

# Manual

## Software Power PMAC BNL ISS

Author	Jürgen Staud
Revision	1.5
Date	20.5.2016
Project	P15009

## **Content**

1	Warnings .....	4
2	System Description .....	5
2.1	About this manual .....	5
2.2	Abbreviations .....	5
2.3	Introduction .....	6
2.4	Controller .....	7
2.4.1	Power Brick LV IMS .....	7
2.4.2	C-887 .....	7
2.4.3	Controller architecture.....	7
2.5	IP Adresses .....	8
2.6	Overview over Axes .....	8
2.7	Omron Safety Controller G9SP-N20S V2 .....	9
2.8	Kinematics.....	10
3	Communication Description .....	11
3.1	Communication tests .....	11
3.2	First steps.....	11
3.3	Command Variable .....	12
3.4	Communication Variables.....	13
3.5	Error codes .....	15
3.5.1	PMAC Motor Errors.....	15
3.5.2	PMAC Motor Status .....	17
3.5.3	PMAC SoftLimit Status .....	18
3.5.4	PMAC Encoder Errors .....	19
3.5.5	PMAC CS Errors.....	20
3.5.6	PMAC CS Status .....	21
3.5.7	PMAC Global Error .....	22
3.5.8	PMAC System Error .....	23
3.5.9	PMAC IO Info.....	24
3.5.10	Program Errors .....	25
3.5.11	Kinematic Errors .....	26
3.5.12	Parallel Kinematic Softlimit Error .....	27
4	Configuration .....	28
4.1	ECT Configuration.....	28
4.2	Motor Configuration .....	29
4.3	Axis Configuration.....	29
4.4	Configuration Coordinate System.....	30
4.5	Configuration Parallel Kinematics .....	30
4.6	Configuration BNL ISS Math .....	31
5	Background Jobs .....	32
5.1	TCPIP Communication with C-887 controller.....	33
5.2	Housekeeping .....	36
5.3	User Shared Memory .....	36

# Manual

## Software Power PMAC

### BNL ISS



6	EPICS Server.....	37
6.1	EPICS Commands.....	37
6.2	EPICS Motors .....	38
6.3	First steps.....	38
6.4	EPICS command combinations.....	39
6.5	EPICS Motor Status .....	44
7	CPU Resources.....	46

## **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**

### **2.1 *About this manual***

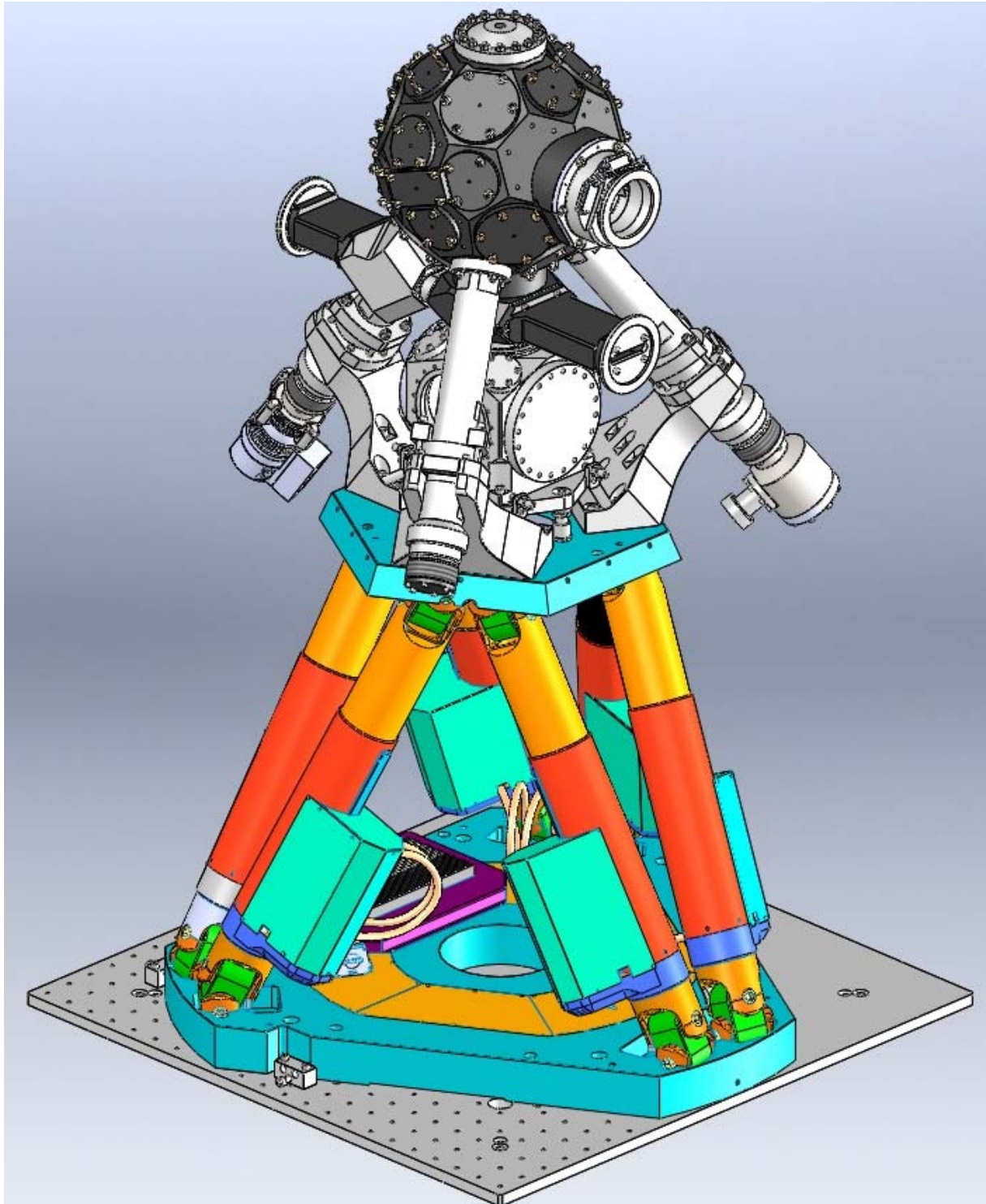
Chapters 3 and 4 of this manual describe the communication with a standard hexapod. The additional EPICS communication layer (which is described in chapter 6) encapsulates most of the standard behaviour. Though the hexapod can still be commanded in the old-fashioned way, most of the content in chapters 3 and 4 can be considered as background information for EPICS users.

### **2.2 *Abbreviations***

CM	Sample Chamber
HX	Hexapod
ISS	Inner Shell Spectroscopy
NEXT	NSLS-II Experimental Tools
NSLS-II	National Synchrotron Light Source II
PCL	Polycapillary Lens
SCHM	Sample Chamber Assembly
SP	Support Structure
WD	Beam Pass-Through Window
XES	X-Ray Emission Spectrometer

### ***2.3 Introduction***

The system architecture consists of a heavy duty parallel kinematic machine which supports the customized monolithic vacuum vessel and the support frame for XES-spectrometer. The vessel's shape is based on a truncated icosahedron for high density of integration. Inside the vessel the Incident PCL Hexapod is placed.



## 2.4 Controller

### 2.4.1 Power Brick LV IMS

Part Number PBL8-JD0-340-8H00000BNL, 0.25A/0.75A amplifier

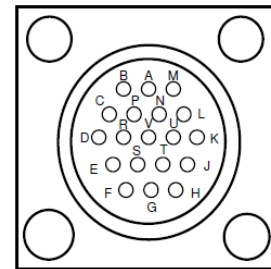
Firmware 2.0.2.14 8. May 2015

IDE 2.1.0.107 18. Feb 2016

CPU PowerPC,APM86xxx

Jumper Limit Switches moved from “Source“ to “Sink”.

In the Abort / STO connector, the pins N and P have to be connected, and pins B and H have to be connected.



Pin #	Symbol	Function	Description
B	Abort	Input	Abort Input 24VDC reference to 0V I/O
H	24V I/O	Output	24V Power Output
N	STO Disable		
P	STO Disable Return		

See also Power\_Brick\_LV\_IMS\_BNL User\_Manual\_Rev\_0.3.pdf.

### 2.4.2 C-887

The following commands can be used to change the IP address of the controller:

IFS 100 IPSTART 0 without DHCP

IFS 100 IPSTART 1 with DHCP

IFS 100 IPADR x.y.z.z:50000

IFS 100 IPMASK 255.255.255.0

RBT

### 2.4.3 Controller architecture

The PowerBrick connects to the C-887 controller with a TCP/IP connection, and sends GCS commands to control the H-850 hexapod.

On the PowerBrick, an EPICS server is running, so you can command all 12 degrees of freedom over the PowerBrick, the C-887 is hidden as a slave controller.

Additionally to the 12 hardware DOFs, another 6 virtual DOFs are available for the special synchronized movements.

## 2.5 IP Adresses

Controller #	IP address	MAC address	Serial No.	SN hexapod
PowerBrick ETH 0	172.20.5.118	00:50:c2:4d:80:41	DTUK0001710	415005906
PowerBrick ETH 1		00:50:c2:4d:70:e1		
PowerBrick RS232	115200 baud			
C-887.532	172.20.5.119	00:c0:08:8a:a4:51	116001423	116000643

## 2.6 Overview over Axes

The Deltatau PowerBrick is used to command the HP-140 directly.

