Manual

Software Spacefab INFN
Gamma Ray Spectrometer Collimator

Author	Jürgen Staud
Revision	1.96
Date	12.10.2016
Project	P15017

Content

1 Warnings

After turning on the system or plugging in the cables,
you must perform the calibration sequence before you start any other movement!
Otherwise, the movements are uncontrolled, and there is the possibility of a collision.

Nach dem Einschalten des Systems müssen als erstes die Endschalter gesucht werden, bevor eine Bewegung gestartet wird. Ansonsten sind die Bewegungen unkontrolliert, und es besteht die Gefahr einer Kollision!

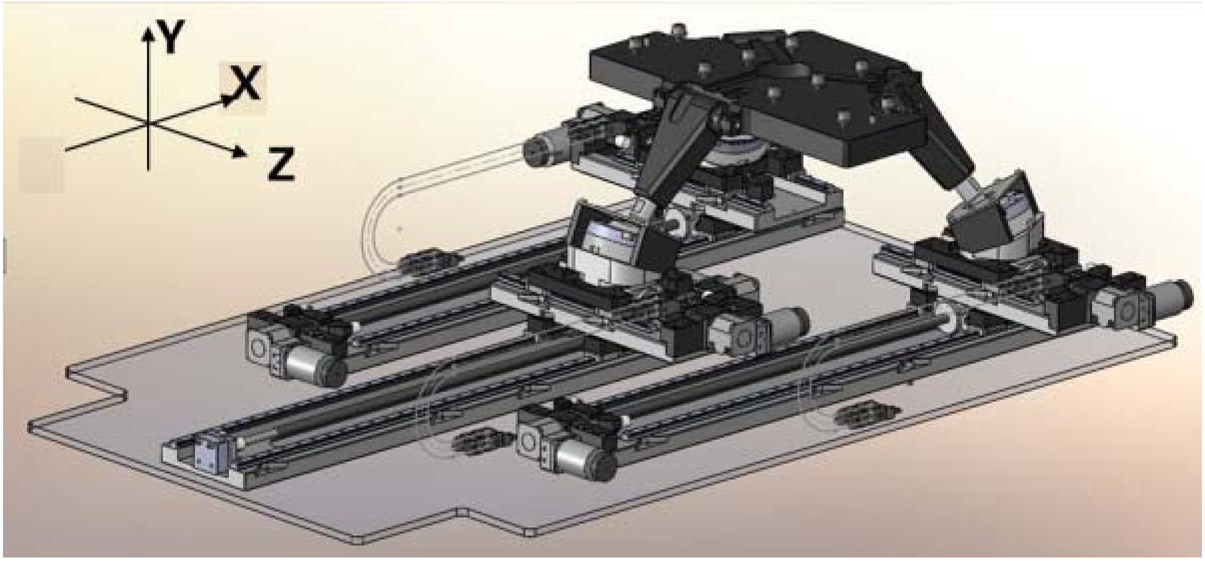
Never connect or disconnect cables which are under power!

Keine Stecker lösen oder verbinden, die unter Spannung stehen.

Under no circumstances disconnect the cable of a rotary or linear encoder if the controller is enabled. Heavy damage at the system as well as injuries of people can occur!

Unter keinen Umständen darf das Kabel eines Messsystems abgezogen werden, solange die Steuerung aktiviert ist. Schwere Schäden an der Mechanik sowie Personenschäden können die Folge sein.

2 System Description

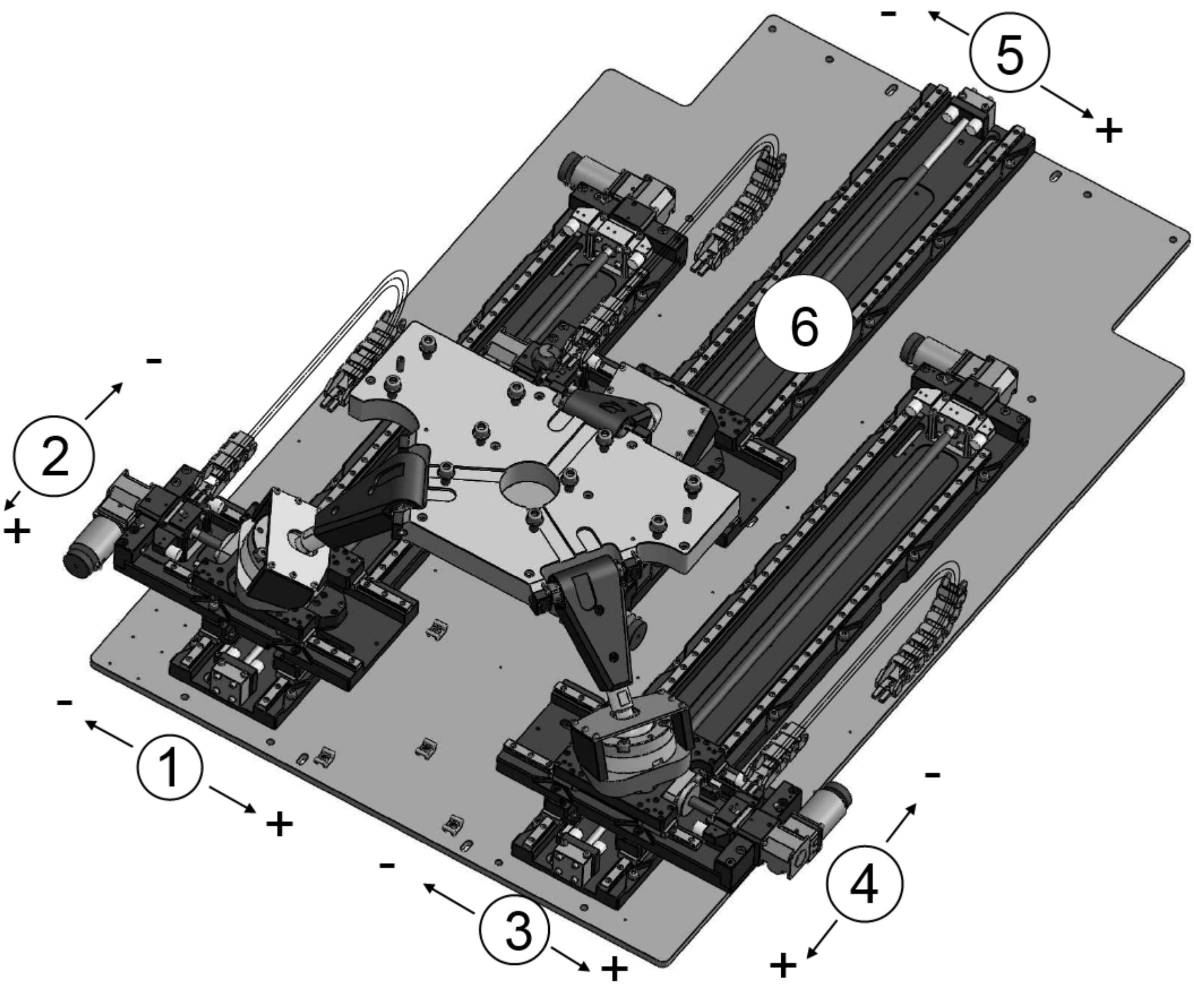


The SpaceFab carries a gamma ray spectrometer collimator assembly with a weight of 200 kg. The gamma ray is propagating in z direction.

