DSP scan board interfacing guide

Last updated: September 7th, 2004.

RHK Technology, Inc.

Table of Contents

1 Introduction

This document describes the command set on the DSP scan board that is built into the STM100 rev.8 and into the VSCAN100 units.

1.1 Hardware

The scan board has an ethernet connection to the control PC. Besides that there are two handshake lines (FLAG_A is an output from the scan board, FLAG_B is an input to the scan board). The ethernet link uses UDP on top of IP. There is a restriction that each command must not exceed one ethernet packet size.

1.2 Interfacing the DSP scan board

The scan board has a kernel that gets loaded into the DSP processor out of an EEPROM chip when the board boots. This kernel controls the ethernet interface and contains some basic functions for loading and executing user programs. At boot time the DSP scan board sends out bootp requests until it receives a valid bootp answer. Under normal conditions this is answered by SPM32. The bootp request can also be answered by a bootp server on a UNIX machine if that got configured for the scan board's hardware ethernet address. The ethernet adress is hand written onto each scan board with a black pen. The address starts with 00-d0-0b-XX-XX-XX with XX-XX-XX being a unique number for each board. The DSP listens to three UDP ports. Packets sent to UDP_REPLY_PORT (0x0a) will be echoed back to the sender. Only full 32 bit cells will be echoed. Port UDP_CTRL_PORT (0x0c) is used for kernel commands. Please look at the file kernel_comm.c in the C_utilities for examples of how to use these commands. Commands for thr user program have to be sent to UDP_CMD_PORT (0x0b). These port numbers are somewhat arbitrary and don't go confirm with the standard port number definitions. For this reason they might get changed in the future.

1.3 Interfacing using C_utilities

Each command has a unique number. For use in C code these numbers have symbolic names that are defined in the file commands.h. For a reference this file is included at the end of this document. A C program has to send the command code first followed by the parameters. The C_utilities contain a program scantest. This porgram is used for scan board code debugging. The command tokens have to be typed in followed by the parameters. For instance for the Ignore speed switch the token "is 1" has to be typed. Scantest parses the command tokens and sends the command number+parameters down to the scan board. The C_utilities do not contain a bootp server. They are written to be compiled on a UNIX machine where a bootp server can be set up quite easily.

1.4 Interfacing using LabVIEW utilities

Each scan board command has a LabVIEW VI (virtual instrument) that represents it's front end. The vi's are built so that they open and close the UDP connection if not some other vi has opened them already. There is a bootp_server.vi that answers the board's bootp requests and a sharc-coff-load.vi that downloads any user program in coff file format.

1.5 Using Lithography scripts

In a lithography file the command token has to be used with it's parameters in front of it. The reason for this is that in future the whole command interface will be done by a FORTH system that uses reverse polish notation. In order for lithography files that you make to be forward compatible then they have to be reverse polish notation. Please have a look at the file RHKlogo.lth in the C_utilities. Both the C_utilities and the LabVIEW_utilities have a front end for WRITE_LITHO_BUF which parses the litho file and sends them down to the scan board where they get stored in a memory area on the scan board. There is only one pointer that is used for the actual write position during WRITE_LITHO_BUF and for the next executed litho instruction when litho runs. Therefore you have to issue a RESET_LITHO_BUF command between loading a litho file and executing it. The litho file parser needs an indication that the litho file ends. Therefore every litho file has to end with an instruction "end". During litho execution two things can happen at the end of a litho script: 1.You want to stop. In this case you have to place a "rlb" instruction in front of the "exit" at the end of the file. If you forget this execution might run into previously loaded longer scripts and do unexpected things or end in an error message. 2.You want to loop over the whole file again. In this case you want to place a jlb instruction at the end of the file. This is shown in RHKlogo.lth. The parameter for jlb gives the offset of the first parameter of the instruction that you want to jump to from the begin of the buffer. The buffer begin is position zero. Please note that each parameter takes two places in the buffer. The parser automatically inserts a PUSH_INT or PUSH_FLOAT in front of it.

The GET_STATUS command returns the lithography pointer position, the last executed command code and the last lithography error message. Error messages get erased after they get read.

2 Global settings and status commands

This chapter describes the global settings and status command. These commands are used for interfacing and setup. For historical reasons there are some commands in here that probably better fit into the scan layer group.

2.1 is (Ignore speed switch)

Command code	IGNORE_SWITCH
Command name in Litho files	is
Number of parameters to send	1
Parameters	u32 FLAG
Examples:	ignore speed front panel switch: is 1
	consider speed front panel switch: is 0

The scanning speed can be set directly by the control PC or from the user via the line time knobs on the SPM100 front panel. Whichever one got changed last will overwrite the speed. This command allows the SPM100 unit to use the front panel switches again after a computer-set speed has been in use.

2.2 wzdac (Write to Z DAC)

Command code	WRITE_Z_DAC
Command name in Litho files	wzdac
Number of parameters to send	1
Parameters	float output_value [physunits]

This command sets the output voltage on the Z DAC on the scan board. The parameter output_voltage is clipped at +-10 V.

2.3 szs (Set Z Scale)

Command code	SET_Z_SCALE
Command name in Litho files	szs
Number of parameters to send	1
Parameters	float Z_Sensitivity

This command defines the meaning of the physical units for the parameter of WRITE_Z_DAC and for the Z sweep functions. Z_Sensitivity is a sensitivity with the unit [PhysUnit/DAC_output_Volt]

2.4 zss (Set Z Speed)

Command code	Z_SET_SPEED
Command name in Litho files	zss
Number of parameters to send	1
Parameters	float Z_Sweep_Speed

This command sets the sweep speed for Z DAC sweeps. The unit of Z_Speed is [Physunits per second].

2.5 abortz (Abort Z Sweep)

Command code	ABORT_Z_SWEEP
Command name in Litho files	abortz
Number of parameters to send	0

This command aborts a running Z sweep.

2.6 sza (Sweep Z Absolute)

Command code	SWEEP_Z_ABSOLUTE
Command name in Litho files	sza
Number of parameters to send	1
Parameters	float Z_Destination

This command sweeps the ZDAC output to the Z_Destination value. The sweep parameters are defined by the SETUP_Z_SWEEP command. The Z_Destination value has to be given in Physical Units as they were defined by the SET_Z_SCALE command.

2.7 setupz (Setup Z Sweep)

Command code	SETUP_Z_SWEEP
Command name in Litho files	setupz
Number of parameters to send	10
Parameters	u32 Z_Sweep_Flags u32 #Sample_groups u32 #samples_per_group u32 ThresholdMode u32 AD_Channels float Z_Sweep_Speed float PreSampleDelay float TimePerSample float UpperThresholdValue float LowerThresholdValue

This command sets the control parameters for all subsequent Z sweeps. It also initializes the A/D subsystem on the scan board to take data if any channels are set up in the AD_Channels parameter. The parameters mean:

`Z_Sweep_Flags:`

This is a set of flags that affect the operation of the Z sweep and the way the Z sweep setup works. The meaning of the bits is:

Bit 0:	DSP opens Feedback	0–>No, 1–>Yes
Bit 1:	Reconfig DSP A/D	0–>No, 1–>Yes

The DSP opens Feedback is not implemented in versions <= 0x2c and requires a Revision 3 DSP scan board. The Reconfig DSP A/D tells whether this command should reset and re-configure the DSP A/D buffers. If set the data buffers will be emptied. If cleared previous data is held in the buffers. Note that AD_Channels can still be changed from previous data acquisitions without this bit set.

`Sample_groups`
: This tells at how many positions along the line samples will be taken. Please note that the begin and the end positions of the sweep are sample positions. If you don't want A/D conversions taken during Z sweeps set this parameter to zero. Zero is the default value for this.

`u32 #samples_per_group`
: This tells how many triggers get issued to the SPM1000 data acquisition board at each sample position. This trigger is also available on pin 3 of the DB15 connector of the STM100/VSCAN100. In order to use these triggers to take samples on the two ADCs on the board you need to connect this pin 3 to pin 6 of the same connector.

`u32 ThresholdMode`
: A value different from 0 in this parameter will cause the sweep to stop when the A/D on the selected scan board A/D channels exceeds one of the values specified in the ThresholdValue parameter. When bit 0 of this parameter is set channel ADC1 will be used to compare to the threshold values. When bit 1 is set channel ADC2 will be used. The channels will only be used when data is taken on them during Z sweeps. When bits 0 and 1 are both set and data is measured on both channels only channel ADC 1 is used. Besides that bit 4 of this value tells whether the ADC values decrease (bit 4 is TRUE) or increase (bit 4 is FALSE) with increasing DAC output voltage.

`u32 AD_Channels`
: This parameter tells which A/D channels on the DSP scan should get sampled during all subsequent Z sweeps. A 1 means only A/D1, a 2 means only channel A/D2 and a three means both channels.

`float Z_Sweep_Speed`
: This sets the Z sweep speed in Physunits/s. The PhysUnit is defined by the SET_Z_SCALE command. The speed gets internally adjusted to match other internal parameters like resolution and maximum DAC update speed. The DSP scan board also takes care that the time betwee sample positions is bigger than TimePerSample in order to prevent A/D conversion errors.

`float PreSampleDelay`
: This number defines a delay time that the DSP scanner waits at each sample position before the trigger sequence is started. The time is given in seconds.

`float TimePerSample`
: This parameter gives the time separation between two samples in the trigger group at each sample position. The user has to make sure that this time is not smaller than the conversion time of the A/D converters used.

`float UpperThresholdValue`
: This gives the upper Threshold Value for the ThresholdMode. When the ADC specified in AD_Channels reads a value that exceeds this the Z sweep output will not be increased beyond that DAC value. Please note that for this to work bit 4 in the AD_Channels parameter of this command has to be set correctly

`float LowerThresholdValue`
: This gives the Threshold Value for the ThresholdMode for up sweeps.

2.8 dioset (Set DIO bits)

Command code	BSET_DIO
Command name in Litho files	dioset
Number of parameters to send	1
Parameters	u32 bit_pattern_to_be_set

All bits that are ONE in bit_pattern_to_be_set will be set in the DIO port of the scan board. The bits are:

DIO_PAGE	PAGE signal for STM100 electronics
DIO_LOOP_INVERT	LOOP INVERT signal for the STM100 electronics
DIO_Z_RETRACT	Z_RETRACT signal for the STM100 electronics
DIO_HOLD_INT	HOLD_INT signal for the STM100 electronics
SCANNING_LED	"Scanning" - LED on the front of the STM100
ADC_CONV_START	This starts the conversion of the 2 scan board ADCs
OVERRIDE_LED	"Override" LED on front of the unit

The coding of these values is defined in commands.h

2.9 dioclr (Clear DIO bit)

Command code	BCLR_DIO
Command name in Litho files	dioclr
Number of parameters to send	1
Parameters	u32 bit_pattern_to_be_cleared

All bits that are ONE in bit_pattern_to_be_cleared will be cleared in the DIO port. See description of [dioset], page 6 for the bit functions.

2.10 diotake (Switch DIO bit to scan board control)

Command code	TAKE_DIO
Command name in Litho files	diotake
Number of parameters to send	1
Parameters	u32 bit_pattern_to_be_taken

All bits that are ONE in bit_pattern_to_be_taken will be made output bits so that the scan board can control them. Either the scan board or SPM32 can control the bits. Please remember to set the control back to SPM32 for all bits. Otherwise SPM32 will not work correclty. This function requires a revision number of 3 and a kernel revision number of 1.4 to work. Please contact support@rhk-tech.com for details of a scan board upgrade.

2.11 trigadc (Trigger DSP ADC)

Command code	TRIGGER_ADC
Command name in Litho files	rigadc
Number of parameters to send	0

This triggers an A/D conversion on the two ADCs on the board. Use READ_ADC_BUF to get the values. The data acquisition parameters are set using SET_DAQ_MODE.

