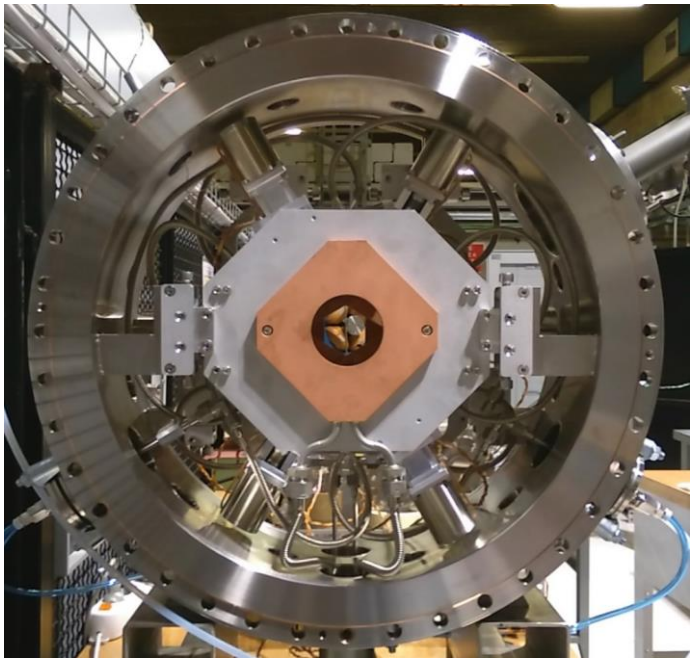
European spallation Source

Documentation

Iris control system



Victor Nadot

September 6, 2017

Abstract

This document has been made to:

- have a quick view of the iris design
- understand the control system
- use and drive it from EPICS
- support it in case of basic issues

Table of contents

Chapter 1

Iris presentation

1.1 Iris goals

The goal of an iris (or diaphragm) is to set the diameter of the beam coming from the source. The iris adjusts light intensity of the beam. It's the same principle in a camera. We can also compare the main of the iris with a resistor: the resistor, in function of its value (geometric shape), adjusts the current which goes through it.

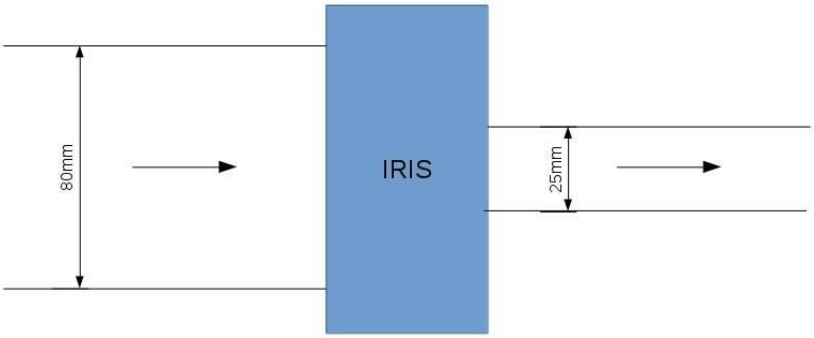


Figure 1.1: goal of the iris, set the diameter of the beam

1.2 ESS specifications

The iris is in LEBT[1], just after the source. Be careful, it's not the job of the iris to set the pulse beam. This is the job of the chopper which is just after the iris. The desired iris aperture (diameter) is between 1 and 80mm. The current after the iris is a function of the aperture. The current is between 6.3mA and 62.5mA. Iris centre is not set. It can be moved thanks to the hardware and the software (see section 1.3.1 page 8).

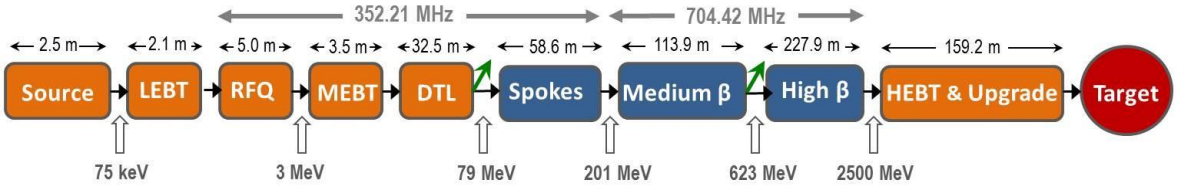


Figure 1.2: ESS line

1.3 Geometry

The diaphragm is composed of 6 blades (orange) which can move independently according to the blue axis. These blades are in a vacuum chamber. The chamber has a diameter of 0,7m.

[1] see appendix the glossary in appendix H page 40

In the figure 1.3, the blades are flat. So the shape of the iris is hexagonal with an aperture between 1 and 80 mm. However, to have an iris's shape as circular as possible, there is another kind of blade which is not flat but triangular. This implicates that the iris's shape is a dodecagon, closer to the circular shape. Because of the geometry of these new blades, the aperture is smaller: from 5 mm to 70mm.

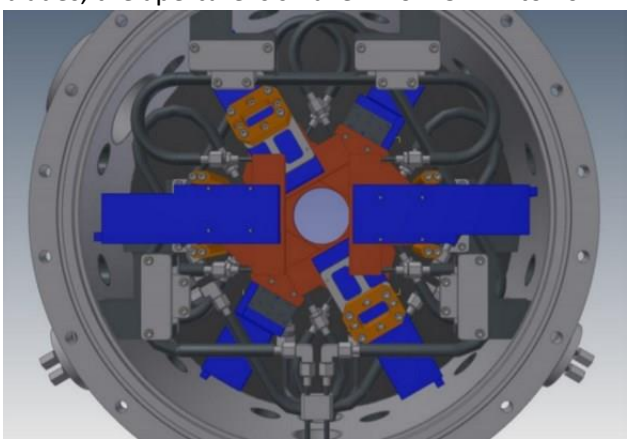


Figure 1.3: iris blades, back of the iris

Each blade is water-cooled. In a water circuit, there are two blades. Moreover, the energy who hits the front of the iris is really powerful. So, at this emplacement, there is a water-cooled shield to protect the iris of high temperature.

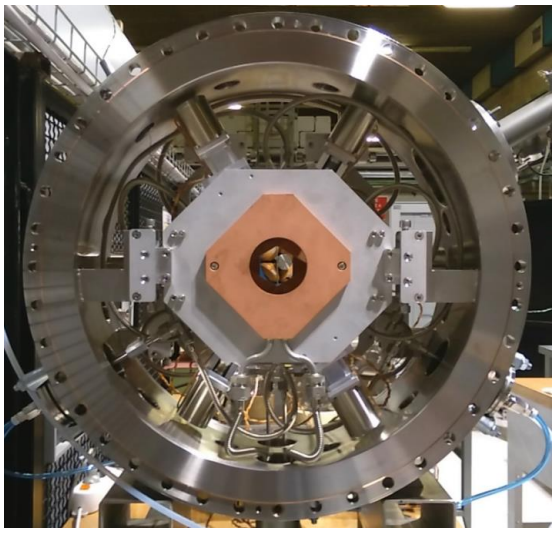


Figure 1.4: iris shield water-cooled (copper piece), front of the iris

Finally, after iris manufacturing, the aperture maximum is 76 mm. The exact travel of motors is 41,66mm. That means that the motor can overtravel of 3,66mm.

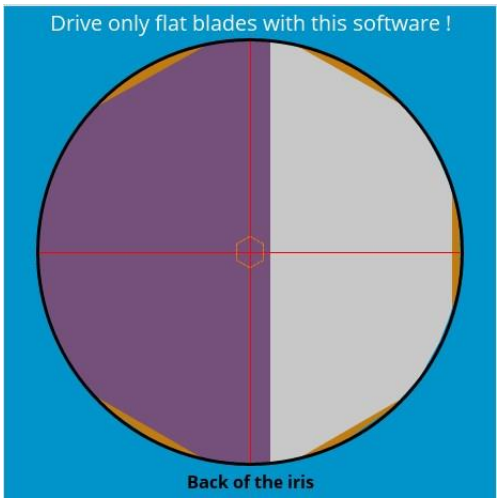


Figure 1.5: blade over travel

The blade is purple because it has hit NLIM switch.

1.3.1 How to decenter the irirs? [ADMIN ONLY]

There are two ways of decentring the iris:

1. hardware: you can move iris center of 2cm according to x and y axis using the screws.

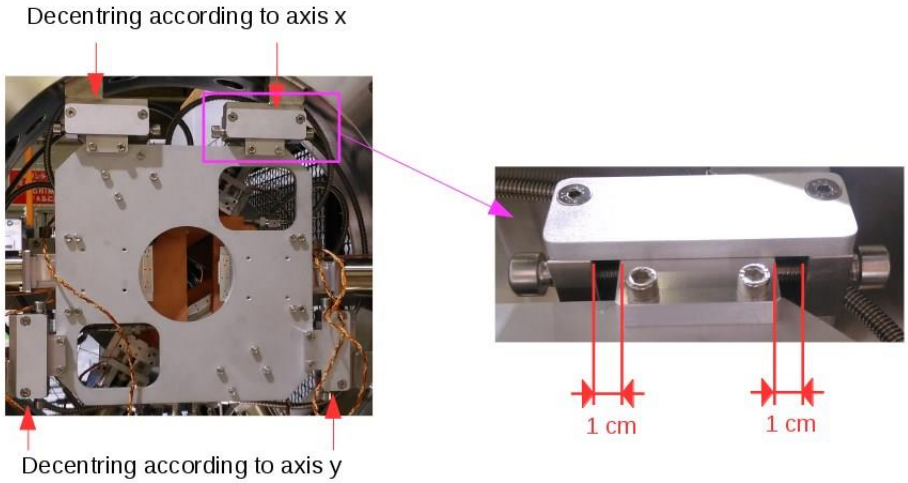


Figure 1.6: hardware iris decentering

2. software: the x and y offset are function of the aperture. Indeed, for example if the aperture is equal to 76 mm (maximum aperture), you can't set an offset. Here is the diagram to show the range values which are possible in function of the aperture. The valide range is between orange and blue curves.

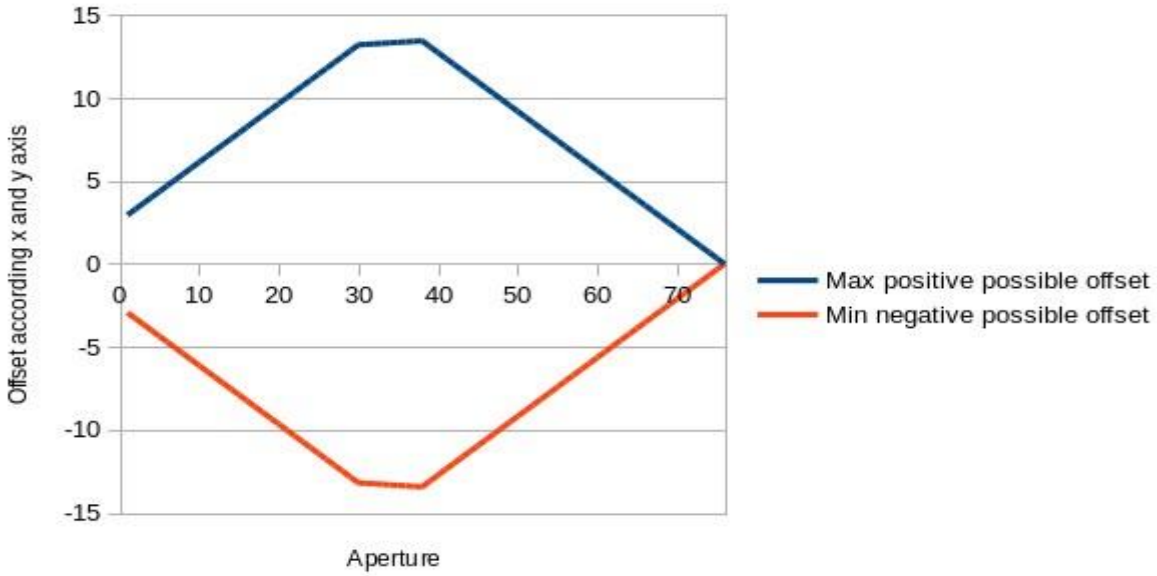


Figure 1.7: range of possible offset in function of the aperture

To decenter the iris, you can use the inputs "x" and "y" in "Admin inputs" panel (see section 3.1 page 15 to learn how to do it properly). Here is an example:

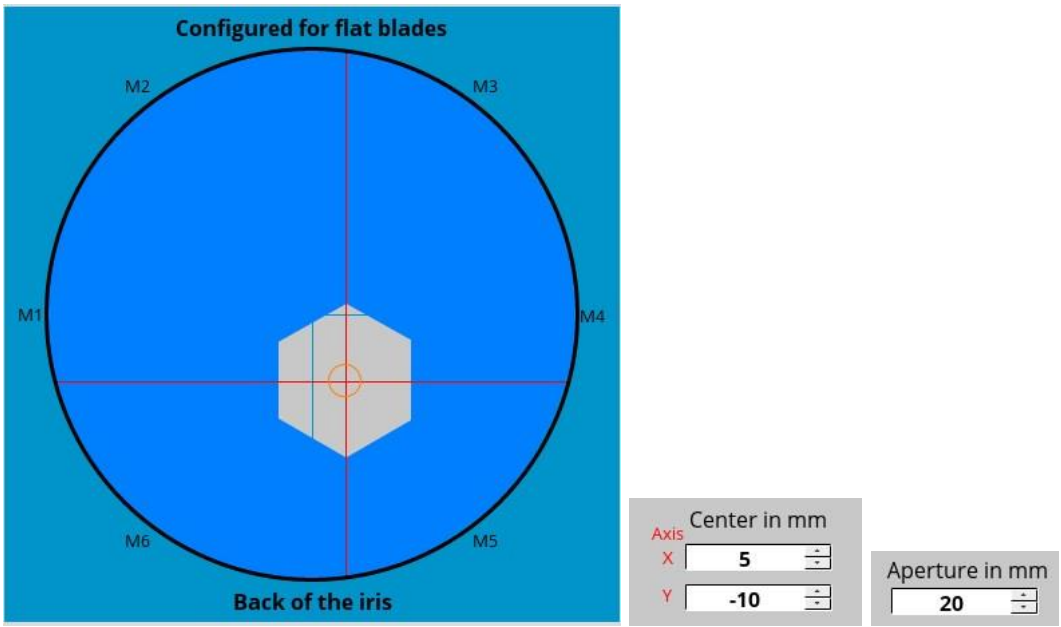


Figure 1.8: iris decenter, 20mm aperture, 5mm offset in x and -10mm in y

The good way to decenter the iris to have it center with the line is to first decenter it with the screws and then to adjust few milimeters, you can use the softwre.

Chapter 2

Control system presentation

2.1 The controller: Geo Brick

There is a tutorial on ICS's Hardware page about this device. Don't hesitate to have a look.

2.1.1 The device

To synchronize the 6 motors, ICS choose to use a Delta Tau's controller: Geo Brick LV-IMS-II. This controller integrates the power supply for the motor and can drive 8 motors (until 32 if you chain several Geo Brick). This device has a TURBO PMAC2 card (Programmable Multi Axis Controller) as brain which is configurable and drivable by TCP/IP (or USB).