#	Axis	CS	Motor	Gear	Resolution	Feedback	Index	Speed	Travel
1	HP-140	1	DC brush	76:1	155429.3	RS422	no	0.8 mm/s	15 mm
2	HP-140	1	DC brush	76:1	155429.3	RS422	no	0.8 mm/s	15 mm
3	HP-140	1	DC brush	76:1	155429.3	RS422	no	0.8 mm/s	15 mm
4	HP-140	1	DC brush	76:1	155429.3	RS422	no	0.8 mm/s	15 mm
5	HP-140	1	DC brush	76:1	155429.3	RS422	no	0.8 mm/s	15 mm
6	HP-140	1	DC brush	76:1	155429.3	RS422	no	0.8 mm/s	15 mm
7									
8									

HP-140: Gear head 76 : 1  $\rightarrow 512 * 4 * 387283 / 5103 = 155429.2738$  cts/mm

The HP-140 has no forward limit switches (E2).

The PI C-887 controller is used to command the H-850K049 hexapod directly.

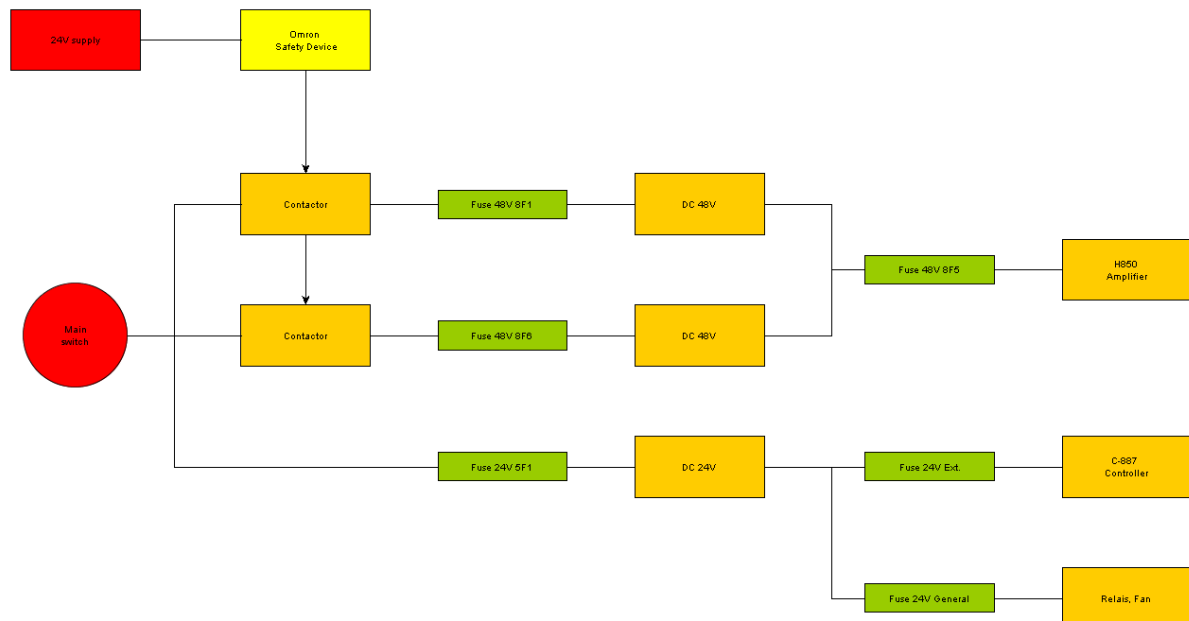
Each degree of freedom has a “zero” position. A command is supplied to send all axes to the “zero” position.

A “start” position is defined, the H-850 moves to -27.3 mm in Y direction, all other DOFs are zero. A command is supplied to send all axes to the “start” position.

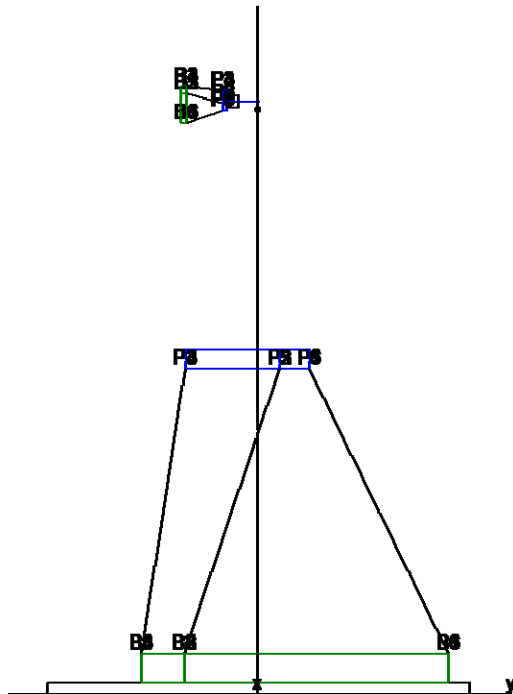


## 2.7 Omron Safety Controller G9SP-N20S V2

Si0	Emergency-Stop Ch2	T1	
Si1	Emergency-Stop Ch1	T0	
Si2	Release E-Stop	T2	Needs to be pressed at least 350 ms to start the system.
Si3	Feedback Main Contactor	T2	Contact welding detection
Si4	Spare		
Si5	Spare		
Si6	Enable-Switch Ch2	T5	Needs to be pressed in service mode
Si7	Enable-Switch Ch1	T4	
Si8	Selector-Switch NC	T3	With the key switch, you can select between work mode and service mode.
Si9	Selector-Switch NO	T3	
Si10	Customer	T1	Needs to be active in work mode
Si11	Customer	T3	
Si12	Light Grid Ch1		Not yet implemented
Si13	Light Grid Ch2		
Si14	Spare		
Si15	Spare		
Si16	Spare		
Si17	Spare		
Si18	Feedback 48V error		
Si19	Feedback 48V enabled		
So0	Main Contactor 1		
So1	Main Contactor 2		
So2	Spare		
So3	Spare		
So4	Lamp Release		Is on when the main contactor is on. It flashes when system cannot be enabled (check emergency stop and enable switch)
So5	Signal-Lamp RD		This beacon is on when the main contactor is off.
So6	Signal-Lamp YE		This beacon is on when system is in service mode.
So7	Signal-Lamp GN		This beacon is on when the system is in work mode and the main contactor is on.



## 2.8 Kinematics



## 3 Communication Description

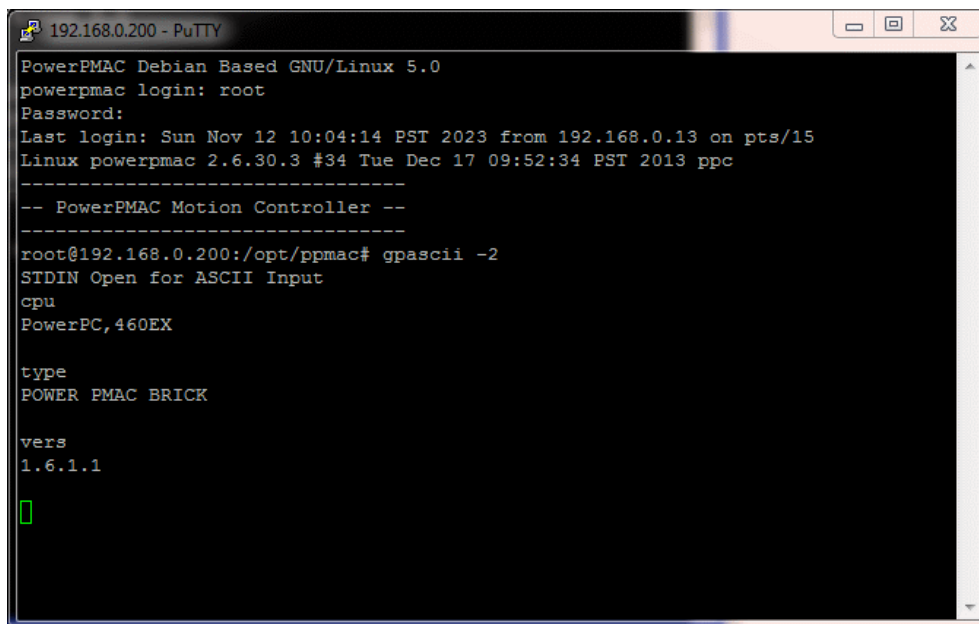
This chapter describes the communication with the DeltaTau Power PMAC controller. For the physical connection, TCP/IP is possible. The communication is based on the Telnet or SSH protocol.

For Windows PCs, the Power PMAC Suite (IDE) can be installed.  
For non-windows PCs, the TCP/IP communication has to be done directly with Telnet.

