BioImaging Zentrum (LMU) / TILL Photonics

Scan Control DSP v1.7.0

Firmware Interface Description

I INDICES

I.1 Table of Contents

I.2 List of Figures

I.3 List of Tables

II INTRODUCTION

This document describes the functionality of the Scan-Control DSP with firmware version 1.6.2 (2006-06-12). The Scan-Control DSP generates the control signal for the *SmartMove* DC900 galvanometer driver boards, an 8 MHz sample clock, 8 digital outputs and currently 4 analog output channels with 16 bit resolution. These output devices are addressed via logical scan channels. The control signal generated by the Scan-Control DSP is defined by a protocol, which is uploaded to the DSP via RS232. There also is a function to set the value of a scan channel directly without changing the protocol.

III CHANGES FROM PREVIOUS VERSIONS

Previous version: v1.6.2

III.1 Changed Features

- Expanded protocol memory from 5,000 to 10,000 entries.

III.2 New Features

- Added a changeable Offset that is added to each of the galvo channels (channels 3, 4, 5, and 6). The offset is specified in counts (not MicroCounts) and can only be changed by a control command when no protocol is running.
- Added DSP-command to show the current value of a channel, interactively.

III.3 Bugfixes

IV SYSTEM OPERATION

The Scan-Control DSP generates the control signal for the *SmartMove* DC900 galvanometer driver boards, an 8 MHz sample clock, 8 digital outputs and currently 4 analog outputs with 16 bit resolution. The galvo driver boards must have a firmware version that reads position data via time multiplexed serial interface (TMSI) with a frame length of 10 µs. The firmware can be obtained from SmartMove. The CPLD has to be programmed accordingly. The recent Smart Move galvo drivers already have the appropriate CPLD programming.

IV.1 System Startup

The Scan Control DSP is powered up by connecting Ground and VCC of a DC power supply to SV10 pins 9 and 10, respectively (see Figure 3, Table 11, and section V.3).
After it is switched on, the DSP initializes. This is signaled by a short lighting up of all LEDs of the DSP board. Initialization takes about 1s. After initialization, only LED #1 (see Figure 2) is on to signal that the board is ready.

IV.2 LED codes

On the DSP board there is a row of five LEDs (see Figure 2). LEDs #1,2,4, and 5 are green and LED #3 is red. These LEDs are used to signal the system state. The Meaning of the LED codes is shown in Table 1.

Table 1: LED codes

LED					System state
1	2	3	4	5	
x	x	x	x	x	System starting up
x					System ready
x				x	Protocol being executed
		x			Severe error

IV.3 Communication PC - Scan-Control DSP

IV.3.1 RS-232 settings

The Scan-Control DSP receives commands (“DSP-commands”), via RS232. For the parameters of the RS232 communication see Table 2. The RS232 interface is located on connector SV6 of the DSP board. See Figure 2 for the location of SV6 on the DSP board and Table 10 for the pin-out of SV6.

Table 2: RS232 communication parameters

57600 baud
8 data bits
no parity
1 stop bit
No flow control

IV.3.2 DSP-commands

DSP-commands are one character long and have a variable number of parameters depending on the specific command. The parameter list immediately follows the command without a separating character or separated by spaces or tabs. Parameters in the parameter list are separated from each other by commas. Leading or trailing spaces or tabs are ignored.
A DSP-command must be terminated by either newline character (0x10) or a carriage return character (0x13) or a semicolon (‘;’ 0x3B). For an overview of the DSP-commands see Appendix A (section VI.1).
When a character is sent to the DSP while the DSP is executing a protocol (following the ‘X’ DSP-command), this character is not interpreted as DSP-command, but is discarded. Its only effect is to stop the currently running protocol (“*stop-character*”).

IV.3.3 DSP echo

Every character sent to the DSP is echoed by the DSP. This includes all spaces, tabs and terminating characters (newlines, carriage returns and semicola). Hence the “local echo” option in a terminal program should be switched off.
There is one exception to this rule: A stop-character, which is sent while the DSP is executing a protocol, is not echoed. All characters following the stop-character are again echoed and interpreted.