2.12 gadc (Read DSP ADC buffer)

Command code	READ_ADC_BUF
Command name in Litho files	gadc
Number of parameters to send	1
Parameters	u32 how_many

This command reads "how_many" conversions out of the on-board A/D conversion buffers and sends them up to the PC. The first u32 word in the reply is the number of A/D samples that get sent up to the control PC. The reply data block contains more than one UDP packet. The first packet returns the total number of samples that will get send back. The reading program has to read udp buffers until that many samples have been read. The A/D conversions can be triggered via the external trigger input as well as via TRIGGER_ADC command. Note that an ADC conversion sample consists of an i16 for ADC1 and another i16 for ADC2. The values are in ADC pixels (-32768...32767). The ADC range is -10V .. 10V. The upper 16 bits of the data words are NOT sign extended.

2.13 wadc (Write DSP ADC buffer)

Command code	WRITE_ADC_BUF
Command name in Litho files	wadc
Number of parameters to send	variable
Parameters	u32 how_many, u32_array data

This command writess "how_many" data points into the on-board A/D conversion buffers as if they were written by the Data Acquisition. The first u32 word in the reply is the number of A/D samples that get sent down to the board. This command is intended for testing and for writing the output data for the replay mode ([srm], page 26).

2.14 gg (Get Gains from chassis EEPROM)

Command code	GET_GAINS
Command name in Litho files	gg
Number of parameters to send	0

This command returns a table that contains the high voltage gain parameters that are burned into the chassis EEPROM of the SPM100 units. Please note that a reading of 6553.5 indicates that the EEPROM does not contain the correct header revision. In this case the information might be in the modifications field of the EEPROM header. The data structure returned by the GET_GAINS command looks like this:

```
[0] u32 the GET_GAINS command code
[1] float X Offset Gain
[2] float X Scan Gain
[3] float Y Offset Gain
[4] float Y Scan Gain
[5] float Z Offset Gain
[6] float Z Scan Gain
```

2.15 gs (Get scan board status)

Command code	GET_STATUS
Command name in Litho files	gs
Number of parameters to send	0
Parameters	none

GET_STATUS returns a block of status information to the control PC. When directly sent the return packet will get sent to the IP address of the machine asking for the status (synchronous send). When called from with in a lithography file the packet will get sent to the "boss" IP address and port number. This can be used to acknowledge that a linedraw has reached it's destination. In detail the structure looks like this:

```
[0] u32 the GET_STATUS command code
[1] u32 machine state
```

Name	Value	Meaning
IDLE	0	nothing happens
LITHO	1	manual drawing mode
SCAN	2	Topgraphy only scanning mode
PREPSCAN	3	Prepare a scan (moving to the scan start corner)
SNGLESCN	4	stop scanning end of this frame (last scan frame)
ARC	5	busy drawing arcs
FEATURE_TRACK	6	busy feature tracking
LITHO_AT_SPECTRO	31	Running lithography on a spectro stop

Besides that when the scanner halts at a spectro stop the top 16 bits will show 0xFFFF except when Litho stuff is done there.

```
[2] float scan speed in physunits/second
[3] float line time in seconds
[4] u32 the next line draw motion mode
```

Name	Value	Meaning
C_SPEED	0	constant speed
SINE_SCAN	1	sine wave speed
NL_LOOKUP	2	nonlinearity table lookup

`[5] float scan rotation angle in deg`
`[6] float X scanner position in physunits`
`[7] float Y scanner position in physunits`

In positioning feedback mode the parameters [6] and [7] give the position of the scanner as measured by the detectors. The physical units will be calculated according to the detector characteristics.

`[8] float X scanner position in Volts`
`[9] float Y scanner position in Volts`

The fields [8] and [9] always give the last DAC output voltage, no matter whether positioning feedback is on.

`[10] float X scanner position offset in physunits`
`[11] float Y scanner position offset in physunits`

These are offsets in the scan signal. This has nothing to do with the SPM1000 front panel offset knobs. They are separate channels.

`[12] u32 actual image pixel number on the line`
`[13] u32 actual scan line number`

These are calculated back from the tip position. When the tip is outside the region defined by SET_BASE these numbers might be negative or bigger than the number of pixels/lines defined by SET_NPIX_NLIN.

`[14] u32 state of the PEN bit right now`

A 0 in this field means pen up, a 0xFFFFFFFF means pen down

`[15] float X piezo sensitivity as set by SET_MATRIX`
`[16] float Y piezo sensitivity as set by SET_MATRIX`
`[17] float X HV gain as set by SET_MATRIX`
`[18] float Y HV gain as set by SET_MATRIX`
`[19] u32 line draw errors`

This shows an error value of 1 when a line draw went off the DAC grid. GET_STATUS resets this to zero after reading it. An error code of 2 means that one of the SET_MATRIX parameters was zero and thus the SET_MATRIX has been ignored.

`[20] float spectroscopy wait time`

A 0 means the board is waiting for s SCAN_UNTIL to continue. A positive values make the board continue after that many seconds A negative values cause FLAG_A -> FLAG_B handshaking

`[21] u32` internal flags

Name	Bit	Meaning
SC_ON	0	scan enable?1=Yes,0=No
QPD	1	Lithography pen down? 1=Yes, 0=No
POSPREP	2	Is the line next to the one currently drawing prepared? 1=Yes 0=No
SPEC_HERE	3	Is this scan line on the spectro grid?

		1=Yes 0=No
POS_FB_ON	4	Is positioning feedback switched on?
		1=Yes 0=No
YSCN	6	enable Y scan? 1=Yes0=No
NOADC	7	Lock ADC triggers? 1=Yes0=No
SINE_LINE	8	Cosine speed line mode for the next line?
NL_LOOKUP_LINE	9	use nonlinearity table for this line?
IGNORE_USR_SPD	10	ignore speed switch settings? 1=Yes 0 = No
QSL	11	Status of the SCAN_LEFT/*RIGHT line during line draw
SPD_TOO_HI	12	speed limited by the scan board due to ADC conditions
LINE_WAITING	13	waiting at a line
PC_SET_SPD	14	Who set the actual speed setting: 0= front panel knob 1= PC
INFINIT_TRIG	15	infinite number of trigger groups for TRIGGER_SERIES
MOVING	16	set whenever the scanning LED is on
PENDING_TRIG	17	Flag/Wait ISR needs to generate the first trigger along the line
FIBER_FB_RUNS	18	Fiber fb is running

`[22] float scan line length in physunits`
`[23] scan Y length in physunits`
`[24] u32 Number of pixels`
`[25] u32 Number of lines`
`[26] u32 number of repetitions for each scan line draw`
`[27] u32 On which scan line repetitions do AD triggers see SET_WHERE_AD`
`[28] u32 SCANNER_FLAGS see SET_SCANNER_FLAGS`
`[29] u32 number of triggers at each image pixel position`
`[30] float trigger period at each image pixel position in seconds`
`[31] float wait time at the end of a forward scan line [s]`
`[32] float wait time at the end of reverse scan line [s]`
`[33] u32 actual line repetition number`
`[34] scanner errors`
`[35] u32 Lithography instruction pointer value`
`[36] u32 Lithography error number`
`[37] u32 The last decoded Litho command`

In case of a bad command the code will be shown here and [28] shows where in the buffer that happened

`[38] float user gain setting from front panel switch`
`[39] float user line time setting from front panel switch`

These contain the front panel switch settings. When the user changes the switches these variables will be updated. The line time shown here is a user

wish. When the scan program got started it will try it's best fulfilling this wish. The resulting line time and scan motion speed can be taken from parameter [2] and [3] of this message. Sending IGNORE_SWITCH 1, SET_SPEED, SET_LINE_TIME or PREPARE_SCAN command will uncouple the actual line speed from the front panel switch. An IGNORE_SPEED 0 will make the board use the switches again... While the speed settings from the switches are ignored the OVERRIDE LED will come on.

[40] u32 Chassis serial number
[41] u32 scan board hardware revision number
[42] u32 Altera firmware revision number
[43] u32 Kernel revision number
[44] u32 Scan software revision number
[45] u32 ADC buffer length
[46] u32 number of samples in ADC buffer
[47] i32 last AD1 value
[48] i32 last AD2 value
[49] float Z DAC sensitivity [Physunits/V]
[50] float Interferometer calibration factor [nm/V]
[51] float Z DAC output value [Physunits]
[52] float Feature track X center position [Physunits]
[53] float Feature track Y center position [Physunits]

These feature track positions are given in coordinates on the rotated coordinate system. TRACK_FEATURE needs to be set up to use POSITION_ABSOLUTE commands for tracking for these numbers to be meaningful. If TRACK_FEATURE moves the offsets the reference coordinate system is moved and these values won't change at all.

[54] u32 number of samples in feature track buffer

2.16 gdm (Get DSP ADC mode)

Command code	GET_DAQ_MODE
Command name in Litho files	gdm
Number of parameters to send	0

This makes the scan board send back its data acquisition mode word. This word contains parameters for data acquisition for the on-board ADCs. The format of the returned word is the same as described in [sdm], page 11.

2.17 sdm (Set DSP ADC mode)

Command code	SET_DAQ_MODE
Command name in Litho files	sdm
Number of parameters to send	1
Parameters	u32 Buffer_Length
	u32 Channels
	u32 ADC_mode

This command sets the Data acquisition buffer length, channels and mode. Use SET_POS_FB to set the positioning feedback... The bits of the ADC_mode word have this meaning:

`bit 0: AD_EXTBUF`

0	Buffer in internal memory
1	use external buffer

The board will try to use the MEM16 bank to put the buffer in. If that is not available it will use the 2nd 64k block in the MEM48 bank.

`bit 1: AD_AUTOTRIG`

0	external triggering
1	trigger generated in ADC_isr

This says whether the interrupt routine that reads the ADCs will generate the next ADC trigger. Timing is somewhat badly determined this way.

`bit 2: AD_AVERAGE`

0	one value for each trigger
1	average values until read

In non-averaging mode each trigger fills two addresses in the ADC buffer. In averaging mode only 4 addresses are used. The lower 2 contain the number of readings for ADC1 and ADC2. The higher two contain the sum of all readings for ADC1 and ADC2. Reading the buffers clears the four cells.

`bit 3: POS_INTEGR`

Use positioning nonlin I feedback algorithm:

0	no
1	yes

For positioning feedback use the POS_INTEGR algorithm. You should use the SET_POS_FB command to set the positioning feedback. That command will take care of all ADC settings automatically.

`bit 16: (0x010) AD_BUF_FULL`

0	still space in buffer
1	buffer full

This bit is set by the DSP when the ADC buffer is full.

`bit 17: (0x011) AD_CONVERTING`

0	not converting
1	ADCs busy...

This bit is set by the DSP when the ADC buffer is full.

2.18 sdc (Set DSP ADC channels)

Command code	SET_DAQ_CHANS
Command name in Litho files	sdc
Number of parameters to send	1
Parameters	u32 channels

This command allows to set the data acquisition channels without having to re-configure the data acquisition. The channels parameter has the same encoding as in [sdm], page 11.

2.19 spf (Set positioning feedback)

Command code	SET_POS_FB
Command name in Litho files	spf
Number of parameters to send	1
Parameters	u32 mode

This command sets the positioning feedback. The modes are:

`NO_POS_FB: 0x0`
: This is the default. The scan board directly outputs to the DACs.

`FB_INTEGRATOR: 0x1`
: This is an I-feedback algorithm that runs in the background of the scanning. It controls the positioning at any time, not just during scanning. The FB time constant can be set using SET_POS_FB_TC. Using this algorithm you should not try to scan faster than 100 ms/line.

2.20 stc (Set position feedback time constant)

Command code	SET_POS_FB_TC
Command name in Litho files	stc
Number of parameters to send	1
Parameters	float gain

This sets the feedback time constant for the positioning feedback in FB_INTEGRATOR mode. The default value that is used on the scan board is 0.02857 s or 1/(35Hz) . This value works for a NANONICS scanner. For everything else it has to be experimentally determined. Please note: FB oscillations might be bad for your scanner, be careful with this.