### 3.1 Communication tests

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

- cpu
- type
- vers



```
192.168.0.200 - PuTTY
PowerPMAC Debian Based GNU/Linux 5.0
powerpmac login: root
Password:
Last login: Sun Nov 12 10:04:14 PST 2023 from 192.168.0.13 on pts/15
Linux powerpmac 2.6.30.3 #34 Tue Dec 17 09:52:34 PST 2013 ppc
-----
-- PowerPMAC Motion Controller --
-----
root@192.168.0.200:/opt/ppmac# gpascii -2
STDIN Open for ASCII Input
cpu
PowerPC,460EX

type
POWER PMAC BRICK

vers
1.6.1.1
█
```

### 3.2 First steps

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

- CommandCode=12                   to start the calibration sequence of the HP-140
- CommandCode=1312               to start the calibration sequence of the H-850

### 3.3 Command Variable

CommandCode is the command variable and is used to start the following functions / programs:

Value	Function	
1	Stop or enable all stages	abort + enable C887 Command = 11
3	start the C background application, see also chapter 5 and 6	system capp1
4	stop the C background application, see also chapter 5 and 6	CAppStop=1
9	Recalculate all matrices	
12	Homing of CS1 (Hexapod HP-140)	5 \$3F CS1Homing
14	Move CS1 to the position specified in &1CSXDestPosX ...	start 1:CS1Move
16	Stop CS1	abort1
17	Power off CS1	disable 1
18	Power on CS1	enable 1
19	Set pivot point to the position specified in CSXPivotpunktX, ...	
20	Get pivot point and write it to CSXPivotpunktX, ...	
1121	Set offset to 0	
1122	Set offset to center height	
1123	Set offset to minimum height	
1134	Simulate Move CS1	
1135	Calculates the 6 encoder values in EncoderPos() back to DOFs	
1312	Homing of slave (Hexapod H-850)	C887 Command = 91
1314	Move slave to the position specified in C887_1DestPos1Send, ...	C887 Command = 87
1316	Stop slave	C887 Command = 11
1317	Power off slave	C887 Command = 31
1318	Power on slave	C887 Command = 21
1351	Dual move calculations, start 1414	
1412	Start homing with all stages simultaneously	5 \$3F CS1Homing C887 Command = 91
1414	Start movement with all stages simultaneously	start 1:CS1Move C887 Command = 87
1416	Stop all stages	abort + enable C887 Command = 11
1417	Power off all stages	disable 1 C887 Command = 31
1418	Power on all stages	enable 1 C887 Command = 21

Do not start movements if the previous movement has not finished yet, always check CSXReady first.

### 3.4 Communication Variables

These variables should be checked periodically from the application software.

CommandCode	see above in 3.3
PMACMotorError0	result of PLC5 Housekeeping, see 3.5.1
PMACMotorStatus0	result of PLC5 Housekeeping, see 3.5.2
PMACSoftLimitStatus0	result of PLC5 Housekeeping, see 3.5.3
PMACEncoderError0	result of PLC5 Housekeeping, see 3.5.4
PMACCCError0	result of PLC5 Housekeeping, see 3.5.5
PMACCSStatus0	result of PLC5 Housekeeping, see 3.5.6
PMACGlobError	result of PLC5 Housekeeping, see 3.5.7
PMACSystemError	result of PLC5 Housekeeping, see 3.5.8
PMACIOInfo0	result of PLC5 Housekeeping, see 3.5.9
&1CSXReady	Shows the coordinate system status of the HP-140, see 3.5.10
&2CSXReady	Shows the coordinate system status of the H850, see 3.5.10
&1CSXSysError	Kinematic error which stopped the movement, see 3.5.11
&1CSXP10Error	Kinematic error concerning the current reported position, see 3.5.11
&1CSXPKSoftLimit	Parallel kinematic softlimit error, see 3.5.12
&1CSXRepPosA	Current reported position, updated by PLC10
&1CSXRepPosB	Current reported position, updated by PLC10
&1CSXRepPosC	Current reported position, updated by PLC10
&1CSXRepPosX	Current reported position, updated by PLC10
&1CSXRepPosY	Current reported position, updated by PLC10
&1CSXRepPosZ	Current reported position, updated by PLC10
&1CSXDestPosA	Destination coordinate used by CommandCode=14
&1CSXDestPosB	Destination coordinate used by CommandCode=14
&1CSXDestPosC	Destination coordinate used by CommandCode=14
&1CSXDestPosX	Destination coordinate used by CommandCode=14
&1CSXDestPosY	Destination coordinate used by CommandCode=14
&1CSXDestPosZ	Destination coordinate used by CommandCode=14
&1CSXDemandSpeed	In mm/s / °/s
&1CSXSoftLimitXMin	Shows the soft limits of the HP-140
&1CSXSoftLimitXMax	Shows the soft limits of the HP-140
&1CSXSoftLimitYMin	Shows the soft limits of the HP-140
&1CSXSoftLimitYMax	Shows the soft limits of the HP-140
&1CSXSoftLimitZMin	Shows the soft limits of the HP-140
&1CSXSoftLimitZMax	Shows the soft limits of the HP-140
&1CSXSoftLimitAMin	Shows the soft limits of the HP-140
&1CSXSoftLimitAMax	Shows the soft limits of the HP-140
&1CSXSoftLimitBMin	Shows the soft limits of the HP-140
&1CSXSoftLimitBMax	Shows the soft limits of the HP-140
&1CSXSoftLimitCMin	Shows the soft limits of the HP-140
&1CSXSoftLimitCMax	Shows the soft limits of the HP-140
&1CSXPivotpunktX	Shows the pivot point of the hexapod, used by CommandCode 19
&1CSXPivotpunktY	Shows the pivot point of the hexapod, used by CommandCode 19

&1CSXPivotpunktZ	Shows the pivot point of the hexapod, used by CommandCode 19
------------------	--

&2CSXReady is created based on information from the C-887 controller.

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

### 3.5 Error codes

#### 3.5.1 PMAC Motor Errors

The following errors can be found in PMACMotorError0, the errors are for motor 1 to 8, the value can be a sum of the errors:

hex value	name	created by
\$00000001	PMACPhase1Error	!(Motor[1].PhaseFound)
\$00000002	PMACPhase2Error	!(Motor[2].PhaseFound)
\$00000004	PMACPhase3Error	!(Motor[3].PhaseFound)
\$00000008	PMACPhase4Error	!(Motor[4].PhaseFound)
\$00000010	PMACPhase5Error	!(Motor[5].PhaseFound)
\$00000020	PMACPhase6Error	!(Motor[6].PhaseFound)
\$00000040	PMACPhase7Error	!(Motor[7].PhaseFound)
\$00000080	PMACPhase8Error	!(Motor[8].PhaseFound)
\$00000100	PMACHome1Error	!(M[1].HomeComplete ) && !(CSXIsHoming)
\$00000200	PMACHome2Error	!(M[2].HomeComplete ) && !(CSXIsHoming)
\$00000400	PMACHome3Error	!(M[3].HomeComplete ) && !(CSXIsHoming)
\$00000800	PMACHome4Error	!(M[4].HomeComplete ) && !(CSXIsHoming)
\$00001000	PMACHome5Error	!(M[5].HomeComplete ) && !(CSXIsHoming)
\$00002000	PMACHome6Error	!(M[6].HomeComplete ) && !(CSXIsHoming)
\$00004000	PMACHome7Error	!(M[7].HomeComplete ) && !(CSXIsHoming)
\$00008000	PMACHome8Error	!(M[8].HomeComplete ) && !(CSXIsHoming)
\$00010000	PMACAmplifier1Error	M[1].FeFatal    M[1].I2tFault    M[1].AmpFault
\$00020000	PMACAmplifier2Error	M[2].FeFatal    M[2].I2tFault    M[2].AmpFault
\$00040000	PMACAmplifier3Error	M[3].FeFatal    M[3].I2tFault    M[3].AmpFault
\$00080000	PMACAmplifier4Error	M[4].FeFatal    M[4].I2tFault    M[4].AmpFault
\$00100000	PMACAmplifier5Error	M[5].FeFatal    M[5].I2tFault    M[5].AmpFault
\$00200000	PMACAmplifier6Error	M[6].FeFatal    M[6].I2tFault    M[6].AmpFault
\$00400000	PMACAmplifier7Error	M[7].FeFatal    M[7].I2tFault    M[7].AmpFault
\$00800000	PMACAmplifier8Error	M[8].FeFatal    M[8].I2tFault    M[8].AmpFault
\$01000000	PMACAltera1Error	BrickLV.Chan[0].OverCurrent
\$02000000	PMACAltera2Error	BrickLV.Chan[1].OverCurrent
\$04000000	PMACAltera3Error	BrickLV.Chan[2].OverCurrent
\$08000000	PMACAltera4Error	BrickLV.Chan[3].OverCurrent
\$10000000	PMACAltera5Error	BrickLV.Chan[4].OverCurrent
\$20000000	PMACAltera6Error	BrickLV.Chan[5].OverCurrent
\$40000000	PMACAltera7Error	BrickLV.Chan[6].OverCurrent
\$80000000	PMACAltera8Error	BrickLV.Chan[7].OverCurrent

The bits “PMACPhaseXError” are set after power on of the controller and show, that the phasing sequence has to be started.

# **Manual**

## **Software Power PMAC**

### **BNL ISS**



The bits „PMACHomeXError“ are set after power on of the controller and show, that the calibration sequence has to be started (CommandCode = 12 / 22).

The bits „PMACAmplifierXError“ should normally not be seen, they show an error. A reset can be tried with CommandCode = 18 / 28.

The bits “PMACAlteraXError” should normally not be seen, they show an error.