IV.3.4 DSP responses

Some commands do not send a response, some return one decimal number in ASCII-representation as status/error code, and others return larger strings. The responses of the DSP-commands are described along with the commands in Appendix A (section VI.1). The status/error codes are listed in Appendix C (section VI.3)

IV.4 Communication with Galvanometer drivers and other devices

Output / Input data is sent to / received from the peripheral devices via a time multiplexed serial interface (“TMSI”). The TMSI transmits 14 bytes of data in each direction during a time frame of 10 µs length, each byte in one “timeslot”. The scan control DSP sends a clock signal, a frame sync, which marks the beginning of a time frame, a data Tx, and receives data via an Rx line. See Table 3 for the communication parameters of the TMSI.

Table 3: TMSI communication parameters

Parameter	value
Cycle length	10 µs
# Time Slots	14
Bits per Time-Slot	8
Shift direction	MSB first
Frame Sync Length	Bit length (One bit before frame start)
Frame Sync Polarity	negative

To facilitate communication with the devices, some timeslots are grouped in channels. Some channels occupy one timeslot, others occupy two timeslots. The values of the channels can be changed by direct DSP-commands or from within precisely timed protocols on the DSP (see section IV.5).
The functions of the channels are displayed in Table 4. The physical connection to the TMSI is established via Pins 1-8 of SV1 of the DSP board. See Figure 2 for the position of SV1 and Table 8 for the pin assignment.

Table 4: TMSI time slots and scan channels

Channel	Timeslot	Function Tx	Function Rx
0	0	reserved	
1	1	Analog out configuration	
2	2	Analog out value high byte	
	3	Analog out value low byte	
3	4	Position Out Galvo 0 low byte	Position In Galvo 0 low byte
	5	Position Out Galvo 0 high byte	Position In Galvo 0 high byte
4	6	Position Out Galvo 1 low byte	Position In Galvo 1 low byte
	7	Position Out Galvo 1 high byte	Position In Galvo 1 high byte
5	8	Position Out Galvo 2 low byte	Position In Galvo 2 low byte
	9	Position Out Galvo 2 high byte	Position In Galvo 2 high byte
6	10	Position Out Galvo 3 low byte	Position In Galvo 3 low byte
	11	Position Out Galvo 3 high byte	Position In Galvo 3 high byte
7	12	8x Digital Out (3 first bits mapped to host interface)	
8	13	reserved	

IV.4.1 The SmartMove DC900 Galvanometer drivers

Position control signals are sent to the galvanometer drivers in channels 3-6. Position values are coded in 16 bit values (“counts”). Therefore, each galvanometer driver uses two timeslots. Position values are transferred in the order LowByte – HighByte. The scan control DSP internally calculates with position values from –(2^35) to (2^35)-1 (i.e. 36 bit range). These values are called MicroCounts (i.e. counts/1024/1024. Only the upper 16 bits of these values are transmitted to the galvanometer control DSPs. It was necessary to implement this to avoid rounding errors in slow raster scans of small scan fields. See Figure 1 and Table 5 for the

relation between the digital signal and the resulting galvo angle. A galvo position can be changed with the scan control commands “V”, “R”, “I”, and “J” in a protocol or with the DSP-command “V”, when no protocol is running.

It is possible to apply a constant offset to a galvo-channel from within a protocol, which is added to the value of the respective channel before the value is transmitted to the galvo driver board. The value of the offset is defined outside the protocol using the DSP-command “O” when no protocol is running, and the offset is switched on and off within the protocol (scan-command scan-command “O”). Its purpose is to quickly move an already loaded scan to a different position without having to load the whole scan, again.
The offset is specified in Counts (not MicroCounts). At system startup its value is 0, and at the start of a protocol run, it is switched off. One can think of it by not changing the value of the scan-channel, but just changing the position signal that is transmitted to the galvo driver.

When the DSP-board is used without DSC mainboard, the DC900 galvo drivers are connected to the DSP-board directly by connecting pins 1-8 of SV1 on the DSP-board to the corresponding pins on the DC900 boards. When the DSP-board is used with the DSC mainboard, the DSC is connected to the TILL Photonics Scan Control Unit (SCU) via the 8-pin RJ45 connector on the DSC front panel with a standard S/FTP Ethernet cable.