Figure 2.1: Geo Brick and its power supply Quad PSU

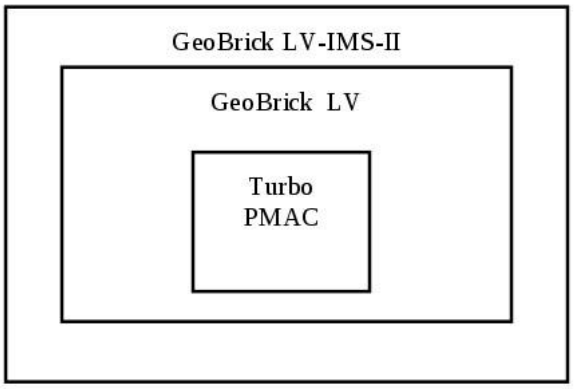


Figure 2.2: turbo PMAC inside the Geo Brick

2.1.2 Features

To use the Geo Brick, you have to configure and program it. To do that, you have to use the variables. There are 4 kinds of variables, for each kind there is 8192 variables:

- I-variables: these are useful to configure the Geo Brick, for example to indicate to the Geo Brick what kind of motors you will use (stepper, brushless, etc.), motor features, etc.
- P-variables: you can use these to stock and process data
- M-variables: mapping variables, makes the link between software and hardware
- Q-variables: (not used in our application) local variables for motion program (see below to have more information about this kind of program).

With the Geo Brick, you can set a Coordinate System (CS) with several motors. This means that the motors will move in function of the other motors. In our application, we want to synchronise six motors. So for this use, it's perfect. There are two kinds of programs:

- PLC program: this kind of program is executed in a fast way and in loop. PLC programs are designed for calculations and actions that are asynchronous to the motion. They are particularly useful for monitoring analogue and digital inputs, setting outputs, sending messages, monitoring motion parameters, issuing commands as if from a host, changing gains, and starting and stopping moves
- Motion program: this program allows you to move a single motor or a CS, for example an iris. Motion programs are Turbo PMAC's chief mechanism for describing the desired motion with the associated math, logic, and I/O operations. They provide a simple, yet powerful and flexible means for describing the motion and operations synchronous to that motion.

2.1.3 Geo Brick programs

You can find in appendix C page 29 the flowchart of Geo Brick programs or the real code in appendix J.1 page 50. There are four PLC programs and one motion program:

- PCL1 protects and clear errors on power-on
- PLC7 sets the coordinate system
- PLC8 is the init procedure, that is what it is doing:
    1. save position
    2. JOG the motors which are in PLIM of 1 mm in direction of NLIM. If PLIM is not anymore on, that means that the motor moved.
    3. JOG all the motors until they reach their PLIM. If after 40 seconds (40cm) of JOG, PLIM is still off, that means that the motor didn't move.
    4. Set all motors positions as maximum position (iris wide open: 76mm of aperture).
    5. save data in the hard memory with a "SAVE" command
    6. come back to the position at the start of PLC8

    This procedure can last until 45 seconds. When the procedure is running, there is a led which is blinking in "User inputs". When it's finished, you passed the "init procedure" if the led "cabling issue" is grey. Otherwise, go to section 4 page 19 to find the solution to get rid of this issue.
- PLC9 : check if there are new order and launch Motion Program 1, if it's the case. It can launch an initialisation move as well.

- PLC10 : stores read back position of motors
- Motion Program 1: move the iris to the wished aperture.

2.2 EPICS driver

You can find the diagram of the driver in appendix D in page 30 or the real code in appendix J.2 page 50. The driver is composed of six files (a xx.db file is a combination of xx.template and xx.substitutions files):

- iris.cmd: ioc file, create the communication with the Geo Brick and launch .db files
- pmacVariables.proto: protocol file to communicate with the Geo Brick
- motor_iris.db : status of the motors and set offset motor by motor (advanced user)
- set_value_pmac.db : set a P-variable
- get_value_pmac.db : get a P-variable
- console.db: console pmac to set or get information of the PMAC. It's useful to debug the control system.

The driver uses "pmac" module which uses "asyn" module to communicate with the *PMAC console*. For example, the driver sends string like "P4801=32" (set the iris aperture at 32 mm) with "set_value_pmac.db" file. It can ask a value as well: "P4831" (motor 1 position) with get_value_pmac.db. Pmac console would return a float value like "20.00012".

2.3 Graphical User Interface

A detailed description of the interface is in appendix F page 35. The things you can do in the interface

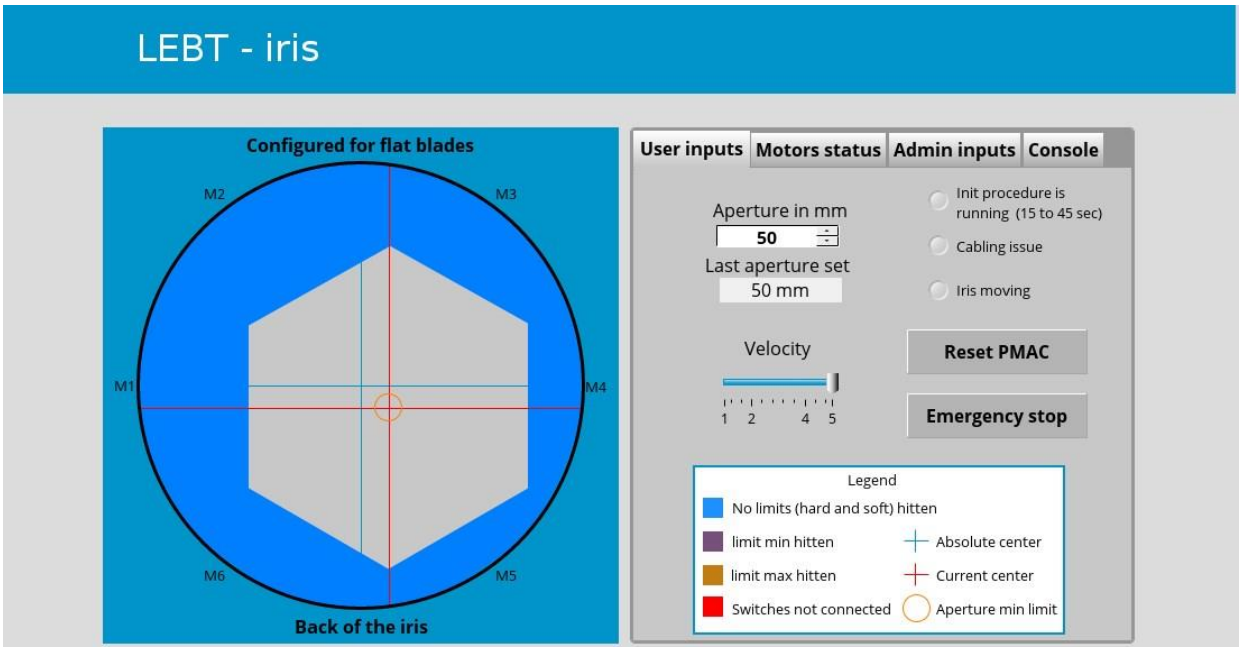


Figure 2.3: GUI, main panel

depends on your knowledge of the general control system and its devices:

- As a user you can have access to :

  - "user inputs" panel. You can set an aperture and velocity. You can also see if the iris is moving (that means that the Geo Brick is running a Motion Program), if an init procedure is running or there is a cabling issue. You can push the "emergency stop" and "reset" buttons as well.
  - "motors status" panel. You can launch an "init procedure", see if motors have reached a hardware limit, if the motor is "in position" or know with which motor is the cabling issue.

- As an advanced user you can:
  - do all the things you can do as a user
  - use the "Admin inputs" panel. You can set an individual offset for each axis. It is useful when you want to calibrate the iris, to make sure its aperture has a regular shape. You can also decenter the iris and choose between driving flat or triangular blades. NB: if you want to choose to drive another kind of blade, you need to open the iris.
  - use "Console" panel. *Use this panel only if you know what you are doing and you are used to deal with the Geo Brick.* It can be useful to debug the control system.

A numerical iris representation is always displayed on the left of the GUI. *Be careful, this is the iris back representation !* Setting an aperture, decentring the iris or launching an init procedure are blocking features. That means that you have to wait for the end of the move before doing something else. If you want to stop the move, you can use the emergency button. Note that a "cabling issue" is blocking as well. You can get a "cabling issue" after running an "init procedure".

2.4 Wiring informations

This section is just general information about the cabling. If you want to wire, please first read this section and then follow the procedure in section 3.1 page 15.

In our architecture, we have 3 devices to connect:

- the computer from which you drive the system
- the Geo Brick controller
- the iris with its motors and switches.

The delicate cabling is between the Geo Brick and the iris. The cabling diagram is page 21, appendix A. The 6 motors are inside the iris, in the vacuum zone. They are connected to the flange with D-sub 15 connectors. The flange is the piece which does the seal between vaccum and atmosphere. Then, there is a cable from there to the Geo Brick. This cable is split in three parts at the end:

- first part goes to AMP connectors (AMP 1 to AMP 6): these wires give the power to the motor
- second part goes to LIM connectors (LIM 1 to LIM 6): these wires tell to the Geo Brick if the motor reached a switch limit
- third part is not connected.

If you want more information about this cable to build it or to understand it, you can have a look at appendix B page 23.

The power supply (Quad PSU), which is under the Geo Brick on the diagram, is connected to the mains with two cables. The Quad PSU can power-on four Geo Brick. Feel free to choose the outputs (2 outputs). In the diagram, it’s outputs four which provides energy to the Geo Brick. The first output "24 I/O 24 LOGIC" is

connected to the connector "24V I/o -24V LOGIC (X2)" of the Geo Brick. The second output "48V DC" is connected to "MOTOR POWER (X1)".

At last, in the front panel, there is an Ethernet connector. It’s thanks to it that we can drive the Geo Brick from the computer.

YOU HAVE TO KEEP IN MIND THAT, IF YOU ARE WRONG ABOUT THE CABLING FOR MOTORS OR LIMITS SWITCHES, CALCULATIONS FOR MOVING MOTORS WOULD BE WRONG AND YOU CAN DAMAGE (OR BREAK) THE IRIS. YOU HAVE TO RESPECT MOTORS NUMERATION WHICH IS IN THE CABLING DIAGRAM PAGE 21.

Chapter 3

How to use the control system to drive the iris?

3.1 Manual

Please, first read sections 2.4 page 13 and section 2.2 page 12.

1. power on the Geo Brick: switch-ON the three switches of the Quad PSU in this order : "LOGIC POWER X" then "I/O POWER X" and then "MOTOR POWER X". If the ethernet cable is not connected, connect it. You should have the same thing as figure 3.1 page 16.
2. In your pc, configure the interface network to talk to the Geo Brick:

   $ ifconfig "name_of_the_interface" 10.10.1.4x netmask 255.255.255.0 up

   Now, you would be able to ping the Geo Brick:

   $ ping 192.6.94.2 #if 10.10.1.40 is the IP of the Geo Brick 64 bytes from 172.16.30.146: icmp_seq=4 ttl=64 time=56 ms

3. start the IOC of the iris:

   $ iocsh /opt/epics/modules/Iris/vnadot/startup/iris.cmd

4. launch the boy of the iris in CSS (you may need to be connected in ssh if it's an IPC and configure CSS to find the PVs).
5. If it's been a while that the iris hasn't been moved, please, launch a "init procedure" to check that the cabling is OK.
6. Now, you are ready to use the GUI. If there are some issues, try section 4 page 19.

Chapter 4

Troubleshooting

DURING MANIPULATION OF CABLES (TO LOOK FOR A BAD CONTACT FOR EXAMPLE) DON'T FORGET TO TURN OFF THE GEO BRICK AND ITS POWER SUPPLY (QUAD PSU) FIRST.

4.1 Control system not responding

It may be that the control system doesn't work. Don't worry, this section is here to help you:

1. check the connection: try to ping the Geo Brick

   $ ping 192.6.94.2 #if 192.6.94.2 is the IP of the Geo Brick
   64 bytes from 172.16.30.146: icmp_seq=4 ttl=64 time=56 ms

   If you are not able to ping it, check that the interface is on with the right IP ($ ifconfig), the ethernet cable is connected, the led 24V LOGIC is green in front of the Geo Brick (if it's not, the cable "24V cables" in the back is bad connected). Sometimes you need to power off then power on the Geo Brick.

2. check the IOC console, is the IOC running?

   $ps -edf || grep iocsh

3. check bad contacts: launch "init procedure" in "motor status" panel. When it's finished (until 45 seconds), if the led "cabling issue" is off in "User inputs" panel, you passed the procedure. If you failed it, go to "motors status" to know which motor is the problem.

4. Reset PMAC: sometimes PMAC crashes. You can reset it whith "reset button" in "User inputs" panel. This button would first save data (motors postions for example) and then reset PMAC.

These are some extra ideas:

1. if input fields are not enabled (you can't set a value in the OPI), you have to run an "init procedure" first, in the "User inputs" panel.

2. check leds on the front of the Geo Brick. 24v LOGIC, 24v I/O, MOTOR POWER, 5V I/O should be green. If no, is the 7-segment "AMP STATUS" displaying something on the front of the Geo Brick? Yes: check the switches of the Quad PSU. They must be ON, ON and RESET and connection cable in the back of the Geo Brick (48V and 24V cables).

3. Maybe the motor has reached software or hardware limits or the last command send to the pmac was already this one. Is your move physically possible? For example a blade could be at a limit (hard or soft). If you ask to the iris to move, this blade could stop the "motion program" and so the iris wouldn't move.

NB: the bad connection can be inside the iris, in the vacuum zone. The Geo brick would move the iris if a (or several) motor power cable (AMPx) is bad connected. However, it would not move the iris if swicthes of one motor are not connected.

4.2 What to do in case of power cut?

There are two cases:

1. Iris was NOT moving when the power cut happend: here is the simple case, just ignore it. You can keep using the GUI. You may have to reconnect the interface with the $ ifconfig command (see section 3.1.2 page 17).