2.21 sdx (Set detector characteristics)

Command code	SET_DETECTOR_CHAR
Command name in Litho files	sdx
Number of parameters to send	8
Parameters	float X_A, X_B, X_C, X_D, X_E float Y_A, Y_B, Y_C, Y_D, Y_E

Most positioning detectors are not linear as well. This command offers the opportunity to linearize and decouple the detector using a 3rd order polynomial. The coefficients starting with X_ are for the X detector, the coefficients starting with Y_ are for the Y detector. The equations (from the viewpoint of the scan board) are:

$XPhysUnits = X_A * (XADC - X_D)^3 + X_B * (XADC - X_D)^2 + X_C * (XADC - X_D) + X_E * (YADC - Y_D)$

and

$YPhysUnits = Y_A * (YADC - Y_D)^3 + Y_B * (YADC - Y_D)^2 + Y_C * (YADC - Y_D) + Y_E * (XADC - X_D)$

Physunits are the physical units that you run the scan board in (set by SET_MATRIX). ADC has Volt units (the range -10V .. 10V) and means the corresponding detector voltage of that position.

2.22 rdx (Read detector characteristics)

Command code	READ_DETECTOR_CHAR
Command name in Litho files	rdx
Number of parameters to send	0
Returns	the X_A, X_B, X_C, X_D, X_E, Y_A, Y_B, Y_C, Y_D, Y_E as they got sent down by a previous SET_DETECTOR_CHAR:
buffer[0]:	READ_DETECTOR_CHAR command code
buffer[1]:	X_A
buffer[2]:	X_B
buffer[3]:	X_C
buffer[4]:	X_D
buffer[5]:	X_E
buffer[6]:	Y_A
buffer[7]:	Y_B
buffer[8]:	Y_C
buffer[9]:	Y_D
buffer[10]:	Y_E

This command is used to verify the XY positioning feedback characteristics that gets sent to the scan board using SET_DETECTOR_CHAR.

2.23 setfib (Setup Fiber Feedback)

Command code	SETUP_FIBER_FB
Command name in Litho files	setfib
Number of parameters to send	3
Parameters	float Setpoint, float TimeConstant, float Interferom_conv

This command sets the Setpoint and Time constant for an interferometer reference channel. This can drive the reference mirror or the fiber piezo of an interferometer that's used as AFM detector. The signal from the interferometer preamp needs to be on ADC1 of the scan board. The fiber piezo drive signal will come out of the ZDAC. The Interferom_conv is just a configuration parameter that gets stored on the scan board. It is not used in any internal calculations but output in the GET_STATUS reply so that SPM32 can have access to this number.

2.24 strtfib (Start fiber feedback)

Command code	START_FIBER_FB
Command name in Litho files	strtfib
Number of parameters to send	1
Parameters	u32 FLAG

This command starts or stops the fiber feedback algorithm with the parameters set up in SETUP_FIBER_FB. A value of zero in the parameter FLAG will stop the feedback. Everything else starts the feedback.

2.25 setwd (Setup Watchdog)

Command code	SETUP_WATCHDOG
Command name in Litho files	setwd
Number of parameters to send	3
Parameters	float lowerlimit, upperlimit, BOOLEAN use_pen_line

This command sets the lower and upper limits for a signal watchdog that checks the DSP ADC2 while the fiber feedback runs. The unit for these parameters is Volts. When lowerlimit is smaller than upperlimit the watchdog checks whether the signal on DSP ADC2 is bigger than lowerlimit and smaller than upperlimit. When lowerlimit is bigger than upper limit the algorithm checks whether the signal is bigger than lowerlimit or smaller thatn upperlimit. In other words: one way it tests whether the signal is within the range and the other way it tests wjhether the signal is outside the range. When the test fails and use_pen_line is not zero the pen_bit on the DB15 connector goes HI. The algorithm resets the pen_bit when the signal gets back into range. Also the scanner errors field (field 34) of the GET_STATUS reply will show a 1. Reading the status resets this error field.

2.26 showwd (Show watchdog)

Command code	SHOW_WATCHDOG
Command name in Litho files	showwd
Number of parameters to send	0

This command displays the upper and lower limit of the signal watchdog algorithm. This will return : float lower_limit float upper_limit BOOLEAN use_pen_line

3 Line draw commands

This chapter describes the lowest software layer of the DSP scan board code. Every motion that the scanner makes is done in straight line segments. The lines are drawn by a timer interrupt routine. These commands set the parameters for the line draw and start or stop it. All higher software layers use these commands. For instance: A scan is composed out of line segments.

3.1 pa (Position Absolute)

Command code	POSITION_ABSOLUTE
Command name in Litho files	pa
Number of parameters to send	2
Parameters	float X
	float Y
Examples	20.0 20.0 pa

For this to work in physical units you need to set the sensitivities using SET_MATRIX.

3.2 pr (Position Relative)

Command code	POSITION_RELATIVE
Command name in Litho files	pr
Number of parameters to send	2
Parameters	float X
	float Y
Examples	20.0 20.0 pr

For this to work in physical units you need to set the sensitivities using SET_MATRIX.

3.3 pp (Position Pure)

Command code	POSITION_PURE
Command name in Litho files	pp
Number of parameters to send	2
Parameters	float X
	float Y
Examples	20.0 20.0 pp

This moves the scanner to the position (X,Y) on the unrotated and un-offset coordinate system. It is mainly used by SPM32 for range checking. X and Y are given in physical units.

3.4 pae (Position Absolute Extended)

Command code	POSITION_ABS_EXTD
Command name in Litho files	pae

Number of parameters to send	11
Parameters	float X float Y float speed float wait_time: Wait time at line start, negative will cause FLAG_A/B handshake. u32 flags: Bit1 PU/PD 0 –> PU 1 –> PD, Bit7 ADC0 –> YES1 –> NO Bit11 SCAN_LEFT DIO 0 –> LOW 1 –> HI u32 TRIGNUM: #AD triggers at each position float TR_PERIOD: trigger period in seconds (a pretty small number...) u32 MotionMode: C_SPEED, SINE_SCAN or NL_LOOKUP u32 first_Ints/AD: where first AD u32 other_Ints/AD: AD trig separation i32 firstspec : first spectro stop after how many A/D position updates firstspec == 0 –> never stop, don't use remainder of previous line firstspec == -1 –> use remainder of counts from previous line
Example	0 5 3 0 0.000001 3 129 0.01 120000.0 20.0 20.0 pae

This function is called by the scan generator. It is the general interface of the line draw section of the program. Using this an external scan generator could take over the control of the board. Please note that the X and Y have to be given in internal scan grid units here.

3.5 swt (Set Wait Time)

Command code	SET_WAIT_TIME
Command name in Litho files	swt
Number of parameters to send	1
Parameters	float wait time
Examples	0.2 swt –> 200ms wait at begin of line draws 0.0 swt –> no additional wait at begin of line draws -0.2 swt –> FLAG_A/B handshake at begin of line draws

Internally this function gets called by POSITION_ABS_EXTD. It might be useful for lithography. A negative wait time results in FLAG_A being set at the end of each line. Drawing continues when FLAG_B is set. Note that the scanner sets this parameter internally. Use SET_FORWARD_WAIT_TIME and SET_REVERSE_WAIT_TIME when scanning.

3.6 slf (Set Line Flags)

Command code	SET_LINE_FLAGS
Command name in Litho files	slf
Number of parameters to send	1
Parameters	u32 flags

Internally this function gets called by POSITION_ABS_EXTD. It might be useful for lithography. See [pae], page 17 for more details on the flags.

3.7 stnum (Set Number of Triggers)

Command code	SET_TRIGNUM
Command name in Litho files	stnum
Number of parameters to send	2
Parameters	u32 TRIGNUM
	float TRIG_PERIOD

This sets the number of triggers at each A/D position along the next lines. DO NOT USE THIS FOR SETTING THE NUMBER OF TRIGGERS FOR THE NEXT SCAN. Use SET_SC_TRIGNUM instead. Internally this function gets called by POSITION_ABS_EXTD. It might be useful for generating ADC triggers on lithography lines. TRIG_PERIOD is the time between trigger events in seconds

3.8 smmod (Set Motion Mode)

Command code	SET_MOTION_MODE
Command name in Litho files	smmod
Number of parameters to send	1
Parameters	u32 motion_mode

Internally this function gets called by POSITION_ABS_EXTD. It sets the motion mode for the next line to be drawn. This might be useful for lithography.

3.9 sfiad (Set First Ints per A/D Trigger)

Command code	SET_FIRST_INTS_AD
Command name in Litho files	sfiad
Number of parameters to send	1
Parameters	u32 first Ints/AD

This function sets the number of scanner position updates alonog the line before the first A/D trigger sequence is generated. Internally this function gets called by POSITION_ABS_EXTD. It might be useful for lithography.

3.10 soiad (Set Other Ints per A/D Trigger)

Command code	SET_OTHER_INTS_AD
Command name in Litho files	soiad
Number of parameters to send	1
Parameters	u32 other_Ints/AD

This function sets the number of scanner position updates alonog the line between all other A/D trigger positions. Internally this function gets called by POSITION_ABS_EXTD. It might be useful for lithography.

3.11 pu (Pen Up)

Command code	PEN_UP
Command name in Litho files	pu
Number of parameters to send	0

This will deactivate the PEN TTL output.

3.12 pd (Pen Down)

Command code	PEN_DOWN
Command name in Litho files	pd
Number of parameters to send	0

This will activate the PEN TTL output.

3.13 ra (Rotate Absolute)

Command code	ROTATE_ABSOLUTE
Command name in Litho files	ra
Number of parameters to send	1
Parameters	float angle
Examples	32.31 ra –> rotation angle = 32.31 degree

This sets the rotation angle of the image coordinate system with respect to the piezo coordinate system. A positive angle means image rotation counterclockwise. The angle is given in degree. This applies to all subsequent line draw actions including scanning except for POSITION_PURE. Note that a change in rotation angle will result in an immediate line draw to the tip position in the rotated coordinate system. The center of the rotation is the position given by the X and Y offsets.

3.14 rr (Rotate Relative)

Command code	ROTATE_RELATIVE
Command name in Litho files	rr
Number of parameters to send	1

Parameters	float angle_change
Examples	0.2 rr –> increase rotation angle by 0.2 degree

This modifies the rotation angle of the image coordinate system with respect to the piezo coordinate system. A positive angle means image rotation counterclockwise. The angle is given in degree. This applies to all subsequent line draw actions including scanning except for POSITION_PURE. Note that a change in rotation angle will result in an immediate line draw to the tip position in the rotated coordinate system. The center of the rotation is the position given by the X and Y offsets.

3.15 ss (Set Drawing Speed)

Command code	SET_SPEED
Command name in Litho files	ss
Number of parameters to send	1
Parameters	float speed (in physunits/s)

Internally this function gets called by POSITION_ABS_EXTD. Speed changes apply to all subsequent line draws. Note that giving a line time in PREPARE_SCAN or SET_LINETIME will overwrite the speed setting. Also SET_BASE will change the speed in an attempt to conserve the line time. If you want to set the speed in a litho file do that after SET_BASE.

3.16 sm (Set Matrix)

Command code	SET_MATRIX
Command name in Litho files	sm
Number of parameters to send	4
Parameters	float X_Piezo_sens
	float Y_Piezo_sens
	float HV_Gain_X
	float HV_Gain_Y
Examples	13.0 13.0 300.0 300.0 sm

This defines the meaning of `physunits` to the board. The piezo sensitivities are given in physunits/V, the HV_Gains are dimensionless. All other operations are based on physunits so probably this is one of the first instructions given to the board.