The actual working position is 9.294 mm higher than the zero position.

The default setting for the pivot point is 0 207.9 -20.

~~For this application, we have a left-handed coordinate system.~~ Changed back to right-handed
The transformation from PI miCos coordinate system to INFN coordinate system is done with
x → z and z → -x.



No axis in this system has limit switches; only soft limits in connection with the home switches are used.

The home switches are configured as normally open instead of normally closed.
A "not connected" switch shows 1 in the variable Mx20, just like a "not touched" switch. An activated switch shows 0 (closed).
So in case of a broken cable, the stages do the homing in positive direction.

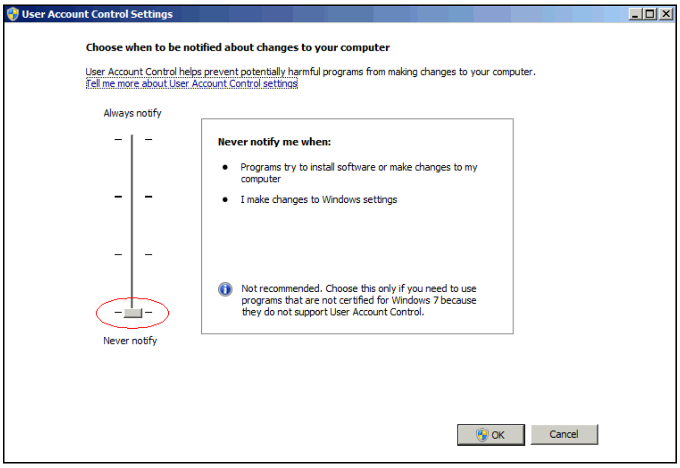
3 Communication Description

This chapter describes the communication with the DeltaTau PMAC Geobrick controller.
For the physical connection, RS-232, USB or TCP/IP is possible.

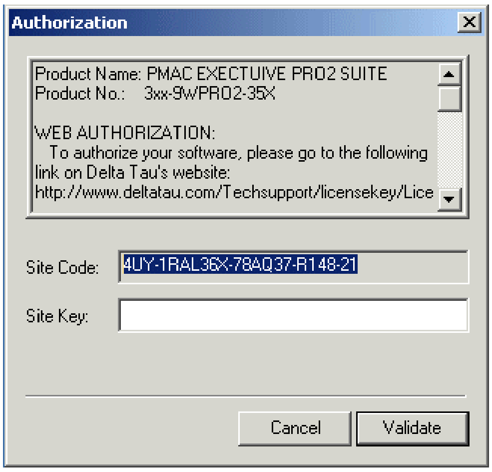
For Windows PCs, the TCP/IP and USB communication is based on the PComm32W.DLL respectively DotNet. The PMAC Executice Pro2 Suite has to be installed for these drivers. This suite also installs the PEWIN32 Pro2 Terminal Software.
For non-windows PCs, the TCP/IP communication has to be done with the help of Berkeley sockets, here a separate document is available (PMAC Ethernet Protocol). Attention: it is not a pure ASCII-communication!
Communication over RS-232 can be done directly without drivers, the default baud rate is 38400 (I54=12).

3.1 Installation of the PMAC Executice Pro2 Suite

- for Win7 PCs, first reduce the “User account control settings” (Benutzerkontensteuerung) to “Never notify” (Nie benachrichtigen)
- then reboot the PC
- install the PMAC Executive Pro2 suite
- double-click the pmac.reg file to create some registry settings (configures the driver automatically for the controller)
- then the commands described below can be sent from the PEWIN32 Pro2 Terminal Software, from the program JS_Terminal or from an application software of your own
- PEWIN32 can be used 30 days in a test period, otherwise it has to be activated. For this, you have to go to http://www.deltatau.com/DT_License/LicenseForm1.aspx and enter the “Serial Number” of your license as well as the “Site Code” from the PEWIN32 authorization window, as a result you will get the “Site Key” which you have to enter in the PEWIN32 authorization window.



User accont control settings



PEWIN32 authorization

3.2 Configuration

The controllers are configured for the following IP addresses:

Controller	IP address (dec)	IP address (hex)	SN Geobrick	SN Spacefab
DeltaTau PMAC Geobrick 1	192.168.10.3	C0.A8.0A.03	C000D99S	416003375
DeltaTau PMAC Geobrick 2	192.168.10.4	C0.A8.0A.04	C000D99J	416003376

3.3 Communication tests

The communication with the DeltaTau PMAC controller can be tested with commands like:

- CPU
- TYPE
- VERSION

3.4 First steps

After switching on the power, send the following commands to prepare the system for operation:

- &2
- Q70=1 to start the calibration sequence of the SpaceFAB

3.5 Demo programs

The following demo programs can be used:

Prg27	demo program

Example:

- Q70=11 to start the program in buffer 27

3.6 Variables

The variables Q0..3000 and P0..600 are used by the SpaceFAB and therefore are not available for the user.

3.7 Command Variable

Q70 is the command variable and is used to start the following functions:

Value	Function	
1	start calibration sequence of CS2	PLC20
4	move CS2 to the position specified in Q71…Q79	Prog21
6	set pivot point specified in Q94 … Q96	
7	get current pivot point and write it to Q94 … Q96	
11	start the demo program	Prog27
16	stop + clear position deviation (P90)	
18	move CS2 to the position specified in Q71…Q79	Prog21
19	start calibration sequence of CS2	PLC20
21	get current offset and write it to Q94 … Q96	
22	set offset specified in Q94 … Q96	
23	set offset based on current position defined in Q77 … Q79	
24	set offset to 0	
25	set offset to 9.294 in y	
42	create rotation matrix based on angles specified in Q94 … Q96	
43	create identity matrix	
44	create matrix with INFN orientation	

Do not start movements if the previous movement has not yet finished, always check P80 first!

3.8 Communication Variables

These variables should be checked periodically from the application software.

