	PAUL SCHERRER INSTITUT	Date of creation	15/12/2008
Title	**MCU software application manual**	Date of last revision	03/04/2009
Authors	J.Destraves	Revision	2

This document is done for people that want to install or work with the MCU system.

The manual begins with a first introduction with an overview and a small move example. Then the application parameters are described. The last part explains how to set-up a motor and make advanced functionalities.

#	List of revision	Date	Authors
1	Initial revision	27/01/2009	JD
2	Homing mode and driver parameter upgrade	03/04/2009	JD

Table of Contents

List of figures

1. A brief overview of the system

The SINQ MCU is a motor controller system that fulfils the requirements of the SINQ Facility motion application. The system is principally based on several industrial parts setup in a 19" rack. The System should be enough flexible and versatile to cover all the SINQ application requirements. In this way the system is programmable, manage several IOs and support a large choice of motors.

MCU main features

- ✓ 3 architectures version:
  - MCU compact 208 8 axis high power
  - MCU HD 316 24 axis low power, limited IO
  - MCU control 416 16 axis controller interface
- ✓ Type of motor:
  - 2 phase stepper - 9Apeak,
  - 3 phase stepper - 6Apeak,
  - Brushless motor-under development
  - Brush motor-under development
- ✓ Maximum power for all motor simultaneously 650W (upgrade 1,3KW possible)
- ✓ IO Inputs Outputs management 10 IOs/axis :
  - Limits +/-, Home
  - Security/Emergency
  - Synchronisation trigger input/output (100mA)
  - Air cushion/break input/output (1A)
  - Inhibit/enable driver command and Error feedback.
- ✓ All IOs are 12-24V level opto-isolated.
- ✓ Standard position feedback supported : Incremental, SSI, Parallel.
- ✓ Expansion possible with resolver, analog 12bit, endat, sincos encoder, MTS, Hyperface.
- ✓ Remote control panel with embedded display and analogue command.
- ✓ Front display for local diagnoses.
- ✓ Ethernet 100MBps interface and IP protocol connectivity.
- ✓ Sequence programming: 1 task/axis available-on the fly programming possible.
- ✓ Work in user unit for goal positioning, acceleration and speed
- ✓ Position feedback close loop architecture
- ✓ Moving mode supported: Point to point – Relative – Jog
- ✓ Axis interpolation and synchronisation possible with special software.

2. MCU Documentation overview

2.1. Project documentation

- MCU Requirements Catalogue RC1 and System Requirements SR1
- MCU Decision making process report
- MCU Evaluation system report

2.2. User documentation

- **MCU USER Software manual**
- MCU Remote control manual

2.3. Hardware setup documentation

- MCU Hardware Manual
- MCU Remote control setup
- MCU Layouts and schematics
    - *IOs interface, feedback interface, Motor interfaces, Driver adaptor for backplane, display interface and driver backplane.*
- MCU Parts list of PSI parts
- MCU 208 internal interconnection overview
- MCU front panel definition
    - *Display, Drivers, Backside connectors*
- MCU interfaces specification
- Suppliers documentation
    - *Controller Layouts and manuals (Delta Tau)*
    - *Display manuals (ATMEL)*
    - *Driver manuals (Phytron,Berger Lahr,etc.)*

2.4. Software setup documentation

- MCU Software specification
- MCU Remote control setup
- Delta Tau documentation
    - IDE manual and Turbo SRM manual
- Source code MCU
- Source code Display
- Source code Remote control

3. Introduction

Letters meaning:

- **I**: System parameter (documentation supply by delta-tau in the SRM manual)
- **M**: Address of a physical hardware object (Output, Register, etc…)
- **P**: Application parameter containing an integer value
- **Q**: Application parameter containing an floating point value

There are 3 parameters categories:

- **[I,M,P,Q][1..2digit]** General parameter: Have an effect on the whole system or application.
    - Example ***P37*** *watchdog time of the remote control*
- **[I,M,P,Q][xx][0..2digit]**Axis sorted parameter or order: it will only affect the respective axis xx 1 to 32.
    - *Example **Ixx00** control the axis activation. I2600 motor 26 activation parameter.*
- **[I][mn][1..2digit]** CompactPCI ACC board parameter. The m is the board number from 2 to 10 (2 is board 1, 3 board 2, etc..) and the n is the axis channel number from 1 to 4 (4 axis /board).

Note: $ before figures means a hexadecimal value, if there is nothing then it is a decimal number.

4. How to order my first move

Order the motor 2 to make move back to 0

Terminal: Q201=0<CR>	*//set goal position for absolute move order Qxx01*
Terminal: M2 =1<CR>	*//Order an absolute move with Mxx=2*
Terminal: P200<CR>	*//Ask for Motor status with Pxx00*
Terminal:5	*//Motor is moving*
Terminal: P200<CR>	*//Ask for Motor status with Pxx00*
Terminal:0	*//Motor is arrived (Standby status*
Terminal: P201<CR>	*//Ask for Motor error status with Pxx01*
Terminal:0	*//no error occurred*

Note: you need a terminal able to communicate with delta tau interface. Look at the PWIN PRO SRM and the UMAC Turbo HRM to install the communication and IDE environment.

