MCS8

DLL Software Interface

User Manual

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

Version 1.01, January 29, 2021

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.

This manual contains proprietary information; no part of it may be reproduced by any means without prior written permission of FAST ComTec, Grünwalder Weg 28a, D 82041 Oberhaching, Germany. Tel: ++49 89 665180-0, FAX: ++49 89 665180-40.

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........6
    1.1 Some hints........6
2 Using the DMCS8 DLL from LabVIEW........8
    2.1 The MCS8test demo VI........8
    2.2 Getting Parameters........9
    2.3 Getting the Status........10
    2.4 Avoiding memory leaks........11
    2.5 Getting the Spectrum Data........12
    2.6 Executing a command........13
    2.7 Getting General MCS8 Settings........13
    2.8 Getting MCS8 System definition........15
    2.9 Getting the MCS8 Data Settings........15
    2.10 Getting the Replay Settings........16
    2.11 Getting the Cnt numbers........16
    2.12 Getting the Strings........17
    2.13 Getting the ROI boundaries........18
    2.14 Getting data from special views in MPANT like projections or slices........18
    2.15 Get data from rectangular ROI........19
    2.16 Get Data from polygonal ROI........20
    2.17 Get ROI Index from ROI name........20
    2.18 Delete ROI with given Index........21
    2.19 Get ROI sum........21
    2.20 Get ROI Name........21
3 Using the DMCS8.DLL from Visual Basic........22
    3.1 The Include File........22
    3.2 The Visual Basic demo program........24
4 Using the DMCS8.DLL from C........28
    4.1 The Include File........29
    4.2 The C demo program........33
5 Using the DMCS8.DLL from C#........38
    5.1 The C# demo program........38
6 Appendix: The DMCS8 DLL........46
    6.1 The Structures........46
    6.2 The Library Functions........53
    6.3 The Ordinal numbers of the functions........53
    6.4 The source code of the functions........54
    6.5 How to compile the DLL........54

Table of Figures

Fig. 2.1: The mcs8test VI..8
Fig. 2.2: Block diagram of mcs8test VI..9
Fig. 2.3: Recycling of memory using shift registers..11
Fig. 2.4 MCS28DAT.VI..12
Fig. 2.5: Data from special MPANT views like slices or projections...18
Fig. 3.1: The Visual Basic MCS8 Demo program...24
Fig. 4.1: The tstmcs8 Demo program written in C..28

1 Introduction

The MPANT software for the MCS8 consists of a hardware-dependent server program MCS8.EXE and a general graphics program MPANT.EXE that controls the hardware via a DLL DMCS8.DLL. Any other Windows application can also control the hardware via the DLL. To support the programming of such customer-specific user interfaces, as an option we deliver this documentation including source code and example programs for LabVIEW, Visual Basic, C and Delphi. The complete source code of the DLL that controls the hardware via the server program is included. A special FMPA3.DLL including source code allows to display calculated spectra including calculated error bars in the MPANT program.

The server program MCS8.EXE is a rather compact Windows application. It controls the hardware and data and allows to perform measurements with the system. MPANT is just a user interface to control the server program. It has access to the data in a shared memory region and can display spectra. It is not necessary that MPANT is running during an acquisition. It can be exited and restarted without stopping a running acquisition. If you don't want MPANT automatically started when starting the server, just rename the file MPANT.EXE in the working directory (default C:\MCS8 or C:\MCS8-x64) for example into MYMPANT.EXE.

The DLL DMCS8.DLL is an interface providing functions to communicate with the server program. Most of these functions send messages to the server as you do it when operating the server program by sending Windows messages via mouse clicks. Please do not expect any functions in this DLL for controlling the hardware directly. All software described in this manual requires that the server program is running. The DLL was mainly developed as an interface between the server program and MPANT, not as a nice developing tool for customers. But by looking at the programming examples and the following hints it should be easy to develop own programs that are able to control the server like MPANT does.

1.1 Some hints

The server program has a built-in command interpreter. The syntax of these commands is described in the MCS8 manual, chapter 5.2, and in the MPANT on-line help (look for: "How to use the command language.."). It is recommended to send commands like "range=16384" to the server via the RunCmd DLL function, i.e. RunCmd(0, "range=16384"); if you want to set parameters like a spectra length. The alternative method is to store all settings parameters into the DLL by calling the DLL function StoreSettingData(setting, 0); and then calling NewSetting(0); to send a message to the server to read new settings from the DLL. This method will not work for changing a spectra length to avoid the problem of any undefined memory pointers. The range parameter should be changed by the server program only (or by sending a "range=.." command). The recommended usage of the DLL is reading parameters like Status, Settings, Strings, Cnt numbers, ROI boundaries using the corresponding DLL functions, but for any actions or setting any parameters the command interpreter should be used.

If your application that controls the MCS8 server via the DLL is a true Windows application with a main window and corresponding message loop handling messages sent to this window, you can fetch a special Windows message to react immediately on actions like an acquisition status change without permanently polling the status:

Declare a global int MM_STATUS and somewhere when initializing your program register a Windows message using a code line like:

```
 MM_STATUS=RegisterWindowMessage("MCS8Status");
```

furthermore, declare somewhere in your headers constants

```
 #define ID_NEWSTATUS   162
 #define ID_NEWSETTING  139
 #define ID_NEWDATA     160
```

I assume your main Window Procedure is declared like

```
 DWORD WINAPI MyMainWndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)
```

you can then insert here code like

```
 if (msg == MM_STATUS) {
   if (wParam == ID_NEWSTATUS) {
      // status change, read acquisition status and react accordingly...
   }
   else if (wParam == ID_NEWDATA) {
     //  release all pointers, the server will reallocate some spectra..
   }
   else if (wParam == ID_NEWSETTING) {
     // the server has reallocated some spectra, get new pointers..
   }
 }
```

On any status change the server sends a NEW_STATUS message. The lParam value is usually zero, but after a stop of an acquisition the lParam is equal to 1, so your program is able to react accordingly.

It is important that the DLL is loaded first by the Server program and that it is loaded from the same path by all programs using it. Otherwise it does not work to access the shared memory. The dmcs8.dll is installed into the Windows\System32 directory or when using the 32-bit Software on a 64-bit Windows into Windows\SYSWOW64\. Please make sure that there is nowhere else any file dmcs8.dll. Start MCS8.exe by hand before starting your program, or by a call from your program for example like

```
{
  STARTUPINFO startupinfo = {0};
  PROCESS_INFORMATION procinfo = {0};
  startupinfo.cb = sizeof(STARTUPINFO);
  return CreateProcess ( "MCS8.EXE", NULL, NULL, NULL, FALSE, NORMAL_PRIORITY_CLASS, NULL,
            NULL, &startupinfo,&procinfo);
}
```

, but before your program loads the DLL. It is recommended not to link the DLL to your program using a dmcs8.LIB file, but explicitely load it at runtime as demonstrated in our example tstmcs8.c.

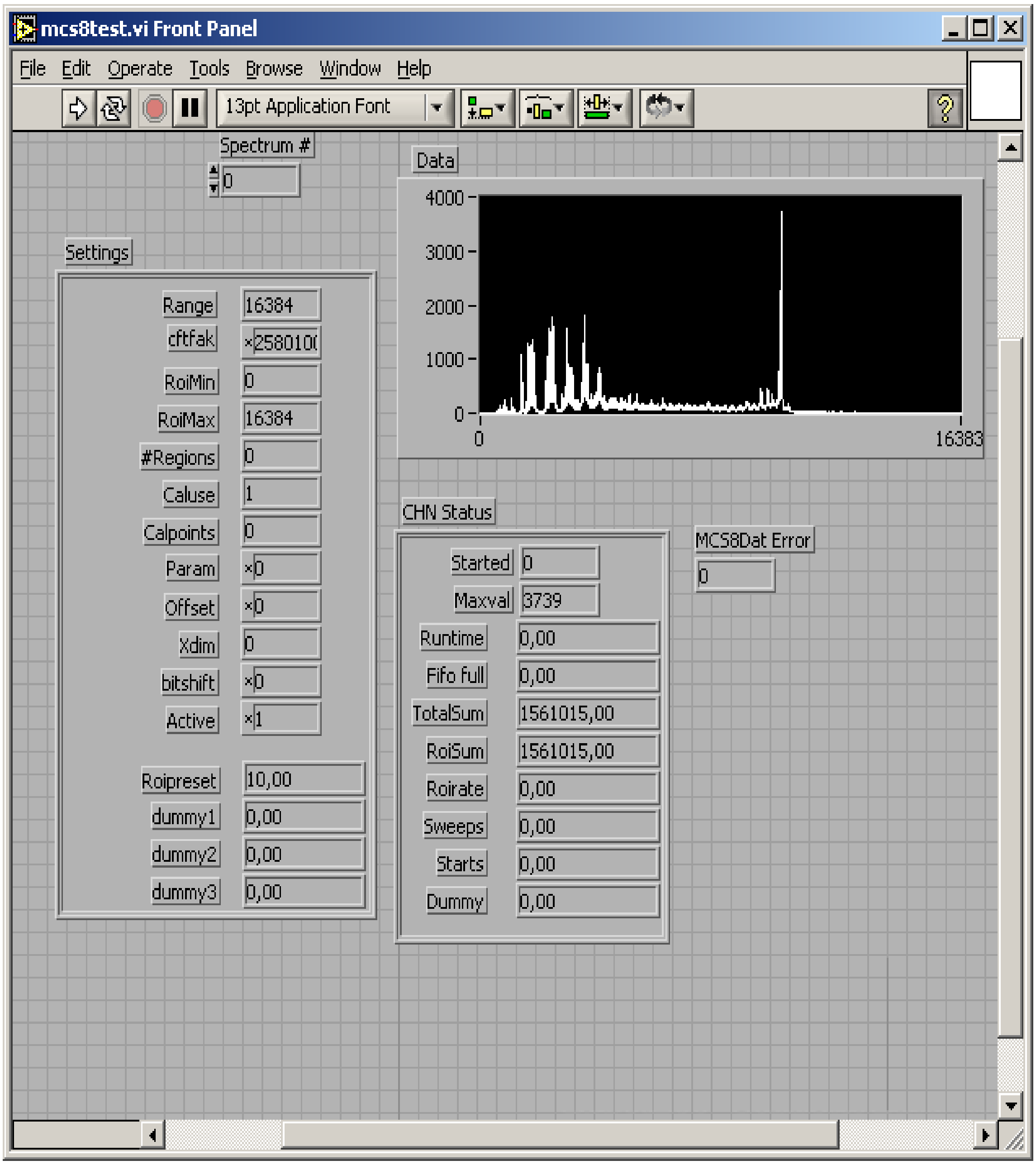
2 Using the DMCS8 DLL from LabVIEW

To access the MCS8 data directly from LabView via the DLL, some LabView VI's ("Virtual Instruments") contained in MCS8LV.LLB and the MCS8TEST.VI are provided.

2.1 The MCS8test demo VI

Files: mcs8lv.llb, mcs8test.vi

The distribution medium contains in a directory \lv2011 the following files: mcs8lv.llb and mcs8test.vi. The mcs8test.vi demonstrates access to spectra data. Please start now MCS8.EXE and then LabVIEW and open the mcs8test.vi (or just double click on mcs8test.vi). You may load some data like spec1.mpa with MPANT or the server and then run the VI.



Spectrum # 0

Settings

Range	16384
cftfak	×258010(
RoiMin	0
RoiMax	16384
#Regions	0
Caluse	1
Calpoints	0
Param	×0
Offset	×0
Xdim	0
bitshift	×0
Active	×1
Roipreset	10,00
dummy1	0,00
dummy2	0,00
dummy3	0,00

CHN Status

Started	0
Maxval	3739
Runtime	0,00
Fifo full	0,00
TotalSum	1561015,00
RoiSum	1561015,00
Roirate	0,00
Sweeps	0,00
Starts	0,00
Dummy	0,00

MCS8Dat Error 0

Fig. 2.1: The mcs8test VI

In the Windows menu, click on Show Diagram to display the diagram, and on Show Help Window to display the Help window.

The Demo VI contains the VI’s to get the settings, status and spectrum data.

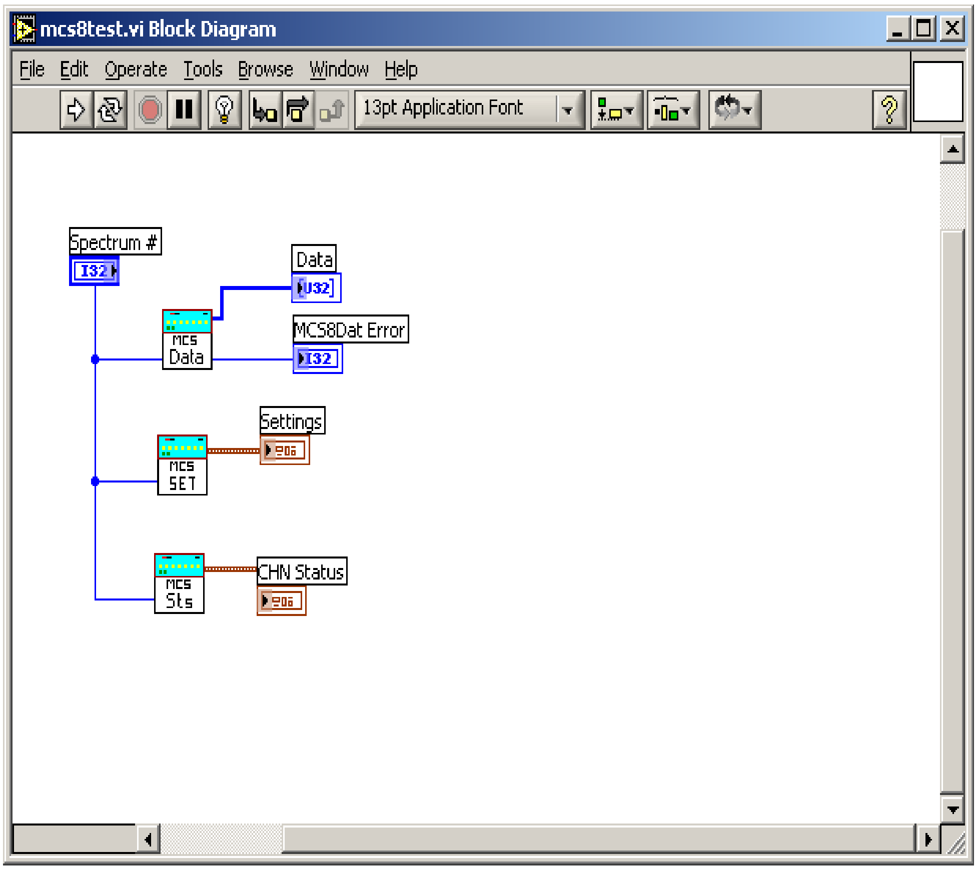
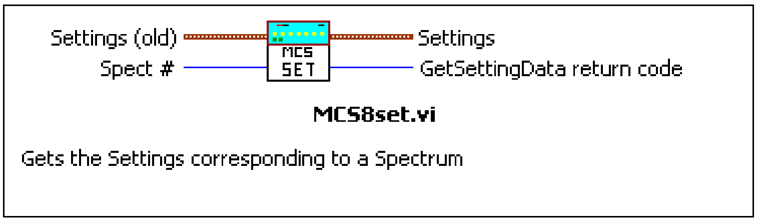


Fig. 2.2: Block diagram of mcs8test VI

2.2 Getting Parameters



MCS8set.vi

Gets the Settings corresponding to a Spectrum

The Settings are obtained with the MCS8set.vi. It results a 32 bit integer as an error code and the settings contained in a cluster.

You can use the help window to get information: on the front panel as the active window, just move the mouse over the item you are interested and observe the help window. The cluster has the components known from the DLL structure definitions:

1. Range (I32)
2. Prena (I32)
3. RoiMin (I32)
4. RoiMax (I32)
5. #Regions (I32)
6. Caluse (I32)
7. Calpoints (I32)
8. Param (hex) (I32)
9. Offset (hex) (I32)
10. Xdim (I32)
11. Timesh (I32)
12. Active(hex) (I32)

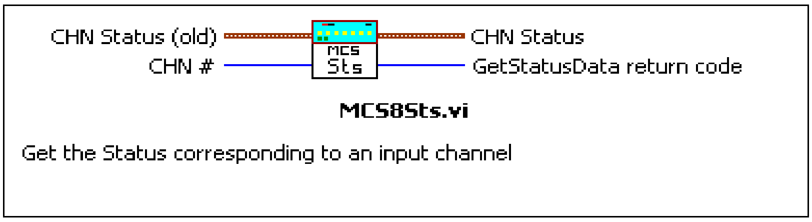
13. Roipreset (DBL) 14. Ltpreset (DBL)
15. TimeOffs (DBL) 16. Dwelltime (DBL)

For the detailed meaning of each parameter please refer to the LabView on-line help window or the DLL description in this manual.

