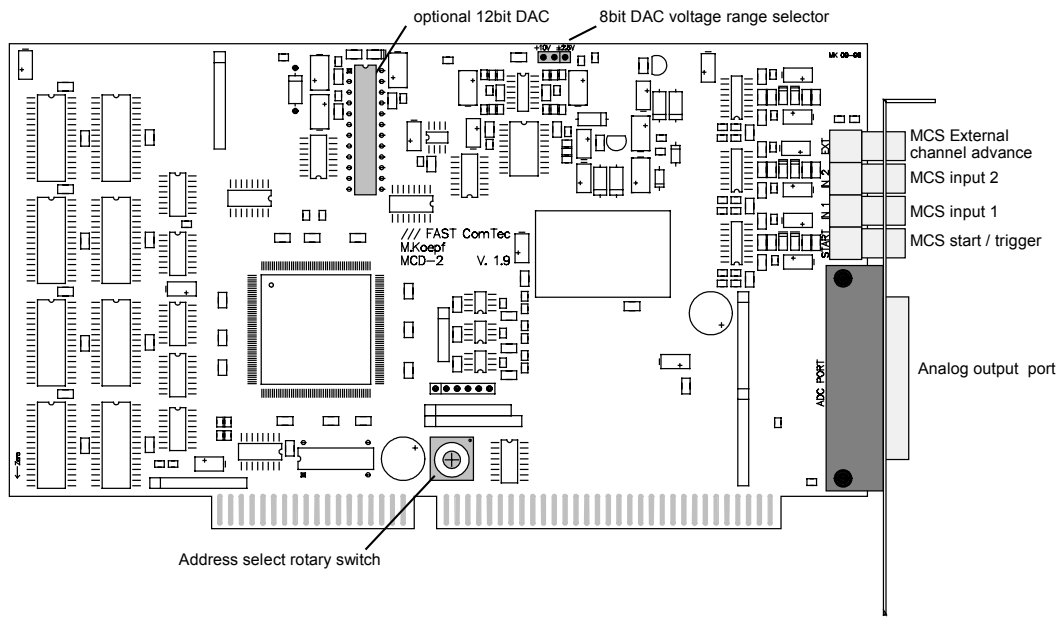


Figure 2.1: 7882 PC card

First you should locate an unused address in the I/O³ address space of your computer. The 7882 has a small rotary switch (ref. Figure 2.1) that determines the base I/O address of the card. The 7882 occupies 12 I/O addresses starting at this base address. The supported base addresses and corresponding switch settings are:

³ I/O: Input / Output

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

Figure 2.2: Table of the base I/O addresses

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

2.3. Software Installation

To install the 7882 software on your hard disk insert the 7882 disk into drive A. Log to drive A: by clicking from the explorer and if you are using any 32 bit Windows (9x, NT, 2000, ME, XP) start the installation program by double clicking the SETUP.EXE. If you are using Windows 3.x change to the WIN3X directory and double click the INSTALL.BAT there.

A directory called C:\7882 is created on the hard disk and all 7882 and MCDWIN files are transferred to this directory. Drive C: is taken as default drive and the 7882 working directory as default directory. It is not mandatory that the 7882 operating software is located in C:\7882. After the installation is completed you may copy the files to any other directory.

For Windows NT/2000/XP it is necessary that you install the device driver FASTMCD.SYS from the WINNT\DRIVER directory. If not already done by the SETUP program, run INSTALL.BAT from this directory and restart the system.

The installation program installs an icon on the desktop for 7882.EXE. 7882.EXE is the 7882 Hardware Server program. This program will automatically call the MCDWIN.EXE program when it is executed. The 7882 Server program controls the 7882 board but provides no graphics display capability by itself. By using the MCDWIN program, the user has complete control of the 7882 along with the MCDWIN display capabilities.

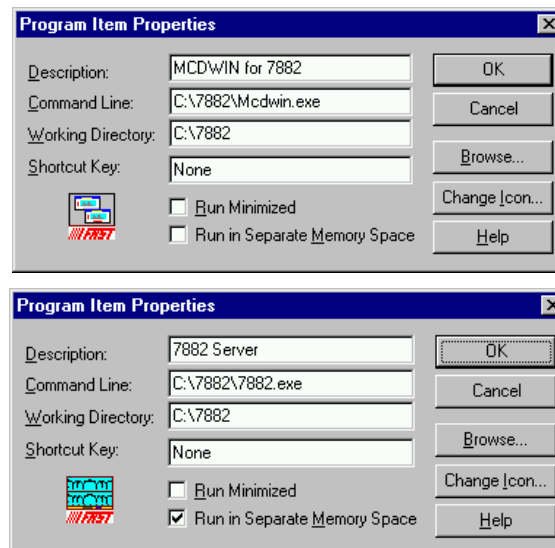


Figure 2.3: WINDOWS program item properties

2.4. Setup a basic MCS measurement

This chapter shows how to setup a basic measurement using a two output TTL frequency generator that might be available in most labs. One output is used to generate the Start/Trigger signal and the other one as count signal connected to Count Input 1.

The 7882 is configured as single MCS, 125 ns dwell time, range 1024, complete sweeps and internal clock. In our example the available TTL signal generator had a 1 kHz (connected to Start/Trigger) and a 100 kHz / 10 μ s (connected to MCS Input 1) output. With the 50 Ω internal termination of the 7882 inputs the resulting pulse height was only 1.5 V thus an input threshold level of 700 mV was chosen (ref. Figure 2.4).

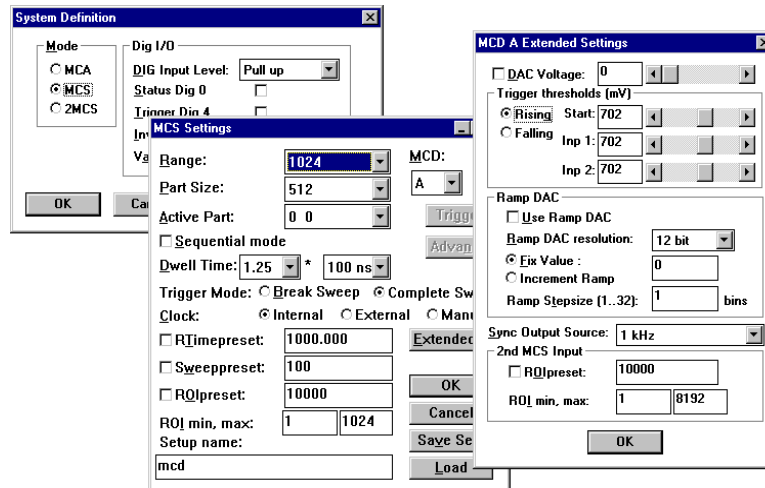


Figure 2.4: Setup for a basic measurement

The obtained spectrum is shown in Figure 2.5. With the 125 ns dwell time and the 100 kHz / 10 μ s pulse repetition there are counts in every 80th channel.

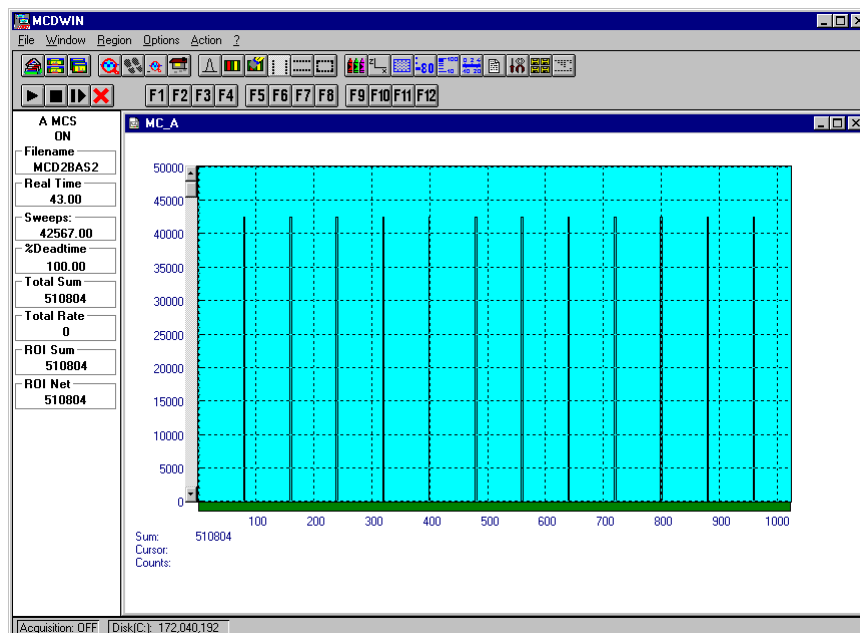


Figure 2.5: Basic spectrum from a 100kHz TTL generator

If there is only one single signal available connect only the Count Input 1 and use a manual trigger. Since the triggers will be asynchronously to the input pulses the spectrum will look like noise after some sweeps (ref. Figure 2.6). To ease the use of the manual trigger configure the

function key F1 as manual Trigger button (refer the MCDWIN menu Options - Function Keys and the online help on "Options - Tool Bar..." and "Use the control language"). The control language command is "SWEEP".

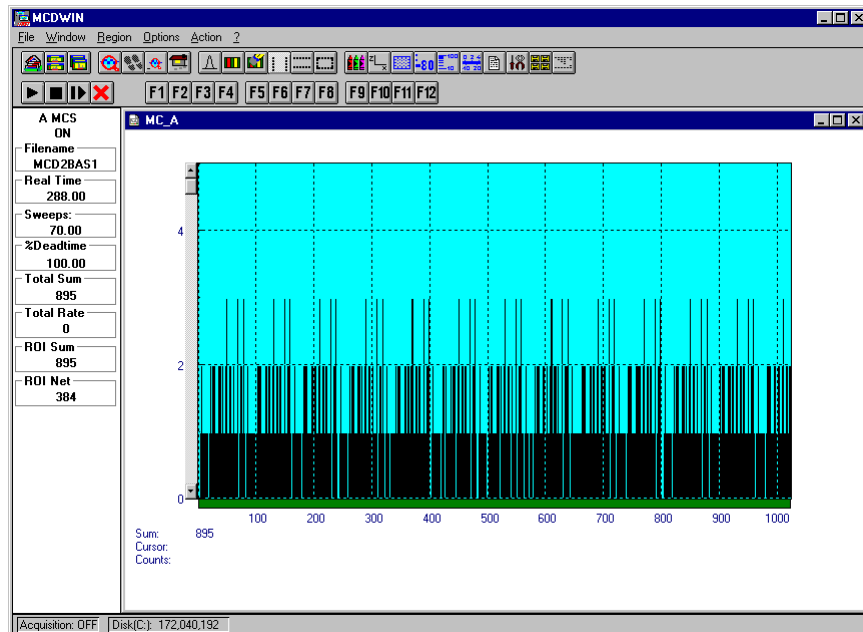


Figure 2.6: Manually triggered spectrum

3. Hardware Description

3.1. Overview

The 7882 is an advanced Dual Input Multiscaler on a $\frac{2}{3}$ length PC-card. Beside the multiscaling inputs it features two analog outputs - one providing the x-ramp - if required - making it a very versatile data analysis and experiment control system.

Due to the 100% background operation capability even a hardware reset of the computer does not affect data acquisition.

3.2. Multiscaler

3.2.1. Trigger/Start, Count 1 & 2 Inputs

All these inputs are terminated with $50\ \Omega$ to ground. The input range is $\pm 5\text{ V}$ for all inputs. The threshold levels of the input discriminators are individually programmable via three onboard 8 bit DACs also in the range of $\pm 5\text{ V}$. The slope of the trigger input is programmable. The count inputs are falling edge sensitive.

3.2.2. Dwell Timer

The dwell time counter is software programmable in steps of

$$N \times 10^M \times 125\text{ ns}$$

where $N = 1 \dots 16$ and $M = 0 \dots 7$. Thus dwell times from 125 ns up to 20 s are supported.

Also an external channel advance input is provided accepting TTL signals.

For test purposes a software / manual channel advance is also possible.

3.2.3. Sweep Counter

The presetable sweep counter is incremented on each start of a new sweep. It is not available in the "Break Sweep" mode of operation.

3.3. Analog Outputs

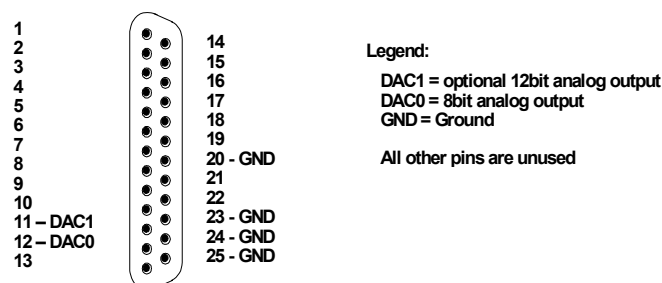


Figure 3.1: 25pin D-SUB connector

3.3.1. 12 Bit / X-Ramp DAC

An optional 12 bit Digital-to-Analog converter provides a programmable output voltage in the range of 0...+10 V.

An internal X-ramp generator allows to alternatively output a X-ramp synchronously to a multiscaling sweep. The lower value of the ramp is programmable. Also the step width is programmable in multiples (1...32) of the bin width. The effective resolution of the DAC can be set to 9, 10, 11 or 12 bit. Thus the X-ramp can be adjusted to different sweep lengths.

3.3.2. 8 Bit DAC

The output voltage range of the 8 bit Digital-to-Analog converter is jumper selectable between ±2.5 V and 0...+10 V.

4. Windows Server Program

The window of the 7882 server program 7882.EXE is shown here. It enables the full control of the 7882 to perform measurements and save data. This program has no own graphic capabilities, 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.