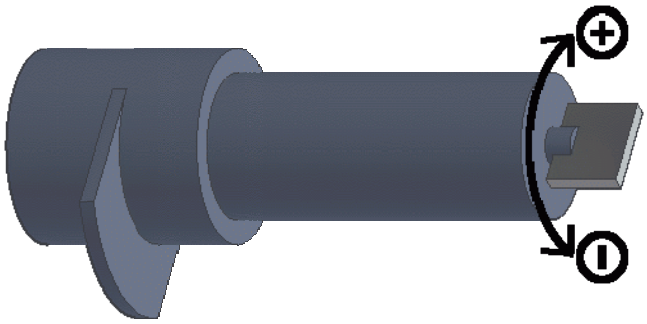

Figure 1: Relation between sign of digital signal and galvanometer turning direction.

Table 5: Conversion digital counts – galvanometer angles

36-bit counts	16-bit counts	Galvo angle mech. / °	Galvo angle opt. / °
+ 34,359,738,367	+ 32,767	+ 15	+ 30
- 34,359,738,368	- 32,767	- 15	- 30
1,048,576	1	0.000458	0.000916
1	0	0	0
2,290,649,225	2,185	1	2

IV.4.2 Digital Outputs

The digital out signal is written to channel 7. The CPLD of the DSC mainboard extracts this byte and sets the status of 8 line drivers accordingly. For applications that do not require the DSC, as a temporal solution the lowest three bits of channel 7 are written to GPIO lines. These lines can be accessed on SV10 of the DSP board. See Table 11 for the pin assignment and Figure 3 for the location of SV10.
Two digital outputs can be accessed via SMB-connectors on the DSC front panel. “T-OUT” is controlled by bit 1 (value 2) and “D-OUT” is controlled by bit 2 (value 4). Voltages are 0 V for low and 5 V for high. The other digital outputs can be accessed via the female Mini D Ribbon connector labeled AOP on the DSC front panel. See Table 12 for the pin-out of this connector.

IV.4.3 *Digital Inputs and Trigger In*

Digital inputs are sensed by line drivers on the DSC board and transmitted to the Scan Control DSP via the input direction of channel 7. Currently, the state of the digital inputs can only be read when no protocol is running by printing out the buffer for channel 7.
There is one exception: Bit 4 of channel 7 (0-based) is used as Trigger-In. When the scan commands “U” (“up”, wait for rising trigger) and “D” (“down”, wait for falling trigger) are used in a protocol, they pause protocol execution until a rising or falling level of this bit is detected. While the protocol is waiting for a trigger, increments are not applied to the scan channels.
Due to a bug in the current version, a trigger event is not sensed if it happens in cycle 0 (zero). Hence, the “U” and “D” commands should not be used in cycle 0.
A trigger line can be connected to the DSC via the “T-IN” SMB-connector on the DSC front panel. Standard TTL levels are expected. A jump from 0 to 5 V is detected as rising trigger, and a jump from 5 to 0 V is detected as falling trigger.

IV.4.4 *Analog Outputs*

Analog Out signals are generated on a separate riser board (“analog board”) on the DSC mainboard. The analog board has 2 DACs, one 16-bit and one 12-bit, with 4 channels each. Currently, only the 16-bit DAC can be used.
The DACs are controlled via TMSI channels 1 (“Analog out configuration”) and 2 (“Analog out value”). TMSI channel 1 configures whether a DAC is updated and which DAC-channel is updated. See Table 6 for the function of TMSI channel 1. The value with that the DAC-channel is updated is written into channel 2. For the 16-bit DAC a value of 0 corresponds to an output-voltage of -10 V and a value of 65535 corresponds to an output-voltage of +10 V.
To change the value of an analog output in a protocol set bit 4 of TMSI channel 1 and the combination of bits 6 and 7 that codes for the desired DAC-channel. In the same cycle of the protocol, set TMSI-channel 2 to the desired value. E.g. to set channel 3 of the 16-bit DAC to +5 V you have to set of TMSI channel 1 to144 (bits 4 and 7) and TMSI channel 2 to 49151. In the next cycle of the protocol, set TMSI channel 1 to 0. If channel 1 is not set to 0, the DAC is updated every cycle. This can produce small voltage spikes on the respective DAC-channel.
The Analog Outputs can be accessed via the female Mini D Ribbon connector labeled AOP on the DSC front panel. See Table 12 for the pin-out of this connector.