Q70	Command variable	see above in 3.7
Q71	Destination CS2 A	destination coordinate used by Q70=4
Q72	Destination CS2 B	destination coordinate used by Q70=4
Q73	Destination CS2 C	destination coordinate used by Q70=4
Q77	Destination CS2 X	destination coordinate used by Q70=4
Q78	Destination CS2 Y	destination coordinate used by Q70=4
Q79	Destination CS2 Z	destination coordinate used by Q70=4
Q81	Report CS2 A	current reported position, updated by PLC10
Q82	Report CS2 B	current reported position, updated by PLC10
Q83	Report CS2 C	current reported position, updated by PLC10
Q87	Report CS2 X	current reported position, updated by PLC10
Q88	Report CS2 Y	current reported position, updated by PLC10
Q89	Report CS2 Z	current reported position, updated by PLC10
Q90	Pivot mode	0 = moving with platform, 1 = fixed in space
Q91	SysError	kinematic error which stopped the movement, see 3.9.8
Q92	P10Error	kinematic error concerning the current reported position, see 3.9.8
Q93	LastIteration	number of iterations the NR took to calculate the reported position
Q94	Pivot/offset/angle X	default pivot 0
Q95	Pivot/offset/angle Y	default pivot 207.9
Q96	Pivot/offset/angle Z	default pivot -20
Q97	Demand Speed CS2	in mm/s / °/s
Q98	TransScaling	1 = mm, 0.001 = µm
Q99	RotScaling	1 = °
P80	CSxReady	is set to 0 by Q70=1/4/18/19 is set to 1 after the program has finished successfully is set to > 1 after the program has finished with an error, see 3.9.7 is set to < -1 after the program has aborted with an error, see 3.9.7 may be set to –1 by the application program to acknowledge the value
P89	PMAC CS Error	result of PLC5 Housekeeping, see 3.9.5
P90	PMAC CS Info	result of PLC5 Housekeeping, see 3.9.6
P91	PMAC Motor Error	result of PLC5 Housekeeping, see 3.9.1
P92	PMAC Motor Status	result of PLC5 Housekeeping, see 3.9.2
P93	PMACEncoderError	result of PLC5 Housekeeping, see 3.9.4
P94	PMAC SoftLimit	result of PLC5 Housekeeping, see 3.9.3
P96	PMAC IO Status	result of PLC5 Housekeeping, see 3.9.9

The file “PMAC Watch.WTB” can be loaded in a watch window of PEWIN32 for an easy access to those communication variables.

While making Q98 and Q99 smaller than 1, it might be necessary to increase the speed of the coordinate system (Q97), see also I5298 CS2MaxFeedRate.

3.9 Error codes

3.9.1 PMAC Motor Errors

The following errors can be found in P91, the value can be a sum of the errors:

hex value	name	created by
$00000001	PMACPhase1Error	M148=1
$00000002	PMACPhase2Error	M248=1
$00000004	PMACPhase3Error	M348=1
$00000008	PMACPhase4Error	M448=1
$00000010	PMACPhase5Error	M548=1
$00000020	PMACPhase6Error	M648=1
$00000040	PMACPhase7Error	M748=1
$00000080	PMACPhase8Error	M848=1
$00000100	PMACHome1Error	M145=0
$00000200	PMACHome2Error	M245=0
$00000400	PMACHome3Error	M345=0
$00000800	PMACHome4Error	M445=0
$00001000	PMACHome5Error	M545=0
$00002000	PMACHome6Error	M645=0
$00004000	PMACHome7Error	M745=0
$00008000	PMACHome8Error	M845=0
$00010000	PMACAmplifier1Error	M142 / M143 / M146 / M147 = 1
$00020000	PMACAmplifier2Error	M242 / M243 / M246 / M247 = 1
$00040000	PMACAmplifier3Error	M342 / M343 / M346 / M347 = 1
$00080000	PMACAmplifier4Error	M442 / M443 / M446 / M447 = 1
$00100000	PMACAmplifier5Error	M542 / M543 / M546 / M547 = 1
$00200000	PMACAmplifier6Error	M642 / M643 / M646 / M647 = 1
$00400000	PMACAmplifier7Error	M742 / M743 / M746 / M747 = 1
$00800000	PMACAmplifier8Error	M842 / M843 / M846 / M847 = 1
$01000000	PMACAltera1Error	M180 &$1C ≠ 4
$02000000	PMACAltera2Error	M280 &$1C ≠ 4
$04000000	PMACAltera3Error	M380 &$1C ≠ 4
$08000000	PMACAltera4Error	M480 &$1C ≠ 4
$10000000	PMACAltera5Error	M580 &$1C ≠ 4
$20000000	PMACAltera6Error	M680 &$1C ≠ 4
$40000000	PMACAltera7Error	M780 &$1C ≠ 4
$80000000	PMACAltera8Error	M880 &$1C ≠ 4

The bits „PMACHomeXError“ are set after power on of the controller and show, that the calibration sequence has to be started (Q70 = 1).

The bits „PMACAmplifierXError“ should normally not be seen, they show an error.
„PMACAmplifierXError“ includes the “Fatal Following Error” (Mx42), the “Amplifier Fault Error” (Mx43), the “Integrated following error fault” (Mx46), and the “I2T Amplifier Fault Error” (Mx47) bits of the motor status.
A reset can be tried with Q70=16.

The bits „PMACAlteraXError“ should normally not be seen, they show an error.
A reset can be tried with PLC 4 and a ^A.

Altera messages:

000	no error, no ready
001	OK, ready
010	bus undervoltage warning
011	over temperature on power card
100	I2T warning or fault
110	over current fault

The bus undervoltage warning is only available for motor 1 – 4, it will not be shown for motor 5 – 8.

If bus undervoltage warning occurs, then all motors are killed.

3.9.2 PMAC Motor Status

The following status informations can be found in P92, the value can be a sum of the informations:

hex value	name	created by
$00000001	Motor 1 moving	M137=1 or (M133=0 and M138=0)
$00000002	Motor 2 moving	M237=1 or (M233=0 and M238=0)
$00000004	Motor 3 moving	M337=1 or (M333=0 and M338=0)
$00000008	Motor 4 moving	M437=1 or (M433=0 and M438=0)
$00000010	Motor 5 moving	M537=1 or (M533=0 and M538=0)
$00000020	Motor 6 moving	M637=1 or (M633=0 and M638=0)
$00000040	Motor 7 moving	M737=1 or (M733=0 and M738=0)
$00000080	Motor 8 moving	M837=1 or (M833=0 and M838=0)
$00000100	Motor 1 OpenLoop	M138=1 / M114=0
$00000200	Motor 2 OpenLoop	M238=1 / M214=0
$00000400	Motor 3 OpenLoop	M338=1 / M314=0
$00000800	Motor 4 OpenLoop	M438=1 / M414=0
$00001000	Motor 5 OpenLoop	M538=1 / M514=0
$00002000	Motor 6 OpenLoop	M638=1 / M614=0
$00004000	Motor 7 OpenLoop	M738=1 / M714=0
$00008000	Motor 8 OpenLoop	M838=1 / M814=0
$00010000	Motor 1 MLIM	M122=1
$00020000	Motor 2 MLIM	M222=1
$00040000	Motor 3 MLIM	M322=1
$00080000	Motor 4 MLIM	M422=1
$00100000	Motor 5 MLIM	M522=1
$00200000	Motor 6 MLIM	M622=1
$00400000	Motor 7 MLIM	M722=1
$00800000	Motor 8 MLIM	M822=1
$01000000	Motor 1 PLIM	M121=1
$02000000	Motor 2 PLIM	M221=1
$04000000	Motor 3 PLIM	M321=1
$08000000	Motor 4 PLIM	M421=1
$10000000	Motor 5 PLIM	M521=1
$20000000	Motor 6 PLIM	M621=1
$40000000	Motor 7 PLIM	M721=1
$80000000	Motor 8 PLIM	M821=1

Mx14: AENA output status
Mx33: desired velocity zero
Mx37: running program
Mx38: open loop mode
Mx21: PLIMx flag input status (real hardware switch)
Mx22: MLIMx flag input status (real hardware switch)

The bits “Motor x moving” are set as long as the axis is moving.

The bits “Motor x open loop” are set if the stage is in open loop mode, Q70=16 brings the stage back to closed loop mode.