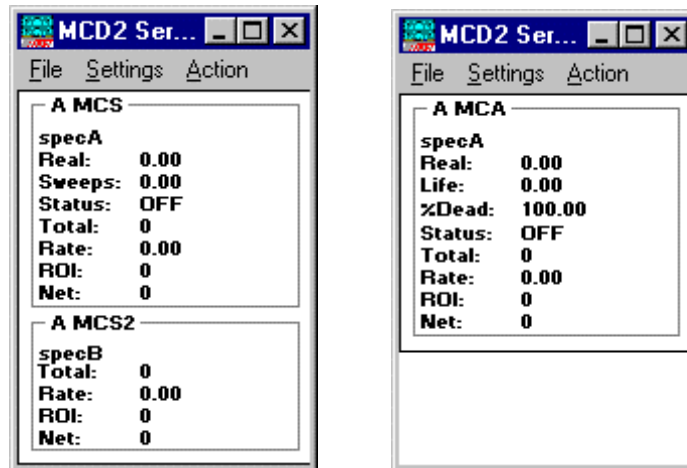


Figure 4.1: Status display in dual MCS mode (left) or MCA mode (right)

At program start the configuration files 7882.INI (contains - for example - the I/O port base address in a format base=320 and the ADC port handshake signal polarities; see Figure 4.2) and MCDA.INF are loaded. Instead of this MCDA.INF file any other setup file can be used if its name - excluding the appendix 'A.INF' - is used as command line parameter (e.g. 7882 TEST to load TESTA.INF) . The server program is normally shown as an icon. After a double click it is opened to show a status window.

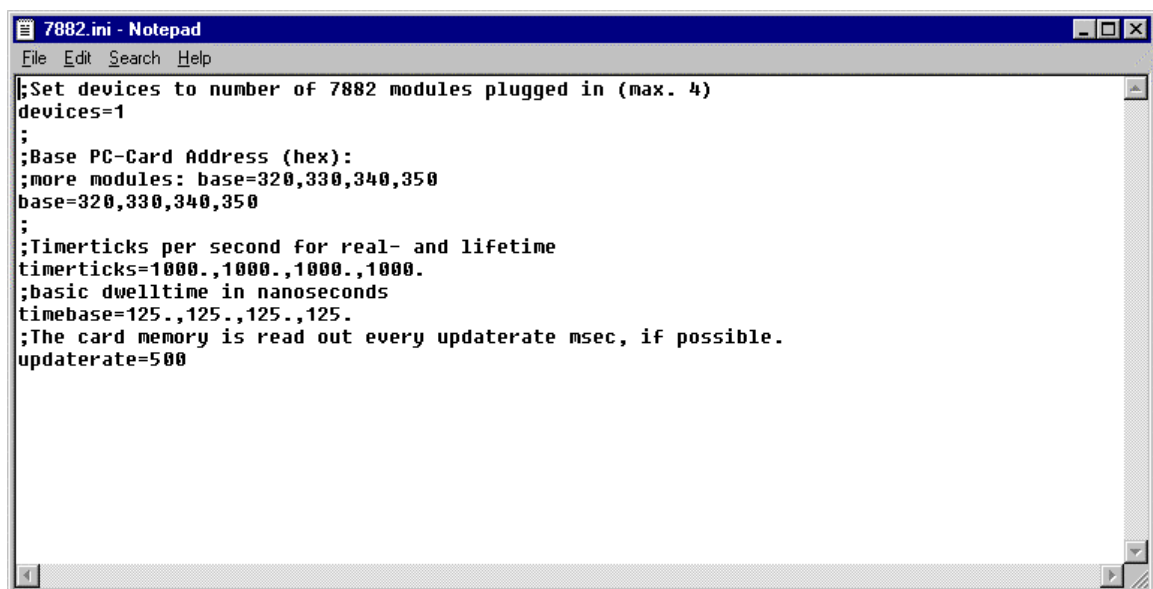


Figure 4.2: Sample 7882.INI file

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.

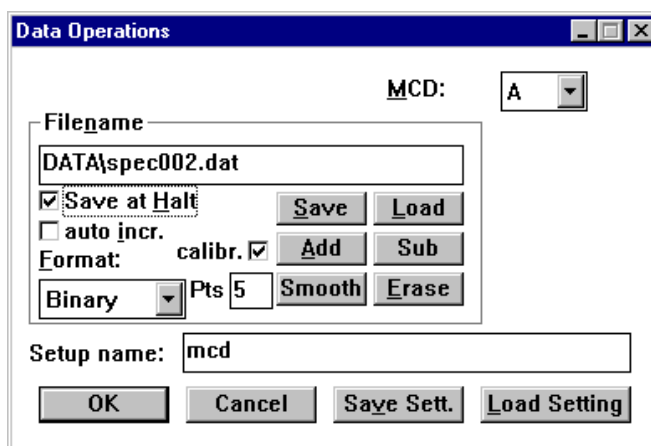


Figure 4.3: Data Operations dialog box

This dialog allows to edit the data settings. Mark the checkbox „**Save at Halt**“ to store a spectrum- and a configuration file at the end of a measurement. The filename can be entered. If the checkbox **auto incr.** is checked, a 3-digit number is appended to the filename that is automatically incremented with each saving. The format of the data file can be ASCII (extension .ASC), binary (.DAT), GANAAS (.SPE) or Dual Binary (.DA2), i.e. both spectra in dual MCS mode in a single data file. The buttons **Save**, **Load**, and **Erase** perform the respective operation. 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 adjusted 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.

Clicking in the Settings menu on MCD... item opens the MCS Settings dialog box. Here parameters like presets, range parameters, dwell time, etc. for the 7882 card can be set.

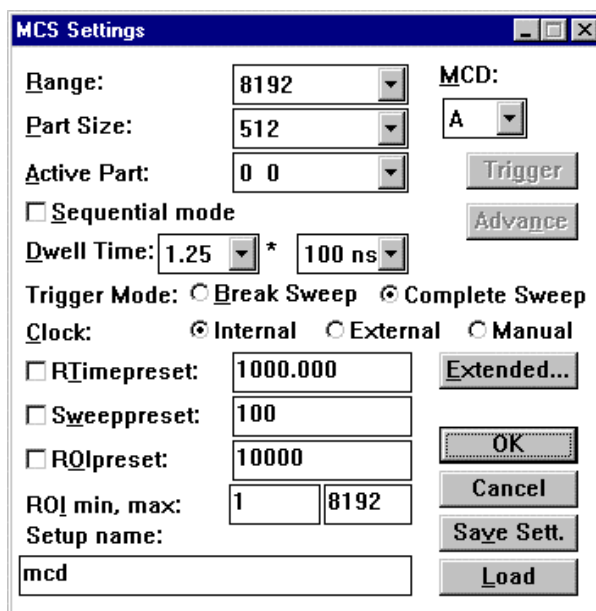


Figure 4.4: 7882 Settings dialog box

In the edit field „**Range**“ the size of the spectrum can be chosen between 256 and 128k. If the checkbox „**ROI preset**“ 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. \geq the lower limit and $<$ the upper limit. Another possibility is to acquire data for a specified amount of sweeps via a „**Sweep preset**“. A measurement will be stopped if the corresponding checkbox is marked. The 7882 has a rather large memory and it is possible to acquire several spectra into separate parts of the memory. The start point of the spectrum in the

memory can be defined using the „**Active Part**“ drop down listbox, the size of the corresponding memory part with the „**Part Size**“. If „**Sequential mode**“ is checked, the label of the „Active Part“ listbox changes to „**Last active part**“. It is then possible to perform a sequential measurement. In this mode it is permanently checked whether the card has stopped due to a preset reached, the active part is incremented and the measurement immediately restarted in the next active memory part. In this mode the range should be equal to the part size and one of the two preset checkboxes for real time and live time or sweeps should be checked.

Extended opens the extended settings dialog box that allows to configure the MCS input thresholds, the analog outputs, the Sync output source and the second MCS input ROI.

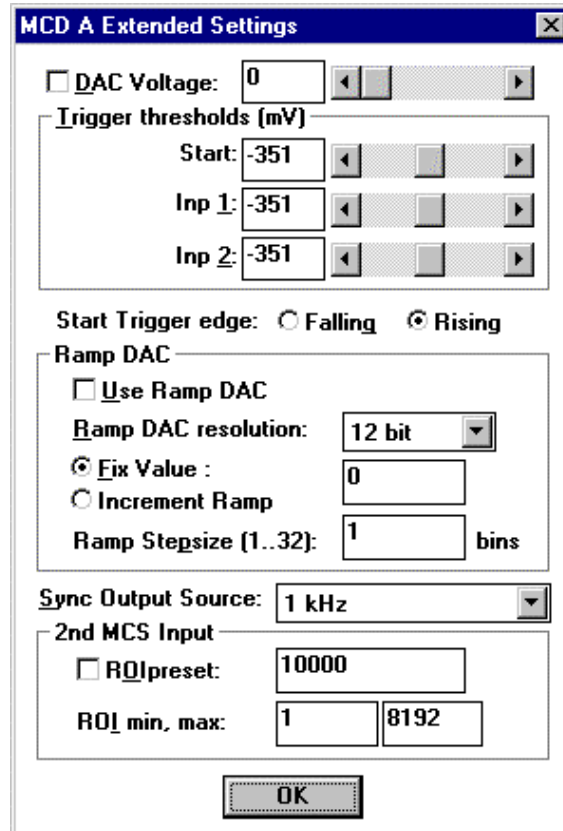


Figure 4.5: Extended Settings dialog box

The 8 bit DAC output can be enabled by checking the box labeled „**DAC Voltage**“. 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. The output voltage range is jumper selectable between ± 2.5 V and 0...+10 V (see 3.5.2).

The „System...“ item in the settings menu opens the System Definition dialog box. Here the configuration of the 7882 for single or dual input MCS mode is made.

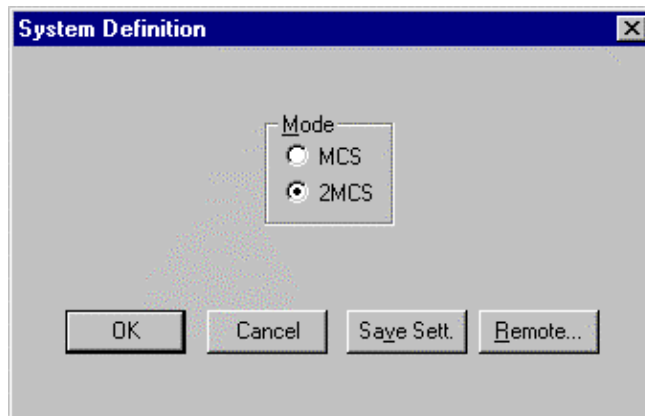


Figure 4.6: System Definition dialog box for a single 7882 card

The digital input / output port can be used either to show the status of the 7882 card in bit 0 (bit 0 set means the 7882 is armed) if the DIG I/O radio button „**Status Dig 0**“ 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 the 7882 is stopped.

Together with using bit 0 as status signal for 7882, it is also possible to use the digital input 4 as an external trigger for starting the system („**DESY control line**“). If the corresponding checkbox is marked, a start command for the system will not immediately start the system. After the start command, the digital input will be permanently polled for its logical level. If the level changes from high to low, any remaining data will be cleared and data accumulation started. Accumulation 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 halt the checking of the digital input.

If more than one 7882 card is used, the system definition dialog box comes up as shown in Figure 4.7. Here the several units can be combined to form up to 4 separate systems that can be started, stopped and erased by one command.

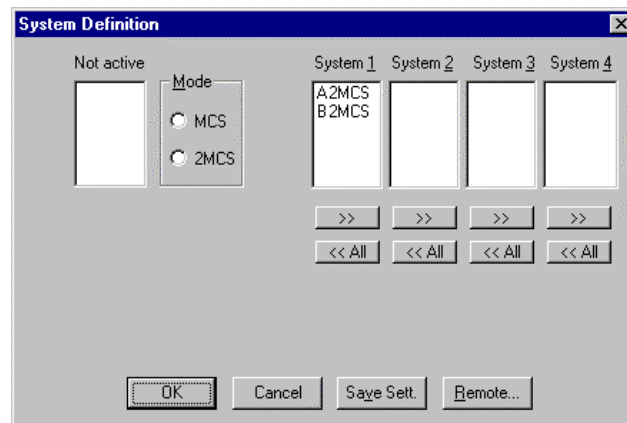


Figure 4.7: System Definition dialog box, two 7882 cards

In the shown setting a single system is formed. The two 7882s A and B are combined, both operate in pulse height analysis mode. System 1 can be started, stopped, erased, and continued with the respective commands in the Action 1 menu. It is also possible for example to form two independent systems 1 and 2: Click on the button labeled **<<All** below the list box „System1“ to remove all units from system 1. They are then shown in the „Not active“ list box. Then select unit A and click on the button labeled **>>** below the „System 1“ list box to include it into system 1 and perform the respective action for unit B and System 2. To switch between modes, the unit must be selected in the „not active“ box and then click on the **MCS** or **MCS2** radio button in the box labeled **Mode**.

Click **OK** to accept all settings. **Cancel** cancels all changes. Clicking „**Save Settings**“ stores all settings in the file **MCDA.INF** in a form:

```
wndwidth=154
wndheight=251
sysdef=0
roi2preset=10000
roi2min=1
roi2max=8192
mcsmode=0
rdacuse=0      ; fix, 12bit, 1 stepsize
rdac=0
qdac=0
thrstart=125
pol1=0
thr1=125
thr2=125
syncsrc=0
diguse=0
digval=0
range=8192
rtpreset=1000
prena=0      ; Preset enable
ltpreset=1000
swpreset=100
roipreset=10000
roimin=1
roimax=8192
mempart=0    ; active:0 size:256
dwelltime=0  ; 2 * 10^2 ns
autoinc=0
datname=specA.asc
savedata=0
fmt=asc
smoothpts=5
```

This file is automatically loaded at the start of the program and the parameters are set. Together with each data file a header file with extension .MCD 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 7882 server program via a serial port.