3.17 so (Set Offset)

Command code	SET_OFFSET
Command name in Litho files	so
Number of parameters to send	2
Parameters	float Xoffs
	float Yoffs
Examples	-30.0 20.0 so

This moves the offset of the line draw coordinate system in respect to the piezo coordinate system. The offset coordinates are given in the unrotated coordinate system (offset moves the center of rotation). Please note: changing the offset also changes the offset for the current position. The result will be a line draw to the current position at the new offset. This line draw will be done using the current line parameters.

3.18 sor (Set Offset Relative)

Command code	SET_OFFSET_REL
Command name in Litho files	sor
Number of parameters to send	2
Parameters	float deltaXoffs
	float deltaYoffs
Examples	-30.0 20.0 so

This moves the offset of the line draw coordinate system in respect to the piezo coordinate system. The offset coordinates are given in the unrotated coordinate system (offset moves the center of rotation) relative to the current offset position. Please note: changing the offset also changes the offset for the current position. The result will be a line draw to the current position at the new offset. This line draw will be done using the current line parameters.

3.19 sswt (Set Spectro Wait Time)

Command code	SET_SPEC_WAIT_TIME
Command name in Litho files	sswt
Number of parameters to send	1
Parameters	float spectro_wait_time

This determines the wait time at each spectro position in table as well as in array mode. At a spectro position FLAG_A is set. When the time is over FLAG_A is cleared automatically for positive wait times. For negative wait times FLAG_B has to be put high in order to make the scanner continue.

3.20 debug (Get Test Array)

Command code	GET_TEST
Command name in Litho files	debug
Number of parameters to send	0

This command returns an array of debugging information. In distribution versions of the software all elements should be zero.

3.21 trs (Generate Trigger Series)

Command code	TRIGGER_SERIES
Command name in Litho files	trs

Number of parameters to send	4
Parameters	float GROUP_PERIOD
	float TRIG_PERIOD
	u32 NGROUP
	u32 TRIG_NUM
Example	3 6 10e-6 100e-6 trs

This command generates a sequence of trigger groups. GROUP_PERIOD defines the period time of the trigger groups. NGROUP defines the number of groups. If this is zero triggers will be generated until told to stop by STOP_SCAN 2. TR_NUM is the number of triggers in each group. TRIG_PERIOD is the trigger period inside the group. The above example will generate 6 trigger groups each containing 3 trigger pulses 1s apart. The trigger pulse groups will have a period of 100 us.

3.22 arc (Draw Arc)

Command code	DRAW_ARC
Command name in Litho files	arc
Number of parameters to send	3
Parameters	float X0, Y0, angle

This command draws an arc starting from the current position around the center point (X0,Y0). The length is determined by the given angle. Negative angle means clockwise.

3.23 arcr (Draw Arc Relative)

Command code	DRAW_ARC_REL
Command name in Litho files	arcr
Number of parameters to send	3
Parameters	float dX0, dY0, angle

This command draws an arc starting from the current position around the center point (X+dX0,Y+dY0). The coordinates of the center point are given relative to the current position. The length is determined by the given angle. Negative angle means clockwise.

3.24 sai (Set Angle Increment)

Command code	SET_ANGLE_INCREMENT
Command name in Litho files	sai
Number of parameters to send	1
Parameters	float angle_increment

This allows to set the angle that the arc command connects with astraight line. The default value is 1 degree. Decreasing the value makes arcs smoother but a little slower.

3.25 sh (Shake Hand)

Command code	SHAKE_HAND
Command name in Litho files	sh
Number of parameters to send	2
Parameters	u32 mode
	float wait_time

This allows to put wait times and handshake events into lithography files. For mode = 0 or mode = 1 a positive wait_time will be interpreted as a wait time. For mode = 2 the wait time is ignored. A negative wait_time will cause handshaking. The handshaking depends on the mode that has been set:

mode = 0	FLAG_A/FLAG_B handshake (TTL lines)
mode = 1	the command suspends until a SHAKE_HAND command with mode=2 gets sent
mode = 2	The line draw gets continued.

Mode	Wait_time	action
0	pos	wait for wait_time
0	neg	FlagA/FlagB
1	pos	wait for wait_time
1	neg	ethernet handshake (wait for SHAKE_HAND with mode=2)
2	pos	continue drawing
2	neg	continue drawing

For mode=1 and a negative wait time no UDP packet gets sent. If that is required a GET_STATUS command needs to be placed in front of this instruction in the litho file.

3.26 tf (Track Feature)

Command code	TRACK_FEATURE
Command name in Litho files	tf
Number of parameters to send	1
Parameters	u32 ON_OFF

This starts or stops a feature tracking as set up with SETUP_TRACKING. The word ON_OFF can have these values: 0 -> shut off feature tracking 1 -> start feature tracking

3.27 sft (Setup Feature Tracking)

Command code	SETUP_TRACKING
Command name in Litho files	sft
Number of parameters to send	10
Parameters	u32 mode

u32 nconv
float Circle_Radius
float Circle_Freq
float TimeConstant
float ConeHeight
float PhaseShift
float Wait_Time
u32 circles/trace_sample
u32 not_touch_trace

This command sets up the feature tracking. In order to start feature tracking mode use the command TRACK_FEATURE. The feature tracking algorithm draws a circle of radius Circle_Radius around the current position and calculates the slope in X and in Y direction using nconv A/D conversions along the circle circumference. Assuming that the feature is a cone of height ConeHeight and that the circle radius is chosen in way that the circle is at half the cone height it calculates an X and Y displacement of the cone center with respect to the circle center. That X and Y displacement is then used to re-position the circle center on top of the cone using an integrator with the time constant TimeConstant. The cone height can be negative. In that case the algorithm tracks a depression in the surface.

The mode word contains the number of circles to be drawn in the lower 16 bits. If this number is zero the algorithm circles until told to stop by a TRACK_FEATURE command. Bit 16 of the mode word determines whether SET_OFFSET (bit 16 ==1) or the POSITION_ABSOLUTE (bit 16 == 0) commands are used to re-position the circle center on top of the feature.

The nconv word contains the number of A/D conversions taken along each circle. This number has to be >= 3, otherwise it will get coerced to 3 by the scan board processor. The Circle_Radius is given in physical units as defined using SET_MATRIX. Circle_Freq is the number of circles drawn per second. The TimeConstant is the Feedback integrator time constant given in seconds. For a stable feedback the relation TimeConstant >= 1/Circle_Freq should be maintained. PhaseShift gives the phase shift between reference and detected signal. This way for fast tracking the pgase shift that is induced by the scanner mechanics can be compensated for.

Wait_Time defines a wait time (in seconds) before each circle gets drawn. When the parameter not_touch_trace is set to a value different from zero this command will not configure (and thus not reset) the tracking trace buffer in external memory. This allows to reconfigure the feature tracking without having to pick up the data of a prevuios run. Using an unconfigured tracking buffer means crashing the DSP board.

3.28 gtd (Get Tracking Data)

Command code	GET_TRACKING_DATA
Command name in Litho files	gtd
Number of parameters to send	1
Parameters	u32 how_many

This command reads the feature tracking trace out of the scan board external memory. It works exactly the same as READ_ADC_BUFFER except that the data that is sent back is in 32 bits IEEE float format.

3.29 pushs (Push Scaling Factor)

Command code	PUSH_SCALE
Command name in Litho files	pushs
Number of parameters to send	1
Parameters	float new_scale_factor

This command scales the subsequent offset and drawing operations using new_scale_factor. The factor new_scale_factor also gets put onto a 16 entries deep stack that stores the 15 previous pushs scaling factors. When more than 16 pushs operations are done without clearing the stack the first factor will get lost first.

3.30 pops (Pop Scaling Factor)

Command code	POP_SCALE
Command name in Litho files	pops
Number of parameters to send	0

This command pops the scaling stack (puts the second entry into the first position etc...) This way the scaling of a previous pushs can be restored. At initialization the scaling factor is 1.0. When the stack gets underflown it will "produce" scale factors of 1.0.

3.31 dumps (Dump Scaling Stack)

Command code	DUMP_SCALING_STACK
Command name in Litho files	dumps
Number of parameters to send	0

This command returns the contents of the scaling stack

3.32 srm (Set Replay Mode)

Command code	SET_REPLAY_MODE
Command name in Litho files	srm
Number of parameters to send	1
Parameters	u32 on/off

This command enables (on/off <>0) or disables (on/off=0) the replay mode. In replay mode the line draw subsystem outputs a data stream out of the A/D buffers on the Z DAC while scanning. This can be used for raster lithography along scan lines. The data can be placed into the A/D buffers with the wadc command ([wadc], page 7). The buffer gets read one reading per A/D trigger that is generated by the line draw.

3.33 srf (Set_Replay_Factor)

Command code	SET_REPLAY_FACTOR
Command name in Litho files	srf
Number of parameters to send	1
Parameters	float replay_factor

This command sets the scale factor that gets multiplied into the ZDAC output signal in Replay Mode.

3.34 lslm (Line Start Litho Mode)

Command code	LS_LITHO_MODE
Command name in Litho files	lslm
Number of parameters to send	1
Parameters	u32 on/off

This command enables (on/off <>0) or disables (on/off = 0) the line start lithography mode. In this mode the lithography buffer is executed at the beginning of a line draw when a wait time <> 0 is defined.

4 Scan layer commands

This chapter describes the scan layer commands. These commands make use of the line draw software layer.

4.1 prepscan (Prepare Scan)

Command code	PREPARE_SCAN
Command name in Litho files	prepscan
Number of parameters to send	13
Parameters	float baselen –> scan line length in physunits float basewidth –> scan area height in physunits float LineTime –> scan line draw time (one way!) float fwd_wait_tm –> stop time at start of fwd line float rev_wait_tm –> stop time at start of rev line u32 npix –> number of image pixels on each line u32 nlin –> number of lines in the scan u32 motionmode –> C_SPEED/ SINE_SCAN/ NL_LOOKUP u32 rept –> # of repetitions for each scan line u32 where_AD –> A/D triggers on which repetitions u32 SCANNER_FLAGS –> see commands.h for coding u32 TRIGNUM –> # of triggers at each A/D position float TRIG_PERIOD –> trigger period in seconds
Example	7e-6 1 4 3 1 0 200 200 0 0 0.002 12.0 12.0 prepscan

This command sets the parameters for a scan. It then moves the tip position to the scan area corner defined in SCANNER_FLAGS from where the scan is supposed to start. The motion along this line will be the speed of the scan as set by LineTime. Use the SCAN_UNTIL command then to start a scan. Please note that the speed of all line draws is set to the speed that corresponds to LineTime. A Linetime of 0.0 will be ignored and the old scan speed will remain unchanged.

4.2 srep (Set Repetitions)

Command code	SET_REPETITIONS

Command name in Litho files	srep
Number of parameters to send	1
Parameters	u32 rept

Internally this function gets called by PREPARE_SCAN. It is intended mostly for testing

4.3 sbas (Set Scan Base)

Command code	SET_BASE
Command name in Litho files	sbas
Number of parameters to send	2
Parameters	float base_length
	float base_width

Internally this function gets called by PREPARE_SCAN. It is intended mostly for testing. The values base_length and base_width give the size of the scan area in physunits. Please note: SET_BASE changes the speed so the line time is conserved!!!

4.4 slt (Set Line Time)

Command code	SET_LINETIME
Command name in Litho files	slt
Number of parameters to send	1
Parameters	float Line_Time

Internally this function gets called by PREPARE_SCAN. It sets the scan line time. The scan line time is the time that just the forward scan line would take, not a forth and back scan... When other parameters change the system tries to conserve the line time.

4.5 sfwt (Set Forward Wait Time)

Command code	SET_FWD_WAIT_TIME
Command name in Litho files	sfwt
Number of parameters to send	1
Parameters	float wait time at the begin of a fwd line [s]

Internally this function gets called by PREPARE_SCAN. It is intended mostly for testing. The time is given in seconds. Negative wait time result in FLAG_A being set at the end of each fwd line. Scanning continues when FLAG_B is set.