The error code has the following meanings:

1: ok
2: No Parameters available

2.3 Getting the Status



MCS8Sts.vi

Get the Status corresponding to an input channel

The Status corresponding to an input channel is obtained with the MCS8Sts.VI. The Input is the Channel number with 0 for STOP1 and so on. It results a 32 bit integer as an error code and the status parameters contained in a cluster.

The cluster has the following components:

1: Started (I32):	0 == OFF, 1 == ON, 3 == READ OUT
2: Maxval (U32):	Maximum value in spectra, only available when stopped
3: Runtime (DBL)	in seconds
4: Fifo full (DBL)	counted number of datalost bits, only available for special dataword formats
5: TotalSum (DBL)	
7: RoiSum (DBL)	
8: RoiRate (DBL)	
9: Sweeps (DBL)	
10: Starts (DBL)	

The error code has the following meanings:

1: ok
0: No data available

2.4 Avoiding memory leaks

All VI’s returning parameters in a cluster structure have an “old” input for the same structure that is reallocated in the vi. When using repeated calling a status it is recommended to use it with a shift register added at the boarders of a “while – loop” structure to recycle the memory as shown in Fig. 2.3

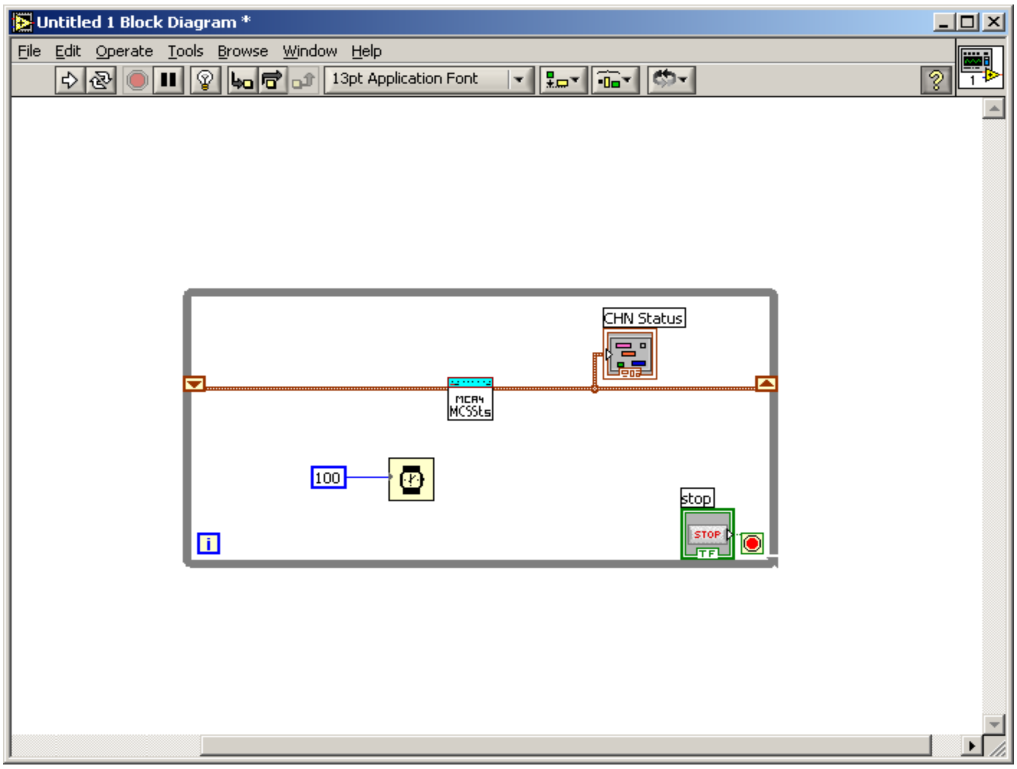
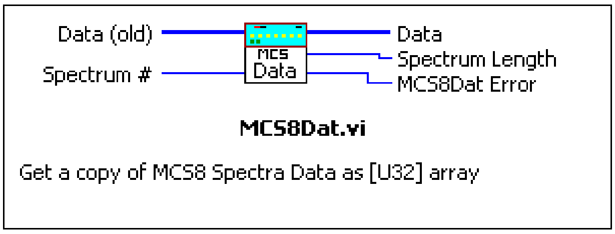

Untitled 1 Block Diagram *
File Edit Operate Tools Browse Window Help
13pt Application Font
CHN Status
MCA4
MCSSts
100
stop
STOP
TF
i

Fig. 2.3: Recycling of memory using shift registers

2.5 Getting the Spectrum Data



MCS8Dat.vi

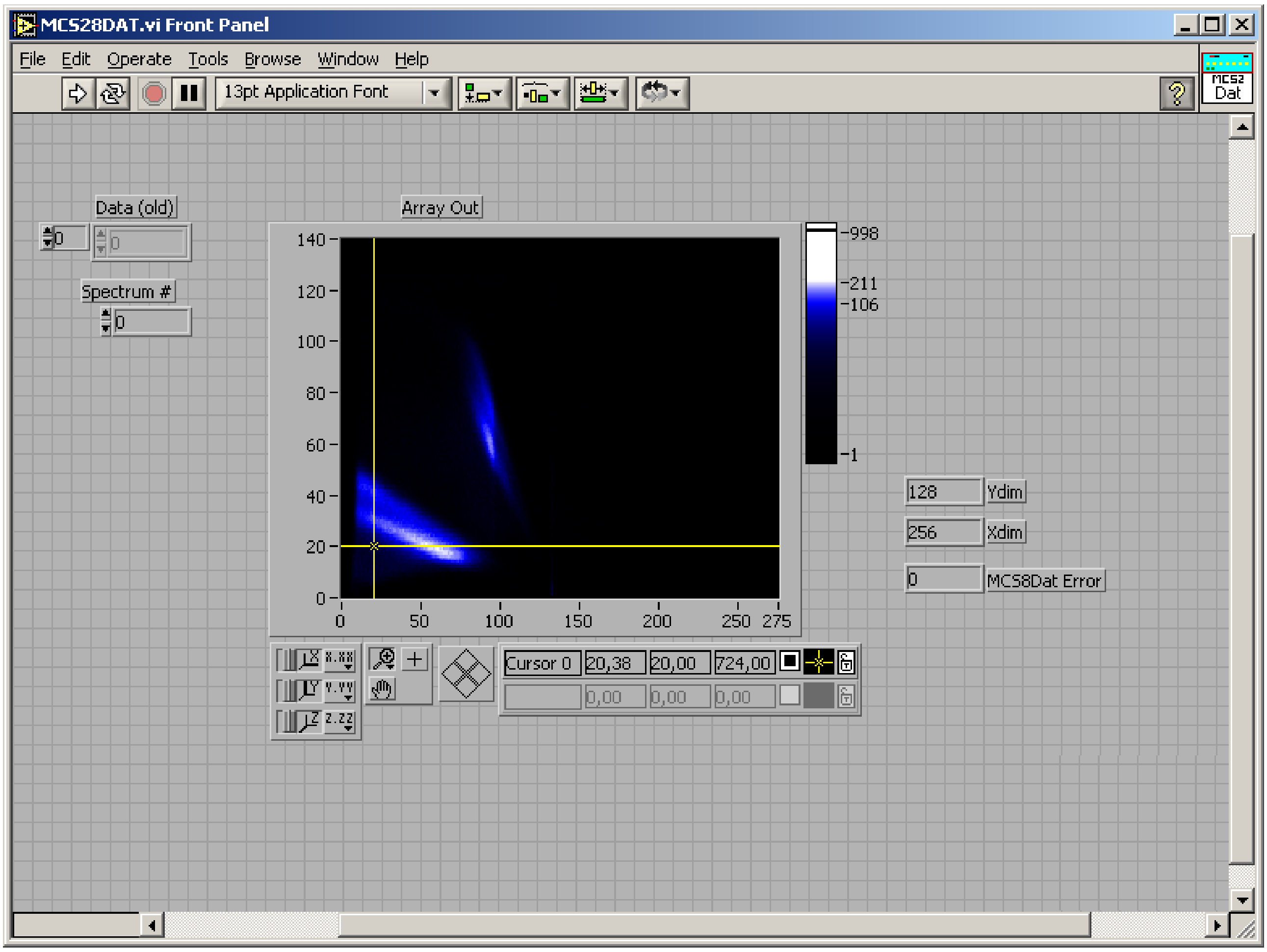
Get a copy of MCS8 Spectra Data as [U32] array

The Spectrum data is obtained with the MCS8Dat.vi. It results spectrum as a [U32] array, the spectrum length and an error codes as a 32 bit integer.

The error code has the following meanings:

0: ok
4: No Data available

How to use MCS8Dat.vi to get a display of a dualparameter spectra in LabVIEW is demonstrated in MCS28Dat.vi.



Data (old) 0 0

Spectrum # 0

128 Ydim

256 Xdim

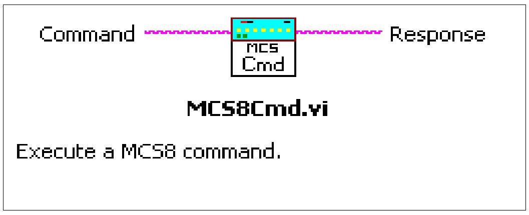
0 MCS8Dat Error

Cursor 0 20,38 20,00 724,00

0,00 0,00 0,00

Fig. 2.4 MCS28DAT.VI

2.6 Executing a command

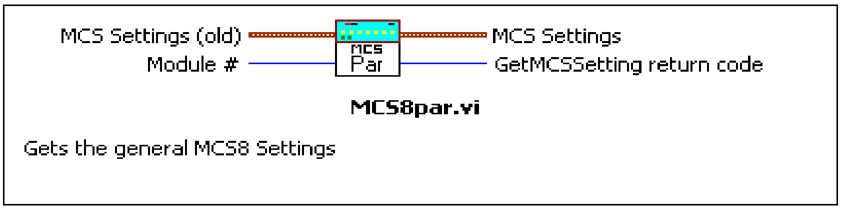


MCS8Cmd.vi

Execute a MCS8 command.

The MCS8LV.LLB contains some more VIs that are not used by MCS8test.VI. Any command for the MCS8 server can be executed by MCS8Cmd.VI. It results an response string.

2.7 Getting General MCS8 Settings



MCS8par.vi

Gets the general MCS8 Settings

The MCS Settings are obtained with the MC8par.vi. It results a 32 bit integer as an error code and the settings contained in a cluster.

You can use the help window to get information: on the front panel as the active window, just move the mouse over the item you are interested and observe the help window. The cluster has the components defined in the DLL structure definition of BOARDSETTING:

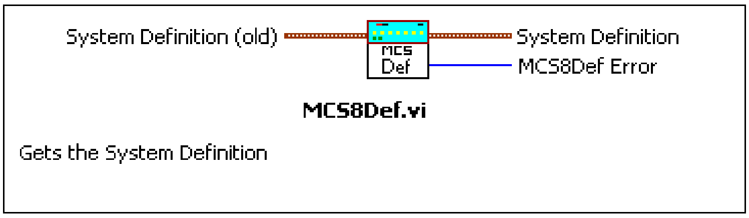
1. sweepmode (I32) // sweepmode & 0xF: 0 = normal,
// 1=differential (relative to first stop in sweep)
// 4=sequential
// 5=seq.+diff (Ch1), bit0 = differential mode
// 6 = CORRELATIONS
// 7 = diff.+Corr.
// 9=differential to stop in Ch2, bit3 = Ch2 ref (diff.mode)
// 0xF = Corr.+diff (Ch2)
// bit 4: Softw. Start
// bit 5: "Don't show" tagbits
// bit 6: Endless
// bit 7: Start event generation
// bit 8: Enable Tag bits
// bit 9: start with rising edge
// bit 10: time under threshold for pulse width
// bit 11: pulse width mode for any spectra with both edges enabled
// bit 12: abandon Sweepcounter in Data
// bit 13: "one-hot" mode with tagbits
// bit 14: start ref (diff.mode)
// bit 15: enable start input
// bit 16..bit 22 ~(input channel enable)
// bit 24: require data lost bit in data
// bit 25:
// bit 27: Folded
// bit 28: Interleaved

2. prena (I32) // bit 0: realtime preset enabled
// bit 1:

```
                                // bit 2: sweep preset enabled
                                // bit 3: ROI preset enabled
                                // bit 4: Starts preset enabled
                                // bit 5: ROI2 preset enabled
                                // bit 6: ROI3 preset enabled
                                // bit 7: ROI4 preset enabled
                                // bit 8: ROI5 preset enabled
                                // bit 9: ROI6 preset enabled
                                // bit 10: ROI7 preset enabled
                                // bit 11: ROI8 preset enabled
3. cycles (U32)                 // for sequential mode, = y-dimension
4. sequences (U32)              // for sequential mode, how often to repeat filling the array
5. syncout (U32)                // LOWORD: sync out; bit 0..5 NIM syncout, bit 8..13 TTL syncout
                                // bit7: NIM syncout_invert, bit15: TTL syncout_invert
                                // 0="0", 1=5 MHz, 2=50 MHz, 3=100 MHz, 4=97.656 MHz,
                                // 5=195.625 MHz, 6= 195 MHz (int ref), 7=Start, 8=Ch1, 9=Ch2, 10=Ch3,
                                // 11=Ch4, 12=Ch5, 13=Ch6, 14=Ch7, 15=GO, 16=Start_of_sweep,
                                // 17=Armed, 18=SWEEP_ON, 19=WINDOW, 20=HOLD_OFF,
                                // 21=EOS_DEADTIME, 22=TIME[0],...,51=TIME[29],
                                // 52...63=SWEEP[0]..SWEEP[11]
6. digio (U32)                  // LOWORD: Use of Dig I/O, GO Line:
                                // bit 0: status dig 0..3
                                // bit 1: Output digval and increment digval after stop
                                // bit 2: Invert polarity
                                // bit 3: Push-Pull output, not possible
                                // bit 4:  Start with Input Dig 4
                                // bit 5:  Start with Input GO
                                // bit 8: GOWATCH
                                // bit 9: GO High at Start
                                // bit 10: GO Low at Stop
                                // bit 11: Clear at triggered start
                                // bit 12: Only triggered start
7. digval (U32)                 // digval=0..255 value for samplechanger
8.  dac0 (U32)                  //  DAC0 value (START) bit 16: Start with rising edge
9.  dac1 (U32)                  //  DAC1 value (STOP 1)
10. dac2 (U32)                  //  DAC2 value (STOP 2)
11. dac3 (U32)                  //  DAC3 value (STOP 3)
12. dac4 (U32)                  //  DAC4 value (STOP 4)
13. dac5 (U32)                  //  DAC5 value (STOP 5)
14. dac6 (U32)                  //  DAC6 value (STOP 6)
                                // bit (14,15) of each word: 0=falling, 1=rising, 2=both, 3=both+CFT
                                // bit 17 of each: pulse width mode under threshold
15. tagbits (U32)       // number of tagbits
16. extclk (U32)        // use external clock
17. periods (U32)       // number of periods in folded mode, sweeplength = range * periods
18. serno (U32)         // serial number
19. ddruse (U32)        // bit0: DDR_USE, bit1: DDR_2GB
                        // bits[2:3]: usb_usage
                        // bits[4:5]: wdlen
20. active (U32)        // module in system
21. holdafter (DBL)     // hold off
22. swpreset (DBL)      // sweep preset value
23. fstchan (DBL)       // acquisition delay
24. timepreset (DBL)    // time preset
```

Information about the parameters can be found in the on-line help window or the description of the BOARDSETTING structure in the DLL description in this manual.

2.8 Getting MCS8 System definition



Gets the System Definition

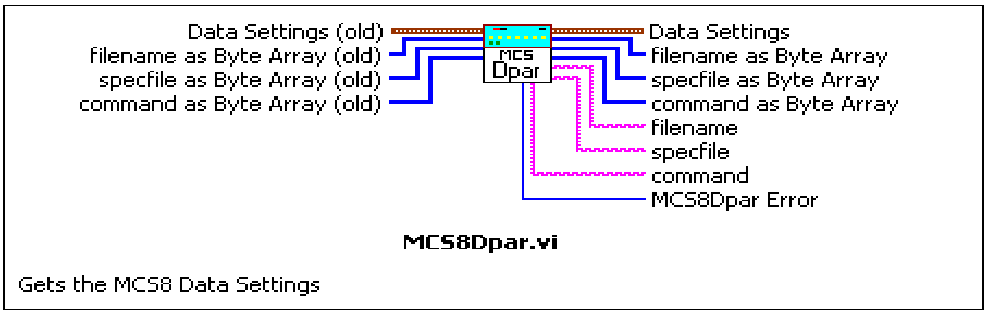
The MCS8 System Definition are obtained by MCS8Def.vi. It results the System Definition parameters in a cluster and a 32 bit integer as an error code. The System Definition has following components:

1. nDevices (I32) nDevices is the number of physical channels (6)
2. nDisplays (I32) nDisplays is the number of spectra = 6 + calc. spectra
3. nSystems (I32) nSystems the number of independent systems which is 1 in the present version
4. bRemote (I32) bRemote indicates whether the MCS8 server is controlled by MPANT
5. sys (I32) System definition word, 0 for applications with only one module.