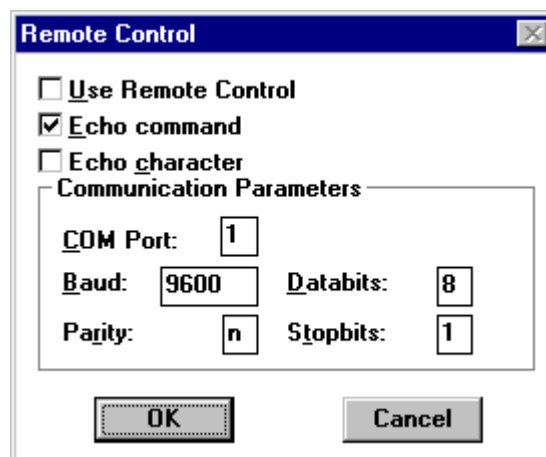


Figure 4.8: Remote Control dialog box

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 7882 server program or MCDWIN with the „Load “ command. Also the configuration files MCDx.INF or the header files with extension .MCD contain such commands to set the parameters. Each command starts at the beginning of a new line with a typical keyword, the case is ignored. Any other characters in a line may contain a value or a comment.

Following methods are available to execute commands:

Load the command file using the Load command in the file menu.

Enable remote mode in the server and send commands via the serial connection. The COMCTL.DLL is necessary which is part of the optional available MCDLAN software.

Open a DDE connection and send the commands via DDE as described in section 4.2. The application name for opening the DDE connection with the standard 7882 server program 7882.EXE is MCD2, the topic is MCD2-. Implemented are the DDE Execute to perform any command, and the DDE Request with items RANGE and DATA.

Send the commands over a TCP/IP net using a remote shell and the optional available MCDLAN software. It is necessary to have a TCP/IP Winsock installed like the Trumpet winsockets and that the remote shell daemon program MCWNET is running. See the readme file on the installation disk.

Send the commands via the DLL interface from LabVIEW, a Visual Basic program or any other application (software including the complete source code of the DLL and examples optional available).

From your own Windows application, register a Windows message and then send the command as can be seen in the DLL source code.

The file MCDA.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=0 (hex)	; Sets system definition word (hexadecimal). Bits 0 and 1 are reserved for unit A, bits 2 and 3 for unit B and so on until bits 15 and 16 for unit H. the bit pattern 00 means the respective unit belongs to System 1, 01 System 2, 10 System 3 and 11 System 4.
roi2preset=10000	; ROI preset value in 2 nd MCS channel
roi2min=1	; ROI in 2 nd MCS channel lower limit (inclusive)
roi2max=4096	; ROI in 2 nd MCS channel upper limit (exclusive)
mcsmode=0	; bit 0: 1=1 channel MCS mode ; bit 1: 1=2 channel MCS mode ; bit 2: 1=Retrigger mode, i.e. the next signal ; breaks the current sweep and starts a new sweep.
rdacuse=1 (hex)	; bit 0: Ramp DAC enable ; bit 1: Ramp DAC increment enable (0=fixed mode) ; bit 2..3: Ramp DAC resolution ; 00 : 12 bit, 01: 11 bit, 10: 10 bit, 11: 9 bit
rdac=0	; Ramp DAC output value

qdac=0 ; bit 0..7: Quad DAC 3 output value (8 bit)
 ; bit 8: qdac enable
 thrstart=-580 ; trigger level for start input in units of 1 mV, i.e. -580 means -580 mV
 pol1=0 ; 0=trigger start input on rising edge, 1=falling edge
 thr1=-580 ; trigger level for input 1 in units of mV (always falling edge)
 thr2=-580 ; trigger level for input 2 in units of mV (always falling edge)
 syncsrc=1 ; bit 0..1: sync output source
 ; 00: 1 kHz, 01: Sweep Start,
 ; 10: End of Sweep, 11: End of time bin
 diguse=0 (hex) ; Usage of DIG I/O (only 1. device stored):
 ; bit 0: DIG I/O pins 0-3 output status of MCD's
 ; bit 1: Invert Polarity
 ; bit 2..5: Input pins 4..7 Trigger System 1..4
 ; bit 6: Output digval and increment digval after stop
 ; bit 7: Input Mode: =0 Resistive, =1 Tristate
 ; bit 8: Input Mode: =0 Pull up, =1 Pull down
 digval=0 ; DIG I/O Output value (only 1. device stored)
 range=4096 ; Sets histogram length
 rtpreset=1000 ; Realtimepreset value (seconds)
 prena=0 (hex) ; bit 0: realtime preset enabled
 ; bit 1: lifetime preset enabled
 ; bit 2: sweep preset enabled
 ; bit 3: ROI preset enabled
 ; bit 4: 2nd MCS input: ROI preset enabled
 ltpreset=1000 ; Lifetime preset value (seconds)
 ltprena=0 ; Lifetime preset enable
 swpreset=1000 ; Sweep preset value
 roipreset=10000 ; ROI preset value
 roimin=1 ; ROI lower limit (inclusive)
 roimax=4096 ; ROI upper limit (exclusive)
 mempart=0 (hex) ; bit 0..7: number of active part
 ; bit 8..10: partsize 0:512, 1:1024,...,7:64k
 dwelltime=1 (hex) ; Dwell time = (N+1) x 10^M x time base (125 ns)
 ; bit 0..3: N
 ; bit 4..7: M
 ; M=8: Extern channel advance
 autoinc=0 ; Enable Auto increment of filename
 datname=SPECA.dat ; Filename
 datname2=SPECB.dat ; Filename for 2nd MCS channel
 savedata=0 ; Save at Halt
 fmt=dat ; Format (ASCII: asc, Binary: dat, GANAAS: spe, Dual Binary: da2)
 smoothpts=5 ; Number of points to average for a smooth operation

A data header file with extension .MCD contains a subset of above parameters and some additional information typical for the special measurement. An example is the file CALIB2.MCD:

```
REPORT-FILE from 11/03/86 08:00:00 written 09/12/96 19:39:00
                                ; the first time is when the measurement was started,
                                ; the 2nd when the data file was written

realtime=4788                  ; real time in seconds
lifetime=4788                  ; life time in seconds
TOTALSUM=0                    ; total sum of counts
ROISUM=0                      ; sum of counts in ROI
NETTOSUM=0                    ; sum in ROI with background subtracted
cmlline0=11/03/86 08:00:00    ; comment lines: the first line always contains the start time
cmlline1=*****CALIBRATION SPECTRA
cmlline2=Cd109, Co57, Ce139, Sn113, Hg203, Sr85, Cs137, Co60
range=4096                    ; subset of parameters as in a MCDx.INF file...
rtpreset=1000
prena=0                      ; Preset enable
ltpreset=1000
swpreset=100
roipreset=10000
roimin=0
roimax=4096
mempart=0                    ; active:0 size:512
dwelltime=0                  ; 125 * 100 ns
autoinc=0
datname=CALIB2.spe
savedata=0
fmt=spe
smoothpts=5                  ; last parameter like in a .INF file
caloff=3.671330               ; calibration parameters: caloff = 0. coefficient
calfact=0.453466              ; 1. coefficient
calfact2=-1.37019e-007        ; 2. coefficient
calfact3=5.369e-011          ; 3. coefficient
calunit=keV                  ; calibration unit
caluse=3                     ; bit 0: Use calibration
                                ; bit 1..2 calibration formula:
                                ; 00=linear (caluse=1), 01=quadratic (caluse=3), 10=cubic (caluse=5)

calch00=186.07                ; calibration points: 0. channel
calvl00=88.034000             ;      0. value
calch01=261.05                ;      1. channel
calvl01=122.061000            ;      1. value...
calch02=357.86
calvl02=165.854000
calch03=607.56
```

```
calvl03=279.197000
calch04=855.83
calvl04=391.688000
calch05=1451.72
calvl05=661.660000
calch06=2932.93
calvl06=1332.500000
calch07=2581.25
calvl07=1173.240000
roi=182 192          ; ROIs set in MCDWIN...
roi=257 268
roi=289 298
roi=352 364
roi=551 559
roi=601 615
roi=849 864
roi=1121 1134
roi=1443 1463
roi=1777 1801
roi=1961 1988
roi=2909 2925
roi=2924 2943
roi=4036 4055
roi=2573 2590
```