5. Software Architecture

The common way to communicate with the MCU is the Ethernet interface, but 2 serial interfaces are also use by the front display and the remote control (joystick). The basic operating system and motion application is store in ROM. The SINQ application is store in EEPROM as “compiled plcc” program.

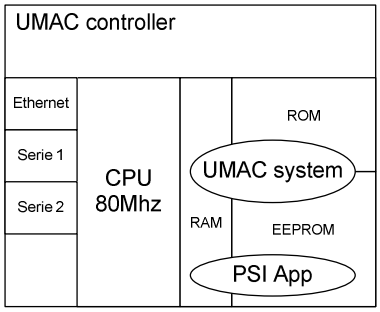


Figure 1 UMAC software architecture

6. Global view of the most important parameter

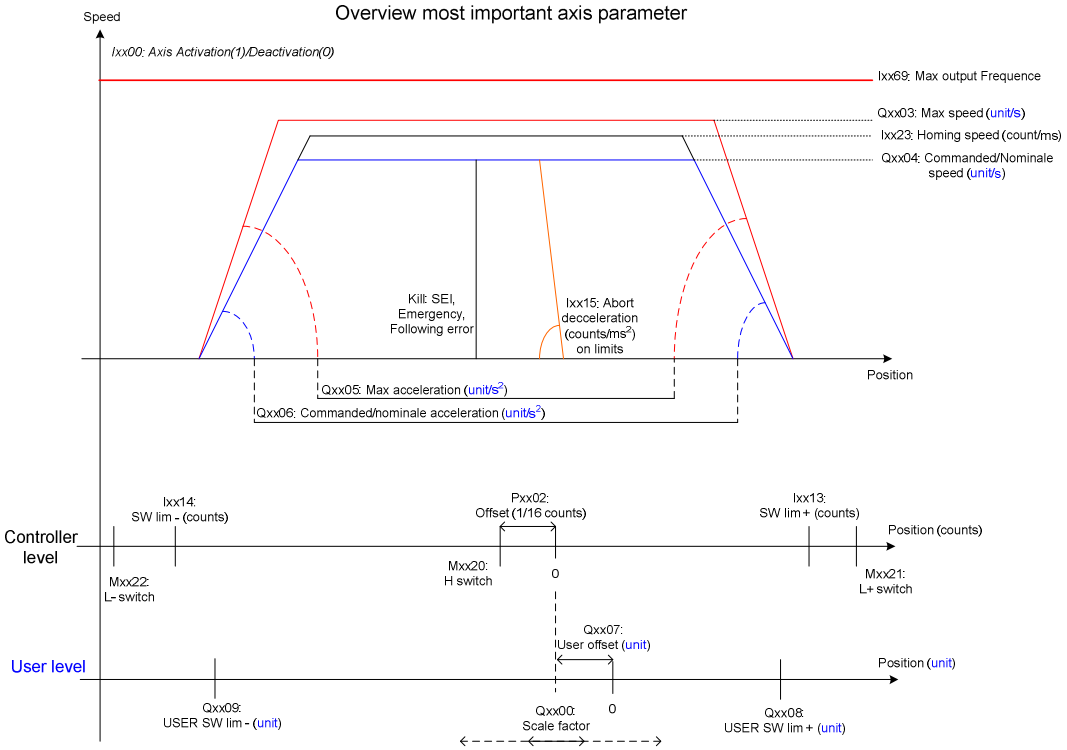


Figure 2 parameters overview

7. Parameters overview table

MCU parameters overview

xx=axis number 0-32 m=board number n=channel/axis number (1 board up to 4 channels)
TS Techn. Staff AD:Admin US:User
Status 1: rarely changed 2: could be changed 3: must be changed for each application