2. Iris was moving when the power cut happend: here is the hard case. The PMAC should be completely crashed. Here is what you have to do:

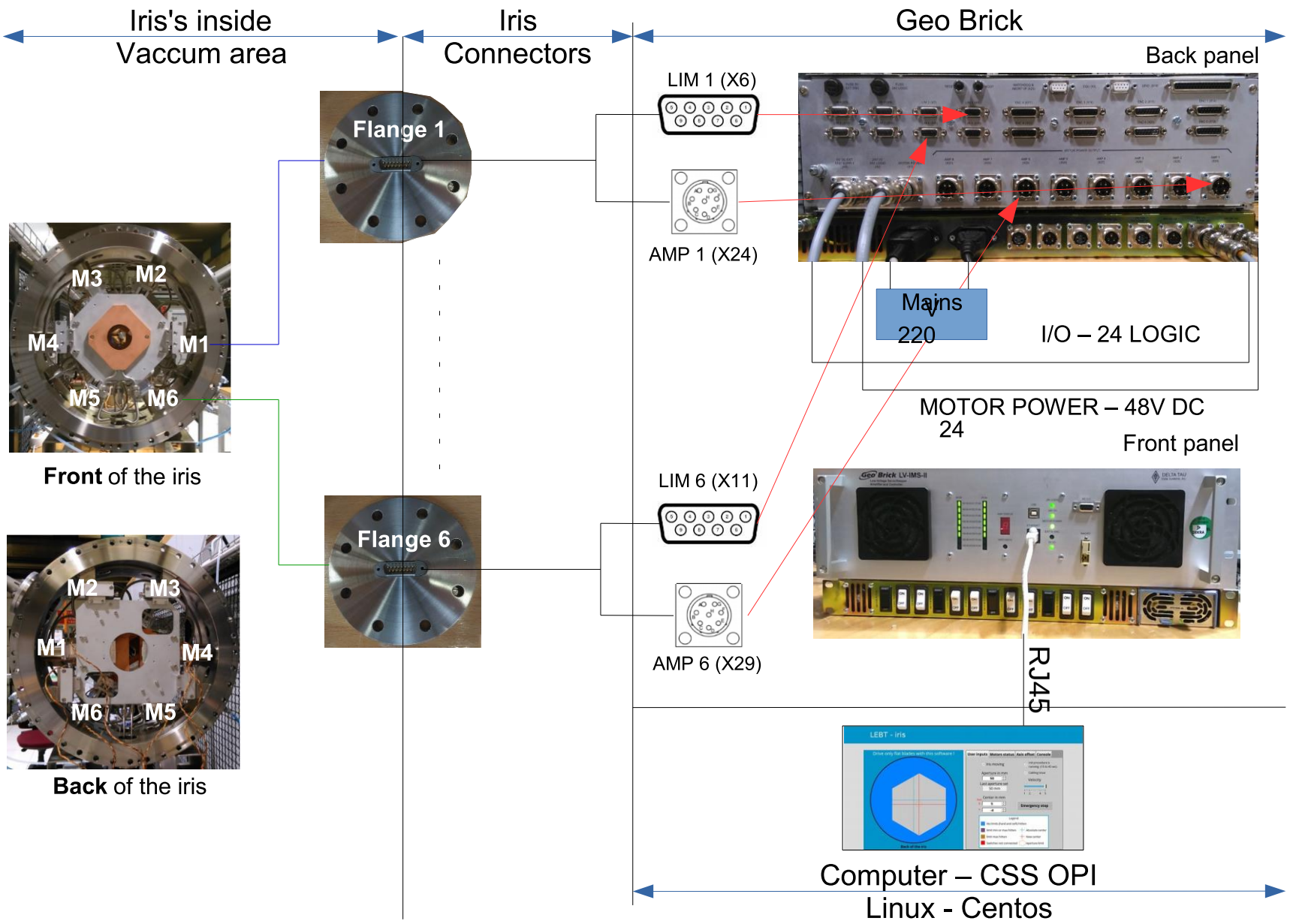
   (a) reset the PMAC doing a Factory Reset: power down the unit then power back up while holding the Reset SW switch down. Release the Reset SW once the unit is powered up. The factory default parameters are now restored from the firmware EEPROM into the active memory. Issue a "SAVE" and a "$$$" to maintain this configuration.

   (b) download again the configuration file. To do that, please follow the procedure section 3g page 17. It may be not work, if it's the case contact Victor Nadot.

Appendix A

Cabling PC - Geo Brick - iris

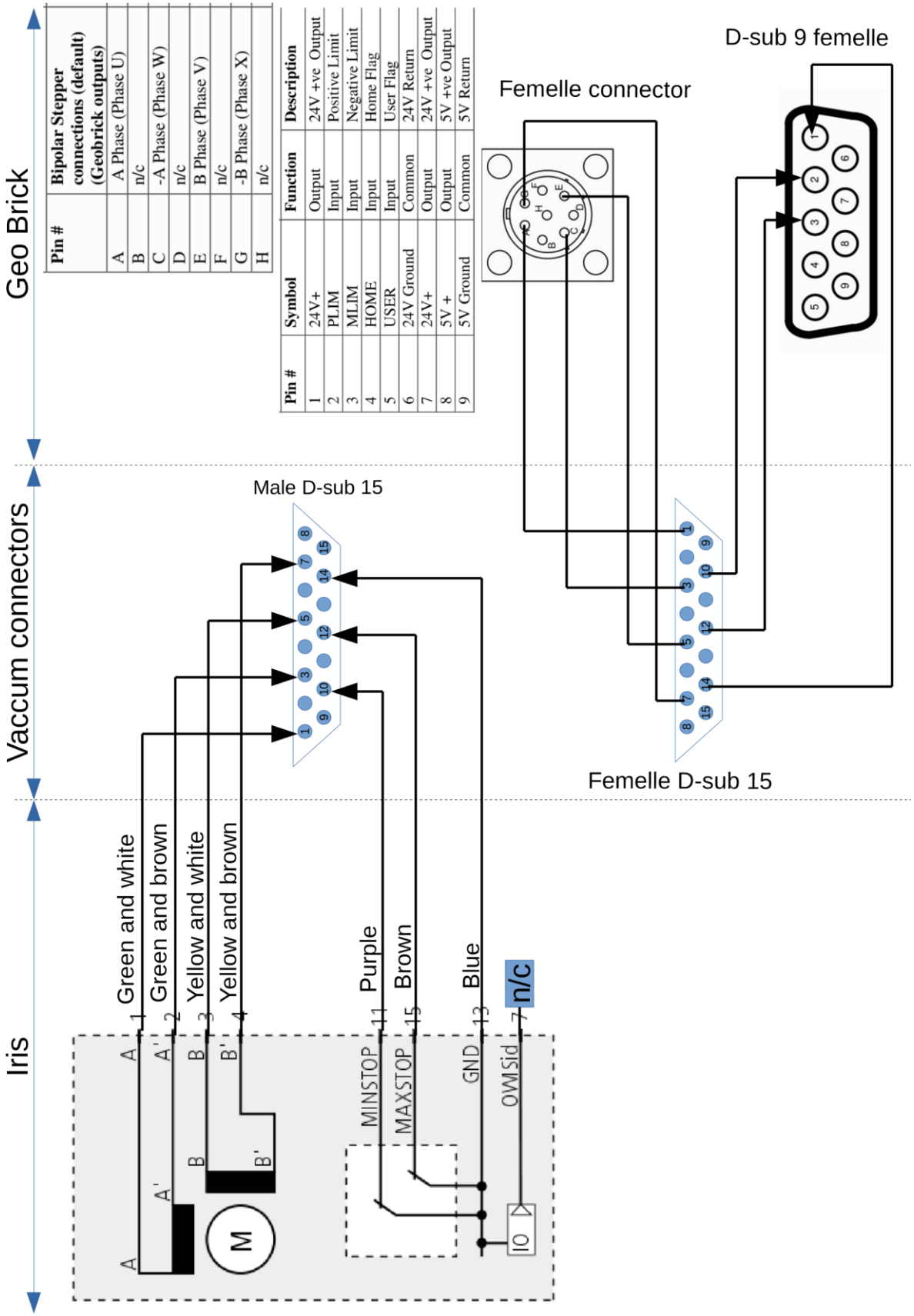
In this appendix, you will find information about the general cabling diagram of the different devices (PC, Geo Brick and iris). It can be useful to connect the system for the first time.


Iris's inside
Vaccum area
Iris
Connectors
Geo Brick
Back panel
LIM 1 (X6)
Flange 1
AMP 1 (X24)
M3
M2
M4
M1
M5
M6
Front of the iris
Mains
220
I/O – 24 LOGIC
MOTOR POWER – 48V DC
24
Front panel
LIM 6 (X11)
Flange 6
AMP 6 (X29)
M2
M3
M1
M4
M6
M5
Back of the iris
RJ45
LEBT - iris
Computer – CSS OPI
Linux - Centos

Appendix B

Electrical cabling

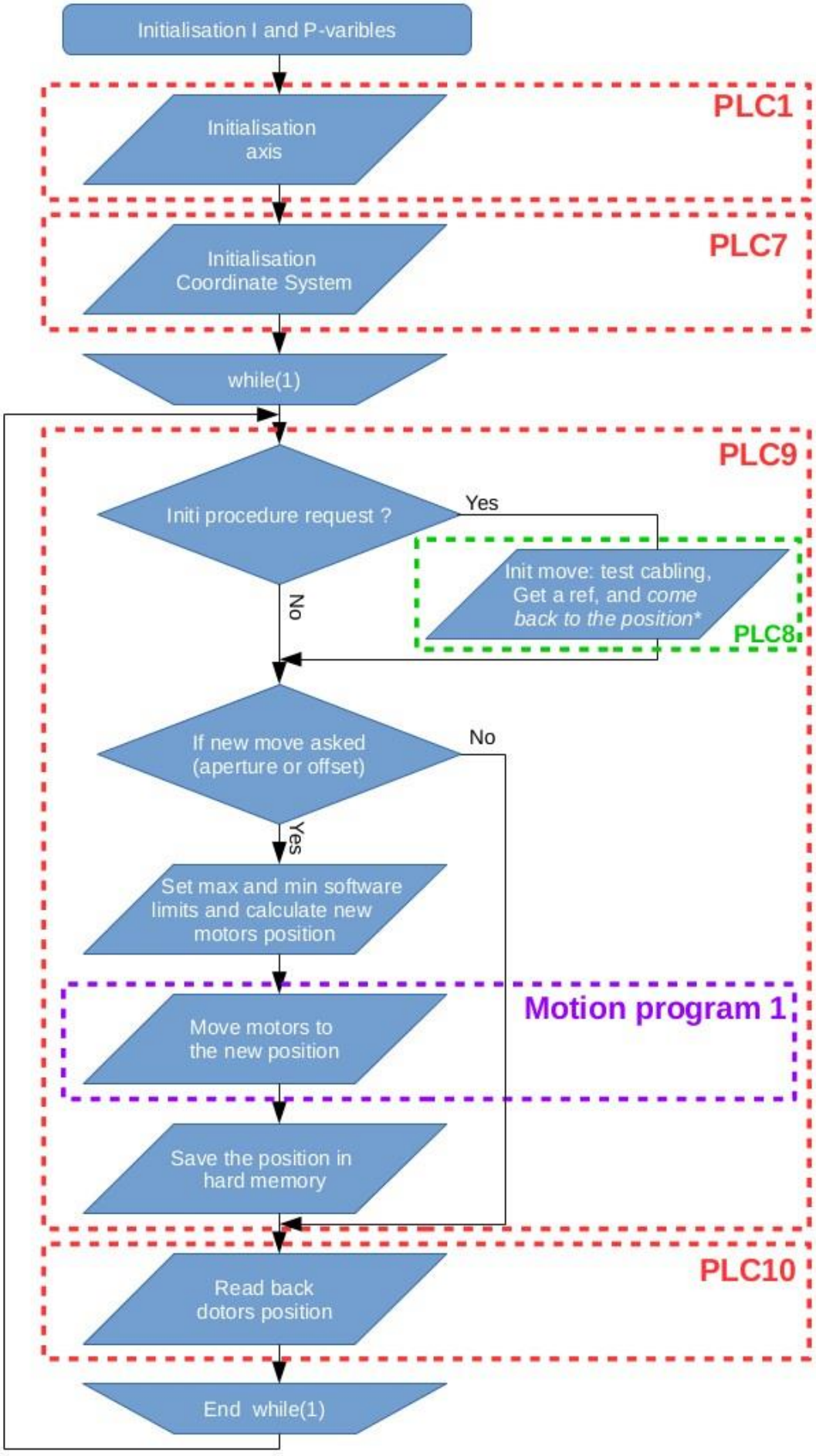
In this appendix, you will find electrical information about the cable between the Geo Brick and the iris. It can be useful if you want to build one or repair one.

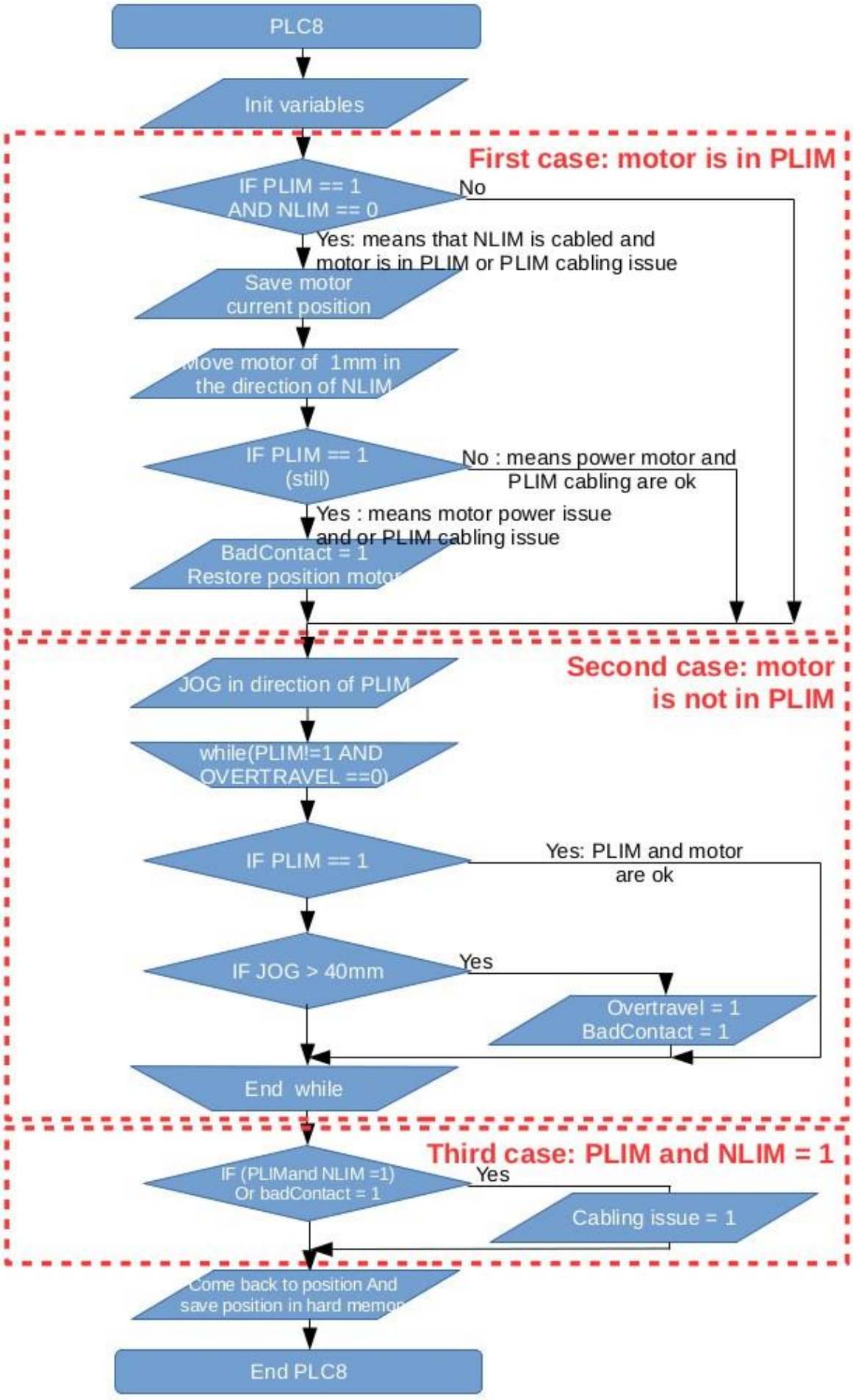

Geo Brick
Vaccum connectors
Iris
Pin # | Bipolar Stepper connections (default) (Geobrick outputs)
A | A Phase (Phase U)
B | n/c
C | -A Phase (Phase W)
D | n/c
E | B Phase (Phase V)
F | n/c
G | -B Phase (Phase X)
H | n/c
Pin # | Symbol | Function | Description
1 | 24V+ | Output | 24V +ve Output
2 | PLIM | Input | Positive Limit
3 | MLIM | Input | Negative Limit
4 | HOME | Input | Home Flag
5 | USER | Input | User Flag
6 | 24V Ground | Common | 24V Return
7 | 24V+ | Output | 24V +ve Output
8 | 5V + | Output | 5V +ve Output
9 | 5V Ground | Common | 5V Return
D-sub 9 femelle
Femelle connector
Male D-sub 15
Femelle D-sub 15
Green and white
Green and brown
Yellow and white
Yellow and brown
Purple
Brown
Blue
n/c
MINSTOP
MAXSTOP
GND
OWSid
M
IO