### 3.5.2 PMAC Motor Status

The following status information can be found in PMACMotorStatus0, the status information are for motor 1 to 8, the value can be a sum of the information:

hex value	name	created by
\$00000001	Motor 1 moving	!(Motor[1].DesVelZero) && Motor[1].ClosedLoop
\$00000002	Motor 2 moving	!(Motor[2].DesVelZero) && Motor[2].ClosedLoop
\$00000004	Motor 3 moving	!(Motor[3].DesVelZero) && Motor[3].ClosedLoop
\$00000008	Motor 4 moving	!(Motor[4].DesVelZero) && Motor[4].ClosedLoop
\$00000010	Motor 5 moving	!(Motor[5].DesVelZero) && Motor[5].ClosedLoop
\$00000020	Motor 6 moving	!(Motor[6].DesVelZero) && Motor[6].ClosedLoop
\$00000040	Motor 7 moving	!(Motor[7].DesVelZero) && Motor[7].ClosedLoop
\$00000080	Motor 8 moving	!(Motor[8].DesVelZero) && Motor[8].ClosedLoop
\$00000100	Motor 1 OpenLoop	!(Motor[1].AmpEna)    !(Motor[1].ClosedLoop)
\$00000200	Motor 2 OpenLoop	!(Motor[2].AmpEna)    !(Motor[2].ClosedLoop)
\$00000400	Motor 3 OpenLoop	!(Motor[3].AmpEna)    !(Motor[3].ClosedLoop)
\$00000800	Motor 4 OpenLoop	!(Motor[4].AmpEna)    !(Motor[4].ClosedLoop)
\$00001000	Motor 5 OpenLoop	!(Motor[5].AmpEna)    !(Motor[5].ClosedLoop)
\$00002000	Motor 6 OpenLoop	!(Motor[6].AmpEna)    !(Motor[6].ClosedLoop)
\$00004000	Motor 7 OpenLoop	!(Motor[7].AmpEna)    !(Motor[7].ClosedLoop)
\$00008000	Motor 8 OpenLoop	!(Motor[8].AmpEna)    !(Motor[8].ClosedLoop)
\$00010000	Motor 1 MLIM	Gate3[0].Chan[0].MinusLimit
\$00020000	Motor 2 MLIM	Gate3[0].Chan[1].MinusLimit
\$00040000	Motor 3 MLIM	Gate3[0].Chan[2].MinusLimit
\$00080000	Motor 4 MLIM	Gate3[0].Chan[3].MinusLimit
\$00100000	Motor 5 MLIM	Gate3[1].Chan[0].MinusLimit
\$00200000	Motor 6 MLIM	Gate3[1].Chan[1].MinusLimit
\$00400000	Motor 7 MLIM	Gate3[1].Chan[2].MinusLimit
\$00800000	Motor 8 MLIM	Gate3[1].Chan[3].MinusLimit
\$01000000	Motor 1 PLIM	Gate3[0].Chan[0].PlusLimit
\$02000000	Motor 2 PLIM	Gate3[0].Chan[1].PlusLimit
\$04000000	Motor 3 PLIM	Gate3[0].Chan[2].PlusLimit
\$08000000	Motor 4 PLIM	Gate3[0].Chan[3].PlusLimit
\$10000000	Motor 5 PLIM	Gate3[1].Chan[0].PlusLimit
\$20000000	Motor 6 PLIM	Gate3[1].Chan[1].PlusLimit
\$40000000	Motor 7 PLIM	Gate3[1].Chan[2].PlusLimit
\$80000000	Motor 8 PLIM	Gate3[1].Chan[3].PlusLimit

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, CommandCode = 18 / 28 / ... or also 118 / 128 / ... 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).

### 3.5.3 PMAC SoftLimit Status

The following status information can be found in PMACSoftLimitStatus0, the status information are for motor 1 to 8, the value can be a sum of the information:

hex value	created by	name
\$00000001	Motor[1].SoftLimit && Motor[1].SoftLimitDir == 1	the motor has stopped because it hit either its negative software overtravel limit or has noted that it needs to stop the present motion at one of these limits, even if it is presently not in that limit
\$00000002	Motor[2].SoftLimit && Motor[2].SoftLimitDir == 1	
\$00000004	Motor[3].SoftLimit && Motor[3].SoftLimitDir == 1	
\$00000008	Motor[4].SoftLimit && Motor[4].SoftLimitDir == 1	
\$00000010	Motor[5].SoftLimit && Motor[5].SoftLimitDir == 1	
\$00000020	Motor[6].SoftLimit && Motor[6].SoftLimitDir == 1	
\$00000040	Motor[7].SoftLimit && Motor[7].SoftLimitDir == 1	
\$00000080	Motor[8].SoftLimit && Motor[8].SoftLimitDir == 1	
\$00000100	Motor[1].SoftLimit && Motor[1].SoftLimitDir == 0	the motor has stopped because it hit either its positive software overtravel limit or has noted that it needs to stop the present motion at one of these limits, even if it is presently not in that limit
\$00000200	Motor[2].SoftLimit && Motor[2].SoftLimitDir == 0	
\$00000400	Motor[3].SoftLimit && Motor[3].SoftLimitDir == 0	
\$00000800	Motor[4].SoftLimit && Motor[4].SoftLimitDir == 0	
\$00001000	Motor[5].SoftLimit && Motor[5].SoftLimitDir == 0	
\$00002000	Motor[6].SoftLimit && Motor[6].SoftLimitDir == 0	
\$00004000	Motor[7].SoftLimit && Motor[7].SoftLimitDir == 0	
\$00008000	Motor[8].SoftLimit && Motor[8].SoftLimitDir == 0	
\$00010000	Motor[1].SoftMinusLimit	the motor has reached or exceeded its negative software limit as set by Motor[x].MinPos and Motor[x].SoftLimitOffset
\$00020000	Motor[2].SoftMinusLimit	
\$00040000	Motor[3].SoftMinusLimit	
\$00080000	Motor[4].SoftMinusLimit	
\$00100000	Motor[5].SoftMinusLimit	
\$00200000	Motor[6].SoftMinusLimit	
\$00400000	Motor[7].SoftMinusLimit	
\$00800000	Motor[8].SoftMinusLimit	
\$01000000	Motor[1].SoftPlusLimit	the motor has reached or exceeded its positive software limit as set by Motor[x].MaxPos and Motor[x].SoftLimitOffset
\$02000000	Motor[2].SoftPlusLimit	
\$04000000	Motor[3].SoftPlusLimit	
\$08000000	Motor[4].SoftPlusLimit	
\$10000000	Motor[5].SoftPlusLimit	
\$20000000	Motor[6].SoftPlusLimit	
\$40000000	Motor[7].SoftPlusLimit	
\$80000000	Motor[8].SoftPlusLimit	

### 3.5.4 PMAC Encoder Errors

The following errors can be found in PMACEncoderError0, the errors are for motor 1 to 8, the value can be a sum of the errors:

hex value	name	created by
\$00000001	PMACEncLoss1Error	Motor[1].EncLoss
\$00000002	PMACEncLoss2Error	Motor[2].EncLoss
\$00000004	PMACEncLoss3Error	Motor[3].EncLoss
\$00000008	PMACEncLoss4Error	Motor[4].EncLoss
\$00000010	PMACEncLoss5Error	Motor[5].EncLoss
\$00000020	PMACEncLoss6Error	Motor[6].EncLoss
\$00000040	PMACEncLoss7Error	Motor[7].EncLoss
\$00000080	PMACEncLoss8Error	Motor[8].EncLoss
\$00000100		
\$00000200		
\$00000400		
\$00000800		
\$00001000		
\$00002000		
\$00004000		
\$00008000		

### 3.5.5 PMAC CS Errors

The following errors can be found in PMACCSError0, the errors are for CS 1 .. 8, the value can be a sum of the errors:

hex value	name	created by
\$00000001	PMACCS1RunTimeError	Coord[1].RunTimeError
\$00000002	PMACCS2RunTimeError	Coord[2].RunTimeError
\$00000004	PMACCS3RunTimeError	Coord[3].RunTimeError
\$00000008	PMACCS4RunTimeError	Coord[4].RunTimeError
\$00000010	PMACCS5RunTimeError	Coord[5].RunTimeError
\$00000020		
\$00000040		
\$00000080		
\$00000100	PMACCS1LimitStop	Coord[1].LimitStop && !(CSXIsHoming)
\$00000200	PMACCS2LimitStop	Coord[2].LimitStop && !(CSXIsHoming)
\$00000400	PMACCS3LimitStop	Coord[3].LimitStop && !(CSXIsHoming)
\$00000800	PMACCS4LimitStop	Coord[4].LimitStop && !(CSXIsHoming)
\$00001000	PMACCS5LimitStop	Coord[5].LimitStop && !(CSXIsHoming)
\$00002000		
\$00004000		
\$00008000		
\$00010000	PMACCS1ErrorStatus	Coord[1].ErrorStatus != 0
\$00020000	PMACCS2ErrorStatus	Coord[2].ErrorStatus != 0
\$00040000	PMACCS3ErrorStatus	Coord[3].ErrorStatus != 0
\$00080000	PMACCS4ErrorStatus	Coord[4].ErrorStatus != 0
\$00100000	PMACCS5ErrorStatus	Coord[5].ErrorStatus != 0
\$00200000		
\$00400000		
\$00800000		
\$01000000	PMACCS1SoftLimit	Coord[1].SoftLimit
\$02000000	PMACCS2SoftLimit	Coord[2].SoftLimit
\$04000000	PMACCS3SoftLimit	Coord[3].SoftLimit
\$08000000	PMACCS4SoftLimit	Coord[4].SoftLimit
\$10000000	PMACCS5SoftLimit	Coord[5].SoftLimit
\$20000000		
\$40000000		
\$80000000		

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

The bits „PMACCSXLimitStop“ are set if a movement exceeds the allowed travel range of a motor and therefore the movement is stopped.