4.6 srwt (Set Reverse Wait Time)

Command code	SET_REV_WAIT_TIME
Command name in Litho files	srwt
Number of parameters to send	1
Parameters	float wait time at the begin of a rev line [s]

Internally this function gets called by PREPARE_SCAN. It is intended mostly for testing. The time is given in seconds. Negative wait time result in FLAG_A being set at the end of each rev line. Scanning continues when FLAG_B is set.

4.7 snp (Set Number of Pixels)

Command code	SET_NPIX_NLIN
Command name in Litho files	snp
Number of parameters to send	2
Parameters	u32 npix
	u32 nlin

Internally this function gets called by PREPARE_SCAN. It is intended mostly for testing. It sets the number of lines in the scan (nlin) and the number of pixels along each line (npix).

4.8 smm (Set Motion Mode)

Command code	SET_SC_MOTMOD
Command name in Litho files	smm
Number of parameters to send	1
Parameters	u32 motion_mode

This function sets the motion for the scan lines of all following scans. The connection lines will be constant speed lines.

4.9 swad (Set Where A/D)

Command code	SET_WHERE_AD
Command name in Litho files	swad
Number of parameters to send	1
Parameters	u32 where_AD

Internally this function gets called by PREPARE_SCAN. It is intended mostly for testing. In where_AD each bit allows AD trigger generation on a certain line repetition number. Bit 0 means first fwd line, bit 1 means 1st rev line etc. A value of where_AD of 2 means only generate A/D trigger sequences at the first backward scan. The bits in here are only considered when the line is drawn. That has to be set using SET_REPETITIONS.

4.10 ssf (Set Scanner Flags)

Command code	SET_SCANNER_FLAGS
Command name in Litho files	ssf
Number of parameters to send	1
Parameters	u32 SCANNER_FLAGS

Internally this function gets called by PREPARE_SCAN. It is intended mostly for testing. The scanner flags describe: Start corner: Bit 0: 1 -> lower 0 -> upper Bit 1: 1 -> right 0 -> left Slow scan direction: Bit 2: 1 -> keep 0 -> alternate

4.11 sctnum (Setup Scan Triggers)

Command code	SET_SC_TRIGNUM
Command name in Litho files	sctnum
Number of parameters to send	2
Parameters	u32 TRIGNUM float TRIG_PERIOD

Internally this function gets called by PREPARE_SCAN. It is intended mostly for testing. It sets the number of A/D triggers generated at each A/D position. Note that this is the value used by the scan generator. If you set TRIGNUM by using SET_TRIGNUM this will be overwritten by the SET_SC_TRIGNUM value at the begin of the next scan. TRIG_PERIOD is the time between trigger events in seconds.

4.12 ms (Move to Scan Start Position)

Command code	MOVE_TO_SCAN_START
Command name in Litho files	ms
Number of parameters to send	0

This moves the tip position to the scan area corner defined in SCANNER_FLAGS from where the scan is supposed to start. The motion along this line will be at the speed of the scan as set by LineTime. Use the SCAN_UNTIL command then to start a scan. Internally this function gets called by PREPARE_SCAN after all other parameters have been set. A MOVE_TO_SCAN_START command can be issued at any time. It will then use the configurations of the last PREPARE_SCAN. When a scan is running a MOVE_TO_SCAN_START command will stop the current scan at the end of the next prepared line and then move to the start corner. Please note that the scan generator works in the background of the line draws that it has "ordered". Any line that got ordered will be drawn. This means that when you issue a MOVE_TO_SCAN_START command the scanner might actually draw another line after the current one before it responds.

4.13 su (Scan Until)

Command code	SCAN_UNTIL
Command name in Litho files	su
Number of parameters to send	1
Parameters	u32 #image_pixels_to_go
Examples	1 stpsc ms 0 su –> stops the old and starts a new continuous scan

SCAN_UNTIL is the command to use when the scanner should move further along the scan lines. The value in #image_pixels_to_go will tell when the scan gets stopped again. A value of 0 means continuous scanning. When the scan is stopped the high word of the machine state word in the GET_STATUS reply is set to 0xffff. The low word still stays at SCAN. SCAN_UNTIL can be used for generating spectroscopy stops. SCAN_UNTIL is also used to get a scan running after the MOVE_TO_START command. When a SCAN_UNTIL is applied to a running line draw it sets the number of A/D conversions from the moment the command came in to the next spectro stop.

4.14 stpsc (Stop the Scan)

Command code	STOP_SCAN
Command name in Litho files	stpsc
Number of parameters to send	1
Parameters	u32 WHERE

This command is used to stop the scan in a defined way. The parameter WHERE says where to abort the scan. The coding is:

`EOSCAN` Stops at the end of a scan frame. Note that this sets the machine state to SNGLESCN for the rest of the frame

`EOLN` Stops the scan at the end of this scan line

`STOP_NOW` Stops the scan immediately, aborting the current line draw

After the STOP_SCAN command the tip position does NOT move to the start corner. You have to use MOVE_TO_SCAN_START for that. Scans stopped at EOSCAN or EOLN can be continued using SCAN_UNTIL. For a scan stoped at STOP_NOW this will give unpredictable results. Use INTERRUPT_SCAN if you want to stop at a previously unknown position along a scan line and then want to continue scanning.

4.15 intsc (Interrupt Scan)

Command code	INTERRUPT_SCAN
Command name in Litho files	intsc
Number of parameters to send	0

This command forces a spectroscopy stop at the position of the scanner. Using SCAN_UNTIL the scan can be continued afterwords. As on any spectro stop position the high word of the machine_state variable is set to 0xffff while the scan is interrupted.

4.16 stopy (Stop Y scanning)

Command code	STOP_Y_SCAN
Command name in Litho files	stopy
Number of parameters to send	1
Parameters	u32 YES_NO –>: STOP Y = 1 RUN Y = 0

This sets or clears a scan generator flag which tells whether the Y line position should be changed during scanning. This allows to scan a whole image along one line. This might be interesting for testing the SPM imaging conditions.

4.17 sws (Set Where Spectro)

Command code	SET_WHERE_SPEC
Command name in Litho files	sws
Number of parameters to send	1

Parameters	u32 where_spectro

Each bit in where_spectro represents a line repetition during scanning. When the bit is set automatic spectroscopy is allowed on that scan repetition.

4.18 sam (Set Array Spectro Mode)

Command code	SPEC_ARRAY_MODE
Command name in Litho files	sam
Number of parameters to send	4
Parameters	u32 first_pix, delta_pix, firstlin, deltalin

This switches on array mode automatic spectroscopy on the scan line repetitions set by SET_WHERE_SPEC. Please note that spectroscopy is only possible at points where A/D conversions are taken. Disabling A/D conversions along a line automatically also disables spectroscopy. The parameter first_pix gives the pixel position of the first spectroscopy position from the line start point. The parameter delta_pix gives the separation of the spectro positions along the line in A/D conversion steps. The parameter firstlin gives the first scan line that contains spectro stops, counted from the begin of the scan. The parameter deltalin gives the separation of spectro scan lines along the image. When firstlin < deltalin the spectro positions of a downward scan and the following upward scan lie exactly on top of each other. Array spectro parameters are:

first_pix	(goes 0 ... npix-1)
delta_pix	(minimum 1)
firstlin	(goes 0 ... nlin-1)
deltalin	(minimum 1)

4.19 stm (Set Spectro Table Mode)

Command code	SPEC_TABLE_MODE
Command name in Litho files	stm
Number of parameters to send	depends on table length
Parameters	spectro position table

This command switches on table spectroscopy mode. This allows to place spectro positions on any A/D trigger position on the image. The table contains line numbers (which have 0x0000 in the high word) and position markers for the spectro position on that line. The position markers are compressed. The Hi word contains a repetition counter and the lower word the position difference to the previous position or to the line start. Note that there is a position difference of 1 needed to reach position 0 on the line. A position difference of zero will shut off spectroscopy for the rest of the line. The table is coded like this:

```
line number ( has 0x0000 in the high word)
 entry 1, has repetition cntr in Hi word and position X diff in lo word
 entry 2, has repetition cntr in Hi word and position X diff in lo word
 entry 3, has repetition cntr in Hi word and position X diff in lo word
 ...
 entry n, has repetition cntr in Hi word and position X diff in lo word
 line number ( has 0x0000 in the high word)
```

```
 entry 1, has repetition cntr in Hi word and position X diff in lo word
 entry 2, has repetition cntr in Hi word and position X diff in lo word
.
.
.

Example:
  0x0
  0x00030001
  0x05
  0x00020003
  0x00010004
  0x0c
  0x00030003
  0x00020001

 will place spectro stops on
 line 0    at positions 0, 1, and 2
 line 5    at A/D positions  2, 5, 9 and
 line 12   at A/D position 2, 5, 8, 9, 10
```

The pixel numbers start with 0 and are in decimal notation here.

4.20 snm (Set no Spectro Mode)

Command code	SPEC_NO_MODE
Command name in Litho files	snm
Number of parameters to send	0

This shuts off automatic spectroscopy. SCAN_UNTIL will still do "manual" spectro positioning.

4.21 linrep (Get Line Repetition)

Command code	N_LINREP
Command name in Litho files	linrep
Number of parameters to send	0

This command puts the number of the line repetition of the scan generator onto the Lithography parameter stack. It can be used to make decisions in lithography files that get started at the beginning of multi line images.

5 lithography commands

This chapter describes the lithography control commands. Most of the commands described in this manual can be placed into a lithography script and executed from the DSP processord memory.

5.1 wlb (Write Litho Buffer)

Command code	WRITE_LITHO_BUF
Command name in Litho files	wlb
Number of parameters to send	variable
Parameters	Instructions to be put into litho buffer

This command can not be put into the lithography buffer. That does not make sense. This is the only command with a variable length. The user has to make sure that the sent packet size does not exceed the ethernet packet size. Otherwise the packet will be split up by the IP layer of the control computer and then ignored by the scan board IP routine. Longer instruction sequences can be downloaded by subsequent WRITE_LITHO_BUF instructions. After the download has succdeeded the lithography pointer points to the first free memory cell after the downloaded commands. This will multiple WRITE_LITHO_BUF instructions to write more instructions than what fits into one ethernet buffer. The WRITE_LITHO_BUF instruction returns the following data structure to the IP address and UDP port number that issued the command:

[0]	WRITE_LITHO_BUF code
[1]	number of written instructions
[2]	remaining free spaces in litho buffer

This is mainly used to confirm that the packet arrived and got processed. (It might get lost in UPD...)

5.2 rlb (Reset Litho Buffer)

Command code	RESET_LITHO_BUF
Command name in Litho files	rlb
Number of parameters to send	0

This is used to stop a running lithography sequence and to set the lithography pointer back to the begin of the buffer. Please note that the lithography buffer controls both where instructions get written to by WRITE_LITHO_BUF and where they are read fron during execution.

5.3 jlb (Jump in Litho Buffer)

Command code	JUMP_BUF
Command name in Litho files	jlb
Number of parameters to send	1
Parameters	u32 position

Examples	0 jlb –> jumps back to the begin and continues

This command simply changes the lithography pointer. This way loops are possible. Conditional jumps are not supported in this version.

5.4 dl (Do Lithography)

Command code	DO_LITHO
Command name in Litho files	dl
Number of parameters to send	0

This executes the commands in the lithography buffer starting at where the lithography pointer points to. During lithography execution the lower word of the machine_state word is set to LITHO.

5.5 + (Plus)

Command code	PLUS
Command name in Litho files	+
Number of parameters to send	0

This command adds the two topmost entries on the Lithography stack. It can only be run from inside a Litho file.