Appendix C

Flowchart program Geo Brick

In this appendix, you will find the flowchart of the program inside the Geo Brick. The PLC8 (init procedure) is also detailed

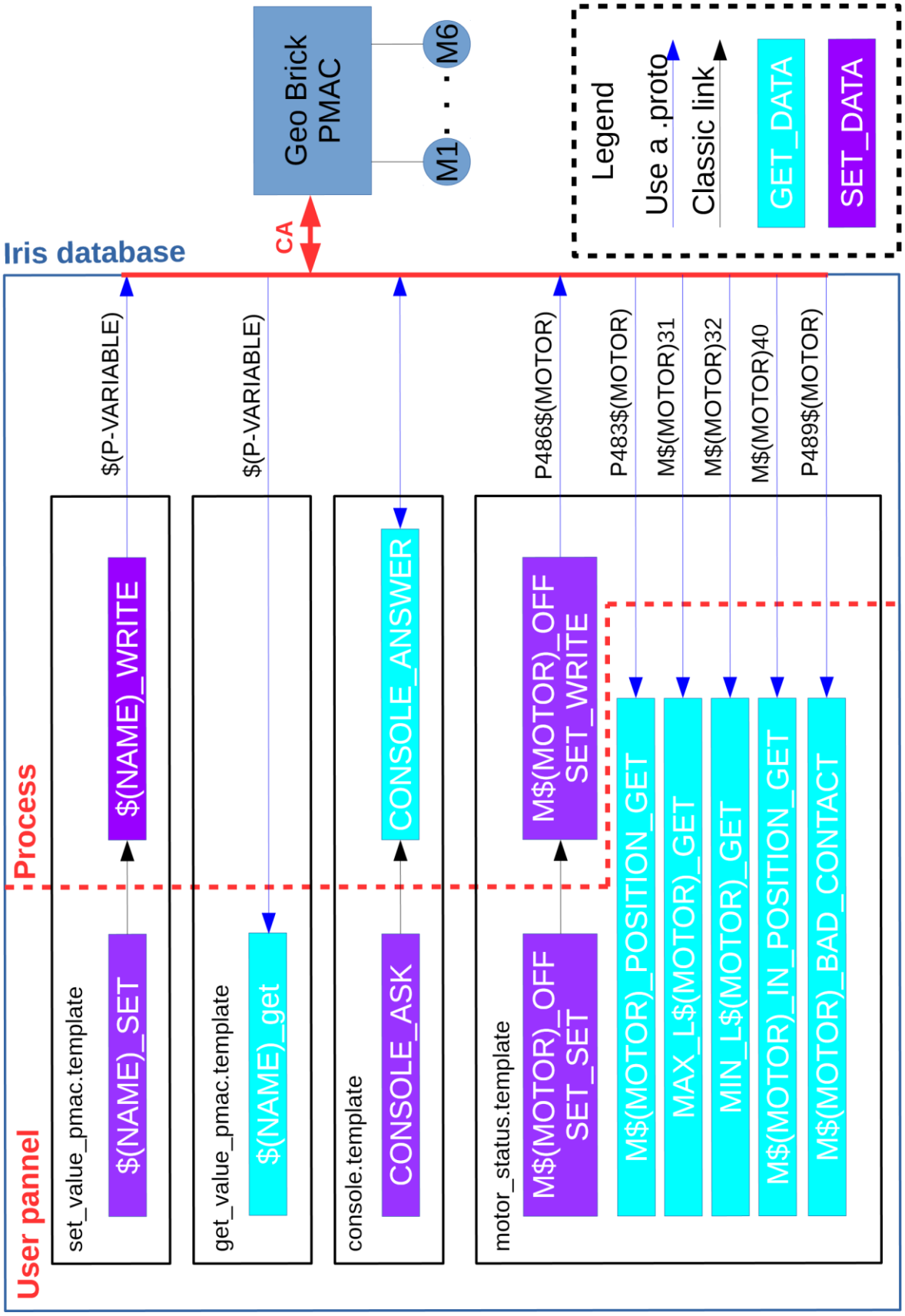

Initialisation I and P-varibles
PLC1
Initialisation axis
PLC7
Initialisation Coordinate System
while(1)
PLC9
Initi procedure request ?
Yes
No
Init move: test cabling, Get a ref, and come back to the position*
PLC8
If new move asked (aperture or offset)
No
Yes
Set max and min software limits and calculate new motors position
Motion program 1
Move motors to the new position
Save the position in hard memory
PLC10
Read back dotors position
End while(1)


PLC8
Init variables
First case: motor is in PLIM
IF PLIM == 1
AND NLIM == 0
No
Yes: means that NLIM is cabled and
motor is in PLIM or PLIM cabling issue
Save motor
current position
Move motor of 1mm in
the direction of NLIM
IF PLIM == 1
(still)
No : means power motor and
PLIM cabling are ok
Yes : means motor power issue
and or PLIM cabling issue
BadContact = 1
Restore position motor
Second case: motor
is not in PLIM
JOG in direction of PLIM
while(PLIM!=1 AND
OVERTRAVEL ==0)
IF PLIM == 1
Yes: PLIM and motor
are ok
IF JOG > 40mm
Yes
Overtravel = 1
BadContact = 1
End while
Third case: PLIM and NLIM = 1
IF (PLIMand NLIM =1)
Or badContact = 1
Yes
Cabling issue = 1
Come back to position And
save position in hard memor
End PLC8

Appendix D

EPICS’s driver

In this appendix, you will find a diagram of the iris EPICS database. It can be useful to understand the driver.


Geo Brick
PMAC
M6
M1
CA
Legend
Use a .proto
Classic link
GET_DATA
SET_DATA
Iris database
Process
User pannel
set_value_pmac.template
$(NAME)_SET
$(NAME)_WRITE
$(P-VARIABLE)
get_value_pmac.template
$(NAME)_get
$(P-VARIABLE)
console.template
CONSOLE_ASK
CONSOLE_ANSWER
motor_status.template
M$(MOTOR)_OFF
SET_SET
M$(MOTOR)_OFF
SET_WRITE
P486$(MOTOR)
M$(MOTOR)_POSITION_GET
P483$(MOTOR)
MAX_L$(MOTOR)_GET
M$(MOTOR)31
MIN_L$(MOTOR)_GET
M$(MOTOR)32
M$(MOTOR)_IN_POSITION_GET
M$(MOTOR)40
M$(MOTOR)_BAD_CONTACT
P489$(MOTOR)

Here are the fields in the substitutions file :

set_value_pmac.substitutions	
$(NAME)	$(P-VARIABLE)
INIT	P4800
APERTURE	P4801
VELOCITY	P4803
OFFSET_X	P4807
OFFSET_Y	P4808
BLADES_KIND	P4838

get_value_pmac.substitutions	
$(NAME)	$(P-VARIABLE)
INIT_PROCESSING	P4800
LAST_COMMAND	P4805
APERTURE_MIN	P4829
INIT_PROCEDURE_DONE	P4837
CABLING_ISSUE	P4889
IRIS_MOVING	M5280

motor_satus.substitutions
$(MOTOR)
1
2
3
4
5
6

Appendix E

Link between Geo Brick and EPICS variables

Here is the table to define which EPICS variable (PV) match with Geo Brick variable (P-variable).

Name in EPICS	Description	P-variables	Name in Geo Brick
$(P)_$(M):INIT_PROCESSING_GET	initialisation position motor	P4800	initialisation
$(P)_$(M):INIT_SET	initialisation position motor	P4800	initialisation
$(P)_$(M):APERTURE_SET	Aperture in mm	P4801	newAperture_mm
	Aperture in counts	P4802	newAperture_counts
$(P)_$(M):VELOCITY_SET	Velocity scale 1 to 5 (1 slow, 5 fast)	P4803	velocity
	indicates if a MP has been launched	P4804	motionProgramHasBeeenLauched
$(P)_$(M):LAST_COMMAND_GET	Last commanded aperture in mm	P4805	lastPosition_mm
	Last commanded aperture in counts	P4806	lastPosition_counts
$(P)_$(M):OFFSET_X_SET	offset in mm axis X to decenter the iris	P4807	offsetX_mm
$(P)_$(M):OFFSET_Y_SET	offset in mm axis Y to decenter the iris	P4808	offsetY_mm
	last offset in x in mm	P4809	lastOffsetX_mm
	last offset in y in mm	P4810	lastOffsetY_mm
	Position for motor 1 in counts	P4811	positionMtr1_counts
	Position for motor 2 in counts	P4812	positionMtr2_counts
	Position for motor 3 in counts	P4813	positionMtr3_counts
	Position for motor 4 in counts	P4814	positionMtr4_counts
	Position for motor 5 in counts	P4815	positionMtr5_counts
	Position for motor 6 in counts	P4816	positionMtr6_counts
	temporary variable	P4817	temp1
	temporary variable	P4818	temp2
	temporary variable	P4819	temp3
	temporary variable	P4820	temp4
	Read back value motor 1 in counts	P4821	positionRBmtr1_counts
	Read back value motor 2 in counts	P4822	positionRBmtr2_counts
	Read back value motor 3 in counts	P4823	positionRBmtr3_counts
	Read back value motor 4 in counts	P4824	positionRBmtr4_counts
	Read back value motor 5 in counts	P4825	positionRBmtr5_counts
	Read back value motor 6 in counts	P4826	positionRBmtr6_counts
	temporary variable	P4827	temp5
	temporary variable	P4828	temp6
$(P)_$(M):APERTURE_MIN_GET	aperture min	P4829	apertureMin_mm
		:	
$(P)_$(M):POSITION_M1_GET	Read back value motor 1 in mm	P4831	positionRBmtr1_mm
$(P)_$(M):POSITION_M2_GET	Read back value motor 2 in mm	P4832	positionRBmtr2_mm
$(P)_$(M):POSITION_M3_GET	Read back value motor 3 in mm	P4833	positionRBmtr3_mm
$(P)_$(M):POSITION_M4_GET	Read back value motor 4 in mm	P4834	positionRBmtr4_mm
$(P)_$(M):POSITION_M5_GET	Read back value motor 5 in mm	P4835	positionRBmtr5_mm
$(P)_$(M):POSITION_M6_GET	Read back value motor 6 in mm	P4836	positionRBmtr6_mm
$(P)_$(M):INIT_PROCEDURE_DONE_GET	indicates if an init procedure has been made	P4837	initProcedureDone
$(P)_$(M):BLADES_KIND_SET	Flat (PV=0) or triangular (PV=1) blades	P4838	blades_kind
		:	
	limit negative in counts of the blade 1	P4841	limitNegativeMtr1_counts
	limit negative in counts of the blade 2	P4842	limitNegativeMtr2_counts
	limit negative in counts of the blade 3	P4843	limitNegativeMtr3_counts
	limit negative in counts of the blade 4	P4844	limitNegativeMtr4_counts
	limit negative in counts of the blade 5	P4845	limitNegativeMtr5_counts
	limit negative in counts of the blade 6	P4846	limitNegativeMtr6_counts
		:	
	limit negative in mm of the blade 1	P4851	limitNegativeMtr1_mm
	limit negative in mm of the blade 2	P4852	limitNegativeMtr2_mm
	limit negative in mm of the blade 3	P4853	limitNegativeMtr3_mm
	limit negative in mm of the blade 4	P4854	limitNegativeMtr4_mm
	limit negative in mm of the blade 5	P4855	limitNegativeMtr5_mm
	limit negative in mm of the blade 6	P4856	limitNegativeMtr6_mm
		:	
	indicates if new offset order for a motor	P4860	offsetOrder