Table 6: Bits TMSI channel 1 ("Analog out configuration")

Bit	Function
0, 1, 2	Reserved. Set to 0 for future compatibility.
3	DAC-selection 16-bit / 12-bit. 0: 16-bit DAC 1: 12-bit DAC Currently only the 16-bit DAC is supported.
4	DAC-update ("LDAC") 0: DAC-value is not updated. 1: DAC-value of selected DAC-channel is updated. Set to 0 when DAC value does not change to avoid glitches.
5	Quick load for the 16-bit DAC. 0: Only selected DAC-channel is updated. 1: All four channels of the 16-bit DAC are updated, if Bit 3 is 0 at the same time.
6, 7	Select the DAC-channel (1-4) to be updated. Bit 6: Low bit Bit 7: High bit

IV.4.5 Clock Output

To synchronize a data-acquisition hardware with the galvanometer boards, an 8 MHz clock and a 4 MHz clock synchronous to the clock of the DSP board can be accessed via SV3. See Table 9 for the pin assignment and Figure 3 for the location of SV10.
The 8 MHz clock can be accesses via the "CLK" SMB-connector on the DSC front panel.

IV.5 Scan Control Protocols

Protocols to define the control signal generated by the Scan-Control DSP can be composed using the **`DSP_CMD_ADD_PROT_CMD`** DSP-command ('**`A`**'). The argument for this command is a string containing one scan control command and its arguments. Scan control commands are listed in Appendix B (section VI.2). All scan control commands have three parameters:

1. cycle (the cycle, in which a command is executed)
2. channel (the scan channel that the command acts on)
3. value (function depends on the command).

All parameters are passed to the DSP as decimal numbers. Cycles and values are internally converted to long integers by the `atol()` C-Function. Channels are internally converted to integers by the `atoi()` C-Function.
In case that a parameter is not used by the command, it can be set to any value, but still has to be specified to ensure the correct order of the parameters.
Protocols contain a sequence of up to 10000 scan control commands with fixed timings on the 10 µs raster of the DSP. Commands are used to set the absolute value of a channel (command "V"), a relative value (increase, decrease; command "R"), an increment of a channel ("1st increment"; "I"), which is executed every 10 µs, an increment of an increment ("2nd increment; "J"), and loop starts and stops ("S" and "E", respectively). Loops can be nested up to 100 levels.

1st increments which are set in one 10 µs cycle are not executed until the next cycle. As an example: When the value of a channel is set to 100 in the first cycle and the increment is set to 50 in the first cycle, a value of 100 will be output to the channels in the first cycle and a value of 150 in the second cycle.
2nd increments are applied to 1st increments after the 1st increment is applied to the channel value. Therefore, the increased 1st increment only takes effect in the following cycle.
Offsets are applied to galvo channels (#3-6) after the value of the channel for the current cycle has been calculated. The offsets are not added to the internal value of the channel, but only to the value that is sent out to the galvos. An offset is added to a galvo-channel starting from the cycle, in which it is switched on and. It will not be applied starting from the cycle, in which it is switched off.
The commands of a protocol have to be loaded to the DSP in the order, in which they are to be executed, i.e. a command scheduled for cycle 101 has to be loaded before a command for cycle 102. A Protocol is executed by issuing the DSP command 'X'. It ends, after the cycle, in which the last command has been executed. To force ending at a later cycle than that of the last meaningful command, use the **`scan_cmd_do_nothing`** (**`0`**) command.

IV.6 TMSI Timing

The TMSI needs a full cycle of 10 µs to transmit one frame of output data. To ensure an accurate timing, the data of the currently transmitted frame must not be changed. Otherwise, it will not be clear, whether the original or the changed data is sent. Therefore, the firmware uses two buffers, both of which are one frame long. During a cycle, the firmware writes the output data for the next frame into one of the buffers, while the data in the other buffer is transmitted to the devices. At the end of a cycle, the buffers are switched and the data in the first buffer is transmitted while the firmware writes to the other buffer.
Reading from the TMSI interface works similarly: During one cycle, data is read from the interface into a buffer, and after the buffer-switch at the end of the cycle, the firmware can access the data.
TODO: describe effect on timing of DOut, DIn and SM-Boards.