The following commands perform actions and therefore usually are not included in a MCDx.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 suspended (see start).
start3         ; Clears and starts system 3. Further execution suspended (see start).
start4         ; Clears and starts system 4. Further execution suspended (see start).
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 prolonged 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 cont).
cont3         ; Continues acquisition of system 3 (see cont).
```

cont4	; Continues acquisition of system 4 (see cont).
savecnf	; Writes the settings into MCDA.INF (,MCDB.INF,...)
MC_A	; Sets actual multichannel analyzer to MC_A for the rest of the ; control file.
MC_B	; Sets actual multichannel analyzer to MC_B for the rest of the ; control file.
MC_C	; Sets actual multichannel analyzer to MC_C for the rest of the ; control file.
MC_D	; Sets actual multichannel analyzer to MC_D for the rest of the ; control file.
savedat	; Saves data of actual multichannel analyzer. An existing file is ; overwritten.
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	; Smooths 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 7882.exe (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.
beep *	; Makes a beep. The character '*' may be replaced with '?', '!' or left ; empty. The corresponding sound is defined in the WIN.INI file in the ; [sounds] section.
delay 4000	; Waits specified time (4000 msec = 4 sec).
pulse 100	; Outputs a pulse of 100 ms duration at dig 3 (pin 11).
waitpin 4000	; Waits 4000 ms for going the level low at dig 7 (pin13). ; After a timeout a Message box warns and waits for pressing OK. ; Can be used for connecting a sample changer.
sweep	; Starts a Sweep (Software-Trigger).
run controlfile	; Runs a sequence of commands stored in control file. This command ; cannot be nested, i.e. it is not possible to execute a run command ; from the control file 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.
exec program	; Executes a Windows program or .PIF file. ; Example: exec notepad test.ctf ; opens the notepad editor and loads test.ctf.
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.
fitrois MC_A	; Similar to the fitrois command, but using the argument allows to specify which spectrum should be evaluated independently of which child window is activated in MCDWIN.
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.
autocal MC_A	; Similar to the autocal command, but using the argument allows to specify which spectrum should be evaluated independently of which child window is activated in MCDWIN.

The following commands make sense only when using the serial line or TCP/IP 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
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 7882 Windows Server via DDE

The 7882 program can be a server for a 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 server program are described together with a demo VI („Virtual Instrument“) for LabVIEW. It is not recommended to use the DDE protocol for LabVIEW, as a DLL interface is (optionally) available which works much faster. The following should be seen as a general description of the DDE conversation capabilities of the server program.

4.2.1. Open Conversation

application: MCD2

topic: MCD2-

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 MCD2 and the topic is MCD2-.

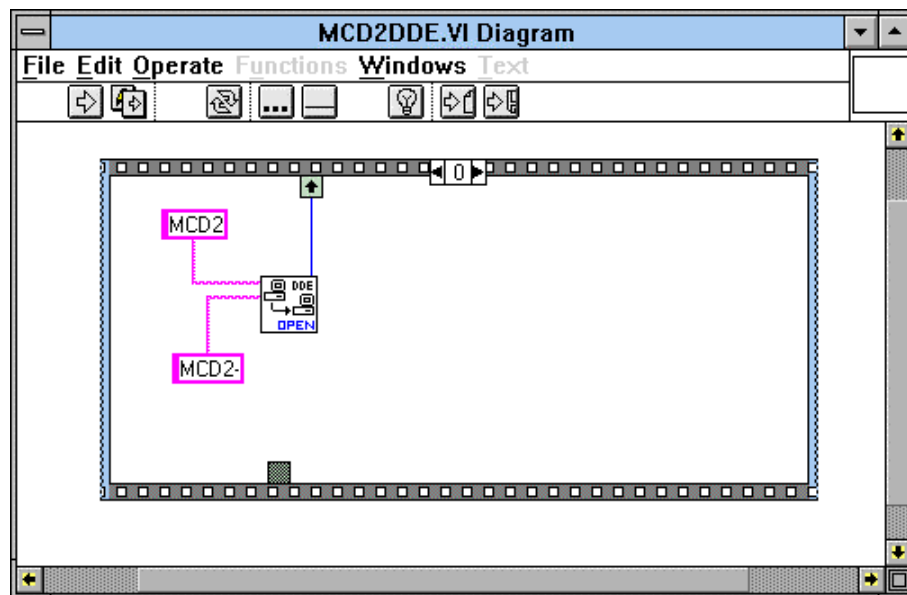


Figure 4.9: Opening the DDE conversation with the 7882 server in LabVIEW

4.2.2. DDE Execute

The DDE Execute command can be used to perform any action of the 7882 program. Any of the Control command lines described in chapter 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 7882 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.

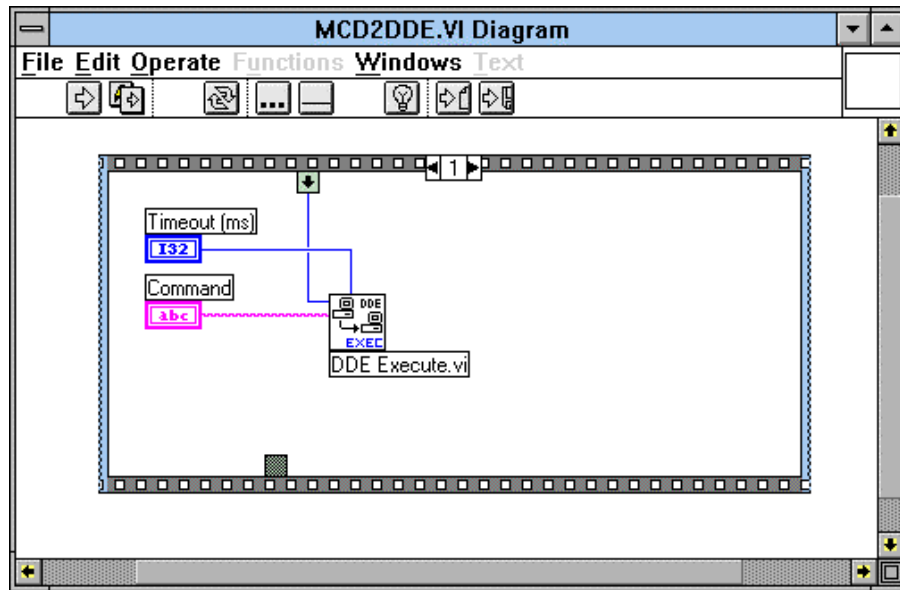


Figure 4.10: Executing a control command from a LabVIEW application

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 DATA item is only available for the 16 bit version). 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 7882 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 desired multichannel analyzer can be selected before by a command MC_A, ..., MC_D.

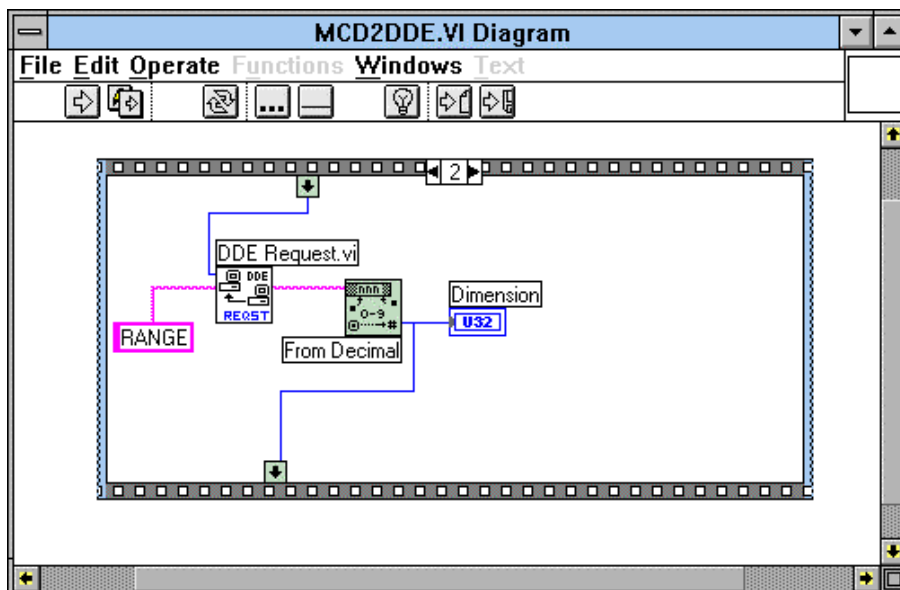


Figure 4.11: Getting the total number of data with LabVIEW

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.

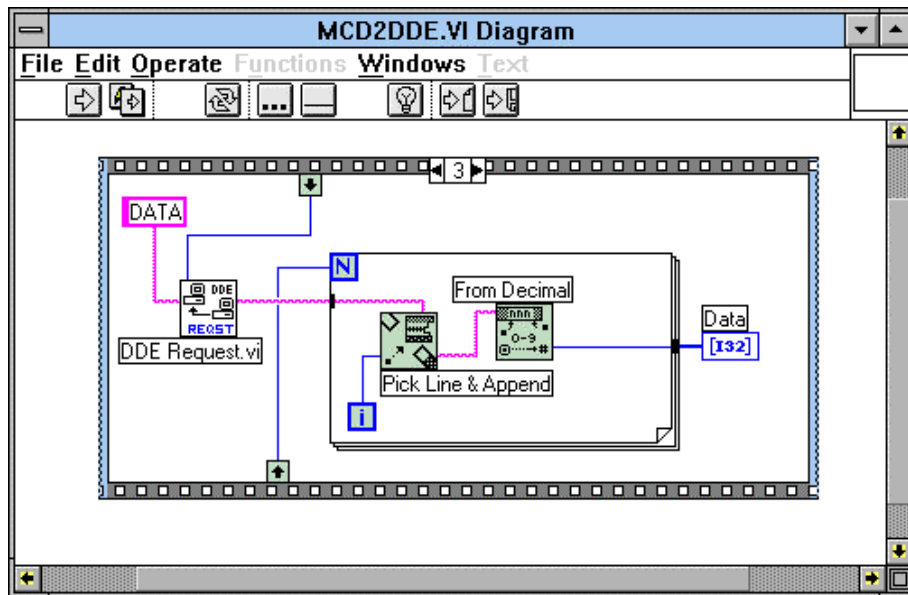


Figure 4.12: Getting the data with LabVIEW

4.2.4. Close Conversation

After finishing the DDE communication with the 7882 program, it must be closed.

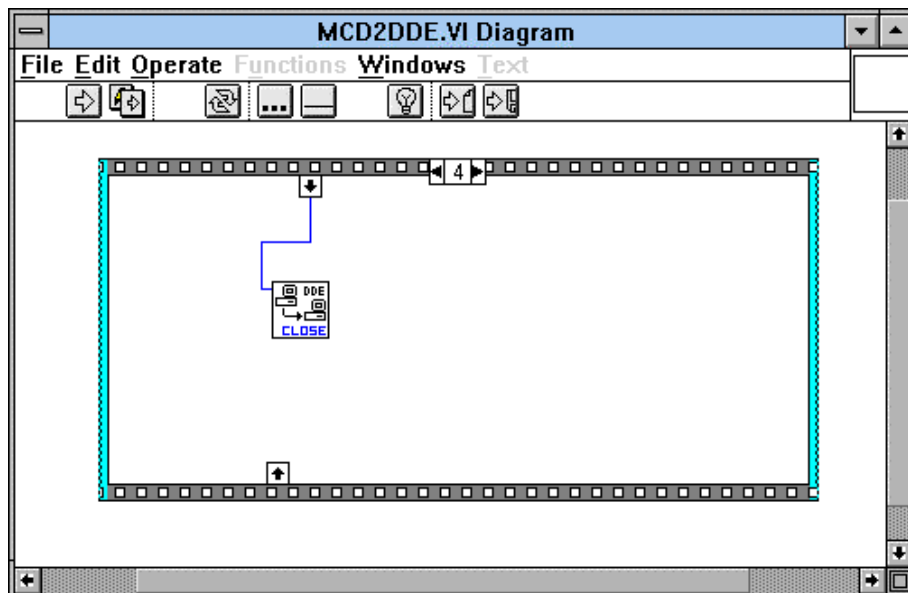


Figure 4.13: Closing the DDE communication in LabVIEW

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

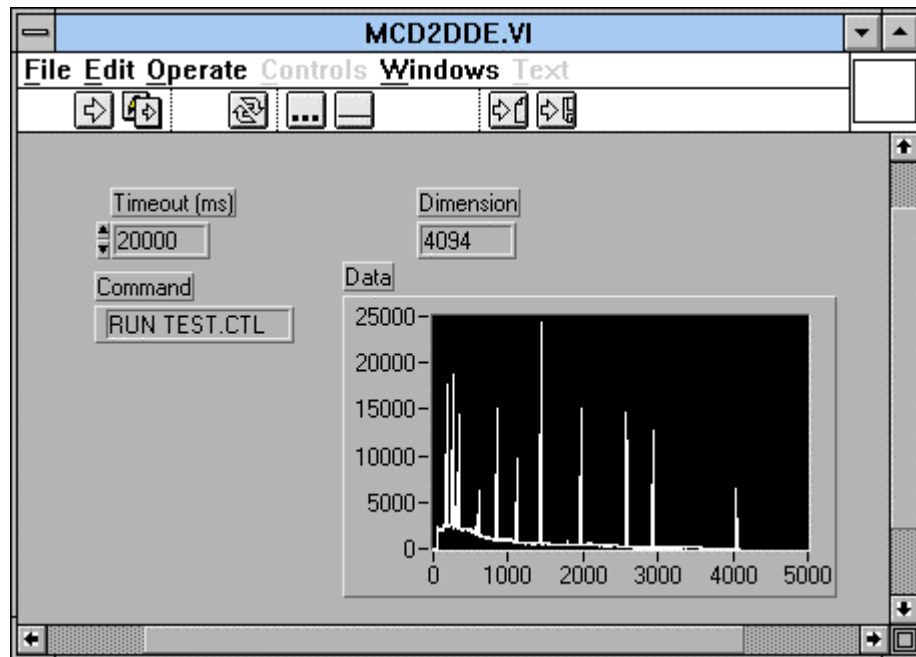


Figure 4.14: Control Panel of the demo VI for LabVIEW

4.3. Controlling the 7882 Windows Server via DLL

The 7882 server program provides - via a DLL („dynamic link library“) - access to all functions, parameters and data. So the server can be completely controlled from the MCDWIN software that provides all necessary graphic displays.

In the following some parts of the header and definition files of the DMCD2.DLL are listed, that may help an experienced programmer to use the DLL for own written applications. Please note that the complete documented source code of the DLL 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;           // acquisition 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;       // Lifetime in seconds
    double deadtime;       // Dead time in percent
    unsigned long maxval;  // Maximum value in spectrum
} ACQSTATUS;

typedef struct{
    long range;            // spectrum length
    int prena;             // bit 0: real time preset enabled
                          // bit 1: lifetime preset enabled
                          // bit 2: sweep preset enabled
                          // bit 3: ROI preset enabled
    int qdac;              // bit 4: 2nd MCS Input: ROI preset enabled
                          // bit 0..7: Quad DAC 3 output value (8 bit)
                          // bit 8: qdac enable
    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 diguse;            // Usage of DIG I/O only Setting[0] stored
                          // bit 0: DIG I/O pins 0-3 output status of
                          // MCD's
                          // bit 1: Invert Polarity
                          // bit 2..5: Input pins 4..7 Trigger System 1..4
                          // bit 6: Output digval and increment
                          // digval after stop
                          // bit 7: Input Mode: =0 Resistive, =1 Tristate
                          // bit 8: Input Mode: =0 Pullup, =1 Pulldown
    int digval;            // DIG I/O Output value only Setting[0] stored
    int rdac;              // Ramp DAC output value (12 bits)
    int mempart;           // bit 0..7: number of active part
                          // bit 8..10: partsize 0:512, 1:1024,...,6:64k
    int dwelltime;         // Dwelltime=(N+1) x 10^M x timebase (125ns)
                          // bit 0..3: N
                          // bit 4..7: M
                          // M=8: Extern channel advance
    int rdacuse;           // bit 0: rdac enable
                          // bit 1: ramp increment enable
                          // bit 2..3: rdac width
                          // 00: 12bit, 01: 11bit, 10: 10 bit, 11: 9bit
    int mcsmode;           // bit 4..8: Divider for ramp increment -1
                          // bit 0: =1: 1 channel MCS-mode
                          // bit 1: =1: 2 channel MCS-mode
                          // bit 2: =1: Retrigger mode
    int syncsrc;           // bit 0..1: sync output source
                          // 00: 1kHz, 01: Sweep Start,
                          // 10: End of Sweep, 11: End of time bin
    double ltpreset;       // lifetime preset value
    int nregions;          // number of regions
}
```

```

    int caluse;           // bit0: 1 if calibration used,
                          // higher bits: formula
    double swpreset;      // Sweep preset
    int active;           // system number 1..4 if active, 0 if not 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 controlled 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 nDisplay);
// Indicate new Settings to Server
UINT FAR PASCAL ServExec(HWND ClientWnd);
// Execute the Server WMCD.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 nDisplay);
// Request actual Status from Server
VOID FAR PASCAL Erase(int nSystem); // Erase spectrum
VOID FAR PASCAL SaveData(int nDisplay); // 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 nDisplay); // Loads data
VOID FAR PASCAL NewData(void); // Indicate new ROI or string Data
VOID FAR PASCAL HardwareDlg(int item);
// item=0: Calls the Settings dialog
// 1: data dialog, 2: system dialog
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 nDisplay, LPSTR Cmd);
// Executes command
VOID FAR PASCAL AddData(int nDisplay); // Adds data