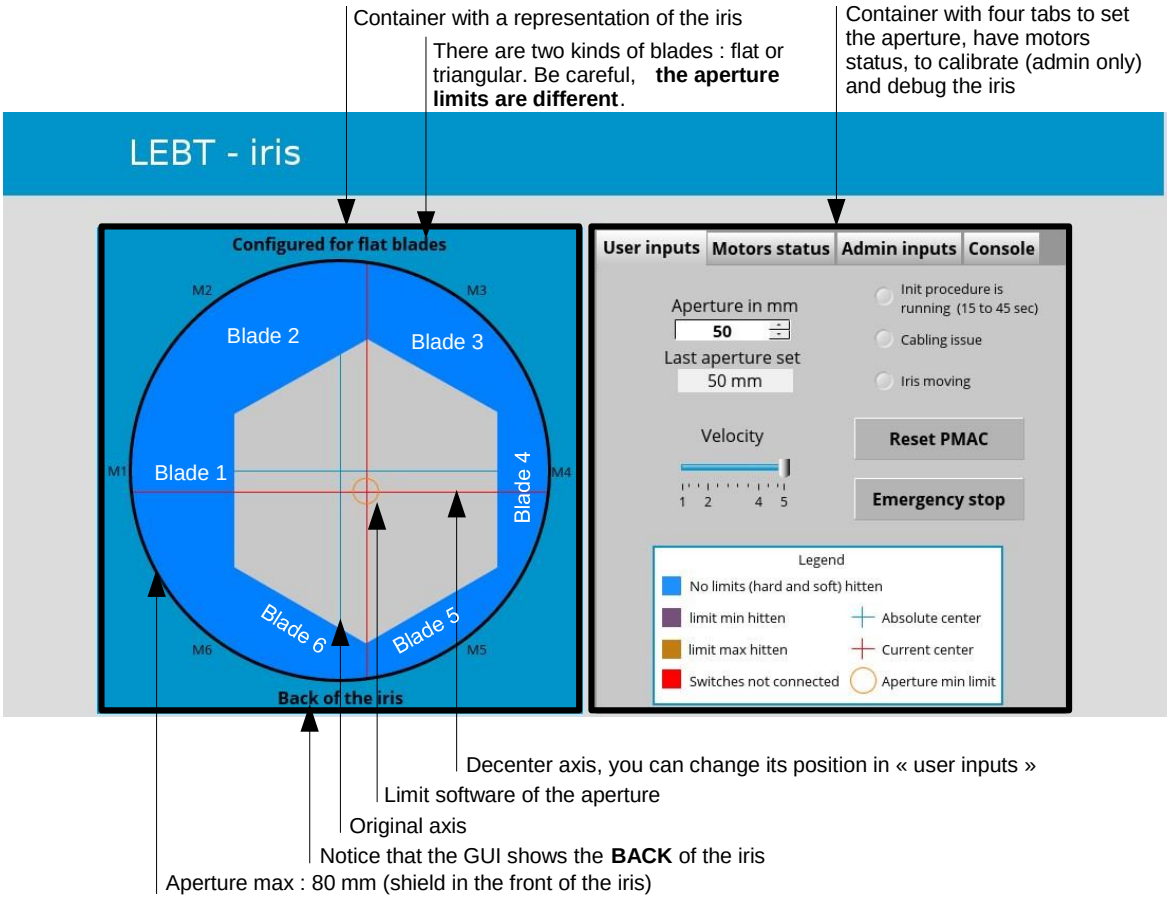
$(P)_$(M):OFFSET_M1_SET	Offset for motor 1 in mm	P4861	mtr1Offtset_mm
$(P)_$(M):OFFSET_M2_SET	Offset for motor 2 in mm	P4862	mtr2Offtset_mm
$(P)_$(M):OFFSET_M3_SET	Offset for motor 3 in mm	P4863	mtr3Offtset_mm
$(P)_$(M):OFFSET_M4_SET	Offset for motor 4 in mm	P4864	mtr4Offtset_mm
$(P)_$(M):OFFSET_M5_SET	Offset for motor 5 in mm	P4865	mtr5Offtset_mm
$(P)_$(M):OFFSET_M6_SET	Offset for motor 6 in mm	P4866	mtr6Offtset_mm
		:	
	Offset for motor 1 in counts	P4871	mtr1Offtset_counts
	Offset for motor 2 in counts	P4872	mtr2Offtset_counts
	Offset for motor 3 in counts	P4873	mtr3Offtset_counts
	Offset for motor 4 in counts	P4874	mtr4Offtset_counts
	Offset for motor 5 in counts	P4875	mtr5Offtset_counts
	Offset for motor 6 in counts	P4876	mtr6Offtset_counts
		:	
	Last offset for motor 1 in mm	P4881	mtr1OffsetLast_mm
	Last offset for motor 2 in mm	P4882	mtr2OffsetLast_mm
	Last offset for motor 3 in mm	P4883	mtr2OffsetLast_mm
	Last offset for motor 4 in mm	P4884	mtr2OffsetLast_mm
	Last offset for motor 5 in mm	P4885	mtr2OffsetLast_mm
	Last offset for motor 6 in mm	P4886	mtr2OffsetLast_mm
		:	
$(P)_$(M):CABLING_ISSUE_GET	Indicates is there is a cabling issue	P4889	cablingIssue
	travel max of a motor in its axis	P4890	travelMax
$(P)_$(M):POWER_BAD_CONTACT_M1_GET	Indicates is there is a bad contact mtr 1	P4891	motor1BadContact
$(P)_$(M):POWER_BAD_CONTACT_M2_GET	Indicates is there is a bad contact mtr 2	P4892	motor2BadContact
$(P)_$(M):POWER_BAD_CONTACT_M3_GET	Indicates is there is a bad contact mtr 3	P4893	motor3BadContact
$(P)_$(M):POWER_BAD_CONTACT_M4_GET	Indicates is there is a bad contact mtr 4	P4894	motor4BadContact
$(P)_$(M):POWER_BAD_CONTACT_M5_GET	Indicates is there is a bad contact mtr 5	P4895	motor5BadContact
$(P)_$(M):POWER_BAD_CONTACT_M6_GET	Indicates is there is a bad contact mtr 6	P4896	motor6BadContact
Name in EPICS	**Description**	**M-variables**	**Name in Geo Brick**
$(P)_$(M):MAX_L1_GET	Swhitch max limit (0=hitten, 1 no-hitten)	M121	PL1
$(P)_$(M):MAX_L2_GET	Swhitch max limit (0=hitten, 1 no-hitten)	M221	PL2
$(P)_$(M):MAX_L3_GET	Swhitch max limit (0=hitten, 1 no-hitten)	M321	PL3
$(P)_$(M):MAX_L4_GET	Swhitch max limit (0=hitten, 1 no-hitten)	M421	PL4
$(P)_$(M):MAX_L5_GET	Swhitch max limit (0=hitten, 1 no-hitten)	M521	PL5
$(P)_$(M):MAX_L6_GET	Swhitch max limit (0=hitten, 1 no-hitten)	M621	PL6
		:	
$(P)_$(M):MIN_L1_GET	Swhitch min limit (0=hitten, 1 no-hitten)	M122	NL1
$(P)_$(M):MIN_L2_GET	Swhitch min limit (0=hitten, 1 no-hitten)	M222	NL2
$(P)_$(M):MIN_L3_GET	Swhitch min limit (0=hitten, 1 no-hitten)	M322	NL3
$(P)_$(M):MIN_L4_GET	Swhitch min limit (0=hitten, 1 no-hitten)	M422	NL4
$(P)_$(M):MIN_L5_GET	Swhitch min limit (0=hitten, 1 no-hitten)	M522	NL5
$(P)_$(M):MIN_L6_GET	Swhitch min limit (0=hitten, 1 no-hitten)	M622	NL6
		:	
$(P)_$(M):IN_POSITION_M1_GET	indicates its is in position – mtr1	M140	
$(P)_$(M):IN_POSITION_M2_GET	indicates its is in position – mtr2	M240	
$(P)_$(M):IN_POSITION_M3_GET	indicates its is in position – mtr3	M340	
$(P)_$(M):IN_POSITION_M4_GET	indicates its is in position – mtr4	M440	
$(P)_$(M):IN_POSITION_M5_GET	indicates its is in position – mtr5	M550	
$(P)_$(M):IN_POSITION_M6_GET	indicates its is in position – mtr6	M650	
		:	
$(P)_$(M):IRIS_MOVING_GET	Motion program running in CS2	M5280	

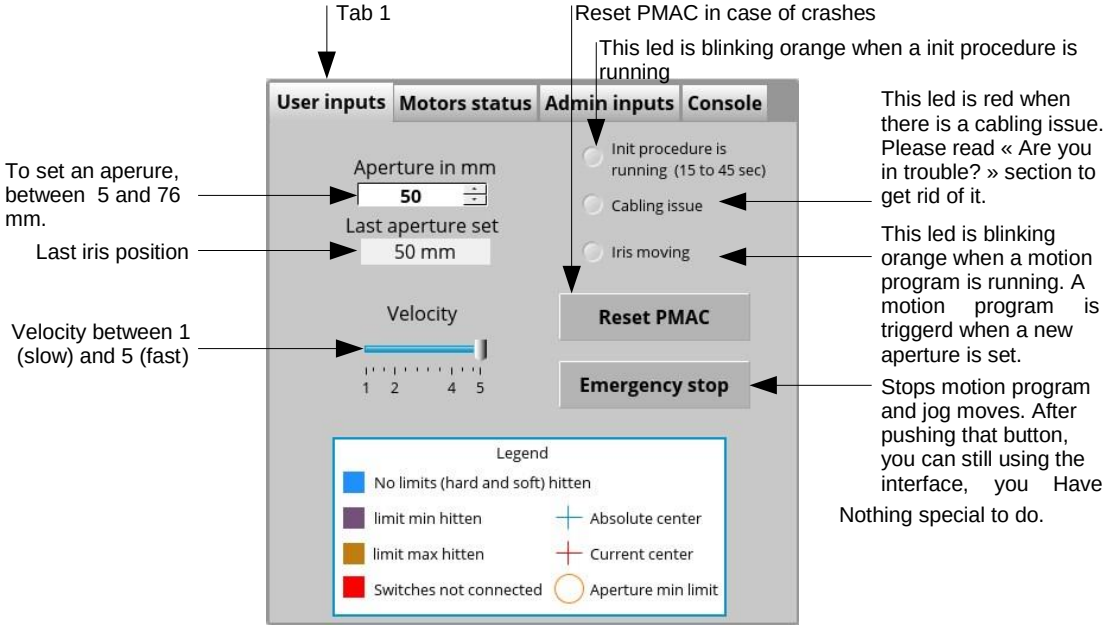
Name in EPICS	**Description**
$(P)_$(M):CONSOLE_ASK	Pmac console input
$(P)_$(M):RESET_ANSWER	Pmac console output

Appendix F

Graphical User Interface

Here is a detailed manual about the GUI.

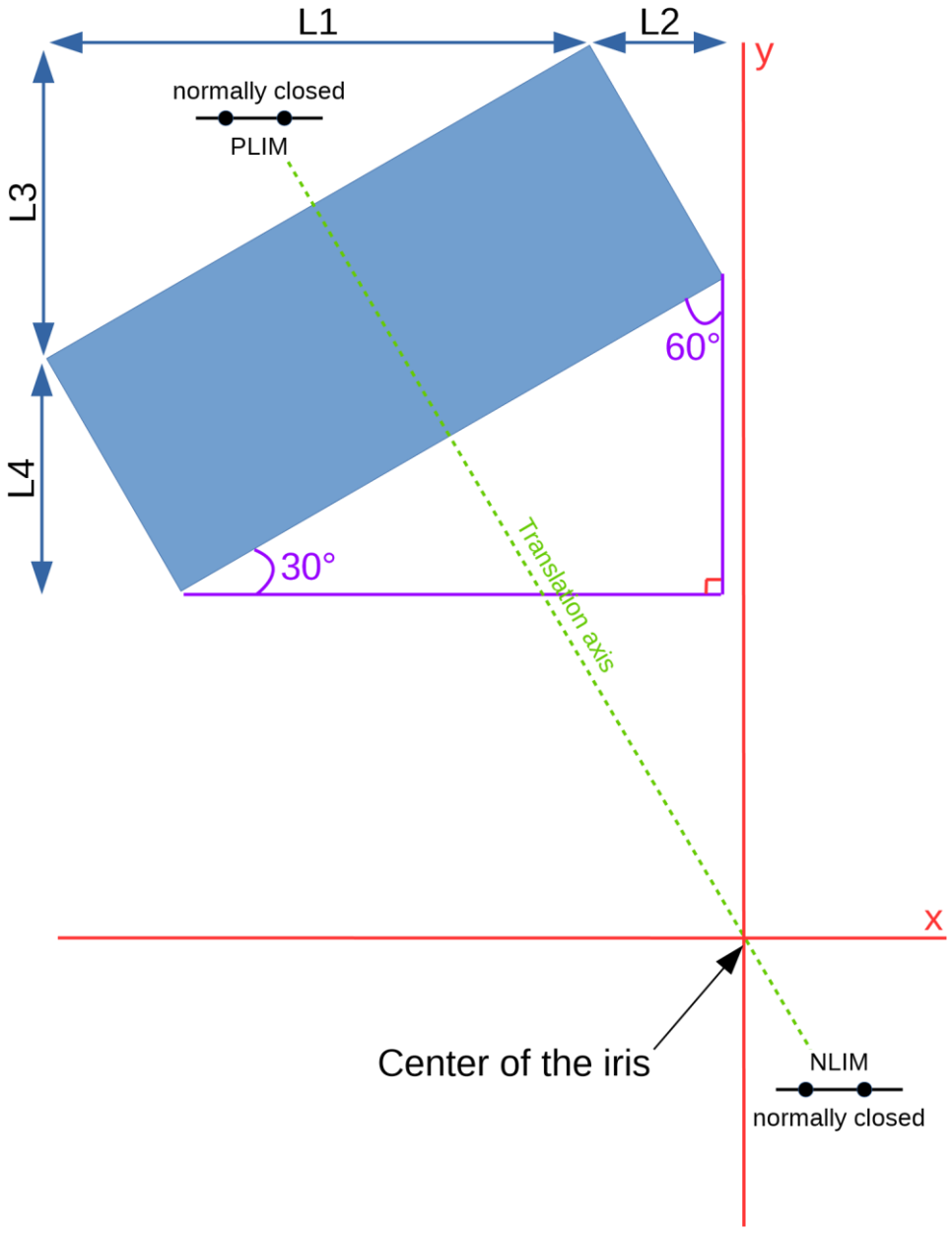

Container with a representation of the iris
There are two kinds of blades : flat or triangular. Be careful, **the aperture limits are different**.
Container with four tabs to set the aperture, have motors status, to calibrate (admin only) and debug the iris
LEBT - iris
Configured for flat blades
M1
M2
M3
M4
M5
M6
Blade 1
Blade 2
Blade 3
Blade 4
Blade 5
Blade 6
Back of the iris
User inputs
Motors status
Admin inputs
Console
Aperture in mm
50
Last aperture set
50 mm
Velocity
1 2 4 5
Init procedure is running (15 to 45 sec)
Cabling issue
Iris moving
Reset PMAC
Emergency stop
Legend
No limits (hard and soft) hitten
limit min hitten
limit max hitten
Switches not connected
Absolute center
Current center
Aperture min limit
Decenter axis, you can change its position in « user inputs »
Limit software of the aperture
Original axis
Notice that the GUI shows the **BACK** of the iris
Aperture max : 80 mm (shield in the front of the iris)


Tab 1
Reset PMAC in case of crashes
This led is blinking orange when a init procedure is running
User inputs
Motors status
Admin inputs
Console
To set an aperure, between 5 and 76 mm.
Aperture in mm
50
Last iris position
Last aperture set
50 mm
Velocity between 1 (slow) and 5 (fast)
Velocity
1 2 4 5
Init procedure is running (15 to 45 sec)
Cabling issue
Iris moving
Reset PMAC
Emergency stop
Legend
No limits (hard and soft) hitten
limit min hitten
limit max hitten
Switches not connected
Absolute center
Current center
Aperture min limit
This led is red when there is a cabling issue. Please read « Are you in trouble? » section to get rid of it.
This led is blinking orange when a motion program is running. A motion program is triggerd when a new aperture is set.
Stops motion program and jog moves. After pushing that button, you can still using the interface, you Have
Nothing special to do.

Appendix G

Geometry blades

This is how the blades 2, 3, 5 and 6 are defined in CSS (GUI design software). It can be useful if want to understand the equations in CSS (.opi file) or the iris geometry.


L1
L2
y
normally closed
PLIM
L3
60°
L4
30°
Translation axis
x
Center of the iris
NLIM
normally closed

glossary

ESS: European Spallation Source

LEBT: Low Energy Beam Transport

ICS : Integrated controlled system

CS : Coordinate System (Geo Brick)

CSS : Control System Studio

GUI : Graphical User Interface

PLIM : positive switch limit of the Geo Brick

NLIM : negative switch limit of the Geo Brick

Contacts

If you have some questions, you can contact:

Designer of the iris
Santi Passarello
INFN-Catania
Via S. Sofia, 62
95125 Catania CT
Italia passarello@lns.infn.it
+39 095 542 395

Designer of the control system
Victor Nadot
CEA Saclay
DSM/IRFU/SIS/LDISC
91191 Gif-Sur-Yvette
France victor.nadot@cea.Fr
+33 169 084 188 37

Code

J.1 Geo Brick: mapping(M-variables), configuration(I-variables), PLCs and Motion Program

For the mapping (M-variables), I used the suggested definition which is in the Turbo PMAC SOFTWARE MANUAL (starting page 610).