### 3.5.6 PMAC CS Status

The following status information can be found in PMACCSStatus0, the status information are for CS 1 .. 8, the value can be a sum of the information:

hex value	name	created by
\$00000001	CS1 program running	Coord[1].ProgActive    PLC[CS1Homing].Active
\$00000002	CS2 program running	Coord[2].ProgActive    PLC[CS2Homing].Active
\$00000004	CS3 program running	Coord[3].ProgActive
\$00000008	CS4 program running	Coord[4].ProgActive
\$00000010	CS5 program running	Coord[5].ProgActive
\$00000020		
\$00000040		
\$00000080		
\$00000100	CS1 not homed	!(CSXHasHomed) && Coord[1].NumMotors != 0
\$00000200	CS2 not homed	!(CSXHasHomed) && Coord[2].NumMotors != 0
\$00000400	CS3 not homed	!(CSXHasHomed) && Coord[3].NumMotors != 0
\$00000800	CS4 not homed	!(CSXHasHomed) && Coord[4].NumMotors != 0
\$00001000	CS5 not homed	!(CSXHasHomed) && Coord[5].NumMotors != 0
\$00002000		
\$00004000		
\$00008000		
\$00010000	CS1 not phased	!(CSXHasPhased) && Coord[1].NumMotors != 0
\$00020000	CS2 not phased	!(CSXHasPhased) && Coord[2].NumMotors != 0
\$00040000	CS3 not phased	!(CSXHasPhased) && Coord[3].NumMotors != 0
\$00080000	CS4 not phased	!(CSXHasPhased) && Coord[4].NumMotors != 0
\$00100000	CS5 not phased	!(CSXHasPhased) && Coord[5].NumMotors != 0
\$00200000		
\$00400000		
\$00800000		
\$01000000	CS1 timebase deviation	abs(Coord[1].DesTimeBase / Sys.ServoPeriod - 1) > 0.1 && Coord[1].NumMotors != 0
\$02000000	CS2 timebase deviation	abs(Coord[2].DesTimeBase / Sys.ServoPeriod - 1) > 0.1 && Coord[2].NumMotors != 0
\$04000000	CS3 timebase deviation	abs(Coord[3].DesTimeBase / Sys.ServoPeriod - 1) > 0.1 && Coord[3].NumMotors != 0
\$08000000	CS4 timebase deviation	abs(Coord[4].DesTimeBase / Sys.ServoPeriod - 1) > 0.1 && Coord[4].NumMotors != 0
\$10000000	CS5 timebase deviation	abs(Coord[5].DesTimeBase / Sys.ServoPeriod - 1) > 0.1 && Coord[5].NumMotors != 0
\$20000000		
\$40000000		
\$80000000		

### 3.5.7 PMAC Global Error

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

hex value	created by
\$00000001	Sys.NoClocks
\$00000002	Sys.WDTFault
\$00000004	Sys.HWChangeErr
\$00000008	
\$00000100	Sys.PhaseErrorCtr
\$00000200	
\$00000400	Sys.ServoBusyCtr
\$00000800	Sys.ServoErrorCtr
\$00001000	Sys.RtIntBusyCtr
\$00002000	Sys.RtIntErrorCtr
\$00004000	
\$00008000	

### 3.5.8 PMAC System Error

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

hex value	name	created by
\$00000001	Check switches of motor 1	Motor.pLimits == 0    both switches pressed
\$00000002	Check switches of motor 2	Motor.pLimits == 0    both switches pressed
\$00000004	Check switches of motor 3	Motor.pLimits == 0    both switches pressed
\$00000008	Check switches of motor 4	Motor.pLimits == 0    both switches pressed
\$00000010	Check switches of motor 5	Motor.pLimits == 0    both switches pressed
\$00000020	Check switches of motor 6	Motor.pLimits == 0    both switches pressed
\$00000040	Check switches of motor 7	Motor.pLimits == 0    both switches pressed
\$00000080	Check switches of motor 8	Motor.pLimits == 0    both switches pressed
\$00000100	C-887 H850 not connected	C887_1IsConnected
\$00000200	C-887 undervoltage	
\$00000400		
\$00000800		
\$00001000		
\$00002000		
\$00004000		
\$00008000		
\$00010000		
\$00020000		
\$00040000		
\$00080000		
\$00100000		
\$00200000		
\$00400000		
\$00800000		
\$01000000		BrickLV.BusOverVoltage
\$02000000		BrickLV.BusUnderVoltage
\$04000000		BrickLV.OverTemp
\$08000000	Power Supply Error	BrickLV.BusUnderVoltage   BusOverVoltage
\$10000000	Motion Abort Pressed	PowerBrick[0].GpioData[0].31.1
\$20000000		
\$40000000		
\$80000000	C app not running	CAppRunning = 0

### 3.5.9 PMAC IO Info

The following information can be found in PMACIOInfo0, the value can be a sum of the information:

hex value	created by	name	
\$00000001	PowerBrick[0].GpioData[0].0.1	Input 1	
\$00000002	PowerBrick[0].GpioData[0].1.1	Input 2	
\$00000004	PowerBrick[0].GpioData[0].2.1	Input 3	
\$00000008	PowerBrick[0].GpioData[0].3.1	Input 4	
\$00000010	PowerBrick[0].GpioData[0].4.1	Input 5	
\$00000020	PowerBrick[0].GpioData[0].5.1	Input 6	
\$00000040	PowerBrick[0].GpioData[0].6.1	Input 7	
\$00000080	PowerBrick[0].GpioData[0].7.1	Input 8	
\$00000100	PowerBrick[0].GpioData[0].8.1	Input 9	
\$00000200	PowerBrick[0].GpioData[0].9.1	Input 10	
\$00000400	PowerBrick[0].GpioData[0].10.1	Input 11	
\$00000800	PowerBrick[0].GpioData[0].11.1	Input 12	
\$00001000	PowerBrick[0].GpioData[0].12.1	Input 13	
\$00002000	PowerBrick[0].GpioData[0].13.1	Input 14	
\$00004000	PowerBrick[0].GpioData[0].14.1	Input 15	
\$00008000	PowerBrick[0].GpioData[0].15.1	Input 16	
\$00010000	PowerBrick[0].GpioData[0].16.1	Output 1	
\$00020000	PowerBrick[0].GpioData[0].17.1	Output 2	
\$00040000	PowerBrick[0].GpioData[0].18.1	Output 3	
\$00080000	PowerBrick[0].GpioData[0].19.1	Output 4	
\$00100000	PowerBrick[0].GpioData[0].20.1	Output 5	
\$00200000	PowerBrick[0].GpioData[0].21.1	Output 6	
\$00400000	PowerBrick[0].GpioData[0].22.1	Output 7	
\$00800000	PowerBrick[0].GpioData[0].23.1	Output 8	
\$01000000			
\$02000000			
\$04000000			
\$08000000			
\$10000000			
\$20000000			
\$40000000			
\$80000000	PowerBrick[0].GpioData[0].31.1	Abort	



### 3.5.10 Program Errors

CSXReady can have the following values:

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

The following positive errors can be found in CSXReady:

Code	Decimal	Error	Meaning
\$10	16	Voltage Error	The power supply for the amplifiers failed.
\$20	32	MotionStopError	An activated motion stop button prevents the movement. Please check all motion stop buttons.
\$100	256	Open Loop Error	An axis went unexpectedly into open loop mode.
\$200	512	E1 not found	Movement out of E1 was stopped after MaxE1RausDistance
\$400	1024	Index not found	An index of a measurement system has not been found.
\$800	2048	Axis not homed	The movement is currently not yet allowed, please start the calibration sequence first.
\$2000	8192	Previous move not ready	The previous move is not yet ready
\$4000	16384	E1 not found	Movement into E1 was stopped after MaxE1ReinDistance()
\$10000	65536	E1 not expected	The calibration sequence was aborted because E1 was hit unexpectedly

The following negative errors can be found in CSXReady:

-2	CSX program aborted
-8	CSXSysError
-16	PMAC error, check PMACCSError0, PMACMotorError0

The negative errors are shown only if CSXReady was 0 before.

### 3.5.11 Kinematic Errors

The following errors can be found in CSXSysError or CSXP10Error, the value in the variable can be the sum of the errors:

hex value	name
\$00000001	LegLength Overflow Leg1
\$00000002	LegLength Overflow Leg2
\$00000004	LegLength Overflow Leg3
\$00000008	LegLength Overflow Leg4
\$00000010	LegLength Overflow Leg5
\$00000020	LegLength Overflow Leg6
\$00000040	LegLength Underflow Leg1
\$00000080	LegLength Underflow Leg2
\$00000100	LegLength Underflow Leg3
\$00000200	LegLength Underflow Leg4
\$00000400	LegLength Underflow Leg5
\$00000800	LegLength Underflow Leg6
\$00001000	Max Base Angle Leg1
\$00002000	Max Base Angle Leg2
\$00004000	Max Base Angle Leg3
\$00008000	Max Base Angle Leg4
\$00010000	Max Base Angle Leg5
\$00020000	Max Base Angle Leg6
\$00040000	Max Plat Angle Leg1
\$00080000	Max Plat Angle Leg2
\$00100000	Max Plat Angle Leg3
\$00200000	Max Plat Angle Leg4
\$00400000	Max Plat Angle Leg5
\$00800000	Max Plat Angle Leg6
\$01000000	SoftLimit Error
\$02000000	
\$04000000	matrix singularity
\$08000000	too many iterations
\$10000000	not homed
\$20000000	
\$40000000	
\$80000000	

### **3.5.12 Parallel Kinematic Softlimit Error**

At the beginning of a movement, a movement vector is defined. In case of a kinematic error, this movement vector is copied into CSXPKSoftLimit.