5.6 jnz (Jump Relative on not Zero)

Command code	JNZ
Command name in Litho files	jnz
Number of parameters to send	0

This command will execute a relative jump inside the lithography buffer. The top of the parameter stack is the offset of the jump, counted from one position behing the jnz instruction. Underneath is the counter parameter. When the counter parameter is zero it gets removed from the stack and no jump is executed. Otherwise the instruction does the relative jump. Please look at the looptest.lth example file in the C_utilities for an example of this.

Appendix A Commands.h

This chapter contains a copy of the file `commands.h` that contains all the parameter definitions for the symbolic names used in this manual. This file would have to be included in a C program that is supposed to talk to the DSP scan board.

```
/*--------------------------------------------------------------------

 This file contains the command number definition for the SHARC
 Scan board.


Steffen Porthun, October 14th, 1998

Copyright: RHK Technology Inc. 1998

$Author: steffen $
$Revision: 1.4 $ ,   $Date: 2004/09/08 00:09:36 $

--------------------------------------------------------------------*/


/* All these commands need to be sent to UDP_CMD_PORT !!! */


/* Global settings and status commands ****************************************/
#define IGNORE_SWITCH           0x01        /* u32 which */
#define IGNORE_SWITCH_LEN       0x02
#define IGN_SPEED               0x0

#define WRITE_Z_DAC             0x02        /* float output_voltage */
#define WRITE_Z_DAC_LEN         0x2

#define BSET_DIO                0x03        /* u32 bit pattern to be set */
#define BSET_DIO_LEN            0x02

#define BCLR_DIO                0x04        /* u32 bit pattern to be cleared */
#define BCLR_DIO_LEN            0x02

/* DIO bit definitions */
#define DIO_PAGE                0x0001
#define DIO_LOOP_INVERT         0x0002
#define DIO_Z_RETRACT           0x0004
#define DIO_HOLD_INT            0x0008
#define SCANNING_LED            0x0010
#define SCANNING_LED_BIT        4
#define ADC_CONV_START          0x0020
#define ADC_CONV_START_BIT      5
#define OVERRIDE_LED            0x0040
#define OVERRIDE_LED_BIT        6
#define SCAN_LEFT_RIGHT         0x0080
#define SCAN_LEFT_RIGHT_BIT     7
#define PEN_DOWN_LINE           0x0100
#define PEN_DOWN_LINE_BIT       8

#define TRIGGER_ADC             0x05        /* void */
#define TRIGGER_ADC_LEN         0x01

#define READ_ADC_BUF            0x06        /* u32 how_many samples */
```

```
#define READ_ADC_BUF_LEN        0x02

#define GET_STATUS              0x07        /* void */
#define GET_STATUS_LEN          0x01

#define GET_DAQ_MODE            0x08    /* void */
#define GET_DAQ_MODE_LEN        0x01

#define SET_DAQ_MODE            0x09    /* u32 buflen, channels, mode */
/* channel definitions */
#define ADC1            0x00
#define ADC2            0x01
/* mode word definitions */
#define AD_EXTBUF       0x00    /* 0 -> Buffer in internal memory     1 -> external buffer */
#define AD_AUTOTRIG     0x01    /* 0 -> needs triggering              1 -> trigger gen-
erated in ADC_isr */
#define AD_AVERAGE      0x02    /* 0 -> one value for each trigger    1 -> average val-
ues until read */
#define POS_INTEGR      0x03    /* Use scan positioning nonlinearity I feedback algorithm 0 -
> no, 1 -> yes */
#define FIBER_FB        0x04    /* Do fiber piezo feedback routine 0 -> no, 1 -> yes */
#define AD_BUF_FULL     0x10    /* 0 -> still space in buffer         1 -> buffer full */
#define AD_CONVERTING   0x11    /* 0 -> not converting                1 -> ADCs busy... */

#define SET_DAQ_MODE_LEN        0x04

#define SET_POS_FB              0x0a    /* u32 mode */
#define NO_POS_FB               0x0
#define FB_INTEGRATOR           0x1

#define SET_POS_FB_LEN          0x02

#define SET_POS_FB_TC           0x0b    /* float TimeConstant */
#define SET_POS_FB_TC_LEN       0x02

#define SET_DETECTOR_CHAR       0x0c    /* float X_A, X_B, X_C, X_D, X_E, Y_A, Y_B, Y_C, Y_D, Y_E */
#define SET_DETECTOR_CHAR_LEN   0x0b

#define READ_DETECTOR_CHAR      0x0d    /* void, returns X_A, X_B, X_C, X_D, X_E, Y_A, Y_B, Y_C, Y_D, Y_E
#define READ_DETECTOR_CHAR_LEN  0x01

#define TAKE_DIO                0x0e   /* u32 BIT_PATTERN ... a 1 means SHARC controls the line. See DIO
initions above   */
#define TAKE_DIO_LEN            0x02

#define SET_Z_SCALE             0x0f   /* float Z_Scale */
#define SET_Z_SCALE_LEN         0x2

#define Z_SET_SPEED             0x10   /* float Z_Sweep_Speed */
#define Z_SET_SPEED_LEN         0x2

#define SETUP_Z_SWEEP           0x11   /* u32 Z_Sweep_Flags, #Sample_groups, #samples_per_group, Thresh-
oldMode, AD_Channels,
                                          float Z_Sweep_Speed, PreSampleDelay, TimePer-
Sample, UpperThresholdValue, LowerThresholdValue */
/* Z_Sweep_Flags */
#define OPEN_FB                 0x0    /* not implemented yet */
#define RESET_ADC               0x1
```

```
#define SETUP_Z_SWEEP_LEN       0xb

#define ABORT_Z_SWEEP           0x12    /* void */
#define ABORT_Z_SWEEP_LEN       0x1

#define SETUP_FIBER_FB          0x13    /* float setpoint, float TimeConstant, float In-
terferom_calib */
#define SETUP_FIBER_FB_LEN      0x4

#define START_FIBER_FB          0x14    /* FLAG  ( 0 => off, everything else => on ) */
#define START_FIBER_FB_LEN      0x2


#define SET_DAQ_CHANS           0x15    /* u32 channels */
#define SET_DAQ_CHANS_LEN       0x2

#define SETUP_WATCHDOG          0x16        /* float lo, float hi , u32 use_pen_bit */
#define SETUP_WATCHDOG_LEN      0x4

#define SHOW_WATCHDOG           0x17        /* void, returns: float lo, float hi, BOOLEAN use_pen_bit */
#define SHOW_WATCHDOG_LEN       0x1

#define WRITE_ADC_BUF           0x18        /* ( m param_1 . . . param_m -- m ) */

#define GET_GAINS               0x19       /* void, returns:
      float X Offset Gain
      float X Scan   Gain
      float Y Offset Gain
      float Y Scan   Gain
      float Z Offset Gain
      float Z Scan   Gain */
#define GET_GAINS_LEN           0x1


/* line draw operation commands ************************************************/

#define POSITION_ABSOLUTE       0x101        /* float X, float Y */
#define POSITION_ABSOLUTE_LEN   0x03

#define POSITION_RELATIVE       0x102        /* float X, float Y */
#define POSITION_RELATIVE_LEN   0x03

#define POSITION_ABS_EXTD       0x103        /* float X, Y, speed, wait_time;
                     u32 flags, TRIGNUM, MotionMode, first_Ints/AD, other_Ints/AD, first-
spec */
               /*
                        float X, float Y   -> destination position X and Y coordinate
                        float speed,       -> drawing speed of this line
                        float wait_time    -> Wait time at the end of this line in seconds
                        u32 flags          -> flags, saying whether pen up and whether A/D trig-
gers on this line
                                Bit1 PU/PD      0 -> PU         1 -> PD,
                                Bit7 ADC        0 -> YES        1 -> NO
                        u32 TRIGNUM        -> number of trigger pulses at each A/D po-
sition in case of ADtrig set
                        float TRIG_PERIOD  -> trigger sequence period time in seconds
                        u32 MotionMode,    -> motion mode C_SPEED/ SINE_SCAN/ NL_LOOKUP
```

```
                        u32 first_Ints/AD, -> First A/D trigger after how many scanner updates
                        u32 other_Ints/AD, -> How many scanner updates between all other A/D po-
sitions along the line
                        i32 firstspec      -> stop for spectro after how many A/D trig-
ger positions
                                                    firstspec == 0  -> never stop, don't change re-
mainder of previous line
                                                    firstspec == -1 -> use remainder of pre-
vious line
                */

/* position of the parameters for the scanner to call this function */
#define         PAE_X           0x0
#define         PAE_Y           0x1
#define         PAE_SPEED       0x2
#define         PAE_WAIT_TIME   0x3
#define         PAE_FLAGS       0x4
#define         PAE_TRIGNUM     0x5
#define         PAE_TRIG_PERIOD 0x6
#define         PAE_MOTIONMODE  0x7
#define         PAE_FIAD        0x8
#define         PAE_OIAD        0x9
#define PAE_FIRSTSPEC           0xa

#define POSITION_ABS_EXTD_LEN   0x0c

#define SET_WAIT_TIME           0x104       /* float wait time */
#define SET_WAIT_TIME_LEN       0x02

#define SET_LINE_FLAGS          0x105       /* u32 flags */
#define SET_LINE_FLAGS_LEN      0x02

#define SET_TRIGNUM             0x106       /* u32 TRIGNUM, float TRIG_PERIOD */
#define SET_TRIGNUM_LEN         0x03

#define SET_MOTION_MODE         0x107       /* u32 motion_mode */
#define SET_MOTION_MODE_LEN     0x02

/* SET_MOTION_MODE codes */
#define C_SPEED            0        /* no speed variations along the lines, no nonlin-
earity tables */
#define SINE_SCAN        0x1        /* vary the speed with a cosine function */
#define NL_LOOKUP        0x2        /* use nonlinearity lookup tables */


#define SET_FIRST_INTS_AD       0x108       /* u32 first Ints/AD */
#define SET_FIRST_INTS_AD_LEN   0x02

#define SET_OTHER_INTS_AD       0x109       /* u32 other_Ints/AD */
#define SET_OTHER_INTS_AD_LEN   0x02

#define PEN_UP                  0x10a       /* void */
#define PEN_UP_LEN              0x01

#define PEN_DOWN                0x10b       /* void */
#define PEN_DOWN_LEN            0x01

#define ROTATE_ABSOLUTE         0x10c       /* float angle */
```

```
#define ROTATE_ABSOLUTE_LEN     0x02