Here is the configuration file with I-Variables and programs.

```
;To use this                                configuration read the documentation FIRST

;************************************************
;                                    IRIS SOFTWARE                                    *
;************************************************
;***************** Description *****************
; configuration                source       f i l e                for DELTA TAU' s Geobrick LV IMS−II
;manage 6 steppers motor (2−phase bipolar ) for ESS' s i r i s ; Aperture : 3−76mm ;1 coordinate system
(CS2)
;                        Works with EPICS:
 ;              PV                                    P−variables              name in Geo Brick                    Description

 ;              $(P)$(M) :APERTURE_SET                P4801                    position                             to set the position                  in counts (360deg
          <=> 102400 counts )
; find other PV in the manual
;************* End description *****************


;******************  CONFIGURATION  *************  ;****************  Motor  s
features ************
;360deg <=> 102400 counts
#define ContCurrent                          1.2                 ; Continuous Current Limit [Amps]
#define PeakCurrent                          2.4    ;                Instantaneous Current Limit [Amps]
#define              defautVelocity          25                  ; counts/msec
#define              limitVelocity                50 ; counts/msec
#define coeff_mmTo_counts 102400 ; means 1 mm travel (one revolution ) = 102400 counts :With a count equal to a micro−step , and 512
; micro−steps per 1.8−degree full step (2048 per cycle ) , you should expect to see 360∗512/1.8= 102 ,400 counts per revolution of the motor .
#define APERTURE_MAX_MM                      74.125              ; (76/2−MAX( offset_blade+offset_X_Y ( here                it   is        96000)/102400)∗2
#define APERTURE_MIN_FLAT_MM                 3
#define APERTURE_MIN_TRIANGULAR_MM                       10
#define OFFSET_APERTURE_TO_AXIS_MM −38 ; it s not −APERTURE_MAX_MM/2 because APERTURE_MAX_MM can be set at a lower value
#define MIN_JOG_MOVE 101000; around 1mm
; these constants are calculated in function of X and Y offset and a calibration
#define OFFSET_MTR1_TO_CENTER 167857
#define OFFSET_MTR2_TO_CENTER 238434
#define OFFSET_MTR3_TO_CENTER 202594
#define OFFSET_MTR4_TO_CENTER 39857
#define OFFSET_MTR5_TO_CENTER 0
#define OFFSET_MTR6_TO_CENTER 46080
;************** End motor s features **********

;****************** Defines ******************
#define              initialisation          P4800          ; user asks to              init motors position
#define newAperture_mm                       P4801                ;new aperture in mm coming from EPICS
#define newAperture_counts                   P4802                ;new aperture in counts
#define              velocity                P4803                ;new value coming from EPICS
#define motionProgramHasBeeenLauched P4804
#define lastPosition_mm                      P4805          ; stock the      last              set−value coming from the IOC in mm
#define              lastPosition_counts P4806                    ; stock the      last              set−value coming from the IOC in COUNTS
#define offsetX_mm                           P4807           ; to decenter the        i r i s      in x−axis
#define offsetY_mm                           P4808           ; to decenter the        i r i s      in y−axis
#define lastOffsetX_mm                       P4809          ; save the              last offsetX_mm to know if          there              is a new value coming from EPICS
#define lastOffsetY_mm                       P4810          ; save the          last offsetX_mm
#define positionMtr1_counts P4811                                ; set              position motor 1 in counts
#define positionMtr2_counts P4812
#define positionMtr3_counts P4813
#define positionMtr4_counts P4814
#define positionMtr5_counts P4815
#define positionMtr6_counts P4816
#define temp1 P4817                          ; temporary variable
#define temp2 P4818
```



```
#define temp3 P4819
#define temp4 P4820
#define positionRBmtr1_counts P4821                                      ; read back position moteur 1 in counts
#define positionRBmtr2_counts P4822
#define positionRBmtr3_counts P4823
```

```
#define positionRBmtr4_counts P4824
#define positionRBmtr5_counts P4825
#define positionRBmtr6_counts P4826
#define temp5 P4827 ; temporary variable
#define temp6 P4828
#define apertureMin_mm P4829
#define positionRBmtr1_mm P4831 ; read back position moteur 1 in mm
#define positionRBmtr2_mm P4832
#define positionRBmtr3_mm P4833
#define positionRBmtr4_mm P4834
#define positionRBmtr5_mm P4835
#define positionRBmtr6_mm P4836 #define
initProcedureDone P4837 #define blades_kind P4838
#define limitNegativeMtr1_counts P4841 ; limit in counts that is calculated in function of offset x and y and the aperture_min
#define limitNegativeMtr2_counts P4842
#define limitNegativeMtr3_counts P4843
#define limitNegativeMtr4_counts P4844
#define limitNegativeMtr5_counts P4845
#define limitNegativeMtr6_counts P4846
#define limitNegativeMtr1_mm P4851 ; limit in mm that is calculated in function of offset x and y and the
aperture_min
#define limitNegativeMtr2_mm P4852
#define limitNegativeMtr3_mm P4853
#define limitNegativeMtr4_mm P4854
#define limitNegativeMtr5_mm P4855
#define limitNegativeMtr6_mm P4856
#define offsetOrder P4860 ; tell to the pmac if there is a new offset order for a motor
#define mtr1Offset_mm P4861 ; offset in mm motor by motor
#define mtr2Offset_mm P4862
#define mtr3Offset_mm P4863
#define mtr4Offset_mm P4864
#define mtr5Offset_mm P4865
#define mtr6Offset_mm P4866
#define mtr1Offset_counts P4871 ; offset in counts motor by motor
#define mtr2Offset_counts P4872
#define mtr3Offset_counts P4873
#define mtr4Offset_counts P4874
#define mtr5Offset_counts P4875
#define mtr6Offset_counts P4876
#define mtr1OffsetLast_mm P4881 ; last offset in mm motor by motor
#define mtr2OffsetLast_mm P4882
#define mtr3OffsetLast_mm P4883
#define mtr4OffsetLast_mm P4884
#define mtr5OffsetLast_mm P4885
#define mtr6OffsetLast_mm P4886
#define cablingIssue P4889
#define travelMax P4890
#define motor1BadContact P4891
#define motor2BadContact P4892
#define motor3BadContact P4893
#define motor4BadContact P4894
#define motor5BadContact P4895
#define motor6BadContact P4896

;PL: positive limit , NL: negative limit
#define PL1 M121 ; if the limit is reached , positiveLimit=1 else positiveLimit=0
#define NL1 M122 ; if the limit is reached , negativeLimit=1 else negativeLimit=0
#define PL2 M221 #define NL2 M222
#define PL3 M321 #define NL3 M322
#define PL4 M421 #define NL4 M422
#define PL5 M521 #define NL5 M522
#define PL6 M621 #define NL6 M622
;***************** End defines ******************

;******** I−variables configuration ************ ;*********** General setup
********************* I5=2 ; enable PLC 1..31 at the start
;*********** End general setup *****************

;*********** Motor setup ************************
I7000=1473 ; Servo IC0 Max Phase/PWM Frequency Control
I7001=3 ; Servo IC0 Phase Clock Frequency Control I7002=1 ; Servo IC0 Servo Clock Frequency Control
I7100=1473 ; Servo IC1 Max Phase/PWM Frequency Control
I7101=3 ; Servo IC1 Phase Clock Frequency Control
I7102=1 ; Servo IC1 Servo Clock Frequency Control
I7003 ,2,100=2258
I7007=0;
I7107=3;
I7010 ,4,10=7
I7110 ,4,10=7
I7012 ,4,10=10
I7112 ,4,10=10
I7013 ,4,10=3
I7113 ,4,10=3

I15=0 ; Trigonometric calculation in degrees
#define MaxPhaseFreq P8000 ;Max Phase Clock [KHz]
#define PWMClk P8001 ; PWM Clock [KHz]
#define PhaseClk P8002 ; Phase Clock [KHz] #define ServoClk P8003 ; Servo Clock [KHz]
```

```
MaxPhaseFreq=117964.8/(2*I7000+3)
PWMClk=117964.8/(4*I7000+6)
PhaseClk=MaxPhaseFreq/( I7001+1) ServoClk=PhaseClk/( I7002+1)

; Encoder Conversion Table
I8000=$6800BF ;                 Parallel read of Y/X:$BF
I8001=$18018 ; Use 24 bits                      starting at X bit 0
I8002=$EC0001 ;             Integrate           result from I8001
I8003=$68013F ;                 Parallel read of Y/X: $13F
I8004=$18018 ; Use 24 bits                      starting at X bit 0
I8005=$EC0004 ;             Integrate           result from I8004
I8006=$6801BF ;                 Parallel read of Y/X:$1BF
I8007=$18018 ; Use 24 bits                      starting at X bit 0
I8008=$EC0007 ;             Integrate           result from I8007
I8009=$68023F ;                 Parallel read of Y/X: $23F
I8010=$18018 ; Use 24 bits                      starting at X bit 0
I8011=$EC000A ;             Integrate           result from I8010
I8012=$6802BF ;                 Parallel read of Y/X:$2BF
I8013=$18018 ; Use 24 bits                      starting at X bit 0
I8014=$EC000D ;             Integrate           result from I8013
I8015=$68033F ;                 Parallel read of Y/X: $33F
I8016=$18018 ; Use 24 bits starting at X bit 0 I8017=$EC0010 ; Integrate result from I8016
I103=$3503 I104=$3503 ; Motor 1 position and velocity                                   feedback
I203=$3506 I204=$3506 ; Motor 2 position and velocity                                   feedback
I303=$3509 I304=$3509 ; Motor 3 position and velocity                                   feedback
I403=$350C I404=$350C ; Motor 4 position and velocity                                   feedback
I503=$350F I504=$350F ; Motor 5 position and velocity                                   feedback
I603=$3512 I604=$3512 ; Motor 6 position and velocity                                   feedback
I100 ,6,100=1 ; Motors 1-8 active : set I100 I200 I300 I400 I500 I600 to 1 I101 ,6,100=1 ; Motors 1-8 Commutation Enabled (from
X-register )

I102=$078002 ; Motor 1 Output Address I202=$07800A ; Motor 2 Output Address
I302=$078012 ; Motor 3 Output Address
I402=$07801A ; Motorf 4 Output Address
I502=$078102 ; Motor 5 Output Address I602=$07810A ; Motor 6 Output Address

I182=$078006 ; Motor 1 Current Feedback Address I282=$07800E ; Motor 2 Current
Feedback Address I382=$078016 ; Motor 3 Current Feedback Address I482=$07801E
; Motor 4 Current Feedback Address I582=$078106 ; Motor 5 Current Feedback
Address
I682=$07810E ; Motor 6 Current Feedback Address
I184 ,6,100=$FFFC00 ; Motors 1-6 Current Loop Feedback Mask, 14-bit (Geo Brick LV Specific )
I172 ,6,100=512 ; Commutation Phase Angle.2-Phase opposite voltage & current sign (Geo Brick LV Specific )

I125=$078000 ; Motor 1 Flag Address
I225=$078008 ; Motor 2 Flag Address
I325=$078010 ; Motor 3 Flag Address
I425=$078018 ; Motor 4 Flag Address
I525=$078100 ; Motor 5 Flag Address
I625=$078108 ; Motor 6 Flag Address I124 ,8,100=$800401

I183=$3503 ; Motor 1 on-going Commutation Address (ECT Integration Result )
I283=$3506 ; Motor 2 on-going Commutation Address (ECT Integration Result )
I383=$3509 ; Motor 3 on-going Commutation Address (ECT Integration Result )
I483=$350C ; Motor 4 on-going Commutation Address (ECT Integration Result )
I583=$350F ; Motor 5 on-going Commutation Address (ECT Integration Result )
I683=$3512 ; Motor 6 on-going Commutation Address (ECT Integration Result )
I170 ,6,100=1 ; Motors 1-6 Single                       cycle       size
I171 ,6,100=65536 ;                     Microsteps per Ixx70 commutation cycles

I169 ,8 ,100=28 ,44 ; Motors 1 thru 8 Output Command Limit

I166=0.1 * I7000 ; Motor #1 PWM Scale Factor ,                                  typical         setting
I266=I166 I366=I166 I466=I166 ; Assuming same motor( s ) as motor #1 I566=I166 I666=I166 ; Assuming same motor( s ) as motor #1

I15=0 ;             Trigonometric           calculation             in degrees
#define MaxADC 33.85                ;       Brick LV full               range ADC reading ( see             electrical              specifications )
#define ServoClk8 8 ;                       [KHz] Computed in Dominant Clock Settings                   Section
#define I2TOnTime 1 ; Time allowed at peak Current [ sec ]
;#define VoltOutLimit P7O7 ; This is Ixx69 normally used in direct digital PWM I157=INT(32767*(ContCurrent*1.414/MaxADC)*cos (30) )
I177=I157/SQRT(2)
I158=INT(32767*(PeakCurrent*1.414/MaxADC)*cos (30) *32767*(PeakCurrent*1.414/MaxADC)*cos (30)-I157*I157 )* ServoClk8*1000*I2TOnTime/(32767*32767)
I257=I157 I277=I177 I258=I158
I357=I157 I377=I177 I358=I158
I457=I157 I477=I177 I458=I158
I557=I157 I577=I177 I558=I158
I657=I157 I677=I177 I658=I158
I180=0 I173=0 I174=0 ;
I280=0 I273=0 I274=0 ;
I380=0 I373=0 I374=0 ;
I480=0 I473=0 I474=0 ;
I580=0 I573=0 I574=0 ;
I680=0 I673=0 I674=0 ;
I780=0 I773=0 I774=0 ;
I880=0 I873=0 I874=0 ;
I181=$3503 ; Motor 1 Power-On Commutation ,                     Integrated Output #1
I281=$3506 ; Motor 2 Power-On Commutation ,                     Integrated Output #2
I381=$3509 ; Motor 3 Power-On Commutation ,                     Integrated Output #3
I481=$350C ; Motor 4 Power-On Commutation ,                     Integrated Output #4
I581=$350F ; Motor 5 Power-On Commutation ,                     Integrated Output #5
I681=$3512 ; Motor 6 Power-On Commutation ,                     Integrated Output #6
I781=$3515 ; Motor 7 Power-On Commutation ,                         Integrated Output #7
I881=$3518 ; Motor 8 Power-On Commutation ,                         Integrated Output #8
I191 ,8,100=$500000 ; Mtrs 1-8 Pwr-on Pos . format Read 16 (11+5) bits                                  of X register           Ixx81
```

```
I130 ,8,100=1024 ; Position−Loop PID Gains
I131 ,8,100=0 ;
I132 ,8,100=85 ;
I133 ,8,100=1024 ;
I134 ,8,100=1 ;
I135 ,8,100=0 ;
I136 ,8,100=0 ;
I137 ,8,100=0 ;
I138 ,8,100=0 ;
I139 ,8,100=0 ;
I161 ,8 ,100=0.16;Motor 1 Current Loop Integral Gain I176 ,8 ,100=1.9;Motor 1 Current Loop
Integral Gain

I7004=1 ; Servo IC 0 PWM Deadtime/PFM Pulse Width Control (PMAC2 Only)
I7104=1 ; Servo IC 1 PWM Deadtime/PFM Pulse Width Control (PMAC2 Only)

I5250 ,15,100=0 ; disenable kinematics I5290=1 ; velocity in counts/msec
for CS2 I5289=limitVelocity ; defaut velocity CS2 I5298=limitVelocity ; max
velocity CS2

I113 ,8,100=0 ; means that there no software positive limits I114 ,8,100=0 ; means that there no software
negative limits I116 ,8,100= limitVelocity ;max program velocity

;********** End Motor setup ******************** ;******** End I−variables
configuration ******* ;************** END CONFIGURATION *************




;***************** PRGOGRAMMES ****************
; There is two kinds of programme : PLC and motion programmes

;*************** PLC programs *****************
;PLC programs are designed                    for                    calculations and actions that are asynchronous to the motion
;PLC programs are particularly useful for monitoring analog and digital inputs , setting outputs , sending ; messages , monitoring motion parameters , issuing commands as if from a host ,
changing gains , and ; starting and stopping moves . CLOSE
END GATHER
DELETE GATHER DELETE TRACE
OPEN PLC 1 CLEAR ; Potection power−on PLC
                DISABLE PLC2..31
                        I5111=20∗8388608/I10                    ; counter
                WHILE(I5111 >0) END WHILE                                                ; wait end of counter
                COMMAND"wx$78014        ,       $f84dfe"
                I5111=20∗8388608/I10 WHILE(I5111 >0)
                END WHILE


                M148=0                  ; Phasing error                 faultbit
                COMMAND"wx$78014        ,       $f94dfe"
                I5111=20∗8388608/I10 WHILE(I5111 >0)
                END WHILE

                M248=0                  ; Phasing error                 faultbit
                COMMAND"wx$78014        ,       $fa4dfe"
                I5111=20∗8388608/I10 WHILE(I5111 >0)
                END WHILE

                M348=0                  ; Phasing error                 faultbit
                COMMAND"wx$78014        ,       $fb4dfe"
                I5111=20∗8388608/I10 WHILE(I5111 >0)
                END WHILE

                M448=0                  ; Phasing error                 faultbit
                COMMAND"wx$78114        ,       $f84dfe"
                I5111=20∗8388608/I10 WHILE(I5111 >0)
                END WHILE

                M548=0                  ; Phasing error                 faultbit
                COMMAND"wx$78114        ,       $f94dfe"
                I5111=20∗8388608/I10 WHILE(I5111 >0)
                END WHILE

                M648=0                  ; Phasing error                 faultbit
                COMMAND"wx$78114        ,       $fa4dfe"
                I5111=20∗8388608/I10 WHILE(I5111 >0)
                END WHILE

                M748=0                  ; Phasing error                 faultbit
                COMMAND"wx$78114        ,       $fb4dfe"
                I5111=20∗8388608/I10 WHILE(I5111 >0)
                END WHILE

                M848=0          ; Phasing error                 fault       bit
                I5111=50∗8388608/I10 WHILE(I5111 >0)
                END WHILE
                ENABLE PLC7
                DISABLE PLC1

CLOSE

; definition                    of the Coordonate System 2 ( execute                    this PLC only one time)
OPEN PLC 7 CLEAR
                ;when the pmac restarts , it comes back to its last value ( position motors) M162=positionRBmtr1_counts∗( I108 ∗32) ; actual position of motor 1 in counts
                M262=positionRBmtr2_counts∗( I208 ∗32)
                M362=positionRBmtr3_counts∗( I308 ∗32)
                M462=positionRBmtr4_counts∗( I408 ∗32)
```

```
M562=positionRBmtr5_counts*( I508 *32)
M662=positionRBmtr6_counts*( I608 *32)
                COMMAND"motor1BadContact ,6,1=0" ; motor1BadContact , motor2BadContact . . . motor6BadContact =0
IF(blades_kind = 0) apertureMin_mm =APERTURE_MIN_FLAT_MM
ELSE apertureMin_mm = APERTURE_MIN_TRIANGULAR_MM END IF
CMD"I122 ,8,100= defautVelocity" ; jog speed
CMD" velocity=limitVelocity" ;CS speed
    ; coorinate system initialisation
     CMD"UNDEFINE ALL" ; Erase definition of all axes in all coordinate sytems

  ; axis initialisation
CMD"&2#1->A" ; assign moteur 1 to axis A in coordinate sytem 2 CMD"&2#2->B" ; assign moteur 2 to axis B in coordinate sytem 2
CMD"&2#3->C"
CMD"&2#4->U"
CMD"&2#5->V"
CMD"&2#6->W" ; axis A, B, C, U, V and W are in coordinate system 2 <=> motor 1 to 6 are in coordinate system 2
    COMMAND"&2$$" ; close loop and phase reference for all motors in CS2

ENABLE PLC9
ENABLE PLC10
DISABLE PLC7
CLOSE

; init procedure
OPEN PLC8 CLEAR cablingIssue=0
COMMAND"I113 ,8,100=0" ; means that there no software positive limits COMMAND"I114 ,8,100=0" ; means that there no
software negative limits
                COMMAND"motor1BadContact ,6,1=0" ; motor1BadContact , motor2BadContact . . . motor6BadContact =0