IV.7 Updating the Firmware

For updating the firmware of the Scan Control DSP the DSP has to be connected to a PC via RS232, and the PC needs to have a terminal-software installed, which supports the Xmodem protocol (e.g. Windows Hyperterminal). The COM port has to be configured with the parameters shown in Table 7. Note that the baud rate is different from the one of the communication parameters (Table 2).

Table 7: RS232 update parameters

115200 baud
8 data bits
no parity
1 stop bit
No flow control

To update the firmware, follow these steps:

1. Copy the firmware image (file extension .dli) to the harddisk of the PC.
2. Switch off the power of the Scan Control DSP.
3. Connect the DSP to the PC via RS232.
4. Open the Hyperterminal software with connection settings as shown in Table 7.

5. Use Transfer->Send File to send the firmware image to the DSP. Select Smodem or 1k Xmodem (faster) as protocol. Select the firmware image file on your hard disk and start sending.
6. Power on the DSP.
7. If the DSP sends the line “Transfer OK”, the update was successful. If not, power off the DSP, check the Hyperterminal configuration and try sending again.

V ELECTRICAL CONNECTIONS

V.1 Board Layouts

CAUTION: The numbering of the connectors and of the pin-outs is different from the numbering in the manual of the *SmartMove* DC900 boards.

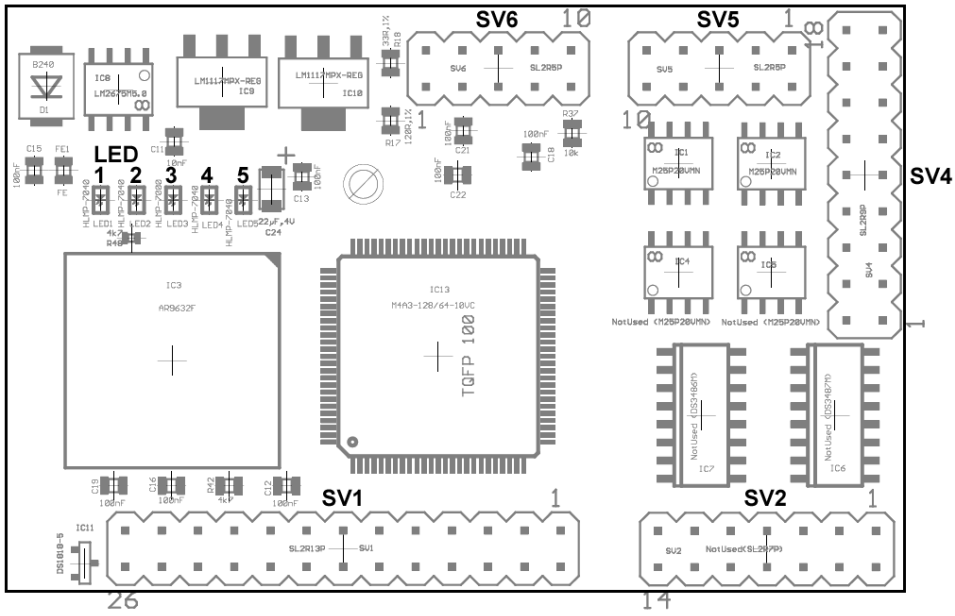