```

```

VOID FAR PASCAL SubData(int nDisplay);          // Subtracts data
VOID FAR PASCAL Smooth(int nDisplay);          // Smooth data
int FAR PASCAL LVGetRoi(unsigned long far *roip, int nDisplay);
                                                // Copies the ROI boundaries to an array
int FAR PASCAL LVGetCnt(double far *cntp, int nDisplay);
                                                // Copies Cnt numbers to an array
int FAR PASCAL LVGetStr(char far *strp, int nDisplay);
                                                // Copies 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 enables the full control of the 7882 card via the server program to perform measurements and save data, and shows the data on-line in several windows.

The server program 7882.EXE automatically starts MCDWIN. If you try to start MCDWIN before the server is started, a message box warns that you should start the server first.

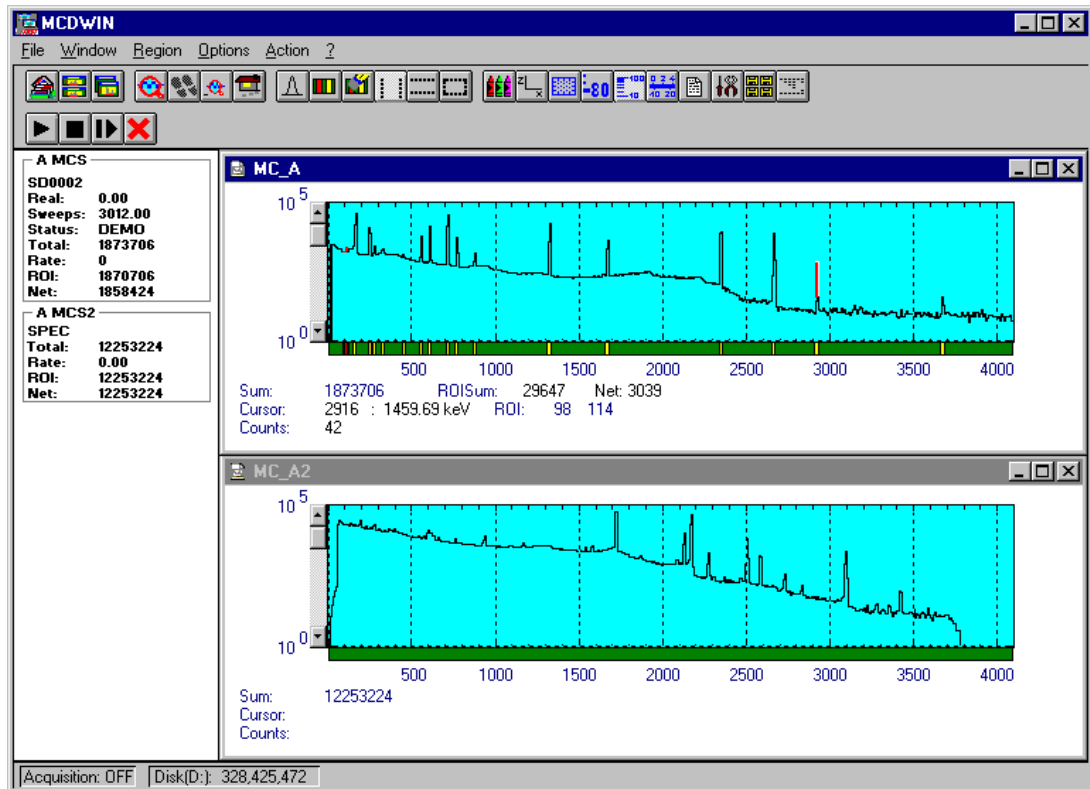


Figure 5.1: MCDWIN main window

A status window at the left side gives all information about the status of the 7882. 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), ASCII (.ASC), GANAAS (.SPE) and DUALBIN (.DA2) format. When you load data, select a header file (extension .MCD). 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.

New Display...

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.

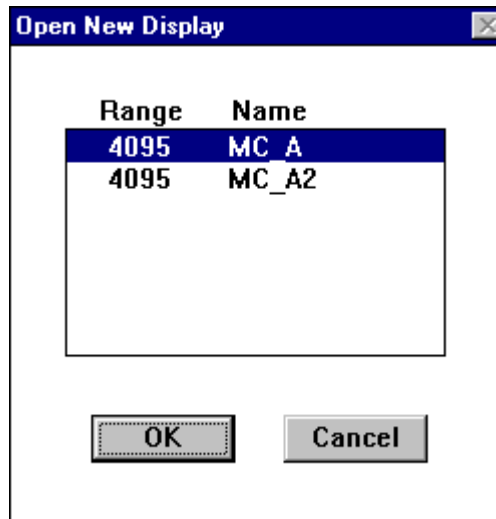


Figure 5.2: File New Display dialog box

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.

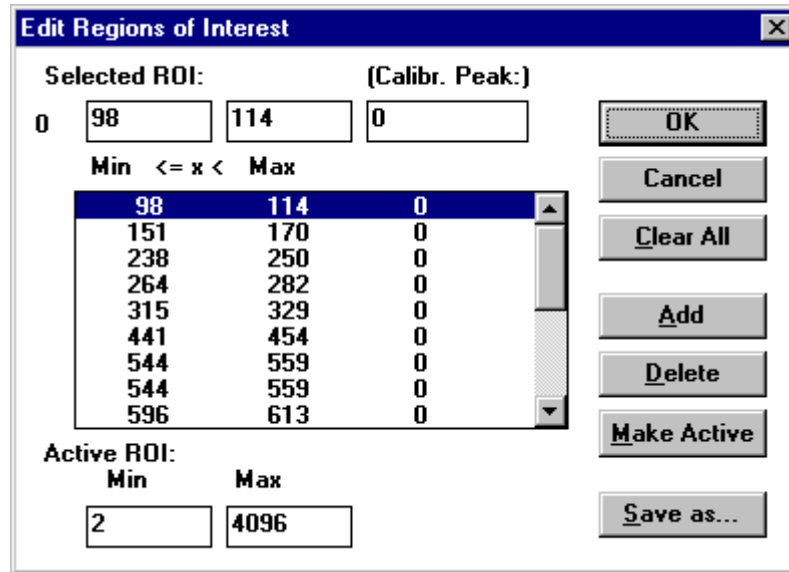


Figure 5.3: ROI Editing dialog box

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 results:

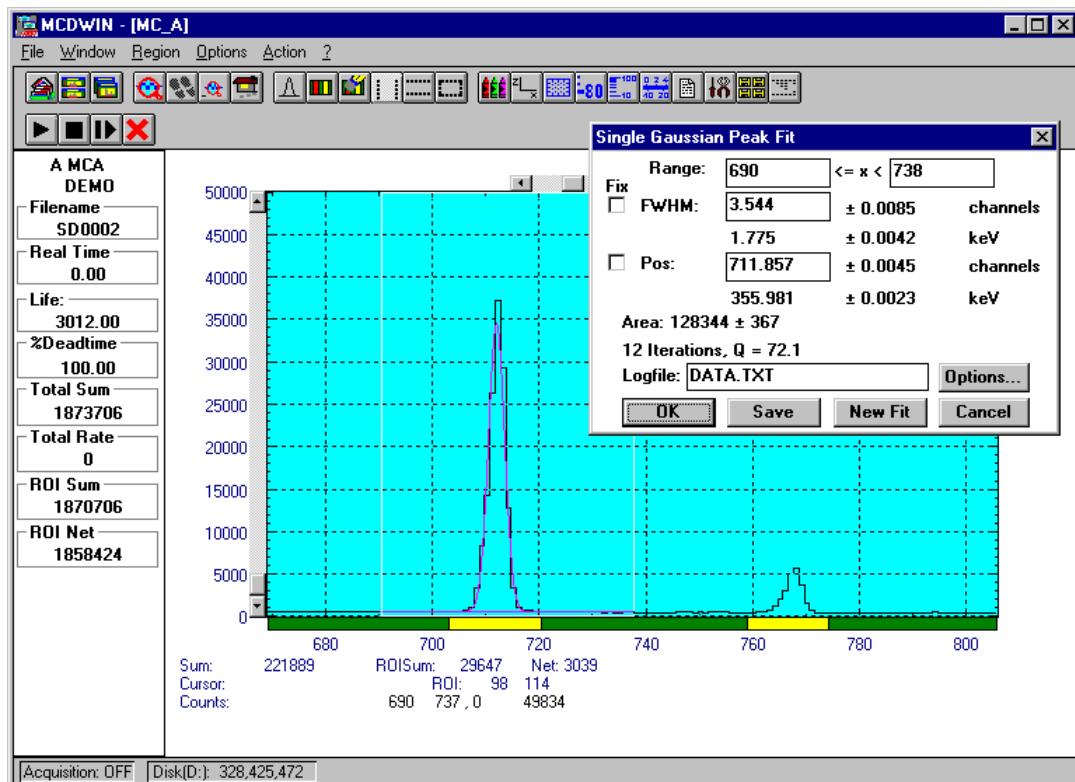


Figure 5.4: Single Gaussian Peak Fit

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 χ^2 . To take into account the systematic error of the line shape, you may multiply the errors with the square root 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 refreshed display. Options... opens a new dialog box to define the information in the logfile:

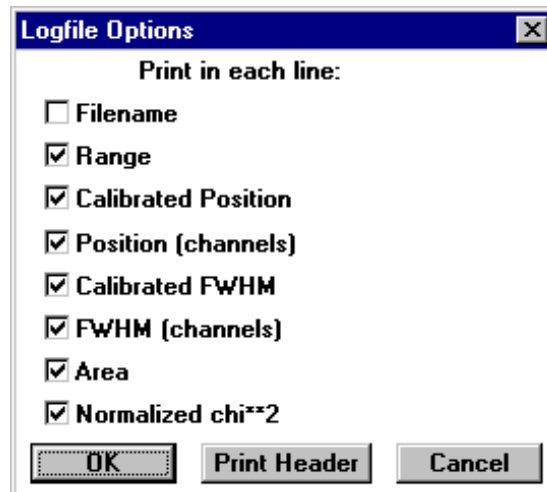


Figure 5.5: Log file Options for the Single Gaussian Peak Fit

The several quantities are written in standard text format with Tabs as separators 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.

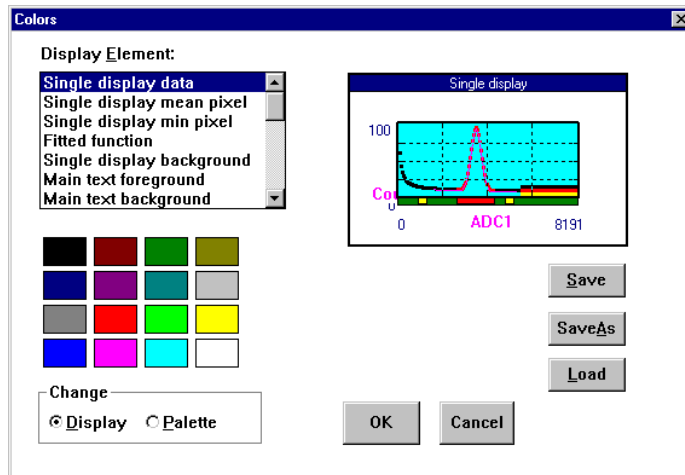


Figure 5.6: Colors dialog box

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.

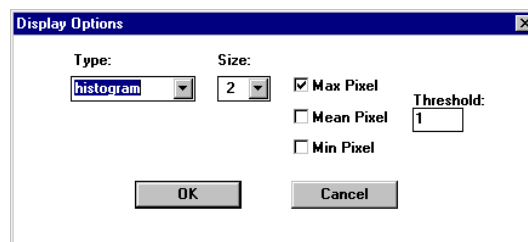


Figure 5.7: Display Options dialog box

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.

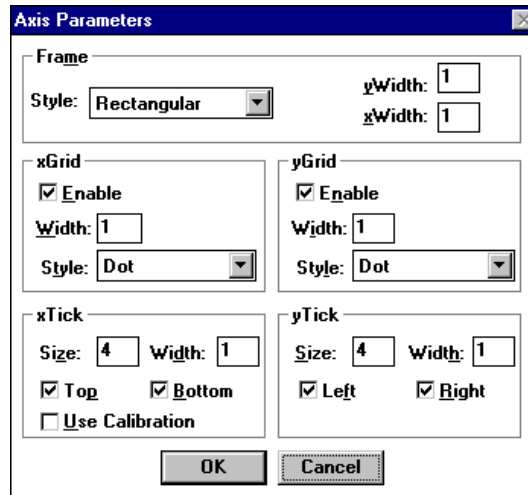


Figure 5.8: Axis Parameter dialog box

Scaling...



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

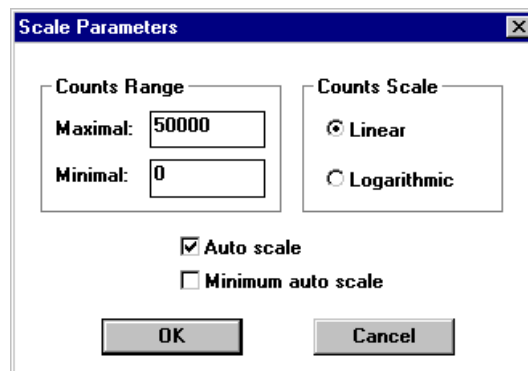


Figure 5.9: :Scale Parameters dialog box

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.

Channel	Value
2344.00	1173.26
2662.00	1332.5

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 7882 settings (See 7882 Server documentation).

MCS Settings

Range: 4096 MCD: A

Part Size: 512

Active Part: 0 0

Dwell Time: 1.25 * 100 ns

Trigger Mode: ☐ Break Sweep ☒ Complete Sweep

Clock: ☒ Internal ☐ External ☐ Manual

☐ ROI preset: 1000

☒ Sweep preset: 100

☐ ROI preset: 10000

ROI min, max: 2 4096

Setup name: mcd

Buttons: Trigger, Advance, Extended..., OK, Cancel, Save Sett., Load

Figure 5.12: Settings dialog box

Via the button **Extended** the Extended Settings dialog box opens. There the MCS input thresholds, the analog outputs, the Sync output source and the 2nd MCS input ROI are configured.

MCD A Extended Settings

☒ DAC Voltage: 65

Trigger level (mV)

☐ Rising Start: 1250

☒ Falling Inp 1: -360

Inp 2: -360

Ramp DAC

☒ Use Ramp DAC

Ramp DAC resolution: 12 bit

☐ Fix Value : 0

☒ Increment Ramp

Ramp Stepsize (1..32): 1 bins

Sync Output Source: 1 kHz

2nd MCS Input

☐ ROI preset: 10000

ROI min, max: 2 4096

Button: OK

Figure 5.13: Extended Settings dialog box

Data...

The Data dialog box allows to make all the respective 7882 settings (See 7882 Server documentation).

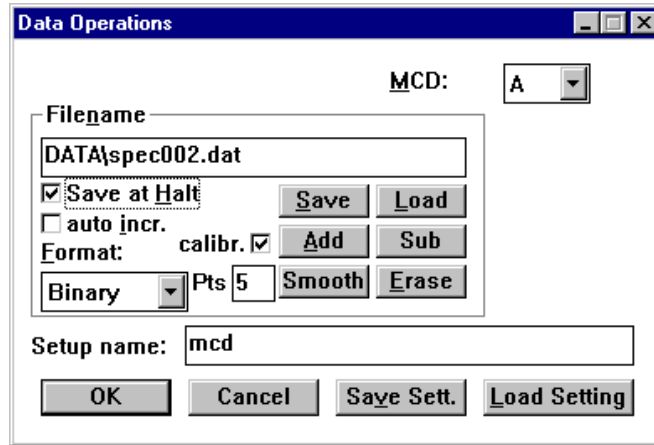


Figure 5.14: Data Operations dialog box

System...

The System Definition dialog box allows to make all the respective 7882 settings (See 7882 Server documentation).

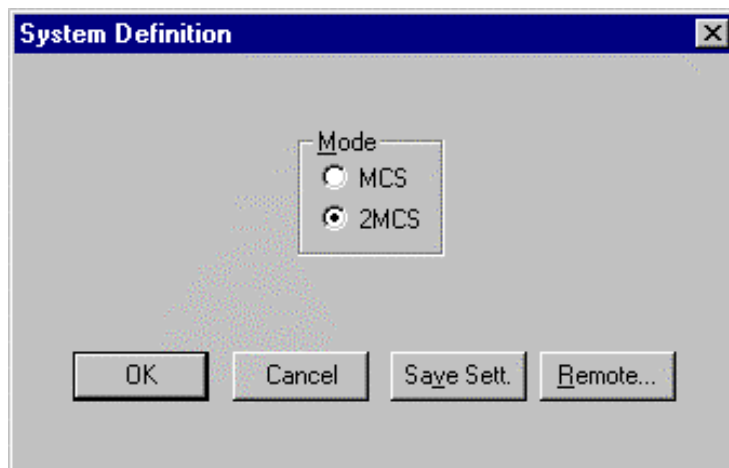


Figure 5.15: System Definition dialog box

Tool Bar...

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

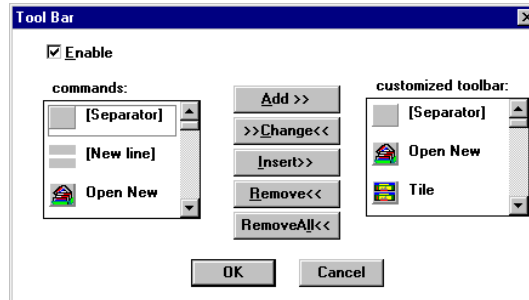


Figure 5.16: Tool Bar dialog box

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 check mark 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. 7882 Programming

6.1. Register Specification

The 7882 is controlled via input and output from/to some I/O port registers. The base address is defined by the rotary switch setting (ref. chapter Hardware Installation; default: 320_{hex}).

Address BASE +	Width [bits]	Write Operation	Read Operation
0	16	CONTROL	CONTROL
1	8	<i>reserved</i>	<i>reserved</i>
2	16	SPEC CONTROL	SPEC CONTROL
3	8	OFFSET	OFFSET
4	16	OUT CONTROL	OUT CONTROL
5	8	DAC CONTROL	DAC CONTROL
6	16	RWA COUNTER	not used
7	8	<i>reserved</i>	<i>reserved</i>
8	16	LOWER DATA WORD	LOWER DATA WORD
9	8	<i>reserved</i>	not used
10	16	HIGHER DATA WORD	HIGHER DATA WORD
11	8	QUAD DAC	not used

Figure 6.1: Register Overview

CONTROL Register**Base + 0**

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

Bit	Default @reset	Name (Write)	Name (Read)	Meaning
0	1	RESET	ARES	Reset state of the 7882 card
1	0	ABORT	ABORT	Abort a running MCS sweep
2	0	MCS	MCS	MCS mode enable
3	0	<i>reserved</i>	<i>reserved</i>	Must be 0
4	0	WRITE	WMODRDY	Write: Write-to-memory enable Read: Write-to-memory mode ready
5	0	SPRE	SPRE	Sweep preset enable
6	0	<i>reserved</i>	<i>reserved</i>	Must be 0
7	0	<i>reserved</i>	<i>reserved</i>	Must be 0
8	0	ARM	ARM	7882 arm