; f ir st case when motors ARE at a PLIM (and NLIM=0) temp1 = M162/( I108 *32) ; save motor 1 position ; actual position of motor 1 in counts temp2 =
M262/( I208 *32) temp3 = M362/( I308 *32) temp4 = M462/( I408 *32) temp5 = M562/( I508 *32) temp6 = M662/( I608 *32) IF(PL1=1 AND NL1=0)
        COMMAND"#1J^-MIN_JOG_MOVE" ; "to be sure motor reach the position (and not MIN_JOG_MOVE)
        WHILE (positionRBmtr1_counts>temp1-(MIN_JOG_MOVE-1000))
                positionRBmtr1_counts = M162/( I108 *32) ; actual positionof motor 1 in counts
        END WHILE ; wait the end of motors move
        IF(PL1=1)
                motor1BadContact=1
                M162=temp1*( I108 *32) ; restore last value knew

        END IF
END IF
IF(PL2=1 AND NL2=0)
        COMMAND"#2J^-MIN_JOG_MOVE" ;"
                WHILE (positionRBmtr2_counts>temp2-(MIN_JOG_MOVE-1000))
                positionRBmtr2_counts = M262/( I208 *32) ; actualposition of motor 1 in counts
        END WHILE ; wait the end of motors move
        IF(PL2=1) motor2BadContact=1
                M262=temp2*( I208 *32)
        END IF
END IF
IF(PL3=1 AND NL3=0)
        COMMAND"#3J^-MIN_JOG_MOVE" ;"
        WHILE (positionRBmtr3_counts>temp3-(MIN_JOG_MOVE-1000))
                positionRBmtr3_counts = M362/( I308 *32) ; actualposition of motor 1 in counts
        END WHILE ; wait the end of motors move
        IF(PL3=1) motor3BadContact=1
                M362=temp3*( I308 *32)
        END IF
END IF
IF(PL4=1 AND NL4=0)
        COMMAND"#4J^-MIN_JOG_MOVE" ;"
        WHILE (positionRBmtr4_counts>temp4-(MIN_JOG_MOVE-1000))
                positionRBmtr4_counts = M462/( I408 *32) ; actualposition of motor 1 in counts
        END WHILE ; wait the end of motors move
        IF(PL4=1) motor4BadContact=1
                M462=temp4*( I408 *32)
        END IF
END IF
IF(PL5=1 AND NL5=0)
        COMMAND"#5J^-MIN_JOG_MOVE" ;"
        WHILE (positionRBmtr5_counts>temp5-(MIN_JOG_MOVE-1000))
                positionRBmtr5_counts = M562/( I508 *32) ; actualposition of motor 1 in counts
        END WHILE ; wait the end of motors move
        IF(PL5=1) motor5BadContact=1
                M562=temp5*( I508 *32)
        END IF
END IF
IF(PL6=1 AND NL6=0)
        COMMAND"#6J^-MIN_JOG_MOVE" ;"
        WHILE (positionRBmtr6_counts>temp6-(MIN_JOG_MOVE-1000))
                positionRBmtr6_counts = M662/( I608 *32) ; actual END WHILE ; wait the end of motorsposition of motor 1 in counts
        move
        IF(PL6=1) motor6BadContact=1
                M662=temp6*( I608 *32)
        END IF
END IF


 ; second case when motors ARE NOT at PLIM
    COMMAND"temp1=#1P" ; save motor 1 position
COMMAND"temp2=#2P"
COMMAND"temp3=#3P"
COMMAND"temp4=#4P"
COMMAND"temp5=#5P"
COMMAND"temp6=#6P"
```

```
COMMAND"#1J+#2J+#3J+#4J+#5J+#6J+" ;move 6 motors until limits travelMax=(90/2)*coeff_mmTo_counts+1000 ;+1000 counts and 90 ( instead of 84) to be sure
WHILE (PL1=0 AND motor1BadContact=0) OR (PL2=0 AND motor2BadContact=0) OR (PL3=0 AND motor3BadContact =0) OR (PL4=0 AND motor4BadContact=0) OR (PL5=0 AND
        motor5BadContact=0) OR (PL6=0 AND motor6BadContact=0)
; wait until          all          motors have reached their                    switch or they are bad wired

                                        positionRBmtr1_counts = M162/( I108 *32)    ; actual    position    of motor 1 in counts
                                        positionRBmtr2_counts = M262/( I208 *32)    ; actual    position    of motor 2
                                        positionRBmtr3_counts = M362/( I308 *32)    ; actual    position    of motor 3
                                        positionRBmtr4_counts = M462/( I408 *32)    ; actual    position    of motor 4
                                        positionRBmtr5_counts = M562/( I508 *32)    ; actual    position    of motor 5
                    positionRBmtr6_counts = M662/( I608 *32)                        ; actual    position    of motor 6
        IF(positionRBmtr1_counts>temp1+travelMax) motor1BadContact=1
                    COMMAND"#1K" ;" k i l l      motor
                      COMMAND"#1J/" ; stop jog
                        M162=temp1*( I108 *32) ; last          value know
        END IF ;
        IF(positionRBmtr2_counts>temp2+travelMax) motor2BadContact=1
                    COMMAND"#2K"
                    COMMAND"#2J/" ;"
                        M262=temp2*( I208 *32) ; last          value know
        END IF ;
        IF(positionRBmtr3_counts>temp3+travelMax) motor3BadContact=1
                    COMMAND"#3K"
                    COMMAND"#3J/" ;"
                        M362=temp3*( I308 *32)
        END IF ;
        IF(positionRBmtr4_counts>temp4+travelMax) motor4BadContact=1
                    COMMAND"#4K"
                    COMMAND"#4J/" ;"
                        M462=temp4*( I408 *32)
        END IF ;
        IF(positionRBmtr5_counts>temp5+travelMax) motor5BadContact=1
                    COMMAND"#5K"
                    COMMAND"#5J/" ;"
                        M562=temp5*( I508 *32)
        END IF ;
        IF(positionRBmtr6_counts>temp6+travelMax) motor6BadContact=1
                    COMMAND"#6K"
                    COMMAND"#6J/" ;"
                        M662=temp6*( I608 *32)
        END   IF   ;   END
WHILE



WHILE(M140=0 OR M240=0 OR M340=0 OR M440=0 OR M540=0 OR M640=0) ; wait end of all jog move END WHILE


COMMAND"&2$$" ; close loop for all motors in CS2 IF (motor1BadContact=0) ; if not bad
contact and . . . .
        IF (PL1=0 AND NL1=0) OR (PL1=1 AND NL1=0) OR(PL1=0 AND NL1=1)    ; . . . . PLIM/NLIM connected CMD"P26=#1P"
                    COMMAND"#1J^-OFFSET_MTR1_TO_CENTER" I5111=100*8388608/I10  ; counter
                        WHILE(I5111 >0)                                                    ; wait end of counter
                    END WHILE
                        WHILE (M133=0) ;M140:          in       position
                    END WHILE ; wait the end of motors move
                    M162=((APERTURE_MAX_MM/2)*coeff_mmTo_counts)*( I108 *32) ; set current position as i r i s wide open , aperture = 76mm END IF
END IF
    IF (motor2BadContact=0) ; if                not bad contact and          . . . .
        IF (PL2=0 AND NL2=0) OR (PL2=1 AND NL2=0) OR(PL2=0 AND NL2=1)    ; . . . . PLIM/NLIM connected CMD"P26=#2P"
                    COMMAND"#2J^-OFFSET_MTR2_TO_CENTER" I5111=100*8388608/I10  ; counter
                        WHILE(I5111 >0)                                                    ; wait end of counter
                    END WHILE
                    WHILE (M233=0)
                    END WHILE ; wait the end of motors move
                    M262=((APERTURE_MAX_MM/2)*coeff_mmTo_counts)*( I208 *32) ; set current position as i r i s wide open , aperture = 76mm END IF
END IF
    IF (motor3BadContact=0) ; if                not bad contact and          . . . .
        IF (PL3=0 AND NL3=0) OR (PL3=1 AND NL3=0) OR(PL3=0 AND NL3=1)    ; . . . . PLIM/NLIM connected CMD"P26=#3P"
                    COMMAND"#3J^-OFFSET_MTR3_TO_CENTER" I5111=100*8388608/I10  ; counter
                        WHILE(I5111 >0)                                                    ; wait end of counter
                    END WHILE
                    WHILE (M333=0)
                    END WHILE ; wait the end of motors move

                    M362=((APERTURE_MAX_MM/2)*coeff_mmTo_counts)*( I308 *32) ; set current position as wide open , aperture = 76mm END IF                i r i s
END IF
IF (motor4BadContact=0) ; if                not bad contact and          . . . .
        IF (PL4=0 AND NL4=0) OR (PL4=1 AND NL4=0) OR(PL4=0 AND NL4=1)    ; . . . . PLIM/NLIM connected CMD"P26=#4P"
                    COMMAND"#4J^-OFFSET_MTR4_TO_CENTER" I5111=100*8388608/I10  ; counter
                        WHILE(I5111 >0)                                                    ; wait end of counter
                    END WHILE
                    WHILE (M433=0)
                    END WHILE ; wait the end of motors move
```

```
                                M462=((APERTURE_MAX_MM/2)*coeff_mmTo_counts)*( I408 *32) ; set current position as wide open , aperture = 76mm END IF                    i r i s
        END IF
        IF (motor5BadContact=0) ; if                            not bad contact and              . . . .
                IF (PL5=0 AND NL5=0) OR (PL5=1 AND NL5=0) OR(PL5=0 AND NL5=1)     ; . . . . PLIM/NLIM connected CMD"P26=#5P"
                        COMMAND"#5J^-OFFSET_MTR5_TO_CENTER" I5111=100*8388608/I10 ; counter
                            WHILE(I5111 >0)                                                                    ; wait end of counter
                        END WHILE
                        WHILE (M533=0)
                        END WHILE ; wait the end of motors move



                                M562=((APERTURE_MAX_MM/2)*coeff_mmTo_counts)*( I508 *32) ; set current position as wide open , aperture = 76mm END IF                    i r i s
        END IF
        IF (motor6BadContact=0) ; if                            not bad contact and              . . . .
                IF (PL6=0 AND NL6=0) OR (PL6=1 AND NL6=0) OR(PL6=0 AND NL6=1)     ; . . . . PLIM/NLIM connected CMD"P26=#6P"
                        COMMAND"#6J^-OFFSET_MTR6_TO_CENTER" I5111=100*8388608/I10 ; counter
                            WHILE(I5111 >0)                                                                    ; wait end of counter
                        END WHILE
                        WHILE (M633=0)
                        END WHILE ; wait the end of motors move



                                M662=((APERTURE_MAX_MM/2)*coeff_mmTo_counts)*( I608 *32) ; set current position as wide open , aperture = 76mm END IF                    i r i s
        END IF


         ; third case PLIM and NLIM =1
         IF     (PL1=1     AND     NL1=1)OR(PL2=1     AND     NL2=1)OR(PL3=1     AND     NL3=1)OR(PL4=1     AND     NL4=1)OR(PL5=1     AND     NL5=1)OR(     PL6=1     AND
                NL6=1)OR(motor1BadContact=1)OR(motor2BadContact=1)OR(motor3BadContact=1)OR(      motor4BadContact=1)OR(motor5BadContact=1)OR(motor6BadContact=1)
                cablingIssue = 1
         ELSE cablingIssue = 0
         END IF

          initialisation=0 ; end of initialisation (newAperture_counts = (newAperture_mm/2) * coeff_mmTo_counts ; convertion mm to counts )

          ; to come back to last aperture with last offset lastPosition_counts = (APERTURE_MAX_MM/2) *
          coeff_mmTo_counts initProcedureDone = 1

          DISABLE PLC8
          COMMAND"#1K#2K#3K#4K#5K#6K"
             COMMAND"SAVE$$$"                                    ;PMAC saves P-variables and other datas in EEPROM and reset
CLOSE

; i n i t i a l i s e s              the motors position          if     it     is     asked
; check      if          there are new values ( of aperture or decentring ) coming from EPICS driver ; if          there is , launch motion program
1
OPEN PLC 9 CLEAR
          IF( initialisation =1) ENABLE PLC8
          END IF

                ; save the position                  of motors after a move
             IF(M5280 = 1)                                  ; M5280 : &2 Program-running bit )
                    IF(motionProgramHasBeeenLauched                =                0)
                    motionProgramHasBeeenLauched = 1 END IF ;
             ELSE ; i r i s            is not moving
                    IF(motionProgramHasBeeenLauched = 1) ; a motion program has been launched motionProgramHasBeeenLauched = 0
                        COMMAND"SAVE" ;PMAC saves P-variables and other datas in EEPROM END IF
          END IF

            ; check     if     there          is a new offset          order          for a motor
          IF(mtr1Offset_mm != mtr1OffsetLast_mm OR mtr2Offset_mm != mtr2OffsetLast_mm OR mtr3Offset_mm != mtr3OffsetLast_mm OR mtr4Offset_mm != mtr4OffsetLast_mm OR
                mtr5Offset_mm != mtr5OffsetLast_mm OR mtr6Offset_mm != mtr6OffsetLast_mm) offsetOrder=1 mtr1OffsetLast_mm = mtr1Offset_mm mtr2OffsetLast_mm =
                mtr2Offset_mm mtr3OffsetLast_mm = mtr3Offset_mm mtr4OffsetLast_mm = mtr4Offset_mm mtr5OffsetLast_mm = mtr5Offset_mm mtr6OffsetLast_mm = mtr6Offset_mm
          ELSE
                    offsetOrder=0
          END IF

          IF(newAperture_mm!=0) ; when PMAC starts all Pxxx variables are 0. so 0 is not a valid command newAperture_counts = (newAperture_mm/2) * coeff_mmTo_counts ; convertion
                    mm to counts
                    IF( lastPosition_counts != newAperture_counts OR offsetX_mm != lastOffsetX_mm OR offsetY_mm != lastOffsetY_mm OR offsetOrder =1) ; if we get a new position or
                            a new offset in x and y
                                ; calcul                of the limits min and max
                                ;MAX
                                COMMAND"I113 ,6 ,100 = (APERTURE_MAX_MM/2)*coeff_mmTo_counts" ; soft positive postion limits motor 1 to 6

                                ;MIN
                                ; blade 1 limitNegativeMtr1_mm = apertureMin_mm/2 - offsetX_mm
                                limitNegativeMtr1_counts = limitNegativeMtr1_mm*coeff_mmTo_counts ;

                                ; blade 2 limitNegativeMtr2_mm = apertureMin_mm/2 - (offsetX_mm)*COS(60) + (offsetY_mm)*COS(30) limitNegativeMtr2_counts =
                                limitNegativeMtr2_mm*coeff_mmTo_counts

                                ; blade 3 limitNegativeMtr3_mm = apertureMin_mm/2 + (offsetX_mm)*COS(60) + (offsetY_mm)*COS(30) limitNegativeMtr3_counts =
                                limitNegativeMtr3_mm*coeff_mmTo_counts

                                ; blade 4 limitNegativeMtr4_mm = apertureMin_mm/2 + (offsetX_mm) limitNegativeMtr4_counts =
                                limitNegativeMtr4_mm*coeff_mmTo_counts

                                ; blade 5 limitNegativeMtr5_mm = apertureMin_mm/2 + (offsetX_mm)*COS(60) - (offsetY_mm)*COS(30) limitNegativeMtr5_counts =
                                limitNegativeMtr5_mm*coeff_mmTo_counts

                                ; blade 6 limitNegativeMtr6_mm = apertureMin_mm/2 - (offsetX_mm)*COS(60) - (offsetY_mm)*COS(30) limitNegativeMtr6_counts =
                                limitNegativeMtr6_mm*coeff_mmTo_counts
```

```
COMMAND"I114=limitNegativeMtr1_counts"
COMMAND"I214=limitNegativeMtr2_counts"
COMMAND"I314=limitNegativeMtr3_counts"
COMMAND"I414=limitNegativeMtr4_counts"
COMMAND"I514=limitNegativeMtr5_counts"
COMMAND"I614=limitNegativeMtr6_counts"


    ; calcul                of the position
; just trigonometry in function of offset in x and y axis positionMtr1_counts = ((newAperture_mm/2) − offsetX_mm)∗ coeff_mmTo_counts
positionMtr2_counts = ((newAperture_mm/2) − offsetX_mm∗COS(60) + offsetY_mm∗COS(30) )∗ coeff_mmTo_counts
positionMtr3_counts = ((newAperture_mm/2) + offsetX_mm∗COS(60) + offsetY_mm∗COS(30) )∗ coeff_mmTo_counts
positionMtr4_counts = ((newAperture_mm/2) + offsetX_mm)∗ coeff_mmTo_counts positionMtr5_counts = ((newAperture_mm/2) +
offsetX_mm∗COS(60) − offsetY_mm∗COS(30) )∗ coeff_mmTo_counts
positionMtr6_counts = ((newAperture_mm/2) − offsetX_mm∗COS(60) − offsetY_mm∗COS(30) )∗ coeff_mmTo_counts

; convertion offset ( axis motor by motor mm) to counts mtr1Offset_counts = mtr1Offset_mm
∗ coeff_mmTo_counts mtr2Offset_counts = mtr2Offset_mm ∗ coeff_mmTo_counts
mtr3Offset_counts = mtr3Offset_mm ∗ coeff_mmTo_counts mtr4Offset_counts =
mtr4Offset_mm ∗ coeff_mmTo_counts mtr5Offset_counts = mtr5Offset_mm ∗
coeff_mmTo_counts mtr6Offset_counts = mtr6Offset_mm ∗ coeff_mmTo_counts

lastOffsetX_mm=offsetX_mm lastOffsetY_mm=offsetY_mm
lastPosition_counts = newAperture_counts
lastPosition_mm = (newAperture_counts/coeff_mmTo_counts)∗2 ; convertion counts to mm

  COMMAND"&2$$"          ; close          loop and phase reference          for      all          motors in CS2
COMMAND"&2B1R" ; launch motion program 1 for Coordinated system 2 END IF
        END IF
CLOSE
;Read−back values of motors position OPEN PLC 10 CLEAR

                positionRBmtr1_counts = M162/( I108 ∗32)          ; actual          position   of motor 1 in counts
                positionRBmtr2_counts = M262/( I208 ∗32)          ; actual          position   of motor 2

                positionRBmtr3_counts = M362/( I308 ∗32)          ; actual          position   of motor 3

                positionRBmtr4_counts = M462/( I408 ∗32)          ; actual          position   of motor 4

                positionRBmtr5_counts = M562/( I508 ∗32)          ; actual          position   of motor 5

                positionRBmtr6_counts = M662/( I608 ∗32)          ; actual          position   of motor 6

        positionRBmtr1_mm =(M162/( I108 ∗32)/coeff_mmTo_counts)                    ; actual          positionof motor 1 in mm
        positionRBmtr2_mm =(M262/( I208 ∗32)/coeff_mmTo_counts)                    ; actual          positionof motor 2
        positionRBmtr3_mm =(M362/( I308 ∗32)/coeff_mmTo_counts)                    ; actual          positionof motor 3
        positionRBmtr4_mm =(M462/( I408 ∗32)/coeff_mmTo_counts)                    ; actual          positionof motor 4
        positionRBmtr5_mm =(M562/( I508 ∗32)/coeff_mmTo_counts)                    ; actual          positionof motor 5
        positionRBmtr6_mm =(M662/( I608 ∗32)/coeff_mmTo_counts)                    ; actual          positionof motor 6
CLOSE
;************* End PLC programs ***************

;************* Motion programm ***************** ; Motion programme are used to move a
coordinate system
; motion prog :                    set the aperture in function of EPICS PVs
;1          coordinate system (CS2)
;6              axis which move simultaneously ( i r i s              application )
; variable          written by EPICS
OPEN PROG 1 CLEAR
        ABS ; absolute location and not incremental (INC) LINEAR ; constant velocitty during the move
            FRAX(A,B,C,U,V,W)                ; axis in          feedrate
          F( velocity )                                    ; velocity= user length unit ( or angle )/Isx90 (ms) ( if                    Isx90 vaut 1000=>
                per sec , 60000=>per minute)
              ; In a coordinate system ,          axis          will move simultaneously          if          there are on the same line
                                                    A( positionMtr1_counts−mtr1Offset_counts )B( positionMtr2_counts−mtr2Offset_counts )C( positionMtr3_counts
            +mtr3Offset_counts )U( positionMtr4_counts+mtr4Offset_counts )V( positionMtr5_counts+ mtr5Offset_counts )W(
positionMtr6_counts−mtr6Offset_counts ) CLOSE
;*********** End motion program ************* ;************* END

PRGOGRAMS *****************
```