Figure 2: Layout of the scan control DSP-board seen from the front

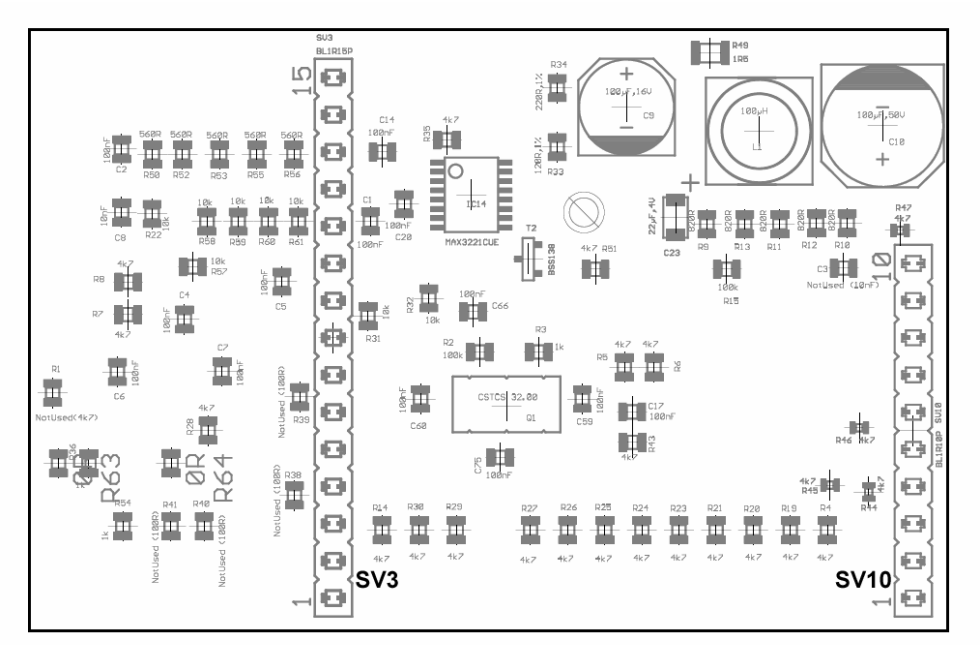

Figure 3: Layout of the scan control DSP-board seen from the back

V.2 Pin-Outs

Table 8: SV1

Pin #	Function on the TILL Scan Control board	Function on the Smart Move Galvo Driver board
1	TMSI Clock	TMSI Clock
3	TMSI Frame Sync	TMSI Frame Sync
5	TMSI Tx	TMSI Rx
7	TMSI Rx	TMSI Tx
9		
11		
13		
15		
17		
19		
21		
23		
25		

All even numbered pins are grounded

Table 9: SV3

Pin #	Function
1	
2	
3	
4	
5	
6	
7	
8	8 MHz CLK
9	4 MHz CLK
10	
11	
12	
13	+5 V out
14	
15	

Table 10: SV6

Pin #	Function
1	
2	
3	RS232 TxD
4	n.c.
5	RS232 RxD
6	n.c.
7	n.c.
8	n.c.
9	GND
10	n.c.

Table 11: SV10

Pin #	Function
1	
2	
3	
4	DO 0
5	DO 1
6	Din
7	DO 2
8	
9	GND
10	VCC

Table 12: Female Mini D Ribbon connector "AOP" on DSC front panel

Pin #	Function
1	Blanking (Bit 0 of TMSI-channel 7)
2	DOut (Bit 3 of TMSI-channel 7)
3	DOut (Bit 4 of TMSI-channel 7)
4	DOut (Bit 5 of TMSI-channel 7)
5	Digital GND
6	Analog GND
7	AOut 1 channel 1 (16-bit)
8	AOut 1 channel 2 (16-bit)
9	AOut 1 channel 3 (16-bit)
10	AOut 1 channel 4 (16-bit)
11	DOut (Bit 6 of TMSI-channel 7)
12	DOut (Bit 7 of TMSI-channel 7)
13	not used
14	not used
15	Analog GND
16	Analog GND
17	AOut 2 channel 1 (12-bit) (not used)
18	AOut 2 channel 2 (12-bit) (not used)
19	AOut 2 channel 3 (12-bit) (not used)
20	AOut 2 channel 4 (12-bit) (not used)