9	0	STRG	ON	Write: Software trigger (not registered) Read: MCS sweep running
10	0	ENRT	ENRT	Retrigger (break sweep) mode enable
11	1	/EN2	/EN2	Input 2 enable (MCS, act. low)
12	0	SADV	DAV	Write: Software channel advance Read: Data valid (read / write memory)
13	0	<i>reserved</i>	<i>reserved</i>	Must be 0
14	0	<i>reserved</i>	<i>reserved</i>	Must be 0
15	[1] 0	PCLR	DIS	Write: Preset clear (not registered) Read: 7882 disabled by preset reached (clears ARM)

Figure 6.2: Control Register

The main modes are Multiscaling (MCS) and Write-to-RAM (WRITE / WMODRDY).

To enable MCS mode the procedure is:

set WRITE = 0

wait until WMODRDY = 0

set MCS = 1

ADC PORT CONTROL Register Base + 1

The ADC PORT CONTROL register is accessed by an 8 bit output or input at base address + 1. The bits in the ADC PORT CONTROL register are defined and used as follows:

Bit	Default @reset	Name	Meaning	
0	1	POL 0	Polarity DRDY:	0 = act. high; 1 = act. low
1	0	POL 1	Polarity DEAD:	0 = act. high; 1 = act. low
2	1	POL 2	Polarity DACC:	0 = act. high; 1 = act. low
3	0	POL 3	Polarity ENC:	0 = act. high; 1 = act. low
4	0	POL 4	Polarity START:	0 = rising; 1 = falling edge
5	0	<i>reserved</i>		
6	0	<i>reserved</i>		
7	0	<i>reserved</i>		

Figure 6.3: ADC PORT CONTROL Register

SPEC CONTROL Register Base + 2

The SPEC CONTROL register is accessed by a 16 bit output or input at base address + 2. The bits written to the SPEC CONTROL register are defined and used as follows: (when reading, N and M are exchanged)

Bit	Default @reset	Name	Meaning
0 1 2 3	0	N(0:3)	Dwell Time = $(N+1) \times 10^M \times 4 \times 1/\text{CLK}$ N = 0...15
4 5 6 7	0	M(0:3)	M = 0...7 8 = select external channel advance 9 = select software channel advance
8 9 10 11	0	RG(0:3)	Range select: 0 = 256 ; 1 = 512 ; 2 = 1k ; 3 = 2k ; 4 = 4k ; 5 = 8k ; 6 = 16k ; 7 = 32k ; 8 = 64k ; 9 = 128k
12 13	0	OSC(0:1)	Onboard crystal oscillator: 0 = 32 MHz ; 1 = 40 MHz ; 2 = 25 MHz ; 3 = 20 MHz
14 15	#	VER(0:1)	Version Number

Figure 6.4: SPEC CONTROL Register

OFFSET Register Base + 3

The OFFSET register is accessed by an 8 bit output or input at base address + 3. The bits in the OFFSET register are defined and used as follows:

Bit	Default @reset	Name	Meaning
0	0	OFFS 0	Spectrum offset = + 512
1	0	OFFS 1	Spectrum offset = + 1k
2	0	OFFS 2	Spectrum offset = + 2k
3	0	OFFS 3	Spectrum offset = + 4k
4	0	OFFS 4	Spectrum offset = + 8k
5	0	OFFS 5	Spectrum offset = + 16k
6	0	OFFS 6	Spectrum offset = + 32k
7	0	OFFS 7	Spectrum offset = + 64k

Figure 6.5: OFFSET Register

OUT CONTROL Register Base + 4

The OUT CONTROL register is accessed by an 16 bit output or input at base address + 4. The bits in the OUT CONTROL register are defined and used as follows:

Bit	Default @reset	Name	Meaning
0 ... 11	0	DAP(0:11)	12 bit DAC fixed mode output voltage or lower voltage limit in X-ramp mode
12 13	0	DSEL(0:1)	Effective DAC resolution in X-ramp mode: 0 = 12 bit ; 1 = 11 bit ; 2 = 10 bit ; 3 = 9 bit
14 15	0	<i>reserved</i>	

Figure 6.6: OUT CONTROL Register

DAC CONTROL Register

Base + 5

The DAC CONTROL register is accessed by an 8 bit output or input at base address + 5. The bits in the DAC CONTROL register are defined and used as follows:

Bit	Default @reset	Name	Meaning
0	31	DACT(0:4)	Divider ratio of End-of-time-bin (EOT) to the X-ramp output generator = $[(0...31)_{dec} + 1]$: In X-ramp mode the 12 bit DAC output is controlled by a counter that is incremented by $EOT / (DACT + 1)$
1			
2			
3			
4			
5	0	ramp	enable X-ramp mode of the 12 bit DAC
6	3	DADR(0:1)	Quad 8 bit DAC select: 0 = Trigger ; 1 = Count 1 ; 2 = Count 2 ; 3 = 8 bit analog output
7			

Figure 6.7: DAC CONTROL Register

The X-ramp output is generated by a 12 bit counter that is incremented by $EOT / (DACT + 1)$. The effective used depth of the counter is defined by DSEL (in any case the most significant bits are used). At EOS (End-of-sweep) the counter is reset to DAP (ref. OUT CONTROL register).

QUAD DAC Register

Base + 11

To write data to one of the four 8 bit DACs first select the appropriate address (DADR) then write the data to base + 11 - QUAD DAC register. The QUAD DAC controls the threshold levels of TRIGGER/START, COUNT 1 & 2 and the 8 bit analog output voltage. Note that this register has no readback capability.

RWA COUNTER

Base + 6

The first RAM address for read or write operations from or to the memory is specified by a 16 bit output of bits 1...16 of the 17 bit RAM address space. Bit 0 (least significant bit) is set to 0. The RWA COUNTER - Read/Write Address Counter - is then automatically incremented on every consecutive read or write operation.

LOWER DATA WORD

Base + 8

The LOWER DATA WORD - 16 bit - of the 32 bit RAM data is accessed via this port address.

HIGHER DATA WORD

Base + 10

The HIGHER DATA WORD - 16 bit - of the 32 bit RAM data is accessed via this port address.

IMPORTANT NOTE:

First the lower, then the higher word must be read or written.

To **write data** into the RAM the procedure is as follows:

set MCS, ARM = 0 and /EN2 = 1

set WRITE = 1

wait for WMODRDY = 1 --> WriteModeReady

wait for DAV = 0
load the RWA COUNTER
wait for DAV = 0 --> 7882 signals that it is waiting for valid data
first write the LOWER then the HIGHER DATA WORD
repeat the last step until finished

To **read data** from the memory the following procedure must be performed:

MCS = 1 and WRITE = 0 is mandatory
wait for DAV = 1
load the RWA COUNTER
wait for DAV = 1 --> 7882 signals that valid data is present in the output register
first read the LOWER then the HIGHER DATA WORD
repeat the last two steps until finished

6.2. The Subroutines for controlling the 7882

In the following the low level subroutines used by the 7882 server program (7882.EXE) for controlling the 7882 are listed and commented. The program language used is C

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