The bits “Motor x MLIM” and “Motor x PLIM” show that a limit switch (plus / minus) is touched, this happens usually only during the calibration sequence.

3.9.3 PMAC SoftLimit

The following status informations can be found in P94, the value can be a sum of the informations:

hex value	name	created by
$00000001	DOF X neg softlimit	Error in inverse kinematics while moving in that direction
$00000002	DOF Y neg softlimit	Error in inverse kinematics while moving in that direction
$00000004	DOF Z neg softlimit	Error in inverse kinematics while moving in that direction
$00000008	DOF Rx neg softlimit	Error in inverse kinematics while moving in that direction
$00000010	DOF Ry neg softlimit	Error in inverse kinematics while moving in that direction
$00000020	DOF Rz neg softlimit	Error in inverse kinematics while moving in that direction
$00000040		
$00000080		
$00000100	DOF X pos softlimit	Error in inverse kinematics while moving in that direction
$00000200	DOF Y pos softlimit	Error in inverse kinematics while moving in that direction
$00000400	DOF Z pos softlimit	Error in inverse kinematics while moving in that direction
$00000800	DOF Rx pos softlimit	Error in inverse kinematics while moving in that direction
$00001000	DOF Ry pos softlimit	Error in inverse kinematics while moving in that direction
$00002000	DOF Rz pos softlimit	Error in inverse kinematics while moving in that direction
$00004000		
$00008000		
$00010000	Motor 1 neg softlimit	M132=1 AND M122=0
$00020000	Motor 2 neg softlimit	M232=1 AND M222=0
$00040000	Motor 3 neg softlimit	M332=1 AND M322=0
$00080000	Motor 4 neg softlimit	M432=1 AND M422=0
$00100000	Motor 5 neg softlimit	M532=1 AND M522=0
$00200000	Motor 6 neg softlimit	M632=1 AND M622=0
$00400000	Motor 7 neg softlimit	M732=1 AND M722=0
$00800000	Motor 8 neg softlimit	M832=1 AND M822=0
$01000000	Motor 1 pos softlimit	M131=1 AND M121=0
$02000000	Motor 2 pos softlimit	M231=1 AND M221=0
$04000000	Motor 3 pos softlimit	M331=1 AND M321=0
$08000000	Motor 4 pos softlimit	M431=1 AND M421=0
$10000000	Motor 5 pos softlimit	M531=1 AND M521=0
$20000000	Motor 6 pos softlimit	M631=1 AND M621=0
$40000000	Motor 7 pos softlimit	M731=1 AND M721=0
$80000000	Motor 8 pos softlimit	M831=1 AND M821=0

Mx21: PLIMx flag input status (real hardware switch)
Mx22: MLIMx flag input status (real hardware switch)
Mx31: Positive-end-limit-set bit (like in the Motor Status)
Mx32: Negative-end-limit-set bit (like in the Motor Status)

P230 shows the movement vector which is defined at the beginning of the movement. In case of a kinematic error, this movement vector is copied into P94.

3.9.4 PMAC Encoder Errors

The following errors can be found in P93, the value can be a sum of the errors:

hex value	name	created by
$00000001	PMACEncoderLossDetected1	M182=0
$00000002	PMACEncoderLossDetected2	M282=0
$00000004	PMACEncoderLossDetected3	M382=0
$00000008	PMACEncoderLossDetected4	M482=0
$00000010	PMACEncoderLossDetected5	M582=0
$00000020	PMACEncoderLossDetected6	M682=0
$00000040	PMACEncoderLossDetected7	M782=0
$00000080	PMACEncoderLossDetected8	M882=0
$00000100		
$00000200		
$00000400		
$00000800		
$00001000		
$00002000		
$00004000		
$00008000		

3.9.5 PMAC CS Errors / Misc Errors

The following errors can be found in P89, the value can be a sum of the errors:

hex value	name	created by
$00000001	PMACCS1RunTimeError	M5182 = 1
$00000002	PMACCS2RunTimeError	M5282 = 1
$00000004	PMACLimitStopCS1	M5185 = 1 desired position limit stop CS1
$00000008	PMACLimitStopCS2	M5285 = 1 desired position limit stop CS2
$00000010	Motor Voltage	M180, M280, … Altera Under-Voltage
$00000020	Motion Abort	M88 = 1
$00000100	PMACCS3RunTimeError	M5382 = 1
$00000200	PMACCS4RunTimeError	M5482 = 1
$00000400	PMACLimitStopCS3	M5385 = 1 desired position limit stop CS3
$00000800	PMACLimitStopCS4	M5485 = 1 desired position limit stop CS4
$00001000	PMACEncoderLossError	Mx82=0
$00004000	PMACAxisNotConnError	Mx21=1 and Mx22=1

The bits „PMACCSXRunTimeError“ are set if the PMAC aborts a running motion program of a coordinate system.

The bits „PMACLimitStopCSX“ are set if a movement exceeds the allowed travel range of a motor (soft limits) and therefore the movement is stopped.

The bit „Motor Voltage“ shows that the power supply for the amplifiers failed.

The bit „Motion Abort“ shows that the interlock (motion stop button) has been activated.

The bit “PMACEncoderLossError” is not cleared automatically, this has to be done with ena plc 2.

The bit “PMACAxisNotConnError” is not cleared automatically, this has to be done with ena plc 2.

3.9.6 PMAC CS Info

The following status informations can be found in P90, the value can be a sum of the informations:

hex value	name	created by
$00000001	CS1 program running	M5180 = 1 or M4024 = 0
$00000002	CS2 program running	M5280 = 1 or M4020 = 0
$00000004	Position deviation	
$00000010	CS3 program running	M5380 = 1 or M4025 = 0
$00000020	CS4 program running	M5480 = 1 or M4026 = 0
$00000100	CS1 not homed	
$00000200	CS2 not homed	
$00000400	CS3 not homed	
$00000800	CS4 not homed	
$00001000	CS1 not phased	
$00002000	CS2 not phased	
$00004000	CS3 not phased	
$00008000	CS4 not phased	
$00010000	CS1 feedrate override	The feedrate is not between 90% and 110%
$00020000	CS2 feedrate override	The feedrate is not between 90% and 110%
$00040000	CS3 feedrate override	The feedrate is not between 90% and 110%
$00080000	CS4 feedrate override	The feedrate is not between 90% and 110%

If a movement is aborted by e.g. a Motion Stop, the current reported position (e.g. Q87) is different than the destination position (e.g. Q77). This is indicated with bit 2 of P90. The threshold for this deviation bit is 0.1 mm / 0.1°.
For following absolute positionings this does not matter because Q77 is overwritten anyway. But for following relative positionings this could lead to unwanted results. Therefore with Q70=16 this bit can be cleared by copying the current reported positions to the destination positions.