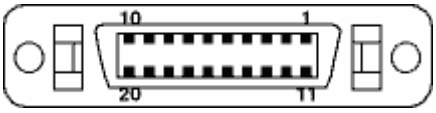

Figure 4: Scheme of a female Mini D Ribbon connector

V.3 Specifications

Supply Voltage (VCC) +15 V - +24 V

VI APPENDIX

VI.1 Appendix A: Commands for addressing the Scan Control DSP

Control commands are sent by the PC as bytes.
Command syntax: A command-line is terminated by a semicolon (';') or a Return-character. Command names are one char long. All spaces are ignored. The first non-space char is interpreted as command. Immediately after that char the parameter list starts. Parameters are separated by commas.
Commands return data or not as indicated in the command description.

```
// Definition of DSP control commands
DSP_CMD_DO_NOTHING     '#'
// does nothing, but keeps the interface busy and signals
// break of protocol execution, can also serve as comment
// marker
// no arguments
// no response

DSP_CMD_RETURN_INFO    'R'
// tells DSP to return null terminated information string
// no arguments
// returns a string "TILL scan control DSP v%s\r", where
// %s is replaced by the version number

DSP_CMD_CLEAR_PROT     'C'
// clears the scan control protocol
// no arguments
//
// returns one number as status/error code

DSP_CMD_ADD_PROT_CMD   'A'
// adds a command to the scan command list
// arguments:
// char <scanner control command (see above)>
// cycles_t <cycle (current type long>
// int <channel>
// int <value>
// all arguments must be present. Even if they don't apply
// to the scan command
//
// returns one number as status/error code

DSP_CMD_INIT_PROT      'I'
// initialize protocol: sort commands within blocks ...(?)
// currently not functional.
// no arguments
// no response
```

```
DSP_CMD_EXECUTE_PROT  'X'
// executes the current protocol
// no arguments
//
// returns one number as status/error code

DSP_CMD_LIST_PROT     'L'
// list protocol
// no arguments
// Returns a listing of the protocol in memory or
// the message "No Protocol in Memory.\n\r".

DSP_CMD_SHOW_BUFFER   'B'
// for debugging purposes.
// shows the contents of the buffer for scan channel "channel"
// argument:
// int <channel>
// Returns the content of the buffer, no error code

DSP_CMD_SHOW_VALUE    '?'
// shows the current value of the scan channel "channel"
// argument:
// int <channel>
// Returns one number: the value, no error code

DSP_CMD_SET_VALUE     'V'
// directly sets the value of a scan channel. Like the 'V'
// command in scan protocols, but the current protocol remains
// unchanged and the value is set immediately after the line
// is parsed.
// arguments:
// int <channel>
// long <value>
//
// returns one number as status/error code

DSP_CMD_SET_OFFSET     'O'
// sets the offset value of a galvo channel (channel 3-6) in
// Counts (not MicroCounts).
// The offset is added to the value of the channel before
// the value is transmitted to the galvo-driver.
// arguments:
// int <channel>
// int <value>, range: -32768 .. 32767
//
// returns one number as status/error code
```

VI.2 Appendix B: Scan control commands

The scan control commands have three parameters:

```
cycle (long; 48 bit)
// number of 10µs cycle, in which command is executed
channel (integer; 24 bit)
// output channel, whose increment is changed
value (long; 48 bit)
// value to set
```

The scan control commands are:

```
scan_cmd_do_nothing = (int) '0',
// dummy command. E.g. to extend execution of commands
//   beyond last "real" command

scan_cmd_set_value = (int) 'V',
// command to directly set the value of an output channel
// cycle: 10 µs cycle, in which value is set
// channel: channel, which is set
// value: value, to which the channel is set

scan_cmd_set_value_relative = (int) 'R',
// command to set value of an output channel relative
//   to the current value

scan_cmd_set_increment_1 = (int) 'I',
// command to set value by which a channel is incremented
//   every cycle (^= 1st derivative)
// cycle: 10 µs cycle, in which value is set
// channel: channel, whose 1st increment is set
// value: increment value

scan_cmd_set_increment_2 = (int) 'J',
// command to set value by which the increment of a channel
//   is incremented every cycle (^= 2nd derivative).
// cycle: 10 µs cycle, in which value is set
// channel: channel, whose 2nd increment is set
// value: increment value

scan_cmd_switch_offset = (int) 'O',
// command to switch the offset of a scan channel (3-6) on
// and off. The offsets are off when the protocol starts.
// cycle: 10 µs cycle, in which the offset is switched
// channel: channel, whose offset is switched
// value: 0: off    1: on
```

```
scan_cmd_loop_start = (int) 'S',
// start of a loop; Up to 100 loops can be nested.
// channel: not used
// value: number of iterations

scan_cmd_loop_end = (int) 'E',
// end of a loop
// channel: not used
// value: not used

scan_cmd_wait_trig_rising = (int) 'U',
// wait for rising trigger
// cycle: execution of commands behind this one is halted
//   at this cycle until a rising trigger is detected (don't
//   use cycle=0
// channel: not used
// value: not used
// While the protocol is waiting for a trigger, increments are
// not applied.
// Should not be used in cycle 0!

scan_cmd_wait_trig_falling = (int) 'D',
// wait for falling trigger
// parameters and usage same as for scan_cmd_wait_trig_rising
```