The following errors can be found in CSXPKSoftLimit, the value in the variable can be the sum of the errors:

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		

## **4 Configuration**

We use 5 kHz Servo Cycle, 10 kHz Phase Cycle, 20 kHz PWM frequency

### **4.1 ECT Configuration**

	Encoder	Motor	C.S.	Usage
1	RS422	DC Brush	1	Hexapod
2	RS422	DC Brush	1	
3	RS422	DC Brush	1	
4	RS422	DC Brush	1	
5	RS422	DC Brush	1	
6	RS422	DC Brush	1	
7	RS422	DC Brush	2	
8	RS422	DC Brush	2	

## 4.2 Motor Configuration

Motor[x].	#1	#2	#3	#4	#5	#6	unit	comment
ServoCtrl	1	1	1	1	1	1		
PhaseCtrl	4	4	4	4	4	4		
PowerOnMode	2	2	2	2	2	2		
AbortTa	-150000	-150000	-150000	-150000	-150000	-150000		6.67 mm/s^2
InvAmax	200000	200000	200000	200000	200000	200000		5 mm/s^2
MaxSpeed	0.0008	0.0008	0.0008	0.0008	0.0008	0.0008	mu/ms	
JogSpeed	0.0006	0.0006	0.0006	0.0006	0.0006	0.0006	mu/ms	
JogTa	100	100	100	100	100	100	ms	
JogTs	100	100	100	100	100	100	ms	
PhaseMode	3	3	3	3	3	3		
PhaseOffset	512	512	512	512	512	512		
PhasePosSf	0	0	0	0	0	0		
MaxPos	15	15	15	15	15	15	mu	
MinPos	0	0	0	0	0	0	mu	
FatalFeLimit	0.1	0.1	0.1	0.1	0.1	0.1		

MaxSpeed is not used automatically for jogging moves !  
 See also 5.2 Housekeeping.

## 4.3 Axis Configuration

MaxE1RausDistance = 1  
 MaxE2RausDistance = 1  
 MaxIndexDistance = 2

	(0)	(1)	(2)	(3)	(4)	(5)
SpeedSwitchRein	0.0006	0.0006	0.0006	0.0006	0.0006	0.0006
SpeedSwitchRaus	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
SpeedMoveOffset	0.0006	0.0006	0.0006	0.0006	0.0006	0.0006
IndexOffset	0	0	0	0	0	0
HomeOffset	0.096018	0.098254	0.138007	0.154824	0.167513	0.075319

#### **4.4 Configuration Coordinate System**

	CS1	
Demand Speed	10	
DemandTA	100	
DemandTS	200	

#### **4.5 Configuration Parallel Kinematics**

The following variables are used as parameter for the kinematics of the hexapod:

	H850K049	HP-140
BaseRadius	500	59.5
BaseAngle	7.5	10.65378
PlatRadius	195	30.0
PlatAngle	13	21.51019
PlatHeight	50.36	12.681658
BaseHeight	77.18	14.0
MaxBaseAngle		20.0
MaxPlatAngle		20.0
MIN_LEG_LEN	682	92.5
MAX_LEG_LEN	982	107.5
SoftLimitXMin	-110	-24
SoftLimitXMax	110	24
SoftLimitYMin	-110	-21
SoftLimitYMax	110	21
SoftLimitZMin	-150	-8
SoftLimitZMax	150	8
SoftLimitAMin	-7.5	-16
SoftLimitAMax	7.5	16
SoftLimitBMin	-7.5	-16
SoftLimitBMax	7.5	16
SoftLimitCMin	-7.5	-33
SoftLimitCMax	7.5	33
MAX_ITERATION		20
EpsilonNR		0.00000010
EpsilonGauss		0.00000010
HomeLen		92.5
MinHeight		111.96
MaxHeight		128.04
CenterHeight	872.29888	120.00

#### **4.6 Configuration BNL ISS Math**

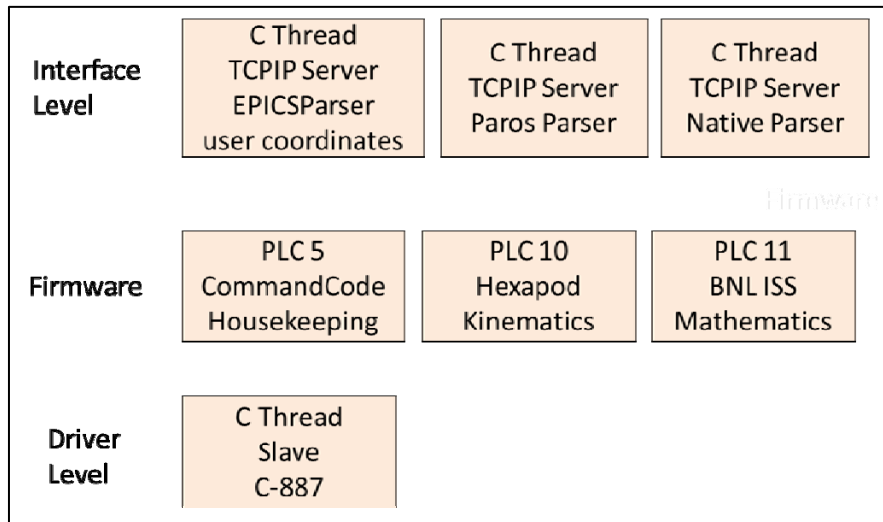
ISS_BasePlate_x	0
ISS_BasePlate_y	0
ISS_BasePlate_z	0
ISS_BasePlate_Rx	0
ISS_BasePlate_Ry	0
ISS_BasePlate_Rz	-90
ISS_BasePlateHeight	30
ISS_H850BaseHeight	77.18
ISS_H850CenterHeight	744.75888
ISS_H850PlatHeight	50.36
ISS_H850PivotX	0
ISS_H850PivotY	0
ISS_H850PivotZ	650
ISS_Bracket_x	$200 + 1.15$
ISS_Bracket_y	0
ISS_Bracket_z	$650 - 0.2$
ISS_Bracket_Rx	0
ISS_Bracket_Ry	-90
ISS_Bracket_Rz	$0 - 0.1776$
ISS_HP140PivotX	0
ISS_HP140PivotY	0
ISS_HP140PivotZ	80
ISS_PayloadDistance	80

## 5 Background Jobs

A C background application starts a thread with slave C-887 communication.

Additionally, several TCPIP servers are started:

- TCPIP server (port 6543) for the EPICS parser (see 6)
- TCPIP server (port 6542) for the Paros parser
- TCPIP server (port 6541) for the native parser



With the Paros parser, you have access to all the variables described in 3.4.

The native parser is a direct access to the Deltatau interface, bypassing the login procedure.

Parallel to the C program, some PLCs are running with housekeeping, command processing, and hexapod and BNL mathematics.



## **5.1 TCPIP Communication with C-887 controller**

A C background application starts a thread for the C-887. This thread essentially runs a state machine in which different information are queried from the C-887 controller.

With the following variables, you have access to these information:

C887_1BGStatus	Shows the state of the state machine
C887_1Freq	Shows the frequency
C887_1IsConnected	Shows if the thread is communicating with the controller
C887_1ActPos1	Shows the position queried with POS?
C887_1ActPos2	Shows the position queried with POS?
C887_1ActPos3	Shows the position queried with POS?
C887_1ActPos4	Shows the position queried with POS?
C887_1ActPos5	Shows the position queried with POS?
C887_1ActPos6	Shows the position queried with POS?
C887_1ErrorAct	Shows the actual error queried with ERR?
C887_1ErrorLast	Shows the last error != 0 queried with ERR? The errors are also shown in plain ASCII in the log file In case of error 5, &2CSXReady is set to \$800 (not homed) In case of error 7, &2CSXReady is set to -8 (CSXSysError)
C887_1Status1	Shows the status queried with SRG?
C887_1Status2	Shows the status queried with SRG?
C887_1Status3	Shows the status queried with SRG?
C887_1Status4	Shows the status queried with SRG?
C887_1Status5	Shows the status queried with SRG?
C887_1Status6	Shows the status queried with SRG?
C887_1DestPos1	Shows the position queried with MOV?
C887_1DestPos2	Shows the position queried with MOV?
C887_1DestPos3	Shows the position queried with MOV?
C887_1DestPos4	Shows the position queried with MOV?
C887_1DestPos5	Shows the position queried with MOV?
C887_1DestPos6	Shows the position queried with MOV?
C887_1Vel1	Shows the system velocity queried with VLS?
C887_1SLMin1	Shows the minimum softlimit queried with TMN?
C887_1SLMin2	Shows the minimum softlimit queried with TMN?
C887_1SLMin3	Shows the minimum softlimit queried with TMN?
C887_1SLMin4	Shows the minimum softlimit queried with TMN?
C887_1SLMin5	Shows the minimum softlimit queried with TMN?
C887_1SLMin6	Shows the minimum softlimit queried with TMN?
C887_1SLMax1	Shows the maximum softlimit queried with TMX?
C887_1SLMax2	Shows the maximum softlimit queried with TMX?
C887_1SLMax3	Shows the maximum softlimit queried with TMX?
C887_1SLMax4	Shows the maximum softlimit queried with TMX?
C887_1SLMax5	Shows the maximum softlimit queried with TMX?
C887_1SLMax6	Shows the maximum softlimit queried with TMX?
C887_1Pivot1	Shows the pivot point queried with SPI?

C887_1Pivot2	Shows the pivot point queried with SPI?
C887_1Pivot3	Shows the pivot point queried with SPI?
C887_FRF1	Shows the referenced value queried with FRF?
C887_FRF2	Shows the referenced value queried with FRF?
C887_FRF3	Shows the referenced value queried with FRF?
C887_FRF4	Shows the referenced value queried with FRF?
C887_FRF5	Shows the referenced value queried with FRF?
C887_FRF6	Shows the referenced value queried with FRF?
C887_DIA1	Shows the diagnosis value queried with DIA?
C887_DIA2	Shows the diagnosis value queried with DIA?
C887_DIA3	Shows the diagnosis value queried with DIA?
C887_DIA4	Shows the diagnosis value queried with DIA?

If C887\_DIA1 is 0 or C887\_DIA2 is 0 and one (or more) axis is in closed loop, then a SVO 0 is send automatically.