The error code has the following meanings:

0: ok
-1: No data available

2.9 Getting the MCS8 Data Settings



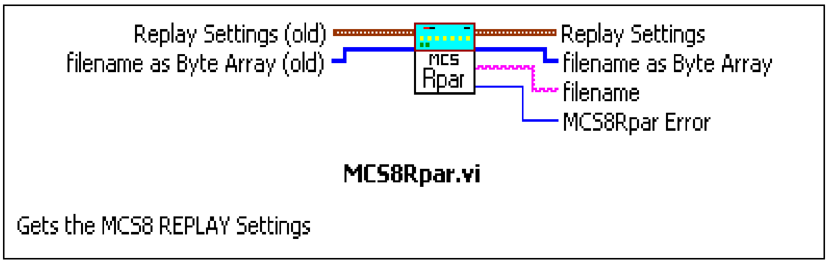
Gets the MCS8 Data Settings

The MCS8 Data Settings can be obtained by MCS8Dpar.vi. It results the Data Settings in a cluster and a 32 bit integer as an error code. The Data Settings are:

1. savedata (I32): 1 means auto save after stop
2. autoinc (I32): autoincrement MPA data filename
3. fmt (I32): MPA format type (0=ASCII, 1=binary, 2=CSV)
4. sepfmt (I32): format type for seperate spectra
5. sephead (I32) 1 means seperate header
6. smpts (I32) number of points for smoothing operation
7. caluse (I32) 1 means using calibration for shifted spectra summing
8. filename (STR) MPA data file name
9. specfile (STR) spectrum file name
10. command (STR)

The error code has the following meanings:
0: ok
-1: No data available

2.10 Getting the Replay Settings



MCS8Rpar.vi

Gets the MCS8 REPLAY Settings

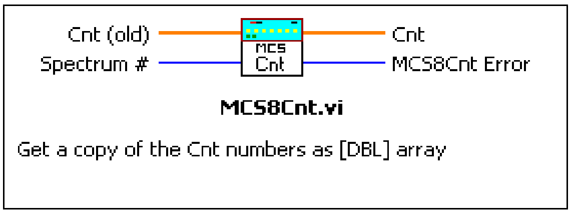
The Replay Settings can be obtained by MCS8Rpar.VI. It results a 32 bit integer as an error code and the Replay Settings:

1. use (I32):	1 if Replay Mode ON
2. modified (I32):	1 if different settings are used from measurement time
3. limit (I32):	0=all, 1 = limited time range
4. speed (I32):	replay speed in units of 100 kByte per sec
5. timefrom (DBL):	first time (sec)
6. timeto (DBL):	last time (sec)
7. timepreset (DBL):	last time - first time
8. filename (STR):	listfile for replay

The return code has the following meanings:

0: ok
-1: No data available

2.11 Getting the Cnt numbers



MCS8Cnt.vi

Get a copy of the Cnt numbers as [DBL] array

The Cnt numbers enable a very fast access to the acquisition status. They are very often actualized and are directly accessible via shared memory, whereas the status structure block is saved into the DLL in fix time intervals and is therefore available only after some delay.

The Cnt numbers can be obtained by MCS8Cnt.VI. It needs as input the spectrum number and results a 32 bit integer as an error code and the Cnt numbers as an array of 448 DBL containing the Cnt numbers.

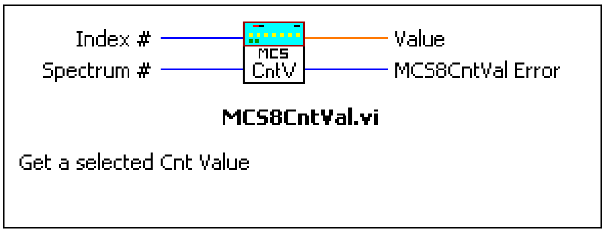
Array of 448 DBL containing the Cnt numbers.

Cnt[0] = Realtime,
Cnt[1] = Totalsum,
Cnt[2] = ROIsum,
Cnt[3] = ROIrate,
Cnt[4]
Cnt[5] = Sweeps,
Cnt[6]
Cnt[7] = cycle cnt
Cnt[8] = sequence cnt
Cnt[9]
Cnt[10] = start time

Cnt[11] = c0 cal. coeff.,
Cnt[12] = c1,
Cnt[13] = c2,
Cnt[14] = c3, for scope displays: vpp (mV) / 65535
Cnt[15] for spectrum# =0: bit 0: system 1 started
bit 1: system 2 started,
bit 2: system 3 started,
bit 3: system 4 started
Cnt[16]
Cnt[17] = FIFOCNT, indicates filling of large FIFO in units of bytes
Cnt[18]
Cnt[19] = calch0,
Cnt[35] = calval0, calib. Points
Cnt[20] = calch1,
Cnt[36] = calval1,
…
Cnt[64]..Cnt[191] : Peak values in corresponding Roi 0..127 for Calibration
Cnt[192], Cnt[193] ,... Cnt[447]: Roi Sum and Roi Net Sum in corresponding Roi 0 (, 1, ..127)
(actualized by MPANT when selected)

The error code has the following meanings:

0: ok
4: No Data available

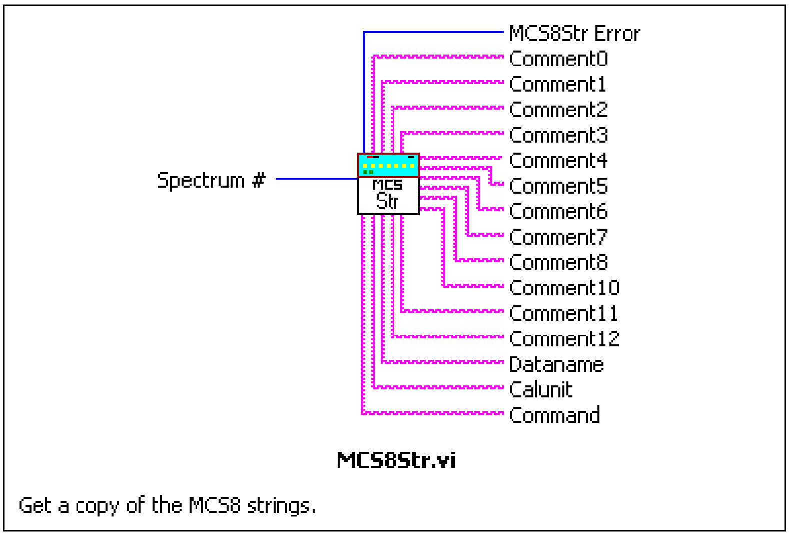


MCS8CntVal.vi

Get a selected Cnt Value

A selected Cnt Value can be obtained by MCS8CntVal.VI. Inputs are the spectrum number and the index into the Cnt array. It results a 32 bit integer as an error code and the selected Cnt value as DBL.

2.12 Getting the Strings



MCS8Str.vi

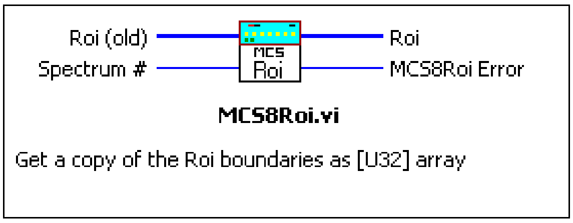
Get a copy of the MCS8 strings.

The Strings can be obtained by MCS8Str.VI. It results a 32 bit integer as an error code and the Strings.

The error code has the following meanings:

0: ok
4: No Data available

2.13 Getting the ROI boundaries



MCS8Roi.vi

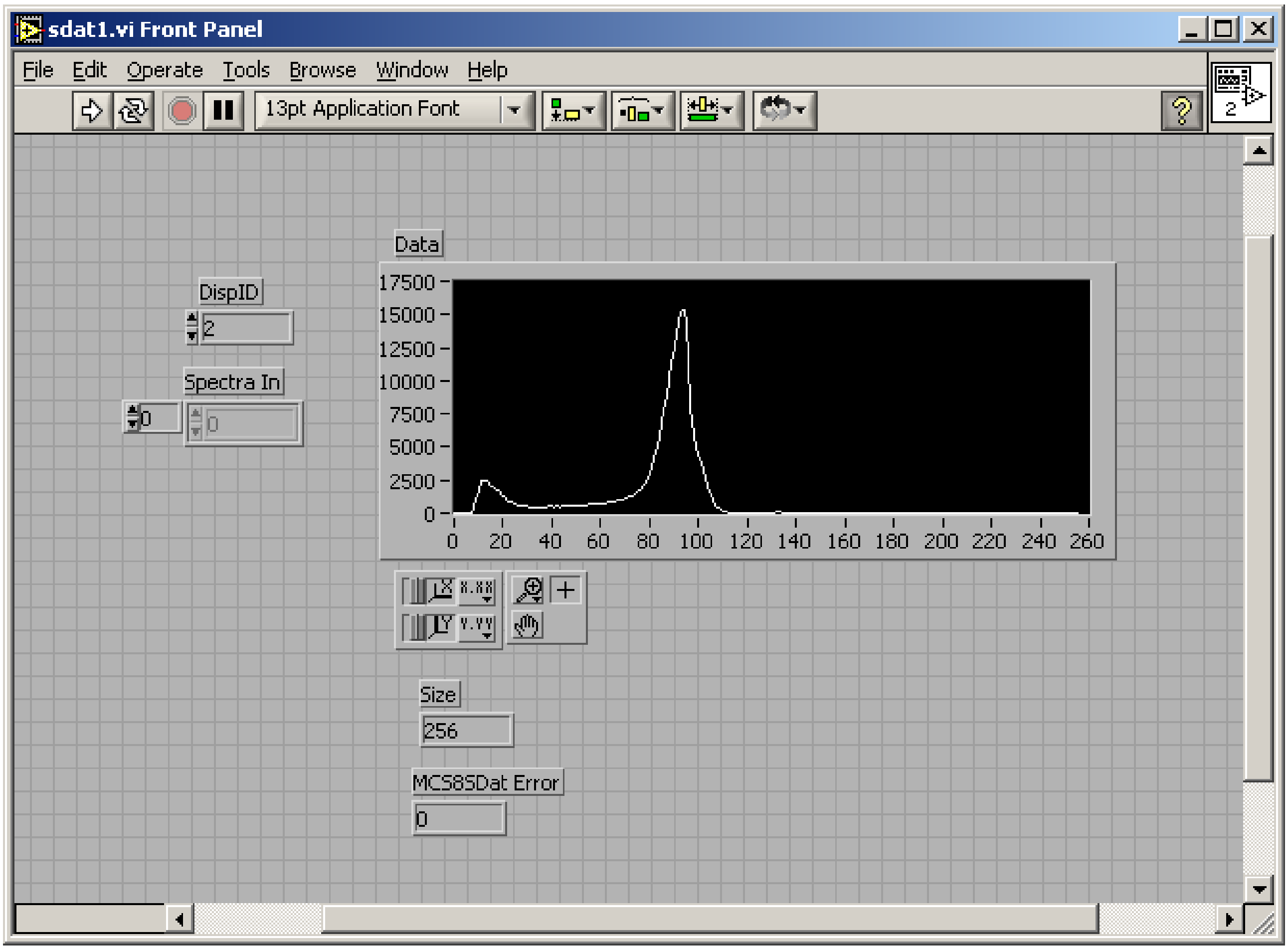
Get a copy of the Roi boundaries as [U32] array

The ROI boundaries can be obtained by MCS8Roi.VI. It results a 32 bit integer as an error code and the ROI boundaries for single spectra or rectangle ROIs for dual parameter spectra, respectively, contained in a [U32] array.

The error code has the following meanings:

0: ok
4: No Data available

2.14 Getting data from special views in MPANT like projections or slices



DispID: 2

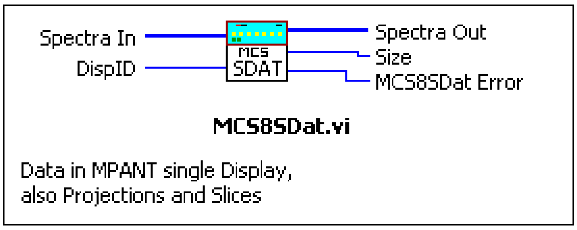
Spectra In: 0 0

Size: 256

MCS8SDat Error: 0

Fig. 2.5: Data from special MPANT views like slices or projections

SDAT1.VI contained in MCS8LV.LLB demonstrates how to access special single spectra like slices or projections of ROIs calculated within the MPANT program. Using the MPANT software create a projection of a ROI or a slice from a dual parameter spectra. Then start this VI, enter the Display Id shown within curved brackets in the title bar of the display window in MPANT and run the VI:The SDAT1.VI demonstrates the use of MCS8SDat.VI contained in MCS8LV.LLB.



MCS8SDat.vi

Data in MPANT single Display,
also Projections and Slices

SDAT.VI has the following inputs:

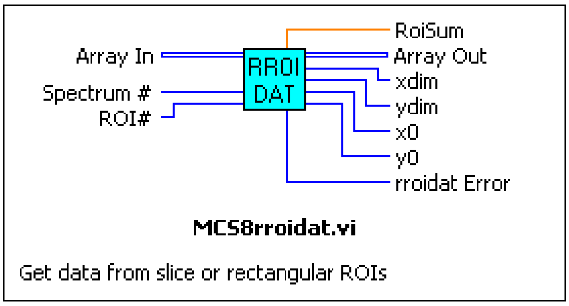
DispID: (I32) ID number of the display view as shown in round brackets in the title bar of the Window in MPANT

Spectra In: One dimensional array [I32] to be resized. May be left open.

The outputs of SDAT.VI are:

Spectra Out: One dimensional array [I32] containing the data.
Size: dimension (I32) of the Array.
Sdat Error (I32): 0: ok
-4: MPANT not running
-5: no data.

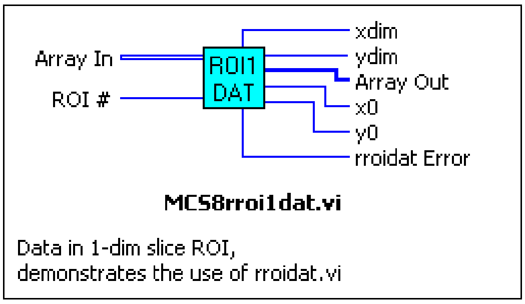
2.15 Get data from rectangular ROI



MCS8rroidat.vi

Get data from slice or rectangular ROIs

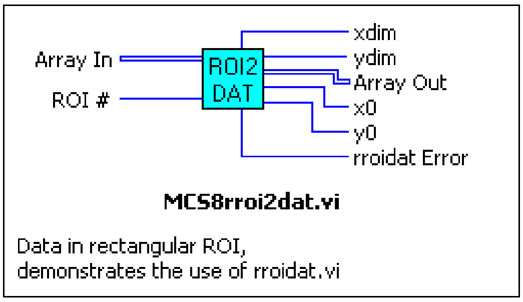
MCS8rroidat can be used to get data inside a rectangular or slice ROI. Enter the Spectrum # and the ROI number, for example 1 for the first TOF spectrum and 1 for the first ROI inside that spectrum.



MCS8rroi1dat.vi

Data in 1-dim slice ROI,
demonstrates the use of rroidat.vi

MC8rroi1dat.vi demonstrates the use of MCS8rroidat.vi for single spectra.



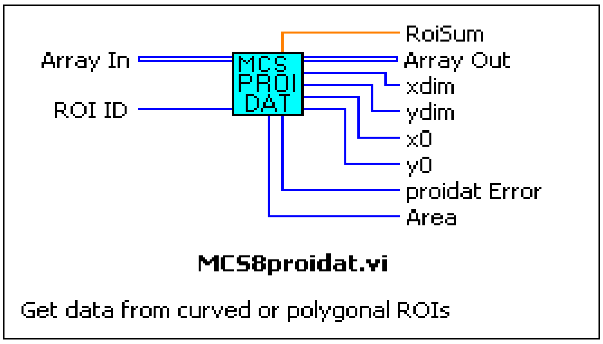
MCS8rroi2dat.vi

Data in rectangular ROI,
demonstrates the use of rroidat.vi

MCS8rroi2dat.vi demonstrates the use of MCS8rroidat.vi for rectangular ROIs in dual parameter spectra.

2.16 Get Data from polygonal ROI

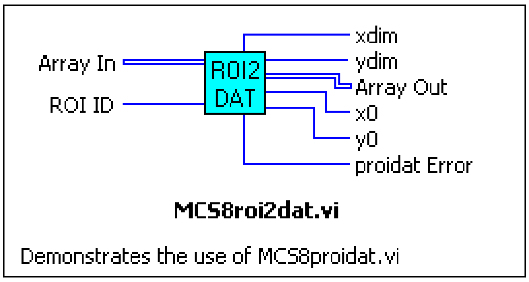
MCS8proidat.vi can be used to get data inside a two-dimensional polygonal ROI. The ROI ID to enter is the number like 900 shown in the MPANT display.