VI.3 Appendix C: Status/Error codes

Codes are returned by the DSP via RS232 as decimal numbers in ASCII representation.

```
SCAN_CMD_NO_ERROR            = 0,
SCAN_CMD_RUN_TIMEOUT         = 1,
  // Calculation took too long for 10 us frame
SCAN_CMD_RUN_ABORTED         = 2,
  // Run aborted by sending character on RS232
SCAN_CMD_LIST_EMPTY          = 3,
  // Trying to run an empty command list.
SCAN_CMD_LIST_NOT_CLOSED     = 4,
  // Trying to run a command list with unclosed loops.

SCAN_CMD_LIST_OVERFLOW       = 10,
  // Tried to add command to a command-list that has
  // already maximum length.
SCAN_CMD_LIST_DISORDER       = 11,
  // Cycle time of newly added command is impossible.
SCAN_CMD_INVALID_CHANNEL     = 12,
  // Command is specified with non-existing channel number.
SCAN_CMD_LOOP_OVERFLOW       = 13,
  // Too many nested loops.
SCAN_CMD_INVALID_ITERATIONS  = 14,
  // Invalid number of loop iterations (<0)
SCAN_CMD_LOOP_IS_CLOSED      = 15,
  // Trying to close a loop, which is already closed.
SCAN_CMD_UNKONWN_COMMAND     = 16,
  // Unknown command
SCAN_CMD_NO_BUFFER           = 17,
  // No debug buffer available for the selected channel
SCAN_CMD_INVALID_SYNTAX      = 18
  // Invalid syntax when adding a scan command or executing
  // a direct command:
  //   not enough parameters or too many parameters.
```

VI.4 Appendix D: Example of a Scan Control Protocol

The following is a small example protocol that runs a sawtooth curve of +/- 5.5° mechanical amplitude with 100Hz.

```
# Test for running a scanfield of +/- 11° optic. ^= +/-5.5° mech.
# The full range of 16 bit (-32768 to +32767)
# covers +/- 30° opt. ^= +/-15° mech.
# => for +/- 5.5° mech. one has to go from -12014.75 to 12014.75
# We use 36 bit in the control DSP and send the highest 16 bit
# (take x1048576).
# => The range is -12598378496 to 12598378496.
#
# 100 Hz frequency means that this range has to be covered
# in 1000 cycles (of 10us). => Increment = 25196757

# Clear old protocol
C

# Add (A) new protocol commands
# Commands have the syntax
# Command, Cycle(10us), Channel, Value
# Commands used here:
# V: Set absolute position value of channel.
# I: Set increment of position of channel.
#    Increment is applied to channel every cycle.
# S: Start a Loop. The channel parameter has no meaning.
#    Value is the number of loop iterations.
# E: End a Loop. Channel and value don't have a meaning.
#    Ending a loop does not take a cycle. The
#    last cycle of one loop iteration and the first cycle
#    of the next iteration are the same.
#

# Set increment so that the ramp is covered in 1000 cycles
A I,0,3,25196757

# Start a loop with 1000 iterations.
A S,0,0,1000

# Reset the position of galvo channel 3 to start of ramp in
# every loop iteration.
A V,0,3,-12598378496

# End loop.
A E,1000,0,0

# Set increment to 0 at end of protocol
# (0 + 1000 iterations * 1000 cycles)
A I,1000000,3,0

# Execute the script
X
```

VI.5 Appendix E: Making an RS232-connector for the Scan control DSP by crimping

... TODO