#define ROTATE_RELATIVE         0x10d       /* float angle_change*/
#define ROTATE_RELATIVE_LEN     0x02

#define SET_SPEED               0x10e       /* float speed ( in Matrix units/s )*/
#define SET_SPEED_LEN           0x02

#define SET_MATRIX              0x10f       /* float X_Piezo_sens, Y_Piezo_sens [V/phys_unit], float HV_
#define SET_MATRIX_LEN          0x05

#define SET_OFFSET              0x110       /* float Xoffs, Yoffs */
#define SET_OFFSET_LEN          0x03

#define SET_SPEC_WAIT_TIME      0x111       /* float spectro_wait_time */
#define SET_SPEC_WAIT_TIME_LEN  0x02

#define GET_TEST                0x112       /* void */
#define GET_TEST_LEN            0x01
#define TEST_ARRAY_LEN          0x4


#define TRIGGER_SERIES          0x113   /* float GROUP_PERIOD, TRIG_PERIOD  u32 NGROUP TRIG_NUM */
#define TRIGGER_SERIES_LEN      0x05

#define DRAW_ARC                0x114   /* float X0, Y0, angle */
#define DRAW_ARC_LEN            0x04

#define SET_ANGLE_INCREMENT     0x115   /* float angle_increment */
#define SET_ANGLE_INCREMENT_LEN 0x02

#define POSITION_PURE           0x116   /* float X, Y */
#define POSITION_PURE_LEN       0x03

#define SHAKE_HAND              0x117   /* u32 mode float time */
#define SHAKE_HAND_LEN          0x03
/* modes */
#define FLAG_HS_MODE            0x00
#define ENET_HS_MODE            0x01
#define ENET_HS_ACK             0x02

#define DRAW_ARC_REL            0x118   /* float dX0, dY0, angle */
#define DRAW_ARC_REL_LEN        0x04

#define TRACK_FEATURE           0x119   /* u32 ON_OFF */
#define TRACK_FEATURE_LEN       0x02

/* ON_OFF */
#define FT_ON                   0x1
#define FT_OFF                  0x0

#define SETUP_TRACKING          0x11a   /* u32 mode, nconv */
                                        /* float Circle_Radius,Circle_Freq,TimeConstant,ConeHeight,PhaseO
                                        /* u32 circ_per_trace_sample, no_trace_touch */
#define SETUP_TRACKING_LEN      0x0b

/* mode contains the number of circles to run in the bits 0 ... 15  all bits zero mean cir-
cle until told to stop */
```

```
#define MOVE_OFFSETS            0x10    /* How to track a feature:     1 => use scan off-
sets   0 => use tip position */
#define PC_SPEED_CTLR           0x11    /* Speed treatment after Ft:   1 => just restore speed 0 => re-
store speed from knobs */

#define PUSH_SCALE              0x11b   /* float scaling_value */
#define PUSH_SCALE_LEN          0x2
#define SCALE_STACK_DEPTH       0x10


#define POP_SCALE               0x11c   /* void */
#define POP_SCALE_LEN           0x1

#define SWEEP_Z_ABSOLUTE        0x11d   /* float destination */
#define SWEEP_Z_ABSOLUTE_LEN    0x2

#define DUMP_SCALE_STACK        0x11e   /* void */
#define DUMP_SCALE_STACK_LEN    0x1

#define SET_OFFSET_REL          0x11f   /* float dX dY */
#define SET_OFFSET_REL_LEN      0x3

#define GET_TRACKING_DATA       0x120   /* u32 how_many samples */
#define GET_TRACKING_DATA_LEN   0x02

#define SET_REPLAY_MODE         0x121   /* u32 on? */
#define SET_REPLAY_MODE_LEN     0x02

#define LS_LITHO_MODE           0x122   /* u32 on? */
#define LS_LITHO_MODE_LEN       0x02

#define SET_REPLAY_FACT         0x123   /* float factor */
#define SET_REPLAY_FACT_LEN     0x02



/* scan layer commands ************************************************/

#define PREPARE_SCAN            0x201       /* float baselen, basewidth, LineTime, fwd_wait_tm, rev_wait
                                           u32 npix, nlin, motionmode, rept, where_AD, SCAN-
NER_FLAGS, TRIGNUM,
                                           float TRIG_PERIOD, wait_time_at_start_position */
        /*
                        float baselen        -> length of a scan line in physical units
                        float basewidth        -> height of the scan area in Y  in physunits
                        float LineTime        -> Time to draw one scan line (one way!)
                        float fwd_wait_tm -> wait time at the end of a forward scan line
                        float rev_wait_tm -> wait time at the end of a reverse scan line
                        u32 npix        -> number of scan trigger positions on each line
                        u32 nlin         -> number of scan lines.
                        u32 motionmode         -> C_SPEED/ SINE_SCAN/ NL_LOOKUP
                        u32 rept           -> how many times repeat each scan line
                        u32 where_AD        -> create A/D triggers on which line repetitions
                                                    Bit 0: 1st forward
                                                    Bit 1: 1st backward
                                                    Bit 2: 2nd forward
                                                    Bit 3: 2nd backward
```

```
                                                        .
                                                        .
                                                        .
                                                Bit31:        16th backward

                        u32 SCANNER_FLAGS ->        Start corner
                                                Bit 0: 1 -> lower  0 -> upper
                                                Bit 1: 1 -> right  0 -> left
                                            Slow scan direction:
                                                Bit 2: 1 -> keep   0 -> alternate
                        u32 TRIGNUM        -> number of A/D triggers at each A/D position
                        float TRIG_PERIOD -> period of each trigger sequence in seconds
        */
#define PREPARE_SCAN_LEN        0x0e

#define SET_REPETITIONS         0x202       /* u32 rept */
#define SET_REPETITIONS_LEN     0x02

#define SET_BASE                0x203       /* float base_length, base_width */
#define SET_BASE_LEN            0x03

#define SET_LINETIME            0x204       /* float LineTime */
#define SET_LINETIME_LEN        0x02

#define SET_FWD_WAIT_TIME       0x205       /* float wait time at the end of a fwd line [s] */
#define SET_FWD_WAIT_TIME_LEN   0x02

#define SET_REV_WAIT_TIME       0x206   /* float wait time at the end of a rev line [s] */
#define SET_REV_WAIT_TIME_LEN   0x02

#define SET_NPIX_NLIN           0x207       /* u32 npix, nlin */
#define SET_NPIX_NLIN_LEN       0x03

#define SET_SC_MOTMOD           0x208   /* u32 scan_line_motion_mode */
#define SET_SC_MOTMOD_LEN       0x2

#define SET_WHERE_AD            0x209       /* u32 where_AD */
#define SET_WHERE_AD_LEN        0x02

#define SET_SCANNER_FLAGS       0x20a       /* u32 SCANNER_FLAGS */
#define SET_SCANNER_FLAGS_LEN   0x02

#define SET_SC_TRIGNUM          0x20b       /* u32 TRIGNUM, float TRIG_PERIOD */
#define SET_SC_TRIGNUM_LEN      0x03

#define MOVE_TO_SCAN_START      0x20c       /* void */
#define MOVE_TO_SCAN_START_LEN  0x01

#define SCAN_UNTIL              0x20d       /* u32 #image_pixels , when zero continu-
ous scan */
#define SCAN_UNTIL_LEN          0x02

#define STOP_SCAN               0x20e       /* u32 WHERE */
#define STOP_SCAN_LEN           0x02
   /* note: a scan stopped NOW! cannot be continued */
/* WHERE can be: */
#define EOSCAN          0x0
#define EOLN            0x1
```

```
#define STOP_NOW        0x2

#define INTERRUPT_SCAN          0x20f        /* void */
#define INTERRUPT_SCAN_LEN      0x01

#define STOP_Y_SCAN             0x210        /* u32 YES_NO : STOP IT = 1   RUN IT = 0 */
#define STOP_Y_SCAN_LEN         0x02

#define SET_WHERE_SPEC          0x211        /* u32 where_spectro */
#define SET_WHERE_SPEC_LEN      0x02

#define SPEC_ARRAY_MODE         0x212        /* u32 first_pix, delta_pix, firstlin, deltalin */
#define SPEC_ARRAY_MODE_LEN     0x05

#define SPEC_TABLE_MODE         0x213        /* u32[] spectro table */
/* length not checked */

#define SPEC_NO_MODE            0x214        /* void */
#define SPEC_NO_MODE_LEN        0x01

#define N_LINREP                0x215        /* only in litho files, returns line rep-
etition there */
#define N_LINREP_LEN            0x01


/* lithography stuff ********************************************************/

#define WRITE_LITHO_BUF         0x301        /* instructions ... */
   /* the length is not checked!!! */

#define RESET_LITHO_BUF         0x302        /* void */
#define RESET_LITHO_BUF_LEN     0x01
   /* next instructions get written to the begin of the buffer */

#define JUMP_BUF                0x303        /* i32 position */
#define JUMP_BUF_LEN            0x02

   /* executes the litho buffer */
#define DO_LITHO                0x304        /* void */
#define DO_LITHO_LEN            0x01

   /* puts data onto the litho stack */
#define PUSH_FLOAT              0x305   /* float parameter to push */
#define PUSH_FLOAT_LEN          0x02

#define PUSH_INT                0x306   /* u32 parameter to push */
#define PUSH_INT_LEN            0x02

#define PLUS                    0x307   /* u32 u1 u2 */
#define PLUS_LEN                0x03

#define JNZ                     0x308   /* i32 where u32 counter */
#define JNZ_LEN                 0x03

#define USER_CMD1               0x0401        /* u32 parameter */
#define USER_CMD1_LEN           0x02        /* the command itselve and the parameter */
```

```
/* Planefit commands */
#define  SET_PLANEFIT_MODE      0x0500
#define  SET_PLANEFIT_MODE_LEN  0x02

#define  GET_PLANEFIT_MODE      0x0501
#define  GET_PLANEFIT_MODE_LEN  0x01

#define  WRITE_PLANEFIT_PAR     0x0502
#define  WRITE_PLANEFIT_PAR_LEN 0x03

#define  READ_PLANEFIT_PAR      0x0503
#define  READ_PLANEFIT_PAR_LEN  0x01

#define  SET_Z_OFFSET           0x0504
#define  SET_Z_OFFSET_LEN       0x02

#define  GET_Z_OFFSET           0x0505
#define  GET_Z_OFFSET_LEN       0x01
```