MCS8proidat.vi

Get data from curved or polygonal ROIs

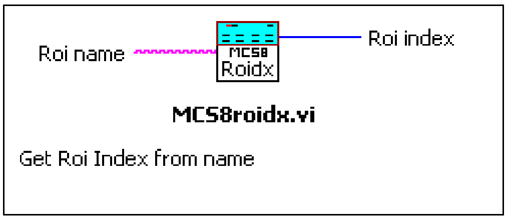
MCS8roi2dat.vi demonstrates the use of MCS8proidat.vi



MCS8roi2dat.vi

Demonstrates the use of MCS8proidat.vi

2.17 Get ROI Index from ROI name



MCS8roidx.vi

Get Roi Index from name

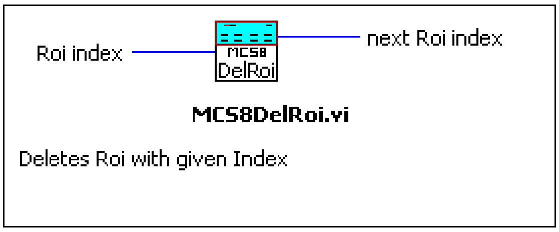
MCS8roidx.vi gets a unique index to address ROIs from named ROI's.

Rectangular or 1D ROIs: LOWORD is the spectra number, HIWORD is the ROI number (1,2,...).

Polygonal ROIs: LOWORD is an entry number, HIWORD is the roiid = 100 * spectra_number + ROI_number.

returns 0 if not found.

2.18 Delete ROI with given Index



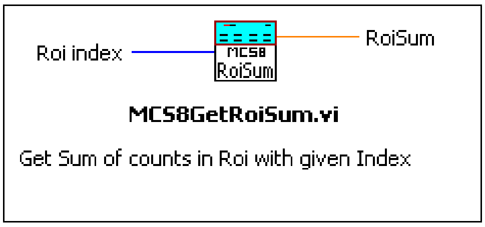
MCS8DelRoi.vi

Deletes Roi with given Index

MCS8DelRoi.vi deletes a ROI with given index.

A combined DelNamedROI.vi in MCS8.LLB can be used to delete a named ROI.

2.19 Get ROI sum



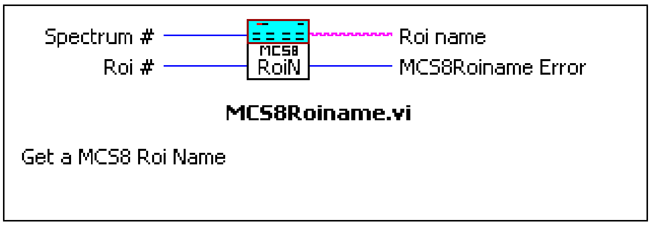
MCS8GetRoiSum.vi

Get Sum of counts in Roi with given Index

MCS8GetRoiSum.vi gets the sum of counts in a ROI with given index.

A combined NamedRoiSum.vi in MCS8.LLB can be used to get the ROISum of a named ROI.

2.20 Get ROI Name



MCS8Roiname.vi

Get a MCS8 Roi Name

MCS8Roiname.vi gets the name of a ROI in a spectra. ROI # 0...127 are 2-point (rectangular or slice) ROIs, ROI #128...191 are polygonal ROIs. The error code has the following meanings:

0: ok

4. no data available

3 Using the DMCS8.DLL from Visual Basic

3.1 The Include File

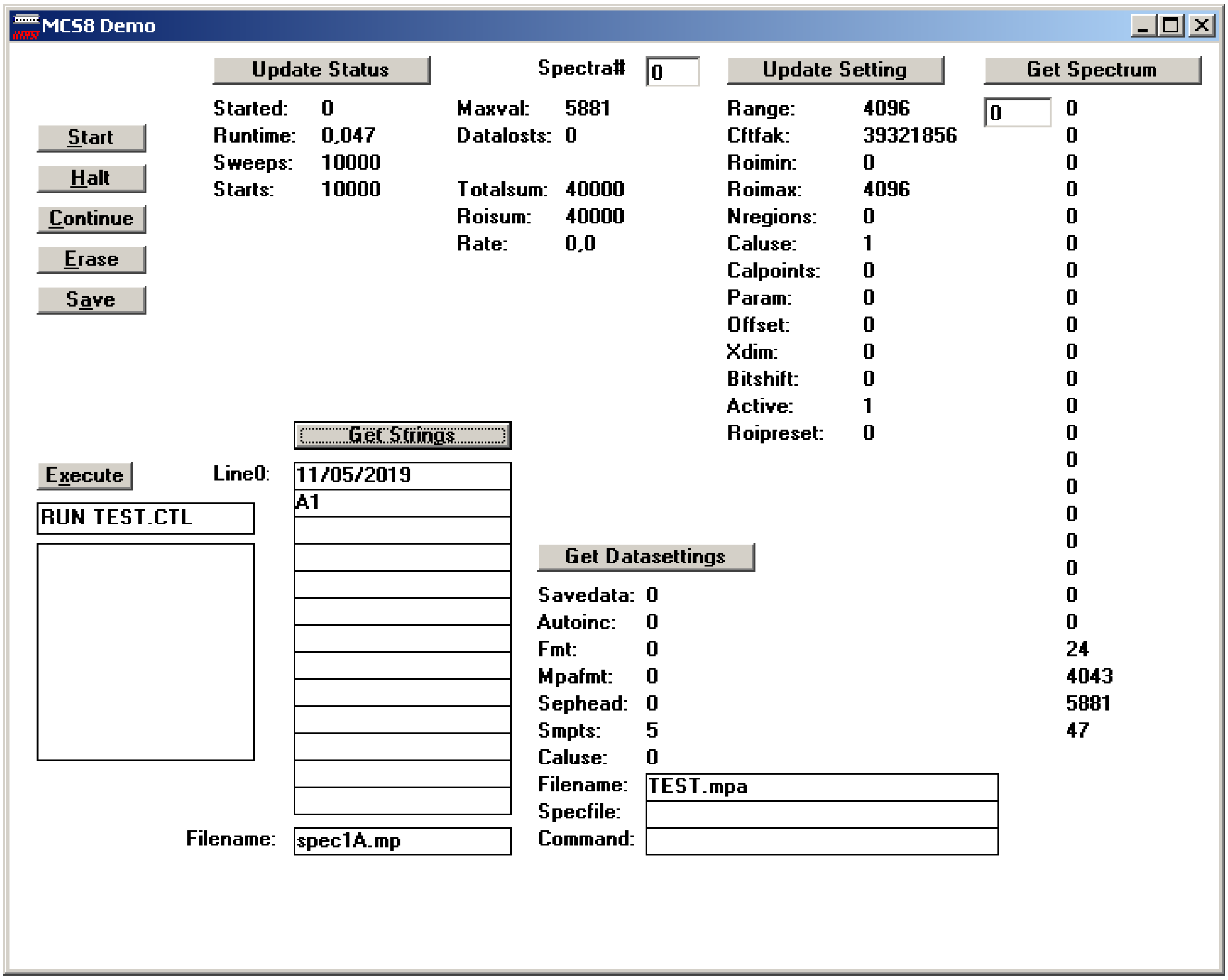
The include file DECLMCS8.BAS contains the structure and function definitions of the DLL.

```
Attribute VB_Name = "DECLMCS8"
Type Acqstatus
    Val As Long
    Val1 As Long
    Cnt(0 To 7) As Double
End Type

Type Acqsetting
    Range As Long
    Cftfak As Long
    Roimin As Long
    Roimax As Long
    Nregions As Long
    Caluse As Long
    Calpoints As Long
    Param As Long
    Offset As Long
    Xdim As Long
    Bitshift As Long
    Active As Long
    Roipreset As Double
    Dummy1 As Double
    Dummy2 As Double
    Dummy3 As Double
End Type

Type Replaysetting
    Use As Long
    Modified As Long
    Limit As Long
    Speed As Long
    Timefrom As Double
    Timeto As Double
    Timepreset As Double
    Filename As String * 256
End Type

Type Datsetting
    SaveData As Long
    Autoinc As Long
    Fmt As Long
    Mpafmt As Long
    Sephead As Long
    Smpts As Long
    Caluse As Long
    Filename As String * 256
    Specfile As String * 256
    Command As String * 256
End Type

Type Boardsetting
    Sweepmode As Long
    Prena As Long
    Cycles As Long
    Sequences As Long
    Syncout As Long
    Digio As Long
    Digval As Long
    Dac0 As Long
    Dac1 As Long
    Dac2 As Long
    Dac3 As Long
    Dac4 As Long
    Dac5 As Long
    Fdac As Long
    Tagbits As Long
    Extclk As Long
    Maxchan As Long
    Serno As Long
    Ddruse As Long
    Active As Long
    Holdafter As Double
    Swpreset As Double
    Fstchan As Double
    Timepreset As Double
End Type

Type Acqdef
  Ndevices As Long
  Ndisplays As Long
  Nsystems As Long
  Bremote As Long
  Sys As Long
  Sys0(56) As Long
  Sys1(56) As Long
End Type

Declare Sub StoreSettingData Lib "DMCS8.DLL" Alias "#2" (Setting As Acqsetting, ByVal Ndisplay As Long)
Declare Function GetSettingData Lib "DMCS8.DLL" Alias "#3" (Setting As Acqsetting, ByVal Ndisplay As
Long) As Long
Declare Function GetStatusData Lib "DMCS8.DLL" Alias "#5" (Status As Acqstatus, ByVal Ndevice As Long)
As Long
Declare Sub Start Lib "DMCS8.DLL" Alias "#6" (ByVal Nsystem As Long)
Declare Sub Halt Lib "DMCS8.DLL" Alias "#7" (ByVal Nsystem As Long)
Declare Sub Continue Lib "DMCS8.DLL" Alias "#8" (ByVal Nsystem As Long)
Declare Sub NewSetting Lib "DMCS8.DLL" Alias "#9" (ByVal Ndisplay As Long)
Declare Function ServExec Lib "DMCS8.DLL" Alias "#10" (ByVal Clwnd As Long) As Long
Declare Function GetSpec Lib "DMCS8.DLL" Alias "#13" (ByVal I As Long, ByVal Ndisplay As Long) As Long
Declare Sub SaveSetting Lib "DMCS8.DLL" Alias "#14" ()
Declare Function GetStatus Lib "DMCS8.DLL" Alias "#15" (ByVal Ndevice As Long) As Long
Declare Sub EraseData Lib "DMCS8.DLL" Alias "#16" (ByVal Nsystem As Long)
Declare Sub SaveData Lib "DMCS8.DLL" Alias "#17" (ByVal Ndevice As Long, ByVal All As Long)
Declare Sub GetBlock Lib "DMCS8.DLL" Alias "#18" (Hist As Long, ByVal Start As Long, ByVal Size As Long,
ByVal Stp As Long, ByVal Ndisplay As Long)
Declare Function GetDefData Lib "DMCS8.DLL" Alias "#20" (Def As Acqdef) As Long
Declare Sub LoadData Lib "DMCS8.DLL" Alias "#21" (ByVal Ndevice As Long, ByVal All As Long)
Declare Sub NewData Lib "DMCS8.DLL" Alias "#22" ()
Declare Sub HardwareDlg Lib "DMCS8.DLL" Alias "#23" (ByVal Item As Long)
Declare Sub UnregisterClient Lib "DMCS8.DLL" Alias "#24" ()
Declare Sub DestroyClient Lib "DMCS8.DLL" Alias "#25" ()
Declare Sub RunCmd Lib "DMCS8.DLL" Alias "#28" (ByVal Ndevice As Long, ByVal Cmd As String)
Declare Sub AddData Lib "DMCS8.DLL" Alias "#29" (ByVal Ndisplay As Long, ByVal All As Long)
Declare Function LVGetRoi Lib "DMCS8.DLL" Alias "#30" (Roi As Long, ByVal Ndisplay As Long) As Long
Declare Function LVGetCnt Lib "DMCS8.DLL" Alias "#31" (Cnt As Double, ByVal Ndisplay As Long) As Long
Declare Function LVGetOneCnt Lib "DMCS8.DLL" Alias "#32" (Cnt As Double, ByVal Ndisplay As Long,
ByVal Cntnum As Long) As Long
Declare Function LVGetStr Lib "DMCS8.DLL" Alias "#33" (ByVal Comment As String, ByVal Ndisplay As
Long) As Long
Declare Sub SubData Lib "DMCS8.DLL" Alias "#34" (ByVal Ndisplay As Long, ByVal All As Long)
Declare Sub Smooth Lib "DMCS8.DLL" Alias "#35" (ByVal Ndisplay As Long)
```

Declare Function GetMCSSetting Lib "DMCS8.DLL" Alias "#39" (Msetting As Boardsetting) As Long
Declare Function GetDatSetting Lib "DMCS8.DLL" Alias "#41" (Dsetting As Datsetting) As Long
Declare Function GetReplaySetting Lib "DMCS8.DLL" Alias "#43" (Rsetting As Replaysetting) As Long

3.2 The Visual Basic demo program

The simple Visual Basic program is shown here: It allows to get Status, Settings, spectrum data and strings for any MCS8 spectra, perform actions like start, halt, continue, erase, or any control command. It is essential that the DLL is loaded from Visual Basic and the MCS8 server program from the same path. Let it in the folder where it was installed by the MPANT setup program and don't copy or move it to another path.



Spectra# 0

Started: 0
Runtime: 0,047
Sweeps: 10000
Starts: 10000

Maxval: 5881
Datalosts: 0
Totalsum: 40000
Roisum: 40000
Rate: 0,0

Range: 4096
Cftfak: 39321856
Roimin: 0
Roimax: 4096
Nregions: 0
Caluse: 1
Calpoints: 0
Param: 0
Offset: 0
Xdim: 0
Bitshift: 0
Active: 1
Roipreset: 0

0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
24
4043
5881
47

RUN TEST.CTL

Line0: 11/05/2019
A1

Filename: spec1A.mp

Savedata: 0
Autoinc: 0
Fmt: 0
Mpafmt: 0
Sephead: 0
Smpts: 5
Caluse: 0
Filename: TEST.mpa
Specfile:
Command:

Fig. 3.1: The Visual Basic MCS8 Demo program

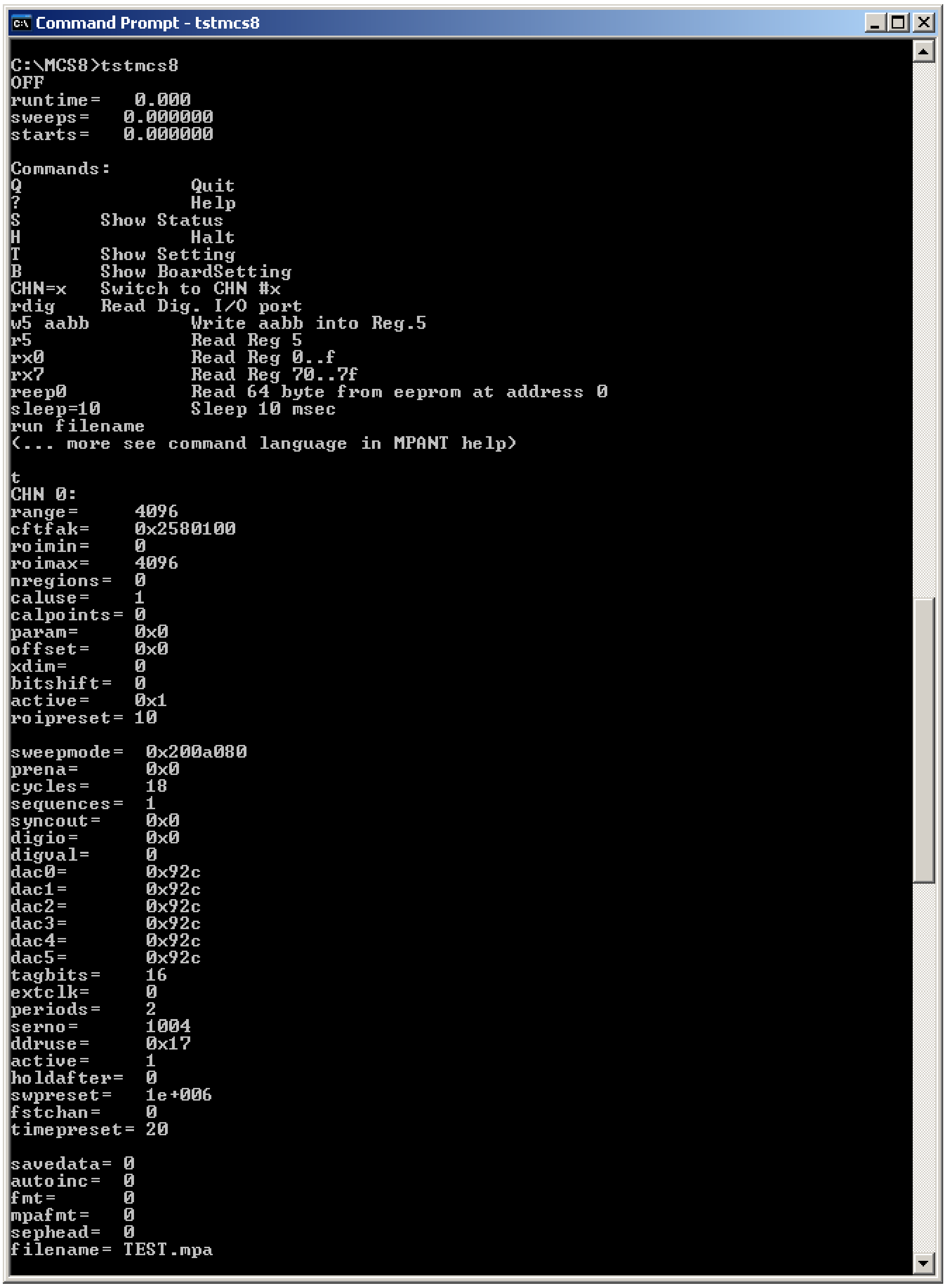
This is the complete program code beside form data:

```
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Dim Status As Acqstatus
Dim Setting As Acqsetting
Dim Dsetting As Datsetting
Dim OldStarted As Integer
Dim Mcano As Long
Dim Sysno As Long
Dim Chan As Long
Dim Hist(24) As Long
Dim Toggle As Integer


Private Sub CommandContinue_Click()
  Call Continue(0)
End Sub

Private Sub CommandDatasettings_Click()
  Call GetDatSetting(Dsetting)
  LabelSavedata.Caption = Dsetting.SaveData
  LabelAutoinc.Caption = Dsetting.Autoinc
  LabelFmt.Caption = Dsetting.Fmt
  LabelMpafmt.Caption = Dsetting.Mpafmt
  LabelSephead.Caption = Dsetting.Sephead
  LabelSmpts.Caption = Dsetting.Smpts
  LabelAddcal.Caption = Dsetting.Caluse
  LabelMpafilename.Caption = Dsetting.Filename
  Labelspecfile.Caption = Dsetting.Specfile
  LabelCommand.Caption = Dsetting.Command
End Sub

Private Sub CommandErase_Click()
  Call EraseData(0)
End Sub

Private Sub CommandExecute_Click()
  Dim a As String * 1024
  Mid$(a, 1) = TextCommand.Text
  Call RunCmd(0, a)
  LabelRespons.Caption = a
  Mcano = Val(TextMC.Text)
  Ret = GetStatus(Mcano)
  Ret = GetStatusData(Status, 0)
  Call UpdateMpStatus
  Ret = GetStatusData(Status, Mcano)
  Call UpdateStatus
End Sub

Private Sub CommandGetspec_Click()
  Mcano = Val(TextMC.Text)
  Chan = Val(TextChan.Text)
  Call GetBlock(Hist(0), Chan, Chan + 24, 1, Mcano)
  For I = 0 To 23 Step 1
    LabelData(I).Caption = Hist(I)
  Next I
End Sub

Private Sub CommandGetstring_Click()
  Dim b As String * 1024
  Mcano = Val(TextMC.Text)
  Ret = LVGetStr(b, Mcano)
  LabelLine(0).Caption = Mid$(b, 1, 60)
  LabelLine(1).Caption = Mid$(b, 61, 60)
  LabelLine(2).Caption = Mid$(b, 121, 60)
  LabelLine(3).Caption = Mid$(b, 181, 60)
  LabelLine(4).Caption = Mid$(b, 241, 60)
  LabelLine(5).Caption = Mid$(b, 301, 60)
  LabelLine(6).Caption = Mid$(b, 361, 60)
  LabelLine(7).Caption = Mid$(b, 421, 60)
  LabelLine(8).Caption = Mid$(b, 481, 60)
  LabelLine(9).Caption = Mid$(b, 541, 60)
  LabelLine(10).Caption = Mid$(b, 601, 60)
  LabelLine(11).Caption = Mid$(b, 881, 60)
  LabelLine(12).Caption = Mid$(b, 961, 60)
  LabelLine(13).Caption = Mid$(b, 661, 100)
End Sub

Private Sub CommandHalt_Click()
  Call Halt(0)
End Sub


Private Sub CommandSave_Click()
  Call SaveData(0, 1)
End Sub

Private Sub CommandSetting_Click()
  Mcano = Val(TextMC.Text)
  If GetSettingData(Setting, Mcano) = 1 Then
      Call UpdateSetting
  End If
End Sub

Private Sub CommandStart_Click()
  Call Start(0)
End Sub

Private Sub CommandUpdate_Click()
  Mcano = Val(TextMC.Text)
  Ret = GetStatus(Mcano)
  If GetStatusData(Status, Mcano) = 1 Then
      Call UpdateStatus
  End If
End Sub

Private Sub Form_Load()
  OldStarted = 0
  Mcano = 0
  Sysno = 0
  Chan = 0
  Ret = ServExec(0)
  Ret = GetStatus(0)
  Ret = GetStatusData(Status, 0)
  Call UpdateMpStatus
  Ret = GetStatusData(Status, Mcano)
  Call UpdateStatus
  Ret = GetSettingData(Setting, 0)
  Call UpdateSetting
End Sub



Private Sub Timer1_Timer()
  Mcano = Val(TextMC.Text)
  If Mcano > 16 Then
    Mcano = 16
  End If
  If Mcano < 0 Then
    Mcano = 0
  End If
  Ret = GetStatus(0)
```

```
 If GetStatusData(Status, 0) = 1 Then
   If Status.Val > 0 Or OldStarted > 0 Then
     Toggle = Not Toggle
     OldStarted = Status.Val
     Call UpdateMpStatus
     If GetStatusData(Status, Mcano) = 1 Then
       Call UpdateStatus
     End If
   End If
 End If
End Sub

Private Sub UpdateMpStatus()
     LabelStarted.Caption = Status.Val
     LabelRuntime.Caption = Status.Cnt(0)
     LabelSweeps.Caption = Status.Cnt(5)
     LabelStarts.Caption = Status.Cnt(6)
End Sub

Private Sub UpdateStatus()
     LabelMaxval.Caption = Status.Val1
     LabelOfls.Caption = Status.Cnt(1)
     LabelTotalsum.Caption = Status.Cnt(2)
     LabelRoisum.Caption = Status.Cnt(3)
     LabelTotalrate.Caption = Format$(Status.Cnt(4), "######0.0#")
End Sub

Private Sub UpdateSetting()
     LabelRange.Caption = Setting.Range
     LabelCftfak.Caption = Setting.Cftfak
     LabelRoimin.Caption = Setting.Roimin
     LabelRoimax.Caption = Setting.Roimax
     LabelNregions.Caption = Setting.Nregions
     LabelCaluse.Caption = Setting.Caluse
     LabelCalpoints.Caption = Setting.Calpoints
     LabelParam.Caption = Setting.Param
     LabelOffset.Caption = Setting.Offset
     LabelXdim.Caption = Setting.Xdim
     LabelBitshift.Caption = Setting.Bitshift
     LabelActive.Caption = Setting.Active
     LabelRoipreset.Caption = Setting.Roipreset
End Sub
```

4 Using the DMCS8.DLL from C

In the following an example is shown how to control the MCS8 from a simple console application written in Microsoft C.



```
C:\MCS8>tstmcs8
OFF
runtime=   0.000
sweeps=   0.000000
starts=   0.000000

Commands:
Q               Quit
?               Help
S       Show Status
H               Halt
T       Show Setting
B       Show BoardSetting
CHN=x   Switch to CHN #x
rdig    Read Dig. I/O port
w5 aabb         Write aabb into Reg.5
r5              Read Reg 5
rx0             Read Reg 0..f
rx7             Read Reg 70..7f
reep0           Read 64 byte from eeprom at address 0
sleep=10        Sleep 10 msec
run filename
(... more see command language in MPANT help)

t
CHN 0:
range=     4096
cftfak=    0x2580100
roimin=    0
roimax=    4096
nregions=  0
caluse=    1
calpoints= 0
param=     0x0
offset=    0x0
xdim=      0
bitshift=  0
active=    0x1
roipreset= 10

sweepmode=  0x200a080
prena=      0x0
cycles=     18
sequences=  1
syncout=    0x0
digio=      0x0
digval=     0
dac0=       0x92c
dac1=       0x92c
dac2=       0x92c
dac3=       0x92c
dac4=       0x92c
dac5=       0x92c
tagbits=    16
extclk=     0
periods=    2
serno=      1004
ddruse=     0x17
active=     1
holdafter=  0
swpreset=   1e+006
fstchan=    0
timepreset= 20

savedata= 0
autoinc=  0
fmt=      0
mpafmt=   0
sephead=  0
filename= TEST.mpa
```

Fig. 4.1: The tstmcs8 Demo program written in C

4.1 The Include File

The include file DMCS8.h contains the function definitions of the DLL. It includes also the structure definitions from struct.h listed in the appendix 6.1.

```
#ifdef __cplusplus
extern "C"
{
#endif
#include "struct.h"

#include "scaler.h"
// pure constant definitions are omitted here

/*** FUNCTION PROTOTYPES (do not change) ***/
#ifdef DLL
BOOL APIENTRY DllMain(HANDLE hInst, DWORD ul_reason_being_called, LPVOID lpReserved);

VOID APIENTRY StoreSettingData(ACQSETTING  *Setting, int nDisplay);
                                // Stores Settings into the DLL

int APIENTRY GetSettingData(ACQSETTING  *Setting, int nDisplay);
                                // Get Settings stored in the DLL

VOID APIENTRY StoreExtSettingData(EXTACQSETTING  *Setting, int nDisplay);
                            // Stores extended Settings into the DLL

int APIENTRY GetExtSettingData(EXTACQSETTING  *Setting, int nDisplay);
                            // Get extended Settings stored in the DLL

VOID APIENTRY StoreStatusData(ACQSTATUS  *Status, int nDisplay);
                                // Store the Status into the DLL

int APIENTRY GetStatusData(ACQSTATUS  *Status, int nDisplay);
                                // Get the Status

VOID APIENTRY Start(int nSystem);       // Start

VOID APIENTRY Halt(int nSystem);        // Halt

VOID APIENTRY Continue(int nSystem);    // Continue

VOID APIENTRY NewSetting(int nDevice);  // Indicate new Settings to Server

UINT APIENTRY ServExec(HWND ClientWnd); // Execute the Server

VOID APIENTRY StoreData(ACQDATA  *Data, int nDisplay);
                                // Stores Data pointers into the DLL

int APIENTRY GetData(ACQDATA  *Data, int nDisplay);
                                // Get Data pointers

int APIENTRY GetSpec(int i, int nDisplay);
                                // Get a spectrum value

VOID APIENTRY SaveSetting(void);        // Save Settings

int APIENTRY GetStatus(int nDevice);    // Request actual Status from Server

VOID APIENTRY Erase(int nSystem);       // Erase spectrum

VOID APIENTRY SaveData(int nDisplay, int all);    // Saves data

VOID APIENTRY GetBlock(int  *hist, int start, int end, int step,  int nDisplay);
                            // Get a block of spectrum data

VOID APIENTRY StoreDefData(ACQDEF  *Def);
                                // Store System Definition into DLL

int APIENTRY GetDefData(ACQDEF  *Def);
                                // Get System Definition
```

```
VOID APIENTRY StoreCDefData(COINCDEF  *Def);

int APIENTRY GetCDefData(COINCDEF  *Def);

VOID APIENTRY LoadData(int nDisplay, int all);     // Loads data

VOID APIENTRY AddData(int nDisplay, int all);      // Adds data

VOID APIENTRY SubData(int nDisplay, int all);      // Subtracts data

VOID APIENTRY Smooth(int nDisplay);       // Smooth data

VOID APIENTRY NewData(void);            // Indicate new ROI or string Data

VOID APIENTRY HardwareDlg(int item);     // Calls the Settings dialog box

VOID APIENTRY UnregisterClient(void);    // Clears remote mode from MCDWIN

VOID APIENTRY DestroyClient(void);       // Close MCDWIN

UINT APIENTRY ClientExec(HWND ServerWnd);
                                // Execute the Client MCDWIN.EXE

int APIENTRY LVGetDat(unsigned int  *datp, int nDisplay);
                                // Copies the spectrum to an array

VOID APIENTRY RunCmd(int nDisplay, LPSTR Cmd);
                                // Executes command

int APIENTRY LVGetRoi(unsigned int  *roip, int nDisplay);
                            // Copies the ROI boundaries to an array

int APIENTRY LVGetOneRoi(int nDisplay, int roinum, int *x1, int *x2);
                              // Get one ROI boundary

int APIENTRY LVGetCnt(double far *cntp, int nDisplay);
                                // Copies Cnt numbers to an array

int APIENTRY LVGetStr(char far *strp, int nDisplay);
                                // Copies strings to an array

VOID APIENTRY StoreMCSSetting(BOARDSETTING *Defmc, int ndev);
                            // Store BOARDSETTING Definition into DLL

int APIENTRY GetMCSSetting(BOARDSETTING *Defmc, int ndev);
                            // Get BOARDSETTING Definition from DLL

VOID APIENTRY StoreDatSetting(DATSETTING *Defdat);
                            // Store Data Format Definition into DLL

int APIENTRY GetDatSetting(DATSETTING *Defdat);
                            // Get Data Format Definition from DLL

VOID APIENTRY StoreReplaySetting(REPLAYSETTING *Repldat);
                            // Store Replay Settings into DLL

int APIENTRY GetReplaySetting(REPLAYSETTING *Repldat);
                            // Get Replay Settings from DLL

int APIENTRY GetDatInfo(int nDisplay, int *xmax, int *ymax);
                            // returns spectra length;

int APIENTRY GetDatPtr(int nDisplay, int *xmax, int *ymax, LPSTR *pt);
                            // Get a temporary pointer to spectra data

int APIENTRY ReleaseDatPtr(void);
                              // Release temporary data pointer

int APIENTRY GetSVal(int DspID, int xval);
                              // Get special display data like projections or slices from MPANT

int APIENTRY GetRoiIndex(LPSTR roiname);

    // get a unique index to address ROIs from named ROI's.
    // rectangular or 1D ROIs:
    //    LOWORD is the spectra number,
    //    HIWORD is the ROI number (1,2,..)
    // polygonal ROIs:
    //    LOWORD is an entry number
    //    HIWORD is the roiid = 100 * spectra_number + ROI_number
    // returns 0 if not found.

int APIENTRY DeleteRoi(DWORD roiindex);
    // deletes ROI with given index

int APIENTRY SelectRoi(DWORD roiindex);
    // selects ROI with given index

int APIENTRY GetRoiSum(DWORD roiindex, double *sum);
        // get sum of counts in ROI,
        // returns roiindex, or 0 if not found

int APIENTRY BytearrayToShortarray(short *Shortarray, char *Bytearray, int length);
                    // auxiliary function for VB.NET to convert strings

int APIENTRY LedBlink(int nDev);
                         // Lets the front leds blink for a while

int APIENTRY DigInOut(int value, int enable);
                         // controls Dig I/0, returns digin

int APIENTRY LVGetSpecLength(int nDisplay);

int APIENTRY LVGetRoinam(char *strp, int nDisplay);

        // get Roi names

int APIENTRY LVGetDatSetting(LVDATSETTING *Defdat, LPSTR filename, LPSTR specfile, LPSTR
command);

int APIENTRY LVGetReplaySetting(LVREPLAYSETTING *Repldat, LPSTR filename);

int APIENTRY LVGetDefData(LVACQDEF *Def);

int APIENTRY LVGetRroiDat(int nDisplay, int roinum, int x0, int y0, int xdim, int ydim, int xmax,
                               double *RoiSum, int *datp, double *area);

int APIENTRY LVGetRoiRect(int nDisplay, int roinum, int *x0, int *y0, int *xdim, int *ydim, int *xmax);

int APIENTRY LVGetProiDat(int roiid, int x0, int y0, int xdim, int ydim, double *roisum, int *datp);

int APIENTRY LVGetCDefData(LVCOINCDEF *Def);

#else

typedef int (WINAPI *IMPAGETSETTING) (ACQSETTING  *Setting, int nDisplay);
                            // Get Spectra Settings stored in the DLL

typedef int (WINAPI *IMPAGETSTATUS) (ACQSTATUS  *Status, int nDisplay);
                            // Get the Status

typedef VOID (WINAPI *IMPARUNCMD) (int nDisplay, LPSTR Cmd);
                            // Executes command

typedef int (WINAPI *IMPAGETCNT) (double  *cntp, int nDisplay);
                            // Copies Cnt numbers to an array

typedef int (WINAPI *IMPAGETROI) (unsigned int  *roip, int nDisplay);
                            // Copies the ROI boundaries to an array

typedef int (WINAPI *IMPAGETDEF) (ACQDEF  *Def);
                            // Get System Definition

typedef int (WINAPI *IMPAGETDAT) (unsigned int  *datp, int nDisplay);
                            // Copies the spectrum to an array

typedef int (WINAPI *IMPAGETSTR) (char  *strp, int nDisplay);
                            // Copies strings to an array
```

```
typedef UINT (WINAPI *IMPASERVEXEC) (HWND ClientWnd);  // Register client at server MCS8.EXE

typedef int (WINAPI *IMPANEWSTATUS) (int nDev);      // Request actual Status from Server

typedef int (WINAPI *IMPAGETMCSSET) (BOARDSETTING *Board, int nDevice);
                              // Get MCSSettings from DLL

typedef int (WINAPI *IMPAGETDATSET) (DATSETTING *Defdat);
                              // Get Data Format Definition from DLL

typedef int (WINAPI *IMPADIGINOUT) (int value, int enable);
                                // controls Dig I/0, returns digin

typedef int (WINAPI *IMPADACOUT) (int value);   // output Dac value as analogue voltage

typedef VOID (WINAPI *IMPASTART) (int nSystem);        // Start

typedef VOID (WINAPI *IMPAHALT) (int nSystem);         // Halt

typedef VOID (WINAPI *IMPACONTINUE) (int nSystem);     // Continue

typedef VOID (WINAPI *IMPAERASE) (int nSystem);        // Erase spectrum

#endif

#ifdef __cplusplus
}
#endif
```