3.9.7 Program Errors

The following positive errors can be found in P80:

Code	Error	Meaning
$8	PrevMoveNotReady	Command was ignored because the previous move was not yet ready (P80=0)
$10	Voltage Error	The power supply for the amplifiers failed.
$20	MotionStopError	An activated motion stop button prevents the movement. Please check all motion stop buttons.
$80	Open Loop Error	An axis went unexpectedly into open loop mode.
$200	Axis not homed	The movement is currently not yet allowed, please start the calibration sequence first (Q70=1).
$800	E1 not found	Movement out of E1 was stopped after MaxE1Distance (P158)

The following negative errors can be found in P80:

-2	CS1 program aborted
-4	CS2 program aborted
-8	inverse kinematic error, check Q91
-16	PMAC error, check P91 and P89
-32	CS3 program aborted
-64	CS4 program aborted

- P80 > 1: program has finished with an error
- P80 < -1: program has aborted with an error
- P80 = 1: program has finished successfully

3.9.8 Kinematic Errors

The following errors can be found in Q91 or Q92, the value in the Q variable can be a sum of the errors:

$1	ErrDivByZero
$2	ErrNegArgSqrt
$4	ErrLegError
$8	ErrTransX
$10	ErrTransY
$20	ErrTransZ
$40	ErrMoveY1
$80	ErrMoveY2
$100	ErrMoveY3
$200	ErrPlatAngle1
$400	ErrPlatAngle2
$800	ErrPlatAngle3
$1000	ErrBaseAngle1
$2000	ErrBaseAngle2
$4000	ErrBaseAngle3
$8000	ErrInternal
$10000	
$20000	ErrMotor1X
$40000	ErrMotor1Z
$80000	ErrMotor2X
$100000	ErrMotor2Z
$200000	ErrMotor3X
$400000	ErrMotor3Z
$800000	ErrMotorDist
$1000000	ErrSoftLimit
$2000000	
$4000000	ErrMatSingu
$8000000	ErrIteration
$10000000	ErrNotHomed

3.9.9 PMAC IO Status

The following status informations can be found in P96, the value can be a sum of the informations:

Hex value	created by	
$00000001	M0	DI1
$00000002	M1	DI2
$00000004	M2	DI3
$00000008	M3	DI4
$00000010	M4	DI5
$00000020	M5	DI6
$00000040	M6	DI7
$00000080	M7	DI8
$00000100	M8	DI9
$00000200	M9	DI10
$00000400	M10	DI11
$00000800	M11	DI12
$00001000	M12	DI13
$00002000	M13	DI14
$00004000	M14	DI15
$00008000	M15	DI16
$00010000	M32	DO1
$00020000	M33	DO2
$00040000	M34	DO3
$00080000	M35	DO4
$00100000	M36	DO5
$00200000	M37	DO6
$00400000	M38	DO7
$00800000	M39	DO8
$01000000		
$02000000		
$04000000		
$08000000		

M0 ... M15 are digital inputs, M32 ... M39 are digital outputs of the group 1 (Basic IO).

3.10 Configuration Variables

To reset all settings, simply type dis plc 10, dis plc 5, Q0..8191=0, P0..8191=0, save, $$$.

3.10.1 System Configuration Variables

If Q12 is 0, then the following P-variables are automatically filled with default values, if Q12 is 1, this is skipped.

		416003375	416003376	
P100	MotorScale1	2048000	2048000	cts/mm
P101	MotorScale2	2048000	2048000	cts/mm
P102	MotorScale3	2048000	2048000	cts/mm
P103	MotorScale4	2048000	2048000	cts/mm
P104	MotorScale5	2048000	2048000	cts/mm
P105	MotorScale6	2048000	2048000	cts/mm
P108	PhasingDelay	200	200	ms
P109	HomingDelay	1000	1000	ms
P128	MaxMotorSpeedCS2	0.5	0.5	mm/s
P129	AbortDecSpeedCS2	0.5	0.5	mm/s
P130	HomeOffset1	-2409600	-1597440	cts
P131	HomeOffset2	-409600	573440	cts
P132	HomeOffset3	-409600	1740800	cts
P133	HomeOffset4	-204800	430080	cts
P134	HomeOffset5	-204800	614400	cts
P135	HomeOffset6	307200	1208320	cts
P140	FlagModeControl1	$820401	$820401	
P141	FlagModeControl2	$820401	$820401	
P142	FlagModeControl3	$820401	$820401	
P143	FlagModeControl4	$820401	$820401	
P144	FlagModeControl5	$820401	$820401	
P145	FlagModeControl6	$820401	$820401	
P158	MaxE1Distance	1	1	mm
P159	ReversalErrorComp	0	0	cts
P261	SpeedSwitchIn1	512	512	cts/ms
P262	SpeedSwitchIn2	512	512	cts/ms
P263	SpeedSwitchIn3	512	512	cts/ms
P264	SpeedSwitchIn4	512	512	cts/ms
P265	SpeedSwitchIn5	512	512	cts/ms
P266	SpeedSwitchIn6	512	512	cts/ms
P271	SpeedSwitchOut1	51	51	cts/ms
P272	SpeedSwitchOut2	51	51	cts/ms
P273	SpeedSwitchOut3	51	51	cts/ms
P274	SpeedSwitchOut4	51	51	cts/ms
P275	SpeedSwitchOut5	51	51	cts/ms
P276	SpeedSwitchOut6	51	51	cts/ms

P281	SpeedMoveOffset1	512	512	cts/ms
P282	SpeedMoveOffset2	512	512	cts/ms
P283	SpeedMoveOffset3	512	512	cts/ms
P284	SpeedMoveOffset4	512	512	cts/ms
P285	SpeedMoveOffset5	512	512	cts/ms
P286	SpeedMoveOffset6	512	512	cts/ms
P300	EMotorScale1	2048000	2048000	cts/mm
P301	EMotorScale2	2048000	2048000	cts/mm
P302	EMotorScale3	2048000	2048000	cts/mm
P303	EMotorScale4	2048000	2048000	cts/mm
P304	EMotorScale5	2048000	2048000	cts/mm
P305	EMotorScale6	2048000	2048000	cts/mm

EMotorScaleN is used in the homing steps to calculate to E1 – Index N variable.
e.g.: E1IndexAx1 = M173 / EMotorScale1
and E1IndexAx1 = M173 / EMotorScale1 - E1IndexAx1

action on fault bits:
$0 : kill all PMAC motors
$2 : kill all motors in same CS
$4 : kill only this motor

Gearhead 20:1, 102400 x 20 = 2048000

Max speed 0.5 mm/s (10 rev/s)

3.10.2 Spacefab Configuration Variables

If Q11 is 1, then after turning on the PMAC, the calibration sequence is started automatically, and each time after the calibration sequence has finished, the demo program 27 is started.
If Q12 is 0, then the variables Q17..64 are automatically filled with default values, if Q12 is 1, this is skipped.
If Q14 is 0, then the softlimits Q240..251 are automatically filled with default values, if Q14 is 1, this is skipped.
If Q15 is 1, the check for the softlimits defined in Q240..251 is skipped.