J.2 EPICS driver: .cmd, .proto, .template and .substitutions

*** iris.cmd ***

```
require asyn , 4.27.0 require streamdevice ,
2.7.1 require tpmac , 3.11.2 require iris , 1.1.0
```

```
# Set environmental variables epicsEnvSet ("ASYN_PORT" ,          "GEOBRICK_ASYN") epicsEnvSet
 ("PMAC_IP" , "10.10.1.40") epicsEnvSet ("PMAC_PORT" , "1025")


#fonctions from TPMAC
# Connection to GEOBRICK, create a asyn port pmacAsynIPConfigure($(ASYN_PORT) , $(PMAC_IP) : $(PMAC_PORT) )

 dbLoadRecords("get_value_pmac                    .db")
 dbLoadRecords("set_value_pmac    .db")    dbLoadRecords("
 motor_iris .db") dbLoadRecords(" console .db")
```

*** pmacVariables.proto ***

```
# Streams protocol            f i l e            for the pmacStreams protocol



# Initial              version NPR 04/2006

# use "streamReload" to reload                    this        f i l e      without restarting the IOC

 ExtraInput = Ignore ;
 ReadTimeout = 500;
 OutTerminator = "";
 InTerminator = ACK; Separator = CR;


 sendString
{
      out "%(\$1) s" CR; in "%s ";
}


 setVar
{
      out "\$1=%f" CR; in ; }

 getVar
{
      out "\$1" CR;
      in "%f ";
}
```

*** get_value_pmac.template ***

```
#get P−variables value from Geo Brick record ( ai , "$(P)_$(M) :
 $(NAME)_GET") { field (DESC, "$(DESC) ") field (EGU, "$(EGU) ")
 field (DTYP, "stream") field (PREC, "$(PREC) ")
       field (INP, "@pmacVariables . proto getVar($(P−VARIABLE) ) $(SPORT) ") field (SCAN, "$(SCAN) ") #I/O Intr
 }
```

*** get_value_pmac.substitutions ***

```
 f i l e          get_value_pmac . template {
pattern {P                M              NAME                              DESC                                         EGU          P−
         VARIABLE           PREC          SCAN                  SPORT}
              {LEBT        IRIS           INIT_PROCESSING                 " inidcates            if a initprocedure        is     running"          boolean P4800
                                 0               ".1 second"              GEOBRICK_ASYN}
              {LEBT        IRIS         LAST_COMMAND                      "LAST_COMMAND of             iris ’ s position send"                 mm          P4805
                                 0               ".1 second"              GEOBRICK_ASYN}
              {LEBT        IRIS          APERTURE_MIN                          "get the aperture min"                                          mm          P4829
                                 0               ".1 second"              GEOBRICK_ASYN}
              {LEBT        IRIS           INIT_PROCEDURE_DONE             " if          this PC=0 => init             procedure not done"          boolean P4837
                                 0               ".1 second"              GEOBRICK_ASYN}
              {LEBT        IRIS          CABLING_ISSUE                    "bit      cacling               issue ( limit or power motor"             boolean P4889
                                 0               ".1 second"              GEOBRICK_ASYN}
              {LEBT        IRIS          IRIS_MOVING                    " i r i s      is          running a program?"                         boolean M5280
```

```
                                0                   ".1 second"              GEOBRICK_ASYN}
}
```

*** set_value_pmac.template ***

```
#set data to send to the Geo brick record (ao , "$(P)_$(M) :
$(NAME)_SET") { field (DESC, "$(DESC) ") field (EGU, "$(EGU) ")
field (DRVL, "$(DRVL) ") field (DRVH, "$(DRVH) ") field (LOPR,
"$(DRVL) ") field (HOPR, "$(DRVH) ") field (VAL, "$(VAL) ") field
(ADEL, "$(ADEL) ") field (MDEL, "$(MDEL) ") field (PREC, "$(PREC)
")
}
#send the data to the Geo brick record ( calcout , "$(P)_$(M) : $(NAME)_WRITE")
{ field (DESC, "send the data to PMAC")
    field (DTYP, "stream") field (CALC, "$(CALC) ")
    field (INPA, "$(P)_$(M) : $(NAME)_SET CP")




    field (OUT, "@pmacVariables . proto setVar ($(P−VARIABLE) ) $(SPORT) ")
}
```

*** set_value_pmac.substitutions ***

```
f i l e     set_value_pmac . template {

pattern {P          M         NAME          DESC                                      EGU      DRVL     DRVH
        CALC      VAL      ADEL      MDEL     PREC      P−VARIABLE       SPORT}
        {LEBT     IRIS     INIT               "launch an init        procedure"          boolean 0          1

                  A        0         −1       −1        0        P4800          GEOBRICK_ASYN}

        {LEBT     IRIS     APERTURE           "set an aperture"                          mm       1         76
                  A        0         0        0         0        P4801          GEOBRICK_ASYN}

        {LEBT     IRIS     VELOCITY           " velocity between 1 ( slow ) and 5 ( fast )"   mm/s     1         5
                  10∗A     5         0        0         0        P4803          GEOBRICK_ASYN}
        {LEBT     IRIS     OFFSET_X           "move the center of the      i r i s "     mm       −20       20
                  A        0         0        0         3        P4807          GEOBRICK_ASYN}
        {LEBT     IRIS     OFFSET_Y           "move the center of the      i r i s "     mm       −20       20
                  A        0         0        0         3        P4808          GEOBRICK_ASYN}
        {LEBT     IRIS     BLADES_KIND        "set    i r i s     blades kind"           boolean 0          1
                  A        0         0        0         0        P4838          GEOBRICK_ASYN}
}
```

*** console.template ***

```
f i l e     console . template
{
pattern {P          M         SPORT}
        {LEBT     IRIS      GEOBRICK_ASYN}
}
```

*** console.substitutions ***

```
f i l e     console . template
{
pattern {P          M         SPORT}
        {LEBT     IRIS      GEOBRICK_ASYN}
}
```