4.2 The C demo program

The source of the simple C program is shown here: It shows how to access the DLL and to get Status, Settings and spectrum data. To perform actions like start, halt, continue, erase, just send the corresponding commands using the command language.

```
// ------------------------------------------------------------------------
// TSTMCS8.C : DMCS8.DLL Software driver C example
// ------------------------------------------------------------------------

#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <time.h>

#undef DLL
#include "dmca4.h"

HANDLE          hDLL = 0;

IMPAGETSETTING      lpSet=NULL;
IMPANEWSTATUS       lpNewStat=NULL;
IMPAGETSTATUS       lpStat=NULL;
IMPARUNCMD          lpRun=NULL;
IMPAGETCNT          lpCnt=NULL;
IMPAGETROI          lpRoi=NULL;
IMPAGETDAT          lpDat=NULL;
IMPAGETSTR          lpStr=NULL;
IMPASERVEXEC        lpServ=NULL;
IMPAGETDATSET       lpGetDatSet=NULL;
IMPAGETMCSSET       lpGetMCSSet=NULL;
IMPADIGINOUT        lpDigInOut=NULL;
IMPASTART           lpStart=NULL;
IMPAHALT            lpHalt=NULL;
IMPACONTINUE        lpContinue=NULL;
IMPAERASE           lpErase=NULL;
IMPALVGETCDEF       lpGetLCDef=NULL;
IMPAGETBRDSET       lpGetBrdSet=NULL;

ACQSETTING          Setting={0};
ACQDATA             Data={0};
ACQDEF              Def={0};
ACQSTATUS           Status={0};
DATSETTING          DatSetting={0};
BOARDSETTING        MCSSetting={0};

int nDev=0;

void help()
{
        printf("Commands:\n");
        printf("Q                   Quit\n");
        printf("?           Help\n");
        printf("S       Show Status\n");
        printf("H                   Halt\n");
        printf("T       Show Setting\n");
        printf("B       Show BoardSetting\n");
        printf("CHN=x  Switch to CHN #x \n");
   printf("(... more see command language in MPANT help)\n");
   printf("\n");
}
```

```
void PrintMpaStatus(ACQSTATUS *Stat)
{
  if(Stat->started == 1) printf("ON\n");
  else if(Stat->started & 0x02)      printf("READ OUT\n");
  else printf("OFF\n");
  printf("runtime=   %.3lf\n", Stat->cnt[ST_RUNTIME]);
  printf("sweeps=    %.3lf\n", Stat->cnt[ST_SWEEPS]);
  printf("starts=   %lf\n\n", Stat->cnt[ST_STARTS]);
}

void PrintStatus(ACQSTATUS *Stat)
{
    printf("totalsum=   %.0lf\n", Stat->cnt[ST_TOTALSUM]);
    printf("roisum=     %.0lf\n", Stat->cnt[ST_ROISUM]);
    printf("rate=    %.2lf\n", Stat->cnt[ST_ROIRATE]);
    printf("ofls=   %.2lf\n", Stat->cnt[ST_OFLS);
}

void PrintDatSetting(DATSETTING *Set)
{
  printf("savedata= %d\n", Set->savedata);
  printf("autoinc=  %d\n", Set->autoinc);
  printf("fmt=      %d\n", Set->fmt);
  printf("mpafmt=   %d\n", Set->mpafmt);
  printf("sephead=  %d\n", Set->sephead);
  printf("filename= %s\n\n", Set->filename);
}

void PrintMCSSetting(BOARDSETTING *Set)
{
  printf("sweepmode=    0x%x\n", Set->sweepmode);
  printf("prena=     0x%x\n", Set->prena);
  printf("cycles=     %d\n", Set->cycles);
  printf("sequences=  %d\n", Set->sequences);
  printf("syncout=    0x%x\n", Set->syncout);
  printf("digio=      0x%x\n", Set->digio);
  printf("digval=     %d\n", Set->digval);
  printf("dac0=       0x%x\n", Set->dac0);
  printf("dac1=       0x%x\n", Set->dac1);
  printf("dac2=       0x%x\n", Set->dac2);
  printf("dac3=       0x%x\n", Set->dac3);
  printf("dac4=       0x%x\n", Set->dac4);
  printf("dac5=       0x%x\n", Set->dac5);
//  printf("fdac=       0x%x\n", Set->fdac);
  printf("tagbits=    %d\n", Set->tagbits);
  printf("extclk=     %d\n", Set->extclk);
  printf("periods=    %d\n", Set->periods);
  printf("serno=      %d\n", Set->serno);
  printf("ddruse=     0x%x\n", Set->ddruse);
  printf("active=     %d\n", Set->active);
  printf("holdafter=  %lg\n", Set->holdafter);
  printf("swpreset=   %lg\n", Set->swpreset);
  printf("fstchan=    %lg\n", Set->fstchan);
  printf("timepreset= %lg\n\n", Set->timepreset);

}

void PrintSetting(ACQSETTING *Set)
{
  printf("range=     %ld\n", Set->range);
  printf("cftfak=    0x%x\n", Set->cftfak);
  printf("roimin=    %ld\n", Set->roimin);
  printf("roimax=    %ld\n", Set->roimax);
  printf("nregions=  %d\n", Set->nregions);
  printf("caluse=    %d\n", Set->caluse);
  printf("calpoints= %d\n", Set->calpoints);
  printf("param=     0x%lx\n", Set->param);
  printf("offset=    0x%lx\n", Set->offset);
  printf("xdim=      %d\n", Set->xdim);
  printf("bitshift=  %d\n", Set->bitshift);
  printf("active=    0x%x\n", Set->active);
  printf("ROIpreset= %lg\n", Set->eventpreset);
}

int run(char *command)
{
        int err;
        if (!stricmp(command, "?"))       help();
        else if (!stricmp(command, "rdig")) {             // read Dig I/O port
          if (lpDigInOut) {
            err = (*lpDigInOut)(0xff,0);
            printf("%x\n", err);
          }
        }
        else if (!strnicmp(command, "wdig=", 5)) {        // write Dig I/O port (open drain)
          if (lpDigInOut) {
                unsigned int val;
                sscanf(command+5, "%x", &val);
            err = (*lpDigInOut)(val,0);
            printf("%x\n", err);
          }
        }
        else if (!strnicmp(command, "pdig=", 5)) {        // write Dig I/O port (push-pull)
          if (lpDigInOut) {
                unsigned int val;
                sscanf(command+5, "%x", &val);
            err = (*lpDigInOut)(val,0xff);
            printf("%x\n", err);
          }
        }
        else if (!stricmp(command,"Q"))
          return 1;
        else if (!stricmp(command,"S")) {
          err = (*lpStat)(&Status, nDev);
          if (nDev) PrintStatus(&Status);
          else PrintMpaStatus(&Status);
          return 0;
        }
        else if (!stricmp(command,"T")) {
                                   // spectra settings
          err = (*lpSet)(&Setting, nDev);
          printf("CHN %d:\n", nDev);
          PrintSetting(&Setting);
          if (nDev==0) {           // MPA settings
            err = (*lpGetMCSSet)(&MCSSetting, 0);
            PrintMCSSetting(&MCSSetting);
                                   // DATSettings
            err = (*lpGetDatSet)(&DatSetting);
            PrintDatSetting(&DatSetting);
          }
          return 0;
        }
        else if (!stricmp(command,"H")) {
          (*lpHalt)(0);
```

```
              return 0;
            }
            else if(!strnicmp(command, "CHN=", 4)) {
              sscanf(command+4, "%d", &nDev);
              (*lpRun)(0, command);
            }
            else if (!stricmp(command,"MPA")) {
              nDev=0;
              (*lpRun)(0, command);
            }
            else {
              (*lpRun)(0, command);
              printf("%s\n", command);
            }
            return 0;
}

int readstr(char *buff, int buflen)
{
  int i=0,ic;

  while ((ic=getchar()) != 10) {
    if (ic == EOF) {
      buff[i]='\0';
      return 1;
    }
    if (ic == 13) ic=0;
    buff[i]=(char)ic;
    i++;
    if (i==buflen-1) break;
  }
  buff[i]='\0';
  return 0;
}

//int PASCAL WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR lpCmd, int nShow)
void main(int argc, char *argv[])
{  long Errset=0;
  char command[80];

  hDLL = LoadLibrary("DMCS8.DLL");
  if(hDLL){
              lpSet=(IMPAGETSETTING)GetProcAddress(hDLL,"GetSettingData");
              lpNewStat=(IMPANEWSTATUS)GetProcAddress(hDLL,"GetStatus");
              lpStat=(IMPAGETSTATUS)GetProcAddress(hDLL,"GetStatusData");
              lpRun=(IMPARUNCMD)GetProcAddress(hDLL,"RunCmd");
              lpCnt=(IMPAGETCNT)GetProcAddress(hDLL,"LVGetCnt");
              lpRoi=(IMPAGETROI)GetProcAddress(hDLL,"LVGetRoi");
              lpDat=(IMPAGETDAT)GetProcAddress(hDLL,"LVGetDat");
              lpStr=(IMPAGETSTR)GetProcAddress(hDLL,"LVGetStr");
              lpServ=(IMPASERVEXEC)GetProcAddress(hDLL,"ServExec");
              lpGetDatSet=(IMPAGETDATSET)GetProcAddress(hDLL,"GetDatSetting");
              lpGetMCSSet=(IMPAGETMCSSET)GetProcAddress(hDLL,"GetMCSSetting");
              lpStart=(IMPASTART)GetProcAddress(hDLL,"Start");
              lpHalt=(IMPAHALT)GetProcAddress(hDLL,"Halt");
              lpContinue=(IMPACONTINUE)GetProcAddress(hDLL,"Continue");
              lpErase=(IMPAERASE)GetProcAddress(hDLL,"Erase");
              lpDigInOut=(IMPADIGINOUT)GetProcAddress(hDLL,"DigInOut");  }
  else return;

  // Initialize parameters
  Errset = (*lpNewStat)(0);
  Errset = (*lpStat)(&Status, 0);
  PrintMpaStatus(&Status);

  help();
  while(TRUE)
          {
                   readstr(command, 80);
                   if (run(command)) break;
          }
  FreeLibrary(hDLL);
  return;
}
```

5 Using the DMCS8.DLL from C#

To get the transfer working of a data block from the DLL to a program written in C#, it is important to change the project properties of the C# program. Under “build” you will find in Visual Studio a check box “allow unsafe code”. This must be checked. For the processor type change the default value “Any CPU” to “x64”. Then the following code example will work with the 64-bit Software.

5.1 The C# demo program

The source of a simple C# console program is shown here: It shows how to access the DLL and to get Status, Settings and spectrum data. To perform actions like start, halt, continue, erase, just send the corresponding commands using the command language.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;

namespace MCS8Demo1
{
   static class Constants
   {
      public const int ST_RUNTIME = 0;
      public const int ST_OFLS = 1;
      public const int ST_TOTALSUM = 2;
      public const int ST_ROISUM = 3;
      public const int ST_ROIRATE = 4;
      public const int ST_SWEEPS = 5;
      public const int ST_STARTS = 6;
      public const int ST_ZEROEVTS = 7;
   }
   // MCS Channel Status
   public struct ACQSETTING
   {
      public int range;            // spectrum length
      public int cftfak;          // LOWORD: 256 * cft factor (t_after_peak / t_to_peak)
      // HIWORD: max pulse width for CFT
      public int roimin;           // lower ROI limit
      public int roimax;           // upper limit: roimin <= channel < roimax
      public int nregions;         // number of regions
      public int caluse;           // bit0: 1 if calibration used, higher bits: formula
      public int calpoints;        // number of calibration points
      public int param;            // (reserved:) for MAP and POS: LOWORD=x, HIWORD=y
      public int offset;          // (reserved:) zoomed MAPS: LOWORD: xoffset, HIWORD, yoffset
      public int xdim;                          //  (reserved:) x resolution of maps
      public int bitshift;    // LOWORD: Binwidth = 2 ^ (bitshift)
      // HIWORD: Threshold for Coinc
      public int active;               // Spectrum definition words for CHN1..6:
      // active & 0xF  ==0 not used
      //             ==1 single
      // bit 8: Enable Tag bits
      // bit 9: start with rising edge
      // bit 10: time under threshold for pulse width
      // bit 11: pulse width mode for any spectra with both edges enabled
      // Spectrum definition words for calc. spectra:
      // active & 0xF  ==3 MAP, ((x-xoffs)>>xsh) x ((y-yoffs)>>ysh)
      //               ((x-xoffs)>>xsh) x ((y-timeoffs)>>timesh)
      //            or ((x-timeoffs)>>timesh x ((y-yoffs)>>ysh)
      //        bit4=1: x zoomed MAP
```

```
    //        bit5=1: y zoomed MAP
    //              ==5 SUM, (x + y)>>xsh
    //              ==6 DIFF,(x - y + range)>>xsh
    //              ==7 ANY, (for compare)
    //              ==8 COPY, x
    //              ==9 DLL  fDLL(x,y,z),
    //              ==0xA Sweep HISTORY, Sweepnum(x)
    // bit 8..11 xsh, bit 12..15 ysh or bit 8..15 xsh
    // HIWORD(active) = condition no. (0=no condition)
    public double eventpreset;      // ROI preset value
    public double dummy1;       // (Livetime preset)
    public double dummy2;               // (Realtime preset)
    public double dummy3;       //
}

unsafe public struct DATSETTING
{
    public int savedata;         // bit 0: auto save after stop
    // bit 1: write listfile
    // bit 2: listfile only, no evaluation
    // bit 5: drop zero events
    public int autoinc;          // 1 if auto increment filename
    public int fmt;            // format type (seperate spectra):
    // 0 == ASCII, 1 == binary,
    // 2 == GANAAS, 3 == EMSA, 4 == CSV
    public int mpafmt;           // format used in mpa datafiles
    public int sephead;          // seperate Header
    public int smpts;
    public int caluse;
    unsafe public fixed byte filename[256];
    unsafe public fixed byte specfile[256];
    unsafe public fixed byte command[256];
}

public struct BOARDSETTING
{
    public int sweepmode;        // sweepmode & 0xF: 0 = normal,
    // 1=differential (relative to first stop in sweep)
    // 4=sequential
    // 5=seq.+diff (Ch1), bit0 = differential mode
    // 6 = CORRELATIONS
    // 7 = diff.+Corr.
    // 9=differential to stop in Ch2, bit3 = Ch2 ref (diff.mode)
    // 0xF = Corr.+diff (Ch2)
    // bit 4: Softw. Start
    // bit 5: "Don't show" tagbits
    // bit 6: Endless
    // bit 7: Start event generation
    // bit 8: Enable Tag bits
    // bit 9: start with rising edge
    // bit 10: time under threshold for pulse width
    // bit 11: pulse width mode for any spectra with both edges enabled
    // bit 12: abandon Sweepcounter in Data
    // bit 13: "one-hot" mode with tagbits
    // bit 14: ch6 ref (diff.mode)
    // bit 16..bit 22 ~(input channel enable)
    // bit 24: require data lost bit in data
    // bit 25: don't allow 6 byte datalength
    // bit 27: Folded
    // bit 28: Interleaved
    public int prena;          // bit 0: realtime preset enabled
    // bit 1:
```