Q11	AutoStart	
Q12	SkipWriteIniValues	=0 → Q17..64, P100..300
Q13		
Q14	SkipWriteSoftLimits	=0 → Q240..251
Q15	SkipSoftLimitCheck	
Q16	ResetNewtonRhapson	
Q17	MaxIteration	20
Q18	CS2MaxFeed	2
Q19	PlatformHeight	22
Q20	StartHeadLX	-125.57368354874360
Q21	StartHeadLY	134.3502884254440
Q22	StartHeadLZ	72.5
Q23	StartHeadRX	125.57368354874360
Q24	StartHeadRY	134.3502884254440
Q25	StartHeadRZ	72.5
Q26	StartHeadBX	0.0
Q27	StartHeadBY	134.3502884254440
Q28	StartHeadBZ	-145.0
Q29	HeadLegAngleL	-60.0° [rad]
Q30	HeadLegAngleR	60.0° [rad]
Q31	HeadLegAngleB	0° [rad]
Q32	LegLengthL	190
Q33	LegLengthR	190
Q34	LegLengthB	190
Q35	MaxHingeAngle	12
Q36	MaxBallPointAngle	10
Q37	MotorMoveMaxX	37
Q38	MotorMoveMaxZ	30
Q39	MotorMoveMinX	-37
Q40	MotorMoveMinZ	-590
Q41	MotorScale1	2048000
Q42	MotorScale2	2048000
Q43	MotorScale3	2048000
Q44	MotorScale4	2048000
Q45	MotorScale5	2048000
Q46	MotorScale6	2048000
Q47	AxisAngle1	0

Q48	AxisAngle2	0
Q49	AxisAngle3	0
Q53	KinematicOffset1	-8.648679
Q54	KinematicOffset2	4.993317
Q55	KinematicOffset3	8.648679
Q56	KinematicOffset4	4.993317
Q57	KinematicOffset5	0
Q58	KinematicOffset6	-9.986635
Q59	EpsilonNR	0.0000001
Q60	EpsilonGauss	0.0000001
Q61	EpsilonIntern	0.0001
Q64	RotaryBufferSize	10000
Q65	TravelMaxX	37 calculated
Q66	TravelMinX	-37 calculated
Q67	TravelMaxZ	30 calculated
Q68	TravelMinZ	-590 calculated
Q69	JointsHeight	134.35 calculated

Default Pivot Point 0 207.9 -20

Q561	AffineMat00	0
Q562	AffineMat01	0
Q563	AffineMat02	-1 → +1 left → right-handed
Q564	AffineMat10	0
Q565	AffineMat11	1
Q566	AffineMat12	0
Q567	AffineMat20	-1
Q568	AffineMat21	0
Q569	AffineMat22	0
Q588	AffineOffsX	0
Q589	AffineOffsY	9.294
Q590	AffineOffsZ	0

3.10.3 Spacefab Softlimit Variables

The following values are written automatically into the Q variables Q240..251 if Q14 = 0.

	SoftLimit	Value	
Q240	XMin	-2.05	The translation DOFs are checked against these soft limits. The soft limit checks happen with the raw DOFs, without the affine transformations.
Q241	XMax	2.05	
Q242	YMin	-2.05 + 9.294	
Q243	YMax	2.05 + 9.294	
Q244	ZMin	-591	
Q245	ZMax	2.05	
Q246	AMin	-2.1	The rotational DOFs are checked against these soft limits.
Q247	AMax	2.1	
Q248	BMin	-2.1	
Q249	BMax	2.1	
Q250	CMin	-2.1	
Q251	CMax	2.1	
Q252	XMinPiv	-10	Rotations with a non-default pivot point create translational movements. The translational DOFs based on a zero pivot point are checked against these soft limits.
Q253	XMaxPiv	10	
Q254	YMinPiv	5	
Q255	YMaxPiv	14	
Q256	ZMinPiv	-591	
Q257	ZMaxPiv	10	

3.11 Affine Transformations

For the reported positions (Q81 … Q89) and the destination coordinates (Q71 … Q79), affine transformations can be used.
The offset defines the translation, and a matrix is used for rotations.
The variables Q98 and Q99 are used for scaling. The scaling should not be done with the matrix!

The softlimits or the pivot point use the raw (untransformed) positions.
The demo programs probably will not work if you use the transformations.

You can use

- Q70=22 to set the offset to the values specified in Q94 … Q96
- Q70=23 to set the offset based on current position defined in Q77 … Q79, after this command, Q77 … Q79 as well as Q87 … Q89 should be zero.
- Q70=24 to set the offset to 0 (default)
- Q70=25 to set the offset to 9.294 in y
- Q70=42 to create a rotation matrix based on the angles specified in Q94 … Q96
- Q70=43 to create an identity matrix (default)
- Q70=44 to create a matrix with INFN orientation

3.12 Reversal Error Compensation

If P159 is different than zero, after a movement with Q70=4 a reversal error compensation is performed, which means all six motors move the specified number of steps into the negative direction, and then back to the destination position.

3.13 Calibration Sequence

The calibration sequence of the Spacefab consists of the following steps:

- All six motors move in direction of the home switches
- All motors where the switch is not touched move in positive direction until all switches are touched (closed)
- All six motors move in negative direction out of the limit switches. If a motor moves more than the distance specified in P158 (MaxE1Distance), then the sequence aborts with an error (P80 = $800)
- All six motors move the distance defined in P130 … P135 (HomeOffset) to be at the starting position, here the position counters of the motors are set to zero. These HomeOffset are measured individually for each Spacefab and belong only to this specific Spacefab.
- The kinematics work with the motor position plus the KinematicOffset.

If a motors goes into open loop during these steps, the sequence is also aborted with an error (P80 = $80).

3.14 Spacefab Information Variables

The following variables might be interesting for debugging:

Q252	P10Counter
Q544	INVBaseAngle1
Q545	INVBaseAngle2
Q546	INVBaseAngle3
Q547	INVPlatAngle1
Q548	INVPlatAngle2
Q549	INVPlatAngle3
Q553	INVTransBasedOn0PX
Q554	INVTransBasedOn0PY
Q555	INVTransBasedOn0PZ
Q2584	P10BaseAngle1
Q2585	P10BaseAngle2
Q2586	P10BaseAngle3
Q2587	P10PlatAngle1
Q2588	P10PlatAngle2
Q2589	P10PlatAngle3
Q2590	P10TransBasedOn0PX
Q2591	P10TransBasedOn0PY
Q2592	P10TransBasedOn0PZ

3.15 Coordinate System Variables

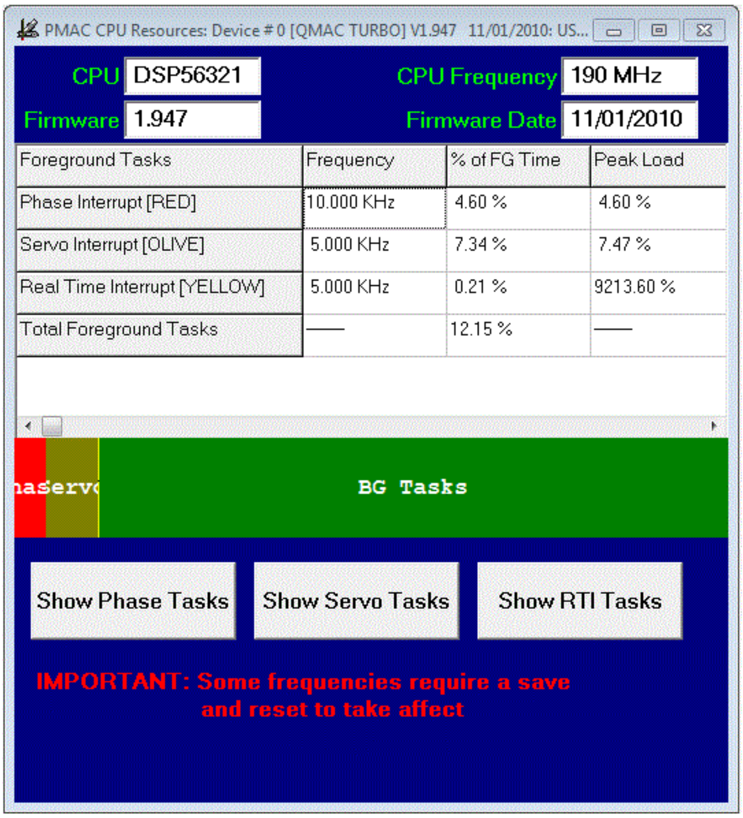
The following variables are used in the coordinate system:

I5213	CS2KinSegTime	10
I5220	CS2LookaheadLength	7 (calculated)
I5250	CS2KinEnable	1
I5287	CS2DefProgAccTime	10
I5288	CS2DefProgSCuTime	100
I5289	CS2DefProgFeedrate	2
I5290	CS2FeedrateTimeUnits	1000
I5298	CS2MaxFeedrate	2

3.16 Timers

I5111	standard delays
I5212	PLC31 Delay

3.17 CPU Resources



GBD8-F3-442-00E00000

4 PMAC Spacefab Background

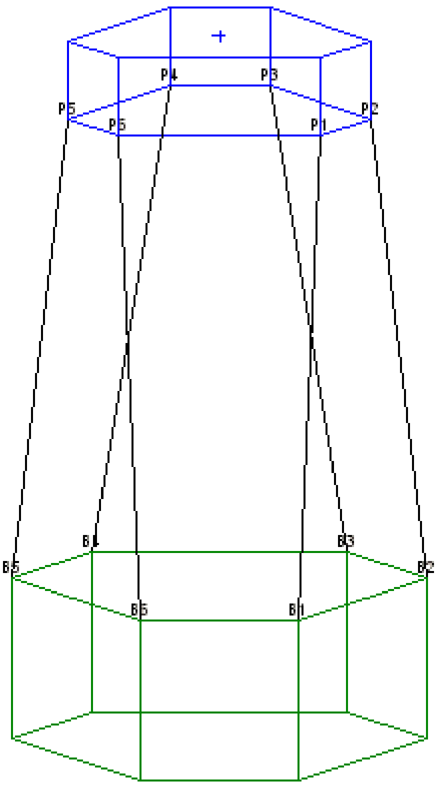
4.1 Kinematic Mode

After the calibration sequence, the kinematic mode is enabled. Here the following three PLCs are used:

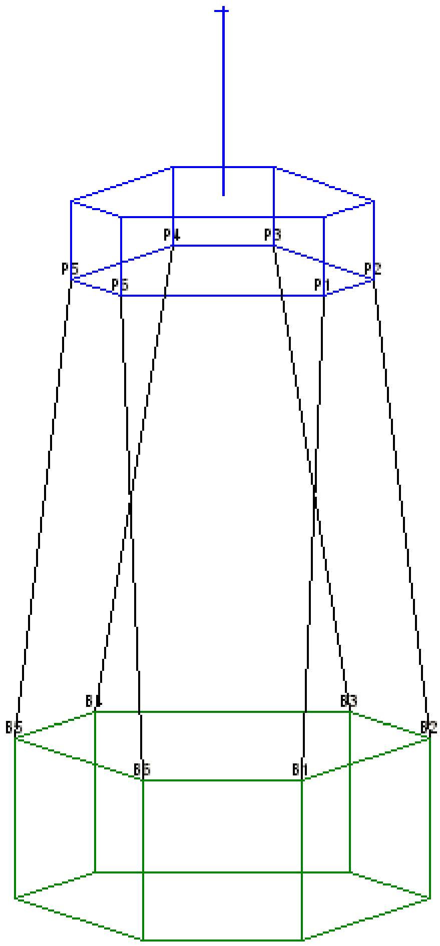
- PLC10 takes the current motor position from M162, … and calculates the degrees of freedom DOF X, Y, Z, A, B and C for the position reporting variables Q81..89. This is done iteratively with a Newton-Raphson algorithm and can take several 100 ms. If the motors are very close to the start position (after calibration), the default DOF positions are used as initial values for the NR, if the motors have been moved some distance, the last result from the NR is used as an initial value for the current NR. In case these calculations get stuck on a bad position, the initial values for the NR can be reset by writing 1 into Q16. The NR usually needs only a few iterations, the number of iterations it took is written to Q93. The last result from the NR in PLC10 is also used in FORWARD as initial value for the NR. PLC10 also does the position check (see below in 4.3) and writes the result into Q92. PLC10 is called continuously as a background task. The frequency of PLC10 can be obtained from Q252.
- FORWARD uses exactly the same mathematical equations like PLC10, but it uses the commanded motor positions from P1..8 to calculate the DOFs for the variables Q1..9. FORWARD is called once when a movement program is started.
- INVERSE calculates for each segment of the move the motor positions P1..8 based on the commanded DOFs Q1..9. It also does the position check and writes the result into Q91. INVERSE is called every 10 ms (segmentation time) during a movement.

	Input	Output	Position check
PLC10	current motor position M162, 262, … last NR position	position report Q81..89 last NR position	yes
FORWARD	commanded motor position P1..8 last NR position	Q1..9	no
INVERSE	Q1..9	P1..8	yes

4.2 *Pivot Point*



a) pivot point at (0 0 0)



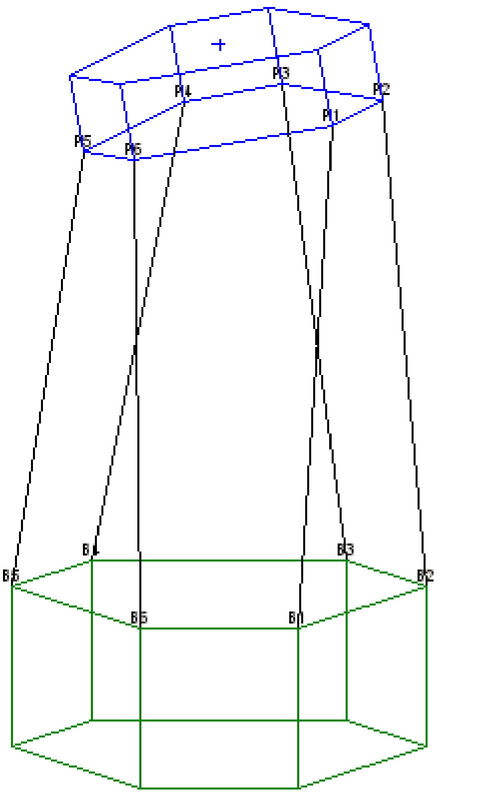
b) pivot point at (0 0 150)

After starting the system, the pivot point is at its default position (0 0 0), which is at the center of the upper surface of the upper platform, see picture a).