Appendix B scan board code change log

This chapter contains the revision log of the code development of the DSP scan board project. The purpose of this is to document at what revision certain features appear in the code.

```
********** version 0x45 ****************************************
March 29th, 2006
* Gave a pending set-base priority over a pending set-offset

********** version 0x44 ****************************************
March 24th, 2006
* scanner Y signal generation redesign

********** version 0x43 ****************************************
March 22nd, 2006
* fixed the table termination in spectro table mode
* increased the spectro table to a length of 350 dwords

********** version 0x42 ****************************************
March 17th, 2006
* Changed the replay mode gain default

********** version 0x41 ****************************************
March 17th, 2006
* inverted replay mode polarity back to original status

********** version 0x40 ****************************************
March 17th, 2006
* fixed the code addition of 0x3E

********** version 0x3F ****************************************
March 16th, 2006
* fixed the code addition of 0x3E

********** version 0x3E ****************************************
March 16th, 2006
* subtracts the first data value from the whole replay line in
  replay mode now

********** version 0x3D ****************************************
Jan  19th, 2006
* changed FLAG handshake timeout from 25 us to one second

Dec 16th, 2004
* allowed scan rotation when Y scan is off

********** version 0x3C ****************************************
Sept 7th, 2004
* added GET_GAINS command, updated manual
Sept 3rd, 2004
* uses separate ISRs now for the Flag_A/Flag_B double handshake


********** version 0x3B ****************************************
May 20th, 2004
* made trs work inside a litho script
```

```
********** version 0x3A ****************************************


********** version 0x39 ****************************************
February 17th, 2004
* increased the repetition rate of the FLAG_ISR

********** version 0x39 ****************************************
January 21st, 2004:
* added WRITE_ADC_BUF (wadc)
* added SET_REPLAY_MODE (srm)
* added SET_REPLAY_FACT (srf)
* added LS_LITHO_MODE (lslm)
* added N_LINREP (linrep)


********** version 0x38 ****************************************
May 21st, 2002:
* Added a FIBER_FB_RUNS (bit 0x40000) to USTAT1 and the internal
  flags (word 21 of the GET_STATUS reply)

May 20th, 2002:
* Added the Interferometer setpoint to the end of GET_STATUS.
* Added SETUP_WATCHDOG and SHOW_WATCHDOG.


********** version 0x37 ****************************************
March 28th, 2002:
* Reset the BR3 and LR3 registers at init, so that IR3 can be used in a
  predictable manner.
* Increased the buffer size for asyncronous GET_STATUS reply.

March 7th, 2002:
* Added SET_DAQ_CHANS to set the data acquisition channels.
* SET_DAQ_MODE doesn't do anything now when the buffer size <= 0.


********** version 0x36 **********************
March 7th, 2002:
* READ_ADC_BUF can send back more that the maximum packet size.
* Added a parameter to SETUP_TRACKING that allows to leave the feature
  track trace untouched.

********** version 0x35 **********************
March 5th, 2002:
* Transformed the feature tracking output to physical units in the
  unrotated coordinate system.
* Rewrote the scan rotation function.

********** version 0x34 **********************
March 1st, 2002:
* fixed a bug in the scan speed that had to do with the new WAIT in the
  1.7 kernel.
* Fixed the feature tracking speed

February 28th.2002:
Added a feature track position tracing
Added the position feedback buffer number of samples to the
```

```
GET_STATUS reply.

February 25th, 2002:
* Fixed the scan speed after return from Feature Tracking
* Added a description of the internal flags to the GET_STATUS manual

************ version 0x33 **********************
February 22nd, 2002:
* Rewrote the command decoding. Saved a lot of code space that way...
* Added the position feedback position to the GET_STATUS output

************ version 0x32 **********************
February 19th, 2002:
* Fixed feature tracking. Added a phase parameter to the sft parameter
  list .

************ version 0x31 **********************
January, 2002:
* For kernels >= 1.7 the scan program does not set the WAIT register any
  more. For lower kernels we change it to fix a memory addressing problem.

************ 0x30 ****************************
June 27th, 2001:
* Appended the ZDAC output value to the GET_STATUS reply

************ 0x2f ****************************
June 15th, 2001:
* Changed the meaning of the ZDAC value in the interferometer fb
  loop. Now the ZDAC value IS the integrator.

************ version 0x2e **********************
June 8th, 2001:
* Added float Interferom_calib to the parameter list of
  SETUP_FIBER_FB and to the GET_STATUS reply.

June 6th, 2001:
* Added SETUP_FIBER_FB and START_FIBER_FB


************ version 0x2d **********************
March 30, 2001:
* Fixed Z positioning inaccuracies
* Fixed the timing pointer save in Z sweeps
* Changed the 40 bit read out of internal memory block 0

March 28, 2001:
* Fixed a bug in SET_DAQ_MODE that screwed up A/D into the external 48bank.
* Changed the Control_Loop parameter of SETUP_Z_SWEEP to
  Z_Sweep_Flags. Now the bits in there have meanings.

************ version 0x2A **********************
Oct. 20th:
* Polished the Threshold mode.

Oct. 19th:
* Added Threshold mode to Z sweeps.
* Fixed a bug that would crash at fast z sweeps when no XY scanning has
  been done before.
```

```
************* version 0x28 ***********************
Oct. 12th:
* Trigadc can be used in a litho file loop. It checks whether the ADC is
  busy before it triggers again.
* Changed the parameters of the SET_DAQ_MODE and GET_DAQ_MODE
  commands. Watch out!!

Oct. 10th:
* Added ABORT_Z_SWEEP, SETUP_Z_SWEEP and Z_SET_SPEED instruction
* Fixed a saturation on overflow bug when the timer ISR counter gets
  bigger than 2^31

************** version 0x27 ***********************
Oct. 6th:
* Added the SET_Z_SCALE parameter to the end of the GET_STATUS reply.
* Added a PLUS and a JNZ instruction. These only work in Litho mode.
* Added a DUMP_SCALE_STACK instruction

*************** version 0x26 ***********************
Oct. 3rd:
* Added a scaling factor to WRITE_Z_DAC. The instruction is SET_Z_SCALE

*************** version 0x25 ***********************
Oct. 2nd:
* Added some Frank wishes to GET_STATUS
* ADC buffer length is given in samples, not triggers any more

*************** version 2-4 ***********************
Sept. 29th:
* added non-functional Z sweep commands, fixed a couple of small scan size bugs

Sept. 20th:
* Changed the meaning of the X_D and Y_D parameters in
  SET_DETECTOR_CHAR and READ_DETECTOR_CHAR.  Also converted all
  factors of these commands so that they represent Volts and not ADC
  bits for the detector reading.

**************** version 2.0 ***********************
Sept. 8th:
* Fixed two bugs that are related to the number of triggers in scans
  with handshaking and resizing during scanning
* Removed the additional trigger at th ebegin of the first frame after
  board reboot.

Aug. 23rd:
* Added TAKE_DIO
* Added PUSH_SCALE and POP_SCALE
* Fixed a bug so that now the scan rotation/offsetting on lines with A/D
  triggers is only disabled when the scan generator runs. This is kind
  of ugly but it works...

Aug. 17th:
* Fixed a bug in the way the X and Y got coerced to the scan area limits

Aug. 16th:
* changed the position feedback gain to a time constant
* removed the POS_PREDICT positioning algorithm
```

```
* added 2 more parameters for XY decoupling to the detector characteristics

Aug. 15th:
* Revised the SETUP_TRACKING interface, fixed some bugs in feature tracking

Aug. 10th:
* Fixed a bug that would NAN the scan speed and Line time for scans with
  less than 1 DAC pixel per A/D trigger.
* Moved the location of the A/D trigger at position 0 behind the
  Flag_A/Flag_B handshake so that SPM32 won't miss that.

Aug. 7th:
* Does not move the offsets on lines with triggers any more.
  This was needed in order to be able to move the offsets while scanning
  without screwing up th enumber of triggers generated
  Now allows to change the scan area while scanning without losing
  triggers.

Aug. 1st:
* Added feature tracking. Still needs extensive testing

July 30th:
* Now the scan area limits take the offsets into account. For a -5V
  offset the scan can now fo from -5V to +15 V. Makes more sense this
  way...

July 26th:
* added ARC_RELATIVE command. This one is used by HPGL conversion

***************** version 1.1 ***********************/
*************** v. 1.1 beta3 below here ***********************
June 15th:
* added ENET_HS_MODE to SHAKE_HAND
* added READ_DETECTOR_CHAR

May 31st:
* Added SHAKE_HAND, removed some test stuff out of arc.asm

May 16th:
* Changed the direction of the scan L/R bit so that it is HI on forward scans.
  It is a Scan Forward/*Backwards bit now.
* Added POSITION_PURE
* fixed STOP_SCAN(STOP_NOW) so that scan LED still works afterwards

May 2nd:
* Added a MOVING bit (bit 0x10000) to USTAT1 and the internal flags (word 21 of
  the GET_STATUS reply)

Feb. 29th:
* added SET_ANGLE_INCREMENT.

*************** version 1.1 beta2 below here ******************
Feb. 28th:
* Added DRAW_ARC. This command uses the current position as the start of
  the arc. Furthermore the coordinates of the center and the angle have
  to be given.

*************** version 1.1 beta1 below here ******************
```

```
Feb. 24th, 2000:
* Changed scantest so that it can handle litho files bigger than one ethernet block.

Feb. 23rd, 2000:
* The Z position gain knob does not shut off the Override LED any more.
* GET_STATUS can now be called from a litho file. This allows to send a
  message via ethernet when some action on the scan board is done. This
  is used in the "calibrate" function in scantest.
* Added SET_LINETIME to the manual and to scantest.
* The line time was initialized to be zero. Pretty bad when SET_BASE conserves the line
  time... That got fixed.
* See SET_SPEED for details on some strange timing behaviour in litho files that I fought
  the whole morning.
* Changed SET_DETECTOR_CHAR so it accepts the coefficients in physical units vs ADC pixels.
  Since the PC gets to see the ADC pixels anyway I think this is the best solution. Let me
  know if it is not...

Feb 22nd, 2000:
* Added Data acquisition and positioning feedback. This resulted in the new
  commands SET_POS_FB, SET_POS_FB_GAIN and SET_DETECTOR_CHAR.
* Basically works but probably needs some more debugging. Also the SET_DETECTOR_CHAR
  should work in physical units...
* I moved the motion mode codes out of scanner.h into commands.h where anybody can see
  and use them now.
* I also added a 'calibrate' function to scantest. This function uses a litho file
  calibrateFB.lth in order to move the scanner to +- max rangeand trigger data acquisition.
  From the data it calculates the detector characteristics and sends it down to the board.

Jan 31st, 2000:
* put under CVS control
* Added GET_DAQ_MODE and SET_DAQ_MODE
* Added averaging data acquisition on-board
* Added auto trigger mode


********** version 1.0  **********************************

********** version 0.3 below here ************************
Aug 25th:
* Added TRIGGER_SERIES

Aug 17th:
* Fixed a bug in the command interpreter that sometimes allowed that a second command
  got called after the first was finished. This made ROTATE_ABSOLUTE change the Z_DAC value...
* Changed SET_SPEED, SET_LINE_TIME and PREPARE_SCAN so they don't set the machine into"
  front panel knob ignore" mode.


Aug 12th:
* Changing the speed by front panel knobs does not screw up triggering any more
* Made the override LED work consistently
* Limits the maximum speed that can be set from the knobs so that the
* Line time is always above  NPIX * TRIGNUM * TRIG_PERIOD.
* Moved the line wait time to the begin of the line.
Please note that PREPARE_SCAN and MOVE_TO_SCAN_START have one less parameter now.
The scan board will ignore the whole command when you send too many parameters!!!

Aug 10th:
```

```
* added GET_TEST.

Aug 4rth:
* Table mode spectro: could not place positions on line 0, that works now. Line
  numbers go from 0 to nlin-1 now. Please read description about the pixel numbering
* Restored the manual spectro mode from a couple of bugs that the auto spectro introduced
* Fixed a bug in the scan speed update. It never checked whether the resolution changed
  before changing the speed. In relation with auto spectro this created some spooky crashes...

Aug 3rd:
* Made the scanning LED to come up with every line draw. It goes off at spectro positions now.
* Automatic spectro positions pull the FLAG_A line HI while at the stop, even when a
  positive wait time is set
* Automatic spectro positions show FFFF in the hi word of the status
* Array spectro parameters are:
   first_pix   (goes 0 ... npix-1)
   delta_pix   (minimum 1)
   firstlin     (goes 0 ... nlin-1)
   deltalin     (minimum 1)

Jul 23rd:
* Added SPEC_TABLE_MODE, SPEC_ARRAY_MODE, SPEC_NO_MODE and SET_WHERE_SPEC to the
* scanner functions

Jul 22nd:
* Added table and array mode spectroscopy. Added the command SET_SPEC_WAIT_TIME.

Jul 16th:
* Fixed a bug that I made on Jul 15th, that ignored user speed settings after the user changed
  them once
* Fortified security in the SET_MATRIX and SET_SPEED so speed == 0 and matrix elements == 0
  are taken care of. Introduced a line draw error of 0x02 in case of zero matrix elements.
* Included the SET_MATRIX parameters into the GET_STATUS return
* Included the spectro wait time into the GET_STATUS return.

Jul 15th:
* Multiple PREPARE_SCANs do not screw up the handshaking at the scan start position
  any more

Jul 14th:
* Corrected the description of SET_MATRIX. The piezo sensitivities are given in
  PhysUnits/V.
* Changed the default AD trigger output pulse width to 500 ns
* Fixed the offset conversion for GET_STATUS output
* Made negative piezo sensitivities reverse the direction of drawing

Jul 12th:
* Made STOP_SCAN NOW! to abort a pending spectro wait.
* Added a  STOP_SCAN NOW! to the begin of the PREPARE_SCAN instruction

Jul 8th, 1999:
* Changed the line draw interrupt routine so that coordinates that exceed the DAC
  area will be clipped to the edge of the DAC area. GET_STATUS will show the line
  draw error 1 when this has happened. The GET_STATUS command clears the errors
  after they have been read.
* Bug fixed: In the GET_STATUS reply the offset values really show physical units now.

Jun 30th, 1999:
```

```
* In the GET_STATUS response the
* Z_gain and the line time are floats in the code and not u32 as stated before...

Jun 29th, 1999
* Added the trigger_period parameter to the description of PREPARE_SCAN and
* to the scan code and the description of the GET_STATUS reply (modified commands.h)

Jun 28th, 1999
* Fixed the GET_STATUS response
* Added more information about timing
```

Index

+

A

C

D

F

G

H

I

J

L

M

P

R

S

T

W

Y

Z