```
    // bit 2: sweep preset enabled
    // bit 3: ROI preset enabled
    // bit 4: Starts preset enabled
    // bit 5: ROI2 preset enabled
    // bit 6: ROI3 preset enabled
    // bit 7: ROI4 preset enabled
    // bit 8: ROI5 preset enabled
    // bit 9: ROI6 preset enabled
    // bit 10: ROI7 preset enabled
    // bit 11: ROI8 preset enabled
    public int cycles;          // for sequential mode
    public int sequences;       // for sequential mode
    public int syncout;                        // LOWORD: sync out; bit 0..5 NIM syncout, bit 8..13 TTL syncout
    // bit7: NIM syncout_invert, bit15: TTL syncout_invert
    // 0="0", 1=5 MHz, 2=50 MHz, 3=100 MHz, 4=97.656 MHz,
    // 5=195.625 MHz, 6= 195 MHz (int ref), 7=Start, 8=Ch1, 9=Ch2, 10=Ch3,
    // 11=Ch4, 12=Ch5, 13=Ch6, 14=Ch7, 15=GO, 16=Start_of_sweep, 17=Armed,
    // 18=SWEEP_ON, 19=WINDOW, 20=HOLD_OFF, 21=EOS_DEADTIME
    // 22=TIME[0],...,51=TIME[29], 52...63=SWEEP[0]..SWEEP[11]
    //
    public int digio;           // LOWORD: Use of Dig I/O, GO Line:
    // bit 0: status dig 0..3
    // bit 1: Output digval and increment digval after stop
    // bit 2: Invert polarity
    // bit 3: Push-Pull output, not possible
    // bit 4:  Start with Input Dig 4
    // bit 5:  Start with Input GO
    // bit 8: GOWATCH
    // bit 9: GO High at Start
    // bit 10: GO Low at Stop
    // bit 11: Clear at triggered start
    // bit 12: Only triggered start
    public int digval;                         // digval=0..255 value for samplechanger
    public int dac0;          // DAC0 value (START)
    //  bit 16: Start with rising edge
    public int dac1;            //  DAC1 value (STOP 1)
    public int dac2;            //  DAC2 value (STOP 2)
    public int dac3;            //  DAC3 value (STOP 3)
    public int dac4;                                   //  DAC4 value (STOP 4)
    public int dac5;                                   //  DAC5 value (STOP 5)
    // bit (14,15) of each word: 0=falling, 1=rising, 2=both, 3=both+CFT
    // bit 17 of each: pulse width mode under threshold
    public int fdac;            // dummy
    public int tagbits;       // number of tagbits
    public int extclk;                // use external clock
    public int periods;                        // number of periods in folded mode, sweeplength = range * periods
    public int serno;                 // serial number
    public int ddruse;              // bit0: DDR_USE, bit1: DDR_2GB
    // bits[2:3]: usb_usage
    // bits[4:5]: wdlen
    public int active;          // module in system
    public double holdafter;          // Hold off
    public double swpreset;      // sweep preset value
    public double fstchan;                     // acquisition delay
    public double timepreset;    // time preset
}

public struct LVCOINCDEF
{
    public int adcnum;      // Number of active ADC's
    public int tofnum;     // Number of active MCS/Scope channels
    public int ntofs0;      // Number of TOF inputs
    public int modules;    // Number of MCS8 modules
    public int nadcs;      // Number of ADCs
}

unsafe public struct ACQSTATUS
{
    public int started;              // aquisition status: 1 if running, 0 else
    public int maxval;
    unsafe public fixed double cnt[8];              // see ST_.. defines above
}
class Program
{
    // imports from DMCS8.DLL
    [DllImport("dmcs8.dll", EntryPoint = "RunCmd")]
    extern static void RunCmd(int nDisplay, string Command);
    //              private static extern void RunCmd(int nDisplay, StringBuilder Command);
    [DllImport("dmcs8.dll", EntryPoint = "GetStatus")]
    extern static int GetStatus(int nDevice);
    [DllImport("dmcs8.dll", EntryPoint = "GetStatusData")]
    extern static int GetStatusData(ref ACQSTATUS Status, int nDevice);
    [DllImport("dmcs8.dll", EntryPoint = "GetSettingData")]
    extern static int GetSettingData(ref ACQSETTING MSetting, int nDisplay);
    [DllImport("dmcs8.dll", EntryPoint = "LVGetCnt")]
    extern static int LVGetCnt(ref double cntp, int nDisplay);
    [DllImport("dmcs8.dll", EntryPoint = "LVGetRoi")]
    extern static int LVGetRoi(ref int roip, int nDisplay);
    [DllImport("dmcs8.dll", EntryPoint = "LVGetDat")]
    extern static int LVGetDat(ref int datp, int nDisplay);
    [DllImport("dmcs8.dll", EntryPoint = "LVGetCDefData")]
    extern static int LVGetCDefData(ref LVCOINCDEF LCDef);
    [DllImport("dmcs8.dll", EntryPoint = "GetMCSSetting")]
    extern static int GetMCSSetting(ref BOARDSETTING BoardSetting, int nDevice);
    [DllImport("dmcs8.dll", EntryPoint = "GetDatSetting")]
    extern static int GetDatSetting(ref DATSETTING DatSetting);
    [DllImport("dmcs8.dll", EntryPoint = "GetBlock")]
    extern static void GetBlock(ref int datp, int from, int to, int step, int nDisplay);

    static void Main(string[] args)
    {
        int nDev = 0;
        ACQSETTING acq;
        ACQSTATUS Status;
        String command;
        LVCOINCDEF LCDef;
        BOARDSETTING BoardSetting;
        LCDef = new LVCOINCDEF();
        BoardSetting = new BOARDSETTING();

        acq = new ACQSETTING();
        Status = new ACQSTATUS();
        GetStatus(0);
        GetStatusData(ref Status, 0);
        LVGetCDefData(ref LCDef);
        GetMCSSetting(ref BoardSetting, 0);
        PrintMPAStatus(ref Status);
        Console.WriteLine();
        GetSettingData(ref acq, nDev);
        PrintSetting(ref acq);
        help();
        while (true)
        {
            command = Console.ReadLine();
            if (run(command, ref nDev) == 1) break;
        }
    }

    unsafe static void PrintMPAStatus(ref ACQSTATUS Status)
    {
        fixed (ACQSTATUS* s = &Status)
        {
            if ((s->started & 0x01) != 0) Console.WriteLine("ON");
            else if ((s->started & 0x02) != 0) Console.WriteLine("READ OUT");
            else Console.WriteLine("OFF");
            Console.Write("runtime= ");
            Console.WriteLine(s->cnt[Constants.ST_RUNTIME]);
            Console.Write("sweeps= ");
            Console.WriteLine(s->cnt[Constants.ST_SWEEPS]);
            Console.Write("starts= ");
            Console.WriteLine(s->cnt[Constants.ST_STARTS]);
        }
    }

    unsafe static void PrintStatus(ref ACQSTATUS Status)
    {
        fixed (ACQSTATUS* s = &Status)
        {
            Console.Write("total= ");
            Console.WriteLine(s->cnt[Constants.ST_TOTALSUM]);
            Console.Write("roi= ");
            Console.WriteLine(s->cnt[Constants.ST_ROISUM]);
            Console.Write("rate= ");
            Console.WriteLine(s->cnt[Constants.ST_ROIRATE]);
            Console.Write("ofls= ");
            Console.WriteLine(s->cnt[Constants.ST_OFLS]);
        }
    }

    unsafe static void PrintDatSetting(ref DATSETTING Set)
    {
        fixed (DATSETTING* d = &Set)
        {
            byte[] f = new byte[255];
            string s;
            int i;
            for (i = 0; i < 255; i++)
                f[i] = d->filename[i];
            s = Encoding.UTF8.GetString(f, 0, 255);
            Console.Write("savedata= ");
            Console.WriteLine(d->savedata);
            Console.Write("autoinc= ");
            Console.WriteLine(d->autoinc);
            Console.Write("fmt= ");
            Console.WriteLine(d->fmt);
            Console.Write("mpafmt= ");
            Console.WriteLine(d->mpafmt);
            Console.Write("sephead= ");
            Console.WriteLine(d->sephead);
            Console.Write("filename= ");
            Console.WriteLine(s);
        }
    }

    static void PrintMCSSetting(ref BOARDSETTING Set)
    {
        Console.Write("sweepmode= ");
        Console.WriteLine(Set.sweepmode);
        Console.Write("prena= ");
        Console.WriteLine(Set.prena);
        Console.Write("cycles= ");
        Console.WriteLine(Set.cycles);
        Console.Write("sequences= ");
        Console.WriteLine(Set.sequences);
        Console.Write("digio= ");
        Console.WriteLine(Set.digio);
        Console.Write("digval= ");
        Console.WriteLine(Set.digval);
        Console.Write("dac0= ");
        Console.WriteLine(Set.dac0);
        Console.Write("dac1= ");
        Console.WriteLine(Set.dac1);
        Console.Write("dac2= ");
        Console.WriteLine(Set.dac2);
        Console.Write("dac3= ");
        Console.WriteLine(Set.dac3);
        Console.Write("dac4= ");
        Console.WriteLine(Set.dac4);
        Console.Write("dac5= ");
        Console.WriteLine(Set.dac5);
        Console.Write("serno= ");
        Console.WriteLine(Set.serno);
        Console.Write("ddruse= ");
        Console.WriteLine(Set.ddruse);
        Console.Write("active= ");
        Console.WriteLine(Set.active);
        Console.Write("holdafter= ");
        Console.WriteLine(Set.holdafter);
        Console.Write("swpreset= ");
        Console.WriteLine(Set.swpreset);
        Console.Write("fstchan= ");
        Console.WriteLine(Set.fstchan);
        Console.Write("timepreset= ");
        Console.WriteLine(Set.timepreset);
    }

    static void PrintSetting(ref ACQSETTING acq)
    {
        Console.Write("range= ");
        Console.WriteLine(acq.range);
        Console.Write("cftfak= ");
        Console.WriteLine(acq.cftfak);
        Console.Write("roimin= ");
        Console.WriteLine(acq.roimin);
        Console.Write("roimax= ");
        Console.WriteLine(acq.roimax);
        Console.Write("nregions= ");
        Console.WriteLine(acq.nregions);
        Console.Write("caluse= ");
        Console.WriteLine(acq.caluse);
        Console.Write("calpoints= ");
        Console.WriteLine(acq.calpoints);
        Console.Write("active= ");
        Console.WriteLine(acq.active);
        Console.Write("roipreset= ");
        Console.WriteLine(acq.eventpreset);
    }
```

```
static int run(string command, ref int nDev)
{

    int[] Spec = new int[30];
    if (command == "H") help();
    else if (command == "Q")
    {
        return 1;
    }
    else if (command == "T")
    {
        ACQSETTING MSetting;
        Console.Write("CHN ");
        Console.WriteLine(nDev);
        MSetting = new ACQSETTING();
        GetSettingData(ref MSetting, nDev);
        PrintSetting(ref MSetting);
    }
    else if (command == "S")
    {
        ACQSTATUS Status;
        Status = new ACQSTATUS();
        GetStatusData(ref Status, nDev);
        PrintStatus(ref Status);
    }
    else if (command == "D")
    {
        ACQSETTING MSetting;
        MSetting = new ACQSETTING();
        GetSettingData(ref MSetting, nDev);
        //              for (i=0; i<30; i++)
        //                  GetBlock(ref Spec[i], i, i+1, 1, nDev);
        GetBlock(ref Spec[0], 0, 30, 1, nDev);
        PrintDat(MSetting.range, ref Spec);
    }
    else if (command == "F")
    {
        DATSETTING DSetting;
        DSetting = new DATSETTING();
        GetDatSetting(ref DSetting);
        PrintDatSetting(ref DSetting);
    }
    else
    {
        RunCmd(0, command);
    }
    return 0;
}

static void help()
{
    Console.WriteLine("Commands:");
    Console.WriteLine("Q    Quit");
    Console.WriteLine("H    Help");
    Console.WriteLine("S    Status");
    Console.WriteLine("T    Setting");
    Console.WriteLine("D    Data");
    Console.WriteLine("F    Datsetting");
    Console.WriteLine("(... more see command language in MPANT help)");
    Console.WriteLine();
}
```

```
        static void PrintDat(int range, ref int[] datp)
        {
            int i;
            Console.Write("first 30 of ");
            Console.Write(range);
            Console.WriteLine(" datapoints:");
            for (i = 0; i < 30; i++)
                Console.WriteLine(datp[i]);
        }
    }
}
```

6 Appendix: The DMCS8 DLL

The Dynamic Link Library DMCS8.DLL provides an interface to the server program MCS8.EXE that is used by the MPANT software, but can also be used by any Windows program. Custom DLL functions allow user-defined calculated parameter spectra. In the following this DLL is described in detail including the complete source code.

6.1 The Structures

In struct.h some important structures are defined. A structure of type ACQSTATUS contains parameters describing the status of an acquisition. There is an array of these structures stored in the DLL, DLLMStatus[ ] contains general mpa status data for each MCA module, and DLLStatus[ ] Spectrum status data.

```
#define GET_WM_COMMAND_ID(w)  LOWORD(w)
#define GET_WM_COMMAND_CMD(w,l) HIWORD(w)
#define GET_WM_COMMAND_HWND(l) l
#define GET_WM_SCRHWND(l) l
#define GET_WM_SCROLLPOS(w,l) (short)HIWORD(w)
#define FIND_WINDOW(a,b) FindWindow(a,b)
#define HUGE
#define _fmemcpy memcpy
#define _fstrcpy strcpy


typedef struct {

        int use;
        int port;
        unsigned int baud;
        int dbits;
        int sbits;
        int parity;
        int echo;
        HWND hwndserver;
        LPSTR cmd;
} COMCTL, far *LPCOMCTL;

#define ST_RUNTIME          0
#define ST_OFLS             1
#define ST_TOTALSUM         2
#define ST_ROISUM           3
#define ST_ROIRATE          4
#define ST_SWEEPS           5
#define ST_STARTS           6
#define ST_ZEROEVTS         7

typedef struct{

  unsigned int started;    // aquisition status
  unsigned int maxval;     // maxval
  double cnt[8];           // status: runtime in msec, ofls,
                           // total sum, roi sum, roi rate, sweeps, starts, zeros
} ACQSTATUS;
```

DATSETTING is a structure type containing data format settings.

```
typedef struct {
  int savedata;            // bit 0: auto save after stop
                           // bit 1: write listfile
                           // bit 2: listfile only, no evaluation
  int autoinc;           // 1 if auto increment filename
  int fmt;               // format type (seperate spectra):
                         // 0 == ASCII, 1 == binary,
                         // 2 == CSV
  int mpafmt;            // format used in mpa datafiles
  int sephead;           // seperate Header
  int smpts;
  int caluse;
  char filename[256];
  char specfile[256];
  char command[256];
} DATSETTING;
```

REPLAYSETTING is a structure type containing Replay settings.