```

    outp    with    outportb    (writes a byte to an I/O port)
    outpw   with    outport     (writes a 16 bit word to an I/O
port)
    inp     with    inportb     (reads a byte from an I/O port)
    inpw    with    inport      (reads a 16 bit word from an I/O port)

```

Variables and constants, if not clear from the context, are discussed when they appear first time. The following functions are listed to illustrate the hardware programming, but do not form a complete executable program. Note that not all variables and functions are declared, if they are not necessary to understand the hardware programming. Standard Microsoft C or Windows functions like fopen, lstrcpy, MessageBox and so on are not documented.

The complete source code of the DLL D7882.DLL that controls the hardware via the server program 7882.exe with example programs for Visual Basic and LabVIEW is available as an option.

cardinit

The cardinit routine defines the base address, initializes the 7882 and performs a basic test whether it is present and OK. It returns TRUE if a 7882 is present and working, otherwise FALSE.

```

int cardinit()
{
    int nDev, demo=0, ibit=1, running=0, ret;
    long i;
    FILE *f;
    char buf[80];

    for (nDev=0; nDev<devices; nDev++) {
        if (!demomod[nDev]) {
            creg = inpw(base[nDev]); // read control register
            if ((creg & 0x01) || // ares==1, power on status
                !(creg & (BITARM | BITENC))) {
                // arm==enc==0, not running
                Status[nDev].started = OFF;
                Status[nDev+devices].started = OFF; // 2nd input spect
                ret=cardreset(nDev);
                if (ret != 1) { // reset card and set parameters
                    demomod[nDev] = 1;
                }
            }
        }
        else { // a measurement is running!
            cardget(nDev); // get card settings
        }
        if (!readmem(nDev,0,256)) { // check whether card
            demomod[nDev]=1; // memory can be read
        }
        if (Status[nDev].started) running=1;
    }
    if (demomod[nDev]) {
        Status[nDev].started = 6; // Status: DEMO
        Status[nDev+devices].started = 6;
        demo |= ibit;
    }
    ibit <=<= 1;
}

if (running) {
    if (f = fopen("MCD2.STS", "r+t")) {
        freadstr(f,buf,80);
        for (i=0; i<devices; i++) {
            fscanf(f,"%lu %lu", rtimoff+i, ltimoff+i);
            if (Status[i].started)
                lstrcpy((LPSTR)Data[i].comment0, (LPSTR)buf);
        }
    }
}

```

```
    }
    fclose(f);
}
}
for (nDev=0; nDev<devices; nDev++) {
    for (i=0; i<memrange[nDev]; i+=256)
        readmem(nDev,i,256);
    if (Setting[nDev].mcsmode & 0x02) {
        int nDev1 = nDev + devices;
        for (i=0; i<memrange[nDev]; i+=256)
            readmem(nDev1,i,256);
    }
}
return demo;
}
```

Variable	Meaning
----------	---------

base[4]	unsigned integer array specifying the i/o port base addresses of the 7882 cards.
---------	--

creg	control register
------	------------------

memrange[8]	long array containing offset + histogram length of the spectra.
-------------	---

Status[8]	an array of structures describing the Status of each MCA (see section 4.3):
-----------	---

Setting[8]	an array of structures describing the settings of each MCA (see section 4.3):
------------	---

devices	number of installed 7882 cards (1..4)
---------	---------------------------------------

Constants	Meaning
-----------	---------

#define BITARES	0x01
#define BITABORT	0x02
#define BITENOWS	0x02
#define BITMCS	0x04
#define BITPHA	0x08
#define BITWRITE	0x10
#define BITWMODRDY	0x10
#define BITSPRE	0x20
#define BITRPRE	0x40
#define BITLPRE	0x80
#define BITARM	0x100
#define BITON	0x200
#define BITSTRG	0x200
#define BITENRT	0x400
#define BITEN2	0x800
#define BITDAV	0x1000
#define BITSADV	0x1000
#define BITENC	0x2000
#define BITDENB	0x4000
#define BITDIS	0x8000
#define BITPCLR	0x8000
#define MASK	0x6DFC

cardreset

The cardreset routine resets the 7882.

```
int cardreset(int nDev)
{
    creg = inpw(base[nDev]);
    creg |= BITARES;
    outpw(base[nDev],creg);           // Reset...
    creg = inpw(base[nDev]);
    if (!creg & BITARES) return -1;   // ares must be ON now
    if (creg & 0x77FE) {               // these bits must be zero
        creg &= MASK;
        outpw(base[nDev],creg);       // ares=0 back to regular state
        return -2;
    }
}
```

```

    }
    creg &= MASK;
    outpw(base[nDev], creg);          // ares=0 back to regular state
    creg = inpw(base[nDev]);
    if (creg & BITARES) return -3;    // must be OFF now
    cardset(nDev);
    return (TRUE);
}

```

cardset

The cardset routine initializes the 7882 parameters.

```

void cardset(int nDev)
{
    int w=0,v,dacctl;
    int baseadr = base[nDev];
    long r=256L;

    v = (pol0[nDev] & 0x0F) | ((pol1[nDev] & 0x01)<<4);
    outp(baseadr+1,v);                // Set polarities
    for (v=0; v<16; v++) {
        if (r==Setting[nDev].range) break;
        r <= 1;
    }
    v <= 8;
    v |= (Setting[nDev].dwelltime & 0xFF);
    v |= oszselect[nDev];             // Oscillator select
    outpw(baseadr+2,v);               // Set dwell time and range
    w = (int)(memstart[nDev] >> 9);
    outp(baseadr+3,w);                // memory offset
    setioreg(nDev);                   // Set out control, dac control, digio,
                                     // digio enable, dac 3
    v = inp(baseadr + 5);              // read dac control
    v &= 0x3F;
    dacctl = v;                       // select dac0
    outp(baseadr+5, dacctl);
    w = ((thrstart[nDev] * 32)/125 + 128);
    if (w > 255) w = 255; if (w < 0) w = 0;
    outp(baseadr+11, w);              // set start threshold
    dacctl = v | 0x40;                // select dac1
    outp(baseadr+5, dacctl);
    w = ((thr1[nDev] * 32)/125 + 128);
    if (w > 255) w = 255; if (w < 0) w = 0;
    outp(baseadr+11, w);              // set inp1 threshold
    dacctl = v | 0x80;                // select dac2
    outp(baseadr+5, dacctl);
    w = ((thr2[nDev] * 32)/125 + 128);
    if (w > 255) w = 255; if (w < 0) w = 0;
    outp(baseadr+11, w);              // set inp2 threshold
    dacctl |= 0xC0;                   // select dac3
    outp(baseadr+5, dacctl);

    creg = inpw(baseadr) & MASK;

    if (!(Setting[nDev].mcsmode & 0x03)) { // MCA mode
        if (Setting[nDev].prena & 0x01) creg |= BITRPRE; // real time preset
        else creg &= (0xFFFF ^ BITRPRE);
        if (Setting[nDev].prena & 0x02) creg |= BITLPRE; // life time preset
        else creg &= (0xFFFF ^ BITLPRE);
    }
    else creg &= 0xFF3F;
    if (Setting[nDev].prena & 0x04) creg |= BITSPRE; // Sweep preset
    else creg &= (0xFFFF ^ BITSPRE);
    if (!(Setting[nDev].mcsmode & 0x03)) { // MCA mode
        if (pol0[nDev] & 0x10) // denb=1
            creg &= (0xFFFF ^ BITDENB);
        else
            creg |= BITDENB;
    }
    outpw(baseadr, creg);              // Set Preset Bits
}

```

```
}

```

setioreg

The setioreg routine sets dig i/o and DAC's available at the i/o connector.

```
void setioreg(int nDev)
{
    int i,j,dacctl,outctl;
    int ioreg = 0;

    if (demomod[nDev] || nDev >= devices) return;
    if (Setting[0].diguse & 0x40) { // output digval
        ioreg = (unsigned char)(Setting[0].digval & 0xFF);
        if (Setting[0].diguse & 0x02) ioreg ^= 0xFF; // invert polarity
        outp(base[nDev]+9, 0xFF); // output enable
        outp(base[nDev]+7, ioreg);
    }
    else if (Setting[0].diguse & 0x01) { // output status
        j=1;
        for (i=0; i<devices; i++) {
            if (Status[i].started & 0x01)
                ioreg |= j;
            j <= 1;
        }
        if (Setting[0].diguse & 0x02) ioreg ^= 0x0F; // invert polarity;
        if (Setting[0].diguse & 0x80) // tristate input
            outp(base[nDev]+9, 0x0F); // bit 0..3 output enable
        else {
            outp(base[nDev]+9, 0xFF); // output enable
            if (!(Setting[0].diguse & 0x100)) // pull up?
                ioreg |= 0xF0;
        }
        outp(base[0]+7, ioreg);
    }
    if (Setting[nDev].rdacuse & 0x01) { // rdac enable
        dacctl = (Setting[nDev].rdacuse & 0x1F0) >> 4; // ramp inc divider
        if ((Setting[nDev].mcsmode & 0x03) &&
            (Setting[nDev].rdacuse & 0x02)) dacctl |= 0x20;
            // ramp increment enable
    }
    else dacctl = inp(base[nDev] + 5);
    dacctl |= 0xC0; // select dac3
    outp(base[nDev]+5, dacctl);
    if (Setting[nDev].qdac & 0x100) { // qdac enable
        outp(base[nDev]+11, Setting[nDev].qdac & 0xFF);
    }
    if (Setting[nDev].rdacuse & 0x01) { // rdac enable
        outctl = Setting[nDev].rdac; // rdac value
        outctl |= ((Setting[nDev].rdacuse & 0x0C)<<10); // rdac width
    }
    else
        outctl = inpw(base[nDev]+4) & 0x3FFF;
    outctl |= (Setting[nDev].syncsrc << 14); // sync output source
    outpw(base[nDev]+4, outctl);
}

```

readmem

The readmem routine reads a block of channels.

```
int readmem(int nDev, long knum, int amount) // reads amount channels
{ // starting from knum
    int i,j,base8,base10,baseadr;
    unsigned int num;
    unsigned long val;
    unsigned *pw;
    int nDev0 = nDev;

```

```
if (!Setting[nDev].active) return TRUE;
if (nDev >= devices) nDev0 -= devices;

if (demomod[nDev0]) return FALSE;
pw = (unsigned *)(&val);
baseadr = base[nDev0];
if (!prepareread(nDev0, baseadr)) return FALSE;
num = (unsigned int)(knum >> 1);
if (nDev >= devices) num |= 0x8000;
outpw(baseadr+6, num);
base8=baseadr+8;
base10=baseadr+10;
for (j=0; j<amount; j++) {
    // if (!waitdav(baseadr)) return FALSE;
    pw[0] = inpw(base8);
    pw[1] = inpw(base10);
    Data[nDev].s0[knum++] = val;
}
return TRUE;
}

int prepareread(int nDev0, int baseadr) // prepare for reading
{
    int i;
    creg = inpw(baseadr);
    if (creg & BITWMODRDY) {                // wmodrdy != 0 ?
        creg &= 0x6DEE;
        outpw(baseadr, creg);                // set write=0
        for (i=0; i<MAXLOOP; i++) {
            creg = inpw(baseadr);
            if (!(creg & BITWMODRDY)) break; // wmodrdy == 0 ?
        }
        if (i==MAXLOOP) return FALSE;
    }
    if (!(creg & BITDAV)) {                // dav != 1?
        creg &= 0x6DE0;
        if (Setting[nDev0].mcsmode && 0x11) creg |= BITMCS;
        else creg |= BITPHA;
        outpw(baseadr, creg);                // set mca or mcs mode
    }
    return TRUE;
}

int waitdav(int baseadr)                // wait for data valid
{
    int i;
    for (i=0; i<MAXLOOP; i++) {
        creg = inpw(baseadr);
        if (creg & BITDAV) break;            // dav == 1 ?
    }
    if (i==MAXLOOP) return FALSE;
    return TRUE;
}
```

Variable Meaning

Data An array of structures containing the pointers to the data of each MCA (see section 4.3).

writemem

The writemem routine writes a block of channels.

```
int writemem(int nDev, long knum, int amount)
{
    int i, j, base8, base10;
    unsigned int num;
```

```

int nDev0 = nDev;
int baseadr;
unsigned int huge *pt;

if (nDev >= devices) nDev0 -= devices;
if (demomod[nDev0]) return TRUE;
baseadr = base[nDev0];
if (!preparewrite(baseadr)) return FALSE;
num = (unsigned int)(knum >> 1);
if (nDev >= devices) num |= 0x8000;
outpwt(baseadr+6, num);
base8=baseadr+8;
base10=baseadr+10;
pt = (unsigned huge *)(&Data[nDev].s0[knum]);
for (j=0; j<amount; j++) {
    outpw(base8, *pt++);
    outpw(base10, *pt++);
}
return TRUE;
}

int preparewrite(int baseadr)    // prepare for writing
{
    int i;
    creg = inpw(baseadr);

    // wmodrdy==0 or dav==1?
    creg &= 0x48E2;
    creg |= BITEN2;
    outpw(baseadr, creg);    // set mcs, pha, arm, enc, enrt=0 and en2=1
    outpw(baseadr+6, 0);    // address counter must be set to clear dav
    creg |= BITWRITE;
    outpw(baseadr, creg);    // set write=1
    for (i=0; i<MAXLOOP; i++) {
        creg = inpw(baseadr);
        if ((creg & BITWMODRDY) && !(creg & BITDAV)) break;
        // wmodrdy==1 && dav==0?
    }
    if (i==MAXLOOP) return FALSE;

    return TRUE;
}

```

clearspec

The clearspec routine clears an MCA.

```

void clearspec(int nDev)
{
    long i;
    int oldstate;
    int nDev0=nDev;

    if (nDev >= devices) nDev0 -= devices;
    oldstate = Status[nDev0].started;
    if (oldstate & 0x01) MCA_Stop(nDev0);

    for (i=memstart[nDev0]; i<memrange[nDev0]; 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].lifetime=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[nDev0]) {

```

```
        for (i=memstart[nDev0]; i<memrange[nDev0]; i+=256) {
            writemem(nDev,i,256);
        }
    }
    rtimoff[nDev0] = 0L;
    ltimoff[nDev0] = 0L;
    writetimers(nDev);
    if (oldstate & 0x01) MCA_Continue(nDev0);
    else ShowStatus(nDev);
}
```

Variable Meaning

memstart[8] memory offset, where the active spectrum starts.

MCA_Newtime, MCA_Newsweeps

The MCA_Newtime routine sets the timers, MCA_Newsweeps the sweepcounter.

```
void MCA_Newsweeps(int nDev, unsigned long sweeps)
{
    Data[nDev].s0[memstart[nDev]] = (unsigned long)sweeps;
    writemem(nDev, memstart[nDev], 1);
}

void MCA_Newtime(int nDev, unsigned long rtim, unsigned long ltim)
{
    int i, j, base8, base10;
    unsigned int num;
    int baseadr;
    unsigned int *pt;
    unsigned int val0, val1;

    if (demomod[nDev]) return;
    baseadr = base[nDev];
    if (!preparewrite(baseadr)) return;
    num = (unsigned int)(memstart[nDev] >> 9) | 0xFF00;
    outpw(baseadr+6, num);
    base8=baseadr+8;
    base10=baseadr+10;
    pt = (unsigned int *)(&rtim);
    val0 = *pt++;
    val1 = *pt++;
    outpw(base8, val0);
    outpw(base10, val1);
    pt = (unsigned int *)(&ltim);
    outpw(base8, *pt++);
    outpw(base10, *pt++);
    return;
}
```

MCA_Gettime, MCA_Getsweeps

The MCA_Gettime routine reads the timers, MCA_Getsweeps the sweepcounter.