Based on those information, the following variables are calculated:

- PMACMotorError1 (home error, amplifier error)
- PMACEncoderError1 (0)
- PMACMotorStatus1 (moving, open loop)
- PMACCSStatus1 (CS1 running)

Bit	15	14	13	12	11	10	9	8
Descrip tion	On Target	Is referen- cing	Is Moving	Servo On	-	-	-	Error flag

Bit	7	6	5	4	3	2	1	0
Descrip tion	Digital Input 4	Digital Input 3	Digital Input 2	Digital Input 1	-	Positive Limit	Reference	Negative Limit

This table shows the meaning of the bits returned with SRG?

# Manual

## Software Power PMAC

### BNL ISS



With the variable C887\_1Command, you can send data to the C-887 controller. Some of the command codes clear the error in the variable C887\_1ErrorLast or set the variable &2CSXReady to 0.

Code	String which is sent	Comment	Error	CSXReady
11	STP 1	Stops the movement		
21	SVO X 1	Turns on the closed loop	= 0	
31	SVO X 0	Turns off the motors, brakes are engaged		
61	SPI R S T	Sets the pivot point defined in C887_1Pivot1Send, C887_1Pivot2Send, C887_1Pivot3Send	= 0	
71	VLS Vel	Vel defined in C887_1Vel1Send	= 0	
81	MOV X Dest	Dest defined in C887_1DestPos1Send	= 0	=0
82	MOV Y Dest	Dest defined in C887_1DestPos2Send	= 0	=0
83	MOV Z Dest	Dest defined in C887_1DestPos3Send	= 0	=0
84	MOV U Dest	Dest defined in C887_1DestPos4Send	= 0	=0
85	MOV V Dest	Dest defined in C887_1DestPos5Send	= 0	=0
86	MOV W Dest	Dest defined in C887_1DestPos6Send	= 0	=0
87	MOV X Y Z U V W	Starts all DOFs at once	= 0	=0
91	FRF X	Starts the homing of the hexapod	= 0	=0

The following variables are used in the command codes mentioned above :

C887_1Vel1Send	System velocity used for the VLS command
C887_1DestPos1Send	Destination used for the MOV commands
C887_1DestPos2Send	Destination used for the MOV commands
C887_1DestPos3Send	Destination used for the MOV commands
C887_1DestPos4Send	Destination used for the MOV commands
C887_1DestPos5Send	Destination used for the MOV commands
C887_1DestPos6Send	Destination used for the MOV commands
C887_1Pivot1Send	Pivot point used for the SPI command
C887_1Pivot2Send	Pivot point used for the SPI command
C887_1Pivot3Send	Pivot point used for the SPI command

## ***5.2 Housekeeping***

The PLC 5 (housekeeping) performs the following actions:

- Calculates RepMotorPos()
- If CheckJogSpeed is 1 → sets Motor[].JogSpeed using Motor[].MaxSpeed
- If CheckKillConditions is 1 and both limit switches are pressed → Axis is killed

## ***5.3 User Shared Memory***

The following elements of the user shared memory are used:

Memory location	Sys.pushm +	Usage
	256	Strings for messaging

## **6 EPICS Server**

The C background application described above starts a TCPIP Server with a parser which can be used to control the whole system.

The server is waiting for connections on port 6543.

### **6.1 EPICS Commands**

The following commands are available:

1	help	Shows a list of all available commands
3	HOMR	Home Reverse
6	STOP	Stop
7	MotorOff	Powers off the axis (not EPICS standard)
8	MotorOn	Powers on the axis (not EPICS standard)
10	REL	Move relatively (not EPICS standard)
13	VELO	Velocity
14	VELO=	Velocity
15	LLS	At Low Limit Switch
16	HLS	At High Limit Switch
17	OFF	User Offset
18	OFF=	User Offset
30	RBV	User Readback Value
31	DRBV	Dial Readback Value
32	VAL	User Desired Value
33	VAL=	User Desired Value
34	DVAL	Dial Desired Value
35	DVAL=	Dial Desired Value
36	DMOV	Done moving to value
37	EGU	Engineering Units
38	MOVN	Motor is moving
41	SET	Set/Use Switch
42	SSET	Set SET Mode
43	SUSE	Set USE Mode
44	SET=	Set/Use Switch
45	SSET=	Set SET Mode
46	SUSE=	Set USE Mode
47	MSTA	Motor Status
48	MSTAStrng	Motor Status (not EPICS standard)

## **6.2 EPICS Motors**

The following degrees of freedom are available:

Offset	DOF
1	H850X
2	H850Y
3	H850Z
4	H850A
5	H850B
6	H850C
7	HP140X
8	HP140Y
9	HP140Z
10	HP140A
11	HP140B
12	HP140C
13	DUALX
14	DUALY
15	DUALZ
16	DUALA
17	DUALB
18	DUALC

## **6.3 First steps**

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

- HP140X HOMR                      to start the calibration sequence of the HP-140
- H850X HOMR                      to start the calibration sequence of the H-850

## 6.4 EPICS command combinations

The following EPICS command combinations are available:

HP140X	RBV	Gets value from CSXRepPosN, N = XYZABC, including the user offset
HP140Y		
HP140Z	DRBV	Gets value from CSXRepPosN, N = XYZABC
HP140A	HOMR	Start homing procedure by sending CommandCode = 12
HP140B	DMOV	Gets the move done information
HP140C	MOVN	Gets the moving information
	MSTA	Gets a motor state (number)
	MSTAString	Gets a motor state (string) (not EPICS standard)
	STOP	Stops axis by sending CommandCode = 16 / 18
	MotorOff	Disable axis by sending CommandCode = 17 (not EPICS standard)
	MotorOn	Enable axis by sending CommandCode = 18 (not EPICS standard)
	VAL	Gets value from CSXDestPosN, N = XYZABC, including the user offset
	DVAL	Gets value from CSXDestPosN, N = XYZABC
	VAL=value VAL value	USE mode : Writes value to CSXDestPosN, and starts movement with CommandCode = 14 SET mode : sets OFF so that VAL shows value
	DVAL=value DVAL value	Writes value to CSXDestPosN, and starts movement with CommandCode = 14
	REL=	Adds value to CSXDestPosN, and starts movement with CommandCode = 14 (not EPICS standard)
	VELO	Gets value from CSXDemandSpeed
	VELO=	Writes value to CSXDemandSpeed
	LLS	Gets bit from PMACMotorStatus0, PMACSoftLimitStatus0 and &1CSXPKSoftLimit.
	HLS	Gets bit from PMACMotorStatus0, PMACSoftLimitStatus0 and &1CSXPKSoftLimit.
	OFF	Gets the position offset from axis n
	OFF=	Sets the position offset for axis n
	EGU	Gets the engineering unit (mm / degree)
	SET	Returns 0 (USE) / 1 (SET)
	SET=	Sets USE (0) or SET (1) mode
	SSET	Returns 0
	SSET=	Switches to SET mode (1)
	SUSE	Returns 0
	SUSE=	Switches to USE mode (0)
H850X	RBV	Gets value from C887_1ActPosN, N = 1 ... 6, including the user offset
H850Y		
H850Z	DRBV	Gets value from C887_1ActPosN, N = 1 ... 6
H850A	HOMR	Start homing procedure by sending CommandCode = 1312
H850B	DMOV	Gets the move done information
H850C	MOVN	Gets the moving information

	MSTA	Gets a motor state (number)
	MSTAString	Gets a motor state (string) (not EPICS standard)
	STOP	Stops axis by sending CommandCode = 1316 / 1318
	MotorOff	Disable axis by sending CommandCode = 1317 (not EPICS standard)
	MotorOn	Enable axis by sending CommandCode = 1318 (not EPICS standard)
	VAL	Gets value from C887_1DestPosN, N = 1 ... 6, including the user offset
	DVAL	Gets value from C887_1DestPosN, N = 1 ... 6
	VAL=value VAL value	USE mode : Writes value to C887_1DestPosNSend, and starts movement with CommandCode = 1314 SET mode : sets OFF so that VAL shows value
	DVAL=value DVAL value	Writes value to C887_1DestPosNSend, and starts movement with CommandCode = 1314
	REL=	Adds value to C887_1DestPosNSend, and starts movement with CommandCode = 1314 (not EPICS standard)
	VELO	Gets value from C887_1Vel1
	VELO=	Writes value to C887_1Vel1Send and sends C887_1Command = 71
	LLS	Gets bit from PMACMotorStatus1 and &2CSXPKSoftLimit
	HLS	Gets bit from PMACMotorStatus1 and &2CSXPKSoftLimit
	OFF	Gets the position offset from axis n
	OFF=	Sets the position offset for axis n
	EGU	Gets the engineering unit (mm / degree)
	SET	Returns 0 (USE) / 1 (SET)
	SET=	Sets USE (0) or SET (1) mode
	SSET	Returns 0
	SSET=	Switches to SET mode (1)
	SUSE	Returns 0
	SUSE=	Switches to USE mode (0)
DUALX DUALY DUALZ DUALA DUALB DUALC	RBV	Gets value from C887_1ActPosN, N = 1 ... 6, including the user offset
	DRBV	Gets value from C887_1ActPosN, N = 1 ... 6
	HOMR	Start homing procedure by sending CommandCode = 1412
	DMOV	Gets the move done information
	MOVN	Gets the moving information
	MSTA	Gets a motor state (number)
	MSTAString	Gets a motor state (string) (not EPICS standard)
	STOP	Stops axis by sending CommandCode = 1416
	MotorOff	Disable axis by sending CommandCode = 1417 (not EPICS standard)
	MotorOn	Enable axis by sending CommandCode = 1418 (not EPICS standard)
	VAL	Gets value from C887_1DestPosN, N = 1 ... 6, including the user offset
	DVAL	Gets value from C887_1DestPosN, N = 1 ... 6
	VAL=value	USE mode : Writes value to C887_1DestPosNSend, and starts