```
typedef struct {
  int use;               // 1 if Replay Mode ON
  int modified;          // Bit 0: 1 if different settings are used
                         // (Bit 1: Write ASCII, reserved)
  int limit;             // 0: all,
                         // 1: limited sweep range
  int speed;             // replay speed in units of 100 kB / sec
  double startsfrom;     // first start#
  double startsto;       // last start#
  double startspreset;   // last start - first start
  char filename[256];
} REPLAYSETTING;
```

ACQSETTING is a structure type containing all the spectra settings

```
typedef struct{
  int range;             // spectrum length
  int cftfak;            // LOWORD: 256 * cft factor (t_after_peak / t_to_peak)
                         // HIWORD: max pulse width for CFT
  int roimin;            // lower ROI limit
  int roimax;            // upper limit: roimin <= channel < roimax
  int nregions;          // number of regions
  int caluse;            // bit0: 1 if calibration used, higher bits: formula
  int calpoints;         // number of calibration points
  int param;             // (reserved:) for MAP and POS: LOWORD=x, HIWORD=y
  int offset;            // (reserved:) zoomed MAPS: LOWORD: xoffset, HIWORD, yoffset
  int xdim;              //  (reserved:) x resolution of maps
  unsigned int bitshift; // LOWORD: Binwidth = 2 ^ (bitshift)
                         // HIWORD: Threshold for Coinc
  int active;            // Spectrum definition words for CHN1..8:
                         // active & 0xF  ==0 not used
                         //               ==1 single
                         // bit 8: Enable Tag bits
                         // bit 9: start with rising edge
                         // bit 10: time under threshold for pulse width
                         // bit 11: pulse width mode for any spectra with both edges enabled
                         // Spectrum definition words for calc. spectra:
                         // active & 0xF  ==3 MAP, ((x-xoffs)>>xsh) x ((y-yoffs)>>ysh)
                         //                 ((x-xoffs)>>xsh) x ((y-timeoffs)>>timesh)
                         //              or ((x-timeoffs)>>timesh x ((y-yoffs)>>ysh)
                         //          bit4=1: x zoomed MAP
                         //          bit5=1: y zoomed MAP
                         //               ==5 SUM, (x + y)>>xsh
                         //               ==6 DIFF,(x - y + range)>>xsh
                         //               ==7 ANY, (for compare)
```

```
                        //              ==8 COPY, x
                        //              ==9 DLL  fDLL(x,y,z),
                        //              ==0xA Sweep HISTORY, Sweepnum(x)
                        // bit 8..11 xsh, bit 12..15 ysh or bit 8..15 xsh
                        // HIWORD(active) = condition no. (0=no condition)
  double eventpreset;   // ROI preset value
  double dummy1;        // (for future use..)
  double dummy2;                //
  double dummy3;                //
} ACQSETTING;

typedef struct{

  int range;            // spectrum length
  int cftfak;           // LOWORD: 256 * cft factor (t_after_peak / t_to_peak)
                        // HIWORD: max pulse width for CFT
  int roimin;           // lower ROI limit
  int roimax;           // upper limit: roimin <= channel < roimax
  int nregions;         // number of regions
  int caluse;           // bit0: 1 if calibration used, higher bits: formula
  int calpoints;        // number of calibration points
  int param;            // (reserved:) for MAP and POS: LOWORD=x, HIWORD=y
  int offset;           // (reserved:) zoomed MAPS: LOWORD: xoffset, HIWORD, yoffset
  int xdim;             //  (reserved:) x resolution of maps
  unsigned int bitshift;  // LOWORD: Binwidth = 2 ^ (bitshift)
                        // HIWORD: Threshold for Coinc
  int active;           // Spectrum definition words for CHN1..8:
                   // active & 0xF  ==0 not used
                   //               ==1 enabled
                   // bit 8: Enable Tag bits
                   // bit 9: start with rising edge
                   // bit 10: time under threshold for pulse width
                   // bit 11: pulse width mode for any spectra with both edges enabled
                   // Spectrum definition words for calc. spectra:
                   // active & 0xF  ==3 MAP, ((x-xoffs)>>xsh) x ((y-yoffs)>>ysh)
                   //        bit4=1: x zoomed MAP
                   //        bit5=1: y zoomed MAP
                   //               ==5 SUM, (x + y)>>xsh
                   //               ==6 DIFF,(x - y + range)>>xsh
                   //               ==7 ANY, (for compare)
                   //               ==8 COPY, x
                   //               ==10 SW-HIS, Sweep History
                   // bit 8..11 xsh, bit 12..15 ysh or bit 8..15 xsh
                   // HIWORD(active) = condition no. (0=no condition)
  double eventpreset;   // ROI preset value
  double dummy1;        // (for future use..)
  double dummy2;                //
  double dummy3;                //
                        // MPANT or Server private saved settings:
  int type;                     // 0=single, 1=MAP, 2=ISO...
  int ydim;                     // y resolution of maps
  int reserved[16];
} EXTACQSETTING;


typedef struct {

  int sweepmode;        // sweepmode & 0xF: 0 = normal,
                        // 1=differential (relative to first stop in sweep)
                        // 4=sequential
                        // 5=seq.+diff (Ch1), bit0 = differential mode
                        // 6 = CORRELATIONS
                        // 7 = diff.+Corr.
```

```
                      // 9=differential to stop in Ch2, bit3 = Ch2 ref (diff.mode)
                      // 0xF = Corr.+diff (Ch2)
                      // bit 4: Softw. Start
                      // bit 5: "Don't show" tagbits
                      // bit 6: Endless
                      // bit 7: Start event generation
                      // bit 8: Enable Tag bits
                      // bit 9: start with rising edge
                      // bit 10: time under threshold for pulse width
                      // bit 11: pulse width mode for any spectra with both edges enabled
                      // bit 12: abandon Sweepcounter in Data
                      // bit 13: "one-hot" mode with tagbits
                      // bit 14: start ref (diff.mode)
                      // bit 15: enable start input
              // bit 16..bit 22 ~(input channel enable)
              // bit 24:  reserved
              // bit 25:  reserved
              // bit 27: Folded
              // bit 28: Interleaved
int prena;            // bit 0: realtime preset enabled
                      // bit 1:
                      // bit 2: sweep preset enabled
                      // bit 3: ROI preset enabled
                      // bit 4: Starts preset enabled
             // bit 5: ROI2 preset enabled
             // bit 6: ROI3 preset enabled
             // bit 7: ROI4 preset enabled
             // bit 8: ROI5 preset enabled
             // bit 9: ROI6 preset enabled
             // bit 10: ROI7 preset enabled
             // bit 11: ROI8 preset enabled
int cycles;           // for sequential mode
int sequences;        //
int syncout;          // LOWORD: sync out; bit 0..5 NIM syncout, bit 8..13 TTL syncout
                      // bit7: NIM syncout_invert, bit15: TTL syncout_invert
                      // 0="0", 1=5 MHz, 2=50 MHz, 3=100 MHz, 4=97.656 MHz,
                      // 5=195.625 MHz, 6= 195 MHz (int ref), 7=Start, 8=Ch1, 9=Ch2, 10=Ch3,
                      // 11=Ch4, 12=Ch5, 13=Ch6, 14=Ch7, 15=GO, 16=Start_of_sweep, 17=Armed,
                      // 18=SWEEP_ON, 19=WINDOW, 20=HOLD_OFF, 21=EOS_DEADTIME
                      // 22=TIME[0],...,51=TIME[29], 52...63=SWEEP[0]..SWEEP[11]
                      //
int digio;            // LOWORD: Use of Dig I/O, GO Line:
              // bit 0: status dig 0..3
              // bit 1: Output digval and increment digval after stop
              // bit 2: Invert polarity
              // bit 3: Push-Pull output, not possible
              // bit 4:  Start with Input Dig 4
              // bit 5:  Start with Input GO
              // bit 8: GOWATCH
              // bit 9: GO High at Start
              // bit 10: GO Low at Stop
              // bit 11: Clear at triggered start
              // bit 12: Only triggered start
int digval;           // digval=0..255 value for samplechanger
int dac0;             //  DAC0 value (START)
                      //  bit 16: Start with rising edge
int dac1;             //  DAC1 value (STOP 1)
int dac2;             //  DAC2 value (STOP 2)
int dac3;             //  DAC3 value (STOP 3)
int dac4;             //  DAC4 value (STOP 4)
int dac5;             //  DAC5 value (STOP 5)
                      // bit (14,15) of each word: 0=falling, 1=rising, 2=both, 3=both+CFT
```

```
                              // bit 17 of each: pulse width mode under threshold
  int fdac;                   // reserved
  int tagbits;                // number of tagbits
  int extclk;                 // use external clock
  int periods;                // number of periods in folded mode, sweeplength = range * periods
  int serno;                  // serial number
  int ddruse;                 // bits[0:1] 1=DDR_USE, 2=DDR_2GB, 3=DDR_4GB
                              // bits[2]: usb_usage (0 means demo mode)
                              // bits[4:5]: wdlen
  int active;                 // module in system
  double holdafter;           // Hold off
  double swpreset;            // sweep preset value
  double fstchan;             // acquisition delay
  double timepreset;          // time preset
} BOARDSETTING;


typedef struct {
  int nDevices;               // Number of channels = number of modules * (6 + 8)
  int nDisplays;              // Number of histograms = nDevices + Positions + Maps
  int nSystems;               // Number of independent systems = 1
  int bRemote;                // 1 if server controlled by MPANT
  unsigned int sys;         // System definition word:
                        // bit0=0, bit1=0: dev#0 in system 1
                        // bit0=1, bit1=0: dev#0 in system 2
                        // bit0=0, bit1=1: dev#0 in system 3
                        // bit0=1, bit1=1: dev#0 in system 4
                        // bit2..bit6: ...
                        // bit6=1, bit7=1: dev#3 in system 4
                        // bit 31: any preset stops all
  int sys0[56];             // (reserved:) System definition words for CHN1..:
                        // bit 0 CHN active
                        // bit 1 =1 CHN coinc, =0 single
                        // bit 2..4 CHN in system1..7
  int sys1[56];             // (reserved:) CHN in System
} ACQDEF;

typedef struct {
  unsigned int adcnum;    // Number of active ADC's (=0)
  unsigned int tofnum;    // Number of active TOF channels
  unsigned int ntofs0;    // Number of TOF inputs
  unsigned int modules;   // Number of modules
  unsigned int nadcs;     // Number of ADCs (=0)
  int sys0[56];                   // (reserved:) System definition words for CHN1..:
                                  // see active definition in ACQSETTING
  int sys1[56];                   // CHN in System (=1)
  int adcs[8];                    // Number of ADCs per module (0)
  int tofs[8];                    // Number of TOF inputs per module
  int speed[8];                   // TOF speed in module: 3=80ps ..100ps, 2=200ps, 1=400ps, 0=800ps
} COINCDEF;
```

ACQDATA is a structure type containing pointers to the data belonging to a measurement. The data is stored in a named memory-mapped file (see DLL source).

```
typedef struct{
  unsigned int HUGE *s0;        // pointer to spectrum
  unsigned int *region;       // pointer to regions
  unsigned char *comment0;    // pointer to strings
  double *cnt;                // pointer to counters
  HANDLE hs0;
  HANDLE hrg;
  HANDLE hcm;
  HANDLE hct;
} ACQDATA;
```

Data[nDisplay].s0 points to a block memory of unsigned long 32-bit numbers containing the spectra data.

Data[nDisplay].region points to a block of 256 unsigned long numbers containing the Roi (Region of interest) boundaries as defined in the MPANT program. The first Roi is:
Data[nDisplay].region[0] <= x < Data[nDisplay].region[1], the second
Data[nDisplay].region[2] <= x <Data[nDisplay].region[3] and so on, 128 Rois are possible.
These Rois have nothing to do with the special Roi defined in the ACQSETTING structure for the Roi Preset.

Data[nDisplay].comment0 points to a block of 1024 bytes containing the strings.
Data[nDisplay].comment0[0] is the first byte of the 0. comment line,
Data[nDisplay].comment0[60] is the first byte of the 1. comment line,
Data[nDisplay].comment0[120] is the first byte of the 2. comment line,
Data[nDisplay].comment0[180] is the first byte of the 3. comment line,
Data[nDisplay].comment0[240] is the first byte of the 4. comment line,
Data[nDisplay].comment0[300] is the first byte of the 5. comment line,
Data[nDisplay].comment0[360] is the first byte of the 6. comment line,
Data[nDisplay].comment0[420] is the first byte of the 7. comment line,
Data[nDisplay].comment0[480] is the first byte of the 8. comment line,
Data[nDisplay].comment0[540] is the first byte of the 9. comment line,
Data[nDisplay].comment0[600] is the first byte of the 10. comment line,
Data[nDisplay].comment0[660] is the first byte of the data file name,
Data[nDisplay].comment0[760] is the first byte of the calibration unit name,
Data[nDisplay].comment0[800] is the first byte of the command string.
Data[nDisplay].comment0[880] is the first byte of the 11. comment line,
Data[nDisplay].comment0[960] is the first byte of the 12. comment line

Data[nDisplay].cnt points to a block of 448 double numbers containing:
Data[nDisplay].cnt[0] = Realtime
Data[nDisplay].cnt[1] = Totalsum
Data[nDisplay].cnt[2] = ROIsum
Data[nDisplay].cnt[3] = Totalrate
Data[nDisplay].cnt[4] = Net ROIsum
Data[nDisplay].cnt[5] = Livetime / Sweeps
Data[nDisplay].cnt[6] = Deadtime (%)
Data[nDisplay].cnt[7] = Cycle counter in sequential mode
Data[nDisplay].cnt[8] = Sequence counter
Data[nDisplay].cnt[11] = c0 Calibration parameter
Data[nDisplay].cnt[12] = c1 Calibration parameter
Data[nDisplay].cnt[13] = c2 Calibration parameter
Data[nDisplay].cnt[14] = c3 Calibration parameter, for scope displays: vpp (mV) / 65535
Data[nDisplay].cnt[15] for nDisplay =0:
bit 0: system 1 started,
bit 1: system 2 started,
bit 2: system 3 started,
bit 3: system 4 started
Data[nDisplay].cnt[19] = Channel number of first calibration Point
Data[nDisplay].cnt[35] = Energy value at first calibration Point
Data[nDisplay].cnt[20] = Channel number of 2. calibration Point
Data[nDisplay].cnt[36] = Energy value at 2. calibration Point...

Data[nDisplay].cnt[64..191] = Energy value for calibration peak in ROI
0..127
Data[nDisplay].cnt[192] = ROI Sum in ROI 0 (actualized by MPANT when
selected in any spectra display)
Data[nDisplay].cnt[193] = ROI Net Sum in ROI 0 …
Data[nDisplay].cnt[447] = ROI Net Sum in ROI 127

```
typedef struct {
  int nDevices;          // Number of channels = number of modules * 6
  int nDisplays;         // Number of histograms = nDevices + Positions + Maps
  int nSystems;          // Number of independent systems = 1
  int bRemote;           // 1 if server controlled by MPANT
  unsigned int sys;      // System definition word:
                       // bit0=0, bit1=0: dev#0 in system 1
                       // bit0=1, bit1=0: dev#0 in system 2
                       // bit0=0, bit1=1: dev#0 in system 3
                       // bit0=1, bit1=1: dev#0 in system 4
                       // bit2..bit6:
                       // bit6=1, bit7=1: dev#3 in system 4
} LVACQDEF;

typedef struct {
  int savedata;     // bit 0: auto save after stop
                    // bit 1: write listfile
                    // bit 2: listfile only, no evaluation
  int autoinc;      // 1 if auto increment filename
  int fmt;          // format type (seperate spectra):
                    // 0 == ASCII, 1 == binary,
                    // 2 == CSV
  int mpafmt;           // format used in mpa datafiles
  int sephead;          // seperate Header
  int smpts;
  int caluse;
} LVDATSETTING;

typedef struct {
  int use;              // 1 if Replay Mode ON
  int modified;         // Bit 0: 1 if different settings are used
                        // (Bit 1: Write ASCII, reserved)
  int limit;            // 0: all,
                        // 1: limited sweep range
  int speed;        // replay speed in units of 100 kB / sec
  double startsfrom;    // first start#
  double startsto;      // last start#
  double startspreset;  // last start - first start
} LVREPLAYSETTING;

typedef struct {
  unsigned int adcnum;     // Number of active ADC's (=0)
  unsigned int tofnum;     // Number of active TOF channels
  unsigned int ntofs0;       // Number of TOF inputs
  unsigned int modules;    // Number of modules
  unsigned int nadcs;      // Number of ADCs (=0)

} LVCOINCDEF;
```

6.2 The Library Functions

In the header file DMCS8.h all functions are declared. It is already listed in chapter 4.1.

6.3 The Ordinal numbers of the functions

In the Definition file DMCS8.def the ordinal numbers of the library fuctions are defined:

```
;*DMCS8.def
;*Hardware:            MCS8
;*Op System:           Windows
;*Compiler:            MSVC++ 6.0
;*

LIBRARY DMCS8

SECTIONS
        dmcs8sh READ WRITE SHARED

EXPORTS
;    Functions in dmcs8.c
     StoreSettingData        @2
     GetSettingData          @3
     StoreStatusData         @4
     GetStatusData           @5
     Start                   @6
     Halt                    @7
     Continue                @8
     NewSetting              @9
     ServExec                @10
     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
     LVGetOneCnt             @32
     LVGetStr                @33
     SubData                 @34
     Smooth                  @35
     StoreExtSettingData     @36
     GetExtSettingData       @37
     StoreMCSSetting         @38
     GetMCSSetting           @39
     StoreDatSetting         @40
     GetDatSetting           @41
     StoreReplaySetting      @42
     GetReplaySetting        @43
```

GetDatPtr	@44
ReleaseDatPtr	@45
LVGetOneRoi	@46
GetSVal	@47
GetDatInfo	@48
BytearrayToShortarray	@49
LedBlink	@50
DigInOut	@51
StoreMStatusData	@52
GetMStatusData	@53
StoreCDefData	@54
GetCDefData	@55
StoreMP4Setting	@60
GetMP4Setting	@61
GetRoiIndex	@62
DeleteRoi	@63
GetRoiSum	@64
SelectRoi	@65
LVGetRoinam	@66
LVGetSpecLength	@67
LVGetDefData	@68
LVGetDatSetting	@69
LVGetReplaySetting	@70
LVGetProiDat	@71
LVGetRoiRect	@72
LVGetRroiDat	@73
LVGetCDefData	@74

6.4 The source code of the functions

In the source file DMCS8.c the body of the library functions is coded. It can be found in the DLL subfolder of the supplied software.

6.5 How to compile the DLL

The 32 bit DLL can be compiled with the Microsoft Visual C/C++ compiler version 6.0 or higher. To recompile the DLL under VC 6.0, use the makefile DMCS8.dsw. For later version create a new DLL project and include the files DMCS8.c and DMCS8.def.

The 64 bit DLL can be compiled with Microsoft Visual Studio 2013 using the file DMCS8.sln.