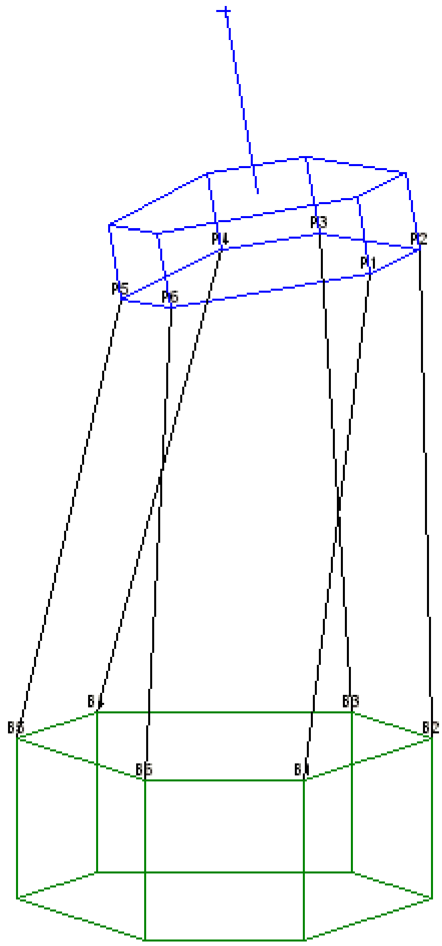
The position of the pivot point can be queried with the command Q70=7 and set with the command Q70=6, using the variables Q94..96.

In picture b) the pivot point has been moved up by 150 mm.

If you rotate the hexapod by 10° around the x axis, then you get the situations shown in pictures c) and d).
Please note that in picture d) the platform has moved much more to the right because it rotates around the much higher pivot point.



c) pivot point at (0 0 0), rotated 10° Rx



d) pivot point at (0 0 150), rotated 10° Rx

The coordinates of the encoders of the motors (which correspond to the lengths of the legs) are continuously transformed to the degrees of freedom x, y, z, Rx, Ry, and Rz of the platform. This transformation depends on the position of the pivot point. So if you change the position of the pivot point while the angles are not 0, the calculated coordinates for x, y, and z will change, too. In the example of picture d), if you move the pivot point back to (0 0 0), then the y coordinate (y axis is to the right) will jump by more than 25 mm.

For picture d) the following y coordinate is calculated:

pivot point	y coordinate
(0 0 150)	0 mm
(0 0 0)	26.05 mm

*) please note that these pictures and numbers have been done with the PAROS II, but the principle is still valid for other Hexapods and SpaceFabs.

The pivot point can be used in two modes:
- moving with platform, then it is specified relatively to the center of the moving platform.
- fixed in space, then it is specified relatively to the platform at the center (start) position.

4.3 Position Check

The kinematic mode uses a segmentation time of 10 ms which means that every 10 ms a trajectory position is calculated. For each trajectory position, the inverse kinematic checks three conditions (with the commanded positions):

- the angles of the joints must be smaller than defined in Q35 / Q36
- the coordinates of the motors must be between the values defined in Q37..40
- the degrees of freedom must be within the softlimits defined in Q240..251

If one of these conditions is not OK, then the movement is stopped, and M5282 (coordinate system 2 run time error) is set to 1. Q91 shows which conditions caused the stop.

The PLC10 position reporting program also performs the position check, but with the current positions instead of the commanded positions. Q92 shows the results of this position check.

Note for the softlimit check:
Since the X Y Z coordinates jump to a different position if you move the pivot point, instead of these coordinates a second set of X Y Z coordinates is used which are based on a zero pivot point.

The softlimits Q240..251 (see 3.10.3) define the maximum travel range.

The maximum ranges for one degree of freedom DOF are only possible when all the other DOFs are not used. For example it is not possible to go to a position y = 13 mm, x = 100 mm. To go to y = 13 mm, the x coordinate has to be smaller.

Please note: The travel range also depends on the position of the pivot point. If you set the pivot point far away from the default position, then the travel ranges shrink.

Without using the lookahead mode (see below in 4.5), usually the mechanics comes to a stop outside the allowed travel range, so the only way of moving back to the allowed travel range is to start the calibration sequence. Therefore it is recommended to use the lookahead mode.

4.4 Speed of the Paros

The speed of the Paros is slightly more complicated than it appears.

The Paros moves in a contouring mode in a six dimensional space. The speed of this movement is one single number and not the single axis speed parameters like for independent single stages. For pure translations, the unit is mm/s, for pure rotations, the unit is °/s. If you move translations and rotations simultaneously, then the unit doesn’t make much sense.

This speed value is set with the variable Q97.

For movements in z direction the maximum speed of the platform is more or less identical with the maximum speed of the motors because the movement of the legs is very similar with the movement of the platform.

For movements in x or y directions, the maximum speed of the platform is much higher (can be a factor of 10 or more) than the maximum speed of the motors because the six legs don’t have to move that much.

Generally speaking: the maximum speed which is possible for a movement depends on the degrees of freedom in which you want to move. A pure translation can be done faster (in mm/s) than a pure rotation (in °/s) with a pivot point which is far away from the upper platform.

The maximum speed also depends on the start point and the end point. A movement in pure x direction around the center can be done faster than a movement in pure x direction around x = 150 mm.

For rotational movements, the maximum speed also depends on the position of the pivot point.

If you set the speed value e.g. 5 times higher than the maximum speed value of the motors, this is absolutely no problem for movements in x and y direction, but for movements in z direction you will get an error.

So you have to set the speed value smaller than the maximum speed value of the motors, but by doing this you waste time for movements in x or y directions. For hexapods without gearhead this is no problem, but for hexapods with a gearhead, this can get boring.

To simplify this, the lookahead mode (see below in 4.5) can be used. In the lookahead mode, Q97 is a constraint instead of a command.

4.5 Lookahead Mode

The lookahead mode is used for two reasons:

- automatic speed handling
- better position check

For the lookahead mode, the following variables are written based on MotorScale, MaxMotorSpeed, and CSKinSegTime: In13, In14, In15, In16, In17, In41, I5220.

“When the lookahead function is enabled, Turbo PMAC will scan ahead in the programmed trajectories, looking for potential violations of its position, velocity, and acceleration limits. If it sees a violation, it will then work backward through the pre-computed buffered trajectories, slowing down the parts of these trajectories necessary to keep the moves within limits. These calculations are completed before these sections of the trajectory are actually executed.” *)

When using the lookahead mode, and the position check finds a problem, the mechanics comes to a stop within the allowed travel range, and you can easily start a new movement.

	without lookahead	with lookahead
speed of platform	constant, defined by Q97	can be changing during the movement
speed of motors	changing during the movement	if one of the motors exceeds its speed limit, the speed of the platform is reduced so that the fastest motor moves with the maximum speed value
violation of travel range	mechanics stops outside of the allowed travel range	mechanics stops inside of the allowed travel range

If you leave Q97 at a small value which is also working without lookahead, then there is no difference between without and with lookahead in the behaviour concerning speed of the platform. In lookahead, you can set Q97 to a higher value, and then the mechanics moves “as fast as possible”.

*) see Turbo PMAC User Manual by Delta Tau