	VAL value	movement with CommandCode = 1351 SET mode : sets OFF so that VAL shows value
	DVAL=value DVAL value	Writes value to C887_1DestPosNSend, and starts movement with CommandCode = 1351
	REL=	Adds value to C887_1DestPosNSend, and starts movement with CommandCode = 1351 (not EPICS standard)
	VELO	Gets value from C887_1Vel1
	VELO=	Writes value to C887_1Vel1Send and sends C887_1Command = 71
	LLS	Gets bit from PMACMotorStatus0, PMACSoftLimitStatus0, &1CSXPKSoftLimit, PMACMotorStatus1 and &2CSXPKSoftLimit
	HLS	Gets bit from PMACMotorStatus0, PMACSoftLimitStatus0, &1CSXPKSoftLimit, PMACMotorStatus1 and &2CSXPKSoftLimit
	OFF	Gets the position offset from axis n
	OFF=	Sets the position offset for axis n
	EGU	Gets the engineering unit (mm / degree)
	SET	Returns 0 (USE) / 1 (SET)
	SET=	Sets USE (0) or SET (1) mode
	SSET	Returns 0
	SSET=	Switches to SET mode (1)
	SUSE	Returns 0
	SUSE=	Switches to USE mode (0)

The following table shows all additional non-EPICS commands:

Stop		Sends CommandCode = 1
Zero		Sends all stages to the zero position, uses CommandCode = 1414
Start		Sends all stages to the start position, uses CommandCode = 1414 (H-850 Z = -27.30)
MSTA		Gets all motor status
RBV		Gets all user readback values
VAL		Gets all user desired values
LLS		Gets all low limit status
HLS		Gets all high limit status
HP140	MotorError	Gets value from PMACMotorError0
	MotorStatus	Gets value from PMACMotorStatus0
	EncoderError	Gets value from PMACEncoderError0
	SystemError	Gets value from PMACSystemError
	SoftLimitStatus	Gets value from PMACSoftLimitStatus0
	CSError	Gets value from PMACCSSError0
	CSStatus	Gets value from PMACCSStatus0
	CS1Ready	Gets value from &1CSXReady
	CS2Ready	Gets value from &2CSXReady
	CS1SysError	Gets value from &1CSXSysError
	CS1P10Error	Gets value from &1CSXP10Error

	CS1PKSoftLimit	Gets value from &1CSXPKSoftLimit
	IOInfo	Gets value from PMACIOInfo0
	Pivot	Gets the current pivot point values
	Pivot=	Sets the current pivot points values with CommandCode = 19
	STOP	Sends CommandCode = 16 / 18
	Zero	Sends all stages to the zero position, uses CommandCode = 14
	Start	Sends all stages to the start position, uses CommandCode = 14
	HOMR	Sends CommandCode = 12
	Encoder	Calculates the 6 encoder values back to DOFs
H850	MotorError	Gets value from PMACMotorError1
	MotorStatus	Gets value from PMACMotorStatus1
	EncoderError	Gets value from PMACEncoderError1
	C887Error	C887_1ErrorLast
	Pivot	Gets the current pivot point values
	Pivot=	Sets the current pivot points values with C887_1Command = 61
	STOP	Sends CommandCode = 1316 / 1318
	Zero	Sends all stages to the zero position, uses CommandCode = 1314
	Start	Sends all stages to the start position, uses CommandCode = 1314 (H-850 Z = -27.30)
	HOMR	Sends CommandCode = 1312
BNLMath	BasePlate_x	Read or write the parameter
	BasePlate_y	Read or write the parameter
	BasePlate_z	Read or write the parameter
	BasePlate_Rx	Read or write the parameter
	BasePlate_Ry	Read or write the parameter
	BasePlate_Rz	Read or write the parameter
	BasePlateHeight	Read or write the parameter
	H850BaseHeight	Read or write the parameter
	H850CenterHeight	Read or write the parameter
	H850PlatHeight	Read or write the parameter
	Bracket_x	Read or write the parameter
	Bracket_y	Read or write the parameter
	Bracket_z	Read or write the parameter
	Bracket_Rx	Read or write the parameter
	Bracket_Ry	Read or write the parameter
	Bracket_Rz	Read or write the parameter
	PayloadDistance	Read or write the parameter
	Parameter	Gets all parameter in the following order: BasePlate_x, BasePlate_y, BasePlate_z, BasePlate_Rx, BasePlate_Ry, BasePlate_Rz, BasePlateHeight, H850BaseHeight, H850CenterHeight, H850PlatHeight, H850PivotX, H850PivotY, H850PivotZ,

		Bracket_x, Bracket_y, Bracket_z, Bracket_Rx, Bracket_Ry, Bracket_Rz, HP140PivotX, HP140PivotY, HP140PivotZ, PayloadDistance
	BasePlateBase	Gets the DOFs and the rot matrix of that plane
	BasePlateAction	Gets the DOFs and the rot matrix of that action
	H850Base	Gets the DOFs and the rot matrix of that plane
	H850Action	Gets the DOFs and the rot matrix of that action
	BracketBase	Gets the DOFs and the rot matrix of that plane
	BracketAction	Gets the DOFs and the rot matrix of that action
	HP140Base	Gets the DOFs and the rot matrix of that plane
	HP140Action	Gets the DOFs and the rot matrix of that action
	PayDistBase	Gets the DOFs and the rot matrix of that plane
	PayDistAction	Gets the DOFs and the rot matrix of that action
	PayloadBase	Gets the DOFs and the rot matrix of that plane
	FORH850Action	Gets the DOFs and the rot matrix of that action
	FORBracketBase	Gets the DOFs and the rot matrix of that plane
	FORHP140Base	Gets the DOFs and the rot matrix of that plane
	FORHP140Action	Gets the DOFs and the rot matrix of that action
	FORPayDistBase	Gets the DOFs and the rot matrix of that plane
	FORPayloadBase	Gets the DOFs and the rot matrix of that plane

## 6.5 EPICS Motor Status

The HP-140 states are determined from PMACMotorError0, PMACMotorStatus0, PMACEncoderError0, and PMACCSStatus0.

The H-850 states are determined from PMACMotorError1, PMACMotorStatus1, PMACEncoderError1, PMACCSStatus1, and C887\_1IsConnected.

The following conditions / errors can be found in the variable MSTA, the value can be a sum of the conditions / errors:

hex value	name	For HP-140 states created by	For H-850 states created by
\$00000001	DIRECTION		
\$00000002	DONE	!(IsMoving0    CSMoving0)	!(IsMoving1    CSMoving1)
\$00000004	PLUS_LS	IsPLIM0	IsPLIM1
\$00000008	HOMELS		
\$00000010	Unused		
\$00000020	POSITION	!IsOpenLoop0	!IsOpenLoop1
\$00000040	SLIP_STALL	AmplError0	AmplError1
\$00000080	HOME		
\$00000100	PRESENT	!EncoderError_0	!EncoderError_1
\$00000200	PROBLEM	AlteraError0   EncoderError_0	AlteraError1   EncoderError_1
\$00000400	MOVING	IsMoving0	IsMoving1
\$00000800	GAIN_SUPPORT	Always on	Always on
\$00001000	COMM_ERR	IsMLIM0 & IsPLIM0	(IsMLIM1 & IsPLIM1)    !IsSlaveConnected
\$00002000	MINUS_LS	IsMLIM0	IsMLIM1
\$00004000	HOMED	!HomeError0	!HomeError1
\$00008000			

hex value	name	For dual move states created by
\$00000001	DIRECTION	
\$00000002	DONE	!(IsMoving0    CSMoving0    IsMoving1    CSMoving1)
\$00000004	PLUS_LS	IsPLIM0    IsPLIM1
\$00000008	HOMELS	
\$00000010	Unused	
\$00000020	POSITION	!IsOpenLoop0 && !IsOpenLoop1
\$00000040	SLIP_STALL	AmplError0    AmplError1
\$00000080	HOME	
\$00000100	PRESENT	!EncoderError_0 && !EncoderError_1
\$00000200	PROBLEM	AlteraError0   EncoderError_0   AlteraError0   EncoderError_1
\$00000400	MOVING	IsMoving0    IsMoving1
\$00000800	GAIN_SUPPORT	Always on
\$00001000	COMM_ERR	(IsMLIM0 & IsPLIM0)    (IsMLIM1 & IsPLIM1)    !IsSlaveConnected
\$00002000	MINUS_LS	IsMLIM0    IsMLIM1

# Manual

## Software Power PMAC

### BNL ISS



\$00004000	HOMED	!HomeError0 && !HomeError1
\$00008000		

From this multi-bit status variable, a string is created which can also be queried. More serious bits overwrite the strings of less serious bits. E.g. if the COMM\_ERR bit is set, all other bits are ignored in this string.

For the PLUS\_LS and MINUS\_LS bits, only the hardware limit switch is used, no kind of softlimit is shown here. LLS and HLS shows both, hard and soft limits.

Condition	Status string
	WAITING
DONE	DONE
! HOMED	NOT HOMED
! POSITION	OPEN LOOP
MOVING	MOVING
SLIP_STALL	SLIP_STALL
PROBLEM	PROBLEM
COMM_ERR	COMM_ERR

# Manual Software Power PMAC BNL ISS



## 7 CPU Resources