```
unsigned long MCA_Getsweeps(int nDev)
{
    readmem(nDev, memstart[nDev], 1);
    return Data[nDev].s0[memstart[nDev]];
}

void MCA_Gettime(int nDev, unsigned long *rtim, unsigned long *ltim)
{
    unsigned long val;
    unsigned *pw;
    unsigned int num;
    int i, base8, base10, baseadr;
    if (!Setting[nDev].active) return;
    if (demomod[nDev]) return;
    baseadr = base[nDev];
```



```

if (!prepareread(nDev, baseadr)) return;
pw = (unsigned *)(&val);
num = (unsigned int)(memstart[nDev] >> 9) | 0xFF00;
outpw(baseadr+6, num);
base8=baseadr+8;
base10=baseadr+10;
if (!waitdav(baseadr)) return;
pw[0] = inpw(base8);
pw[1] = inpw(base10);
*rtim = val;
if (!waitdav(baseadr)) return;
pw[0] = inpw(base8);
pw[1] = inpw(base10);
*ltim = val;
}

```

setpreset

The setpreset routine sets real- life time and sweep presets.

```

long rtim0, ltim0;
int setpreset(int nDev)
{
    int ret, nDev0=nDev;
    if (nDev0 >= devices) nDev0 -= devices;
    if (!(Setting[nDev0].mcsmode & 0x03)) {
        MCA_Gettime(nDev0, &rtim0, &ltim0);
        if (Setting[nDev0].prena & 0x01) {
            if (rtim0 < Setting[nDev0].rtpreset * dtimerticks[nDev0]) {
                rtimoff[nDev0] = (long)(Setting[nDev0].rtpreset *
                    dtimerticks[nDev0]);
                rtim0 -= rtimoff[nDev0];
            }
            else {
                if (!ctlrunning) {
                    ret = MessageBox(hwndMCD,
                        "Realtime Preset reached! \nNew measurement without clear?",
                        szAppName, MB_YESNO);
                    if (ret==IDNO) return 1;
                }
                rtimoff[nDev0] = rtim0 + (long)(Setting[nDev0].rtpreset *
                    dtimerticks[nDev0]);
                rtim0 = - (long)(Setting[nDev0].rtpreset * dtimerticks[nDev0]);
            }
        }
        else rtimoff[nDev0] = 0L;
        if (Setting[nDev0].prena & 0x02) {
            if (ltim0 < Setting[nDev0].ltpreset * dtimerticks[nDev0]) {
                ltimoff[nDev0] = (long)(Setting[nDev0].ltpreset *
                    dtimerticks[nDev0]);
                ltim0 -= ltimoff[nDev0];
            }
            else {
                if (!ctlrunning) {
                    ret = MessageBox(hwndMCD,
                        "Lifetime Preset reached! \nNew measurement without clear?",
                        szAppName, MB_YESNO);
                    if (ret==IDNO) return 1;
                }
                ltimoff[nDev0] = ltim0 + (long)(Setting[nDev0].ltpreset *
                    dtimerticks[nDev0]);
                ltim0 = - (long)(Setting[nDev0].ltpreset * dtimerticks[nDev0]);
            }
        }
        else ltimoff[nDev0] = 0L;
        MCA_Newtime(nDev0, rtim0, ltim0);
    }
}

```

```

else {
    ltim0 = MCA_Getsweeps(nDev0);
    if (Setting[nDev0].prena & 0x04) {
        if (ltim0 < Setting[nDev0].swpreset) {
            ltimoff[nDev0] = (long)(Setting[nDev0].swpreset+1);
            ltim0 -= ltimoff[nDev0];
        }
        else {
            if (!ctlrunning) {
                ret = MessageBox(hwndMCD,
                    "Sweep Preset reached! \nNew measurement without clear?",
                    szAppName, MB_YESNO);
                if (ret==IDNO) return 1;
            }
            ltimoff[nDev0] = ltim0 + (long)(Setting[nDev0].swpreset+1);
            ltim0 = - (long)(Setting[nDev0].swpreset+1);
        }
    }
    else ltimoff[nDev0] = 0L;
    MCA_Newsweeps(nDev0, ltim0);
    if ((Setting[nDev0].prena & 0x01) &&
        (Status[nDev0].realtime >= Setting[nDev0].rtpreset)) {
        if (!ctlrunning) {
            ret = MessageBox(hwndMCD,
                "Realtime Preset reached! \nNew measurement without clear?",
                szAppName, MB_YESNO);
            if (ret==IDNO) return 1;
        }
        rtime0[nDev0] = 0.;
        Status[nDev0].realtime = 0.;
        Data[nDev0].cnt[CN_REALTIME] = Status[nDev0].realtime;
        if (Setting[nDev0].mcsmode & 0x02) {
            int nDev1 = nDev0 + devices;
            Status[nDev1].realtime = 0.;
            Data[nDev1].cnt[CN_REALTIME] = Status[nDev1].realtime;
        }
    }
}
return 0;
}

```

MCA_Start

The MCA_Start routine starts an acquisition.

```

void MCA_Start(int nDev)
{
    time_t    tnow;
    struct tm *tmnow;
    char buf[80];
    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);
    lstrcpy((LPSTR)Data[nDev].comment0, (LPSTR)buf);
    if (Setting[nDev].mcsmode & 0x02)
        lstrcpy((LPSTR)Data[nDev+devices].comment0, (LPSTR)buf);
    cardreset(nDev);
    //cardset(nDev);
    MCA_Continue(nDev);
}

void MCA_Continue(int nDev0)
{
    int baseadr = base[nDev0];
    int nDev1 = nDev0;
    if (demomod[nDev0]) return;
    setpreset(nDev0);
    prepareread(nDev0, baseadr); // set write=0, chose mcs or pha mode
}

```

```

if (Setting[nDev0].mcsmode & 0x02) { // 2 inputs mcsmode
    creg &= 0x01EE; // denb=0, enc=0, en2=0,
                    // write=0, enrt=0
    if (Setting[nDev0].mcsmode & 0x04) // retrigger mode
        creg |= BITENRT; // enrt=1
    creg |= BITARM; // arm=1
    nDev1 += devices;
}
else if (Setting[nDev0].mcsmode & 0x01) { // 1 input mcsmode
    creg &= 0x0D6E; // denb=0, enc=0, write=0
    if (Setting[nDev0].mcsmode & 0x04) // retrigger mode
        creg |= BITENRT; // enrt=1
    creg |= 0x0900; // arm=1, en2=1
}
else { // pha mode
    creg &= 0x2CE2; // denb=0, arm=0, write=0
    if (pol0[nDev0] & 0x10)
        creg |= BITDENB; // denb=1
    creg |= (BITENC | BITPHA); // enc=1
}
outpw(baseadr, creg);
startclock[nDev0] = clock();
Status[nDev0].started = ON;
Status[nDev1].started = ON;
dumpsts();
(*lpStoreStatusData)(&Status[nDev0], nDev0); //store status data in DLL
if (Setting[nDev0].mcsmode & 0x02)
    (*lpStoreStatusData)(&Status[nDev1], nDev1);
}

void NEAR dumpsts()
{
    int i;
    FILE *f;

    f = fopen("MCD2.STS", "w+t");
    fprintf(f, "%s\n", Data[0].comment0);
    for (i=0; i<devices; i++)
        fprintf(f, "%lu %lu\n", rtimoff[i], ltimoff[i]);
    fclose(f);
}

```

MCA_Stop

The MCA_Stop finishes an acquisition.

```

void MCA_Stop(int nDev)
{
    int nDev0 = nDev;
    if (nDev >= devices) nDev0 -= devices;
    MCA_Halt(nDev0);
    ShowStatus(nDev0);
    if (Setting[nDev0].mcsmode & 0x02) ShowStatus(nDev0+devices);
    if ((Setting[nDev0].mcsmode & 0x03) && (Status[nDev0].started & 0x01))
    {
        if (IDNO == MessageBox(hwndMCD,
            "Sweep still running! \nAbort Sweep?",
            szAppName, MB_YESNO)) return;
        abortsweep(nDev0); ShowStatus(nDev0);
    }
}

void MCA_Halt(int nDev0)
{
    int baseadr = base[nDev0];
    if (demomod[nDev0]) {
        return;
    }
    creg = inpw(baseadr);
    creg &= 0xCCEE; // enc=0, arm=0, write=0
}

```

```

    if (!(Setting[nDev0].mcsmode & 0x03)) {
        if (pol0[nDev0] & 0x10) // denb=1
            creg &= (0xFFFF ^ BITDENB);
        else
            creg |= BITDENB;
    }
    outpw(baseadr, creg); // set (evt. pclr=1), enc=0, arm=0, write=0
}

void abortsweep(int nDev)
{
    if (demomod[nDev]) return;
    creg = inpw(base[nDev]);
    creg &= 0x6EFC; // (MASK & 0xFEFF); arm=0
    creg |= BITABORT; // abort = 1
    outpw(base[nDev], creg);
    for (i=0; i<450; i++) {
        creg = inpw(base[nDev]);
        if (creg & BITON) waitmsec(100); // wait 100 msec if still running
        else break;
    }
    creg &= 0x6EFC; // abort = 0
    outpw(base[nDev], creg);
    setioreg(nDev);
}

```

makesweep, makeadvance

The makesweep starts a software sweep, makeadvance advances a new time bin.

```

void makesweep(int nDev)
{
    if (demomod[nDev]) return;
    creg = inpw(base[nDev]);
    creg &= MASK;
    creg |= BITSTRG; // strg = 1
    outpw(base[nDev], creg);
}

void makeadvance(int nDev)
{
    if (demomod[nDev]) return;
    creg = inpw(base[nDev]);
    creg &= MASK;
    creg |= BITSADV; // sadv = 1
    outpw(base[nDev], creg);
}

```

7. Appendix

7.1. Absolute maximum ratings

Input voltage (Trigger, Count 1 & 2): -7 to +7 V

Input voltage (external channel advance): -0.5 to +5.5 V

WARNING !!!

The Trigger, Count 1 and Count 2 inputs are very high speed inputs. To obtain this very high speed no ESD protection circuitry is provided on these inputs. Be very careful handling the 7882 card to avoid high voltage on these inputs.

7.2. Connectors

7.2.1. MCS Inputs

Trigger Input

Connector type: LEMOSA Series 00 NIM-CAMAC

Input voltage range: ± 5 V

Input threshold range: ± 5 V

Input threshold step size: 39 mV

Input impedance: 50 Ω

Slope: programmable falling or rising edge

Count 1 & 2 Inputs

Connector type: LEMOSA Series 00 NIM-CAMAC

Input voltage range: ± 5 V

Input threshold range: ± 5 V

Input threshold step size: 39 mV

Input comparator bandwidth: (20mV overdrive voltage) 600 MHz

Input slew rate requirement: min. 0.5 V/ μ s

Input impedance: 50 Ω

External Channel Advance

Connector type: LEMOSA Series 00 NIM-CAMAC

Input voltage range: TTL

Input comparator bandwidth: (20mV overdrive voltage) 600 MHz

Input slew rate requirement: min. 0.5 V/ μ s

Input impedance: 4.7 k Ω pull-up

7.2.2. Analog Output

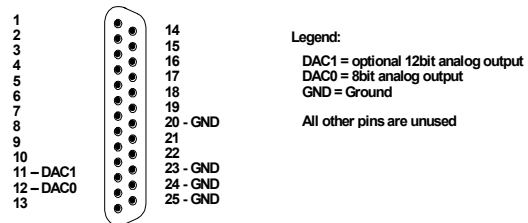


Figure 7.1: Analog Output Connector

RAMP DAC – (DAC1) analog signal output featuring a 12 bit analog voltage or X-ramp output

Voltage range 0 to + 10 V

Resolution 12 bit / 2.44 mV

8 Bit DAC – (DAC0) analog signal output

Voltage range jumper selectable \pm 2.5 V or 0 to + 10 V

Resolution 8 bit / 19.5 mV or 39 mV

7.3. Performance

7.3.1. General

Memory:	128k x 32 bit
Basic Operating Modes:	Multiscaling
Spectra Ranges:	256, 512, 1k, 2k, 4k, 8k, 16k, 32k, 64, or 128k
Spectra Offset:	512 to 130560 (= 128k - 512) programmable in steps of 512
Temperature Range:	0 - 50°C

7.3.2. MCS Mode

Dwell Time Modes:	internal, external, manual
Internal Dwell Time:	1 MCS input channel min. 125 ns 2 MCS input channels min. 250 ns max. 20 s step size: $N \times 10^M \times 125 \text{ ns}$ $N = 1 \text{ to } 16; M = 0 \text{ to } 7$
External Channel Advance:	rising edge sensitive
Pulse Width External Clock:	min. 50 ns synchronized to the internal oscillator clock
Count Rate Capability:	COUNT 1 & 2: falling edge sensitive min. 150 MHz
Pulse Width COUNT IN 1&2:	TBD
Pulse Width TRIGGER IN:	TBD
Interchannel Dead Time:	$< \pm 0.5 \text{ ns}$
End of Sweep Dead Time:	1 MCS input channel 93.75 ns 2 MCS input channels 218.75 ns Retrigger mode no additional dead time
Trigger Delay:	Complete sweep mode 171.875 ns Retrigger mode 546.875 ns
Trigger Uncertainty:	$\pm 15.625 \text{ ns}$
Trigger Modes:	complete sweeps, instant retrigger (break sweep), manual
Sweep Preset Range:	2 to 2^{32}

All technical data are for $\pm 5 \text{ V}$ power supply, FAST NIM input pulses and a 32 MHz system clock.

7.4. Power Requirements

Computer Power Supply	+ 5 V / $\pm 0.25 \text{ V}$ - 5 V / $\pm 0.25 \text{ V}$ @ 150mA
-----------------------	--

7.5. Physical

Size	$\frac{2}{3}$ length ISA bus PC-card, $\approx 218 \times 135 \text{ mm}$
------	---