	status	axis parameter	fonction	Read/Write	Access level	scope	System name	default value	range	Comment
1	3	Motor activation		R/W	TS	axis #	Ixx00	1	0-1	motor desactivated
2	1	command output @		R/W	TS	axis #	Ixx02	-	$0-$FFFFFF	see IC board datasheet
3	2	position address	position feedback	R/W	TS	axis #	Ixx03	-	$0-$FFFFFF	
4	2	velocity address	= position address	R/W	TS	axis #	Ixx04	-	$0-$FFFFFF	
5	1	position scale factor		R/W	TS	axis #	Ixx08	96	0-8388607	
6	1	velocity scale factor	=position factor	R/W	TS	axis #	Ixx09	96	0-8388607	
7	2	Power on address	position feedback	R/W	TS	axis #	Ixx10	$0	$0-$FFFFFF	only for absolut encoder
8	3	following error	abort move if true	R/W	TS	axis #	Ixx11	32000	0-8388607	
9	3	Software Limit +		R/W	TS	axis #	Ixx13	0	-2^35 +2^35	depend of the application
10	3	Software limit -		R/W	TS	axis #	Ixx14	0	-2^35 +2^35	depend of the application
11	2	Abord deceleration	call on limit or abort move	R/W	TS	axis #	Ixx15	0.1	unsigned float	
12	1	S-curve time for jog		R/W	TS	axis #	Ixx21	10	0-8388607	
13	1	Home speed/direction		R/W	TS	axis #	Ixx23	|2|	float	
14	2	Flag mode control		R/W	TS	axis #	Ixx24	-	$0-$FFFFFF	
15	2	Flag address		R/W	TS	axis #	Ixx25	-	$0-$FFFFFF	
16	3	in position band		R/W	TS	axis #	Ixx28	15	0-8388607	
17	3	PID prop gain		R/W	TS	axis #	Ixx30	15000	+/-8388607	
18	2	PID deriv. gain		R/W	TS	axis #	Ixx31	0	+/-8388607	
19	2	PID int gain		R/W	TS	axis #	Ixx33	0	+/-8388607	
20	3	Deadband gain		R/W	TS	axis #	Ixx64	-16	+/-32767	
21	3	Deadband size		R/W	TS	axis #	Ixx65	8	+/-32767	
22	3	ouput command limit	max frequence for an axis	R/W	TS	axis #	Ixx69	500	0-32767	about 2,3Khz
23	2	Power on position format		R/W	TS	axis #	Ixx95	$0	$0-$FFFFFF	only for absolut encoder
24	3	Incremental encoder param.	decode type	R/W	TS	axis #	I7mn0	8	0-15	open loop
25	1	gated index		R/W	TS	axis #	I7mn4	0	0-1	
26	1	getex index control		R/W	TS	axis #	I7mn5	0	0-3	
27	1	output mode select	DAC/PWM/PFM	R/W	TS	axis #	I7mn6	2	0-3	PWM+PFM
28	2	Invert output control	pulse invertion	R/W	TS	axis #	I7mn7	0	0-3	no invertion
29	2	PFM signal invertion	direction invertion	R/W	TS	axis #	I7mn8	0	0-1	no invertion
30	1	Board frequency param.	max phase frequency set	R/W	TS	IC board	I7m00	6527	0-32767	about 9Khz
31	1	Board frequency param.	max phase frequency divider	R/W	TS	IC board	I7m01	0	0-15	about 9Khz
32	1	Board frequency param.	Servo clock divider	R/W	TS	IC board	I7m02	3	0-15	about 2.25KHz
33	2	Max Ouput Frequency	Max Frequency for motors	R/W	TS	IC board	I7m03	7(2298)	1-7	
34	2	Pulse wilde control	pulse wilde definition	R/W	TS	IC board	I7m04	1	0-255	about 3us
35	1	Serial port mode	handshake protocole	R/W	TS	system	I1	1	0-3	no handshake
36	1	Serial port control	handshake char select	R/W	TS	system	I3	2	0-3	<CR>+<ACK>
37	1	communication mode	integrity mode	R/W	TS	system	I4	0	0-3	checksum disable
38	2	PLC control	start/stop plc's	R/W	TS	system	I5	3	0-3	plc's and plcc's on
39	1	Error reporting mode		R/W	TS	system	I6	1	0-3	
40	1	RTI period	real time interrupt period	R/W	TS	system	I8	2	0-255	
41	1	Watchdog sensivity		R/W	TS	system	I40	0	0-65000	
42	1	Aux. serial port baudrate	baud rate aux port	R/W	TS	system	I53	12	0-15	38400 bauds
43	1	Main serial port baudrate	baud rate aux port	R/W	TS	system	I53	12	0-15	38400 bauds
44	3	Scale factor	user unit sclae	R/W	TS	axis #	Qxx00	0	float	48 bit floating point
45	3	Absolut goal position	in user unit	R/W	US	axis #	Qxx01	0	float	48 bit floating point
46	3	Relative goal position	in user unit	R/W	US	axis #	Qxx02	0	float	48 bit floating point
47	3	Max Speed		R/W	TS	axis #	Qxx03	500	float	48 bit floating point
48	3	Commanded speed	axis nominale speed	R/W	AD	axis #	Qxx04	500	float	48 bit floating point
49	3	Max acceleration		R/W	TS	axis #	Qxx05	0	float	48 bit floating point
50	3	Commanded acceleration	axis nominale acceleration	R/W	AD	axis #	Qxx06	10	float	48 bit floating point
51	3	Offset user scale	offset in user unit	R/W	AD	axis #	Qxx07	0	float	48 bit floating point
52	3	User SW Limit +	in user unit	R/W	AD	axis #	Qxx08	0	float	48 bit floating point
53	3	User SW Limit -	in user unit	R/W	AD	axis #	Qxx09	0	float	48 bit floating point
54	3	current position	in user unit	R	US	axis #	Qxx10	0	float	48 bit floating point
55	3	current speed	in user unit	R	US	axis #	Qxx11	0	float	48 bit floating point
56	1	watchdog delay	for joystick	R	TS	system	p37	0	float	integer
57	1	locked mcu		R/W	TS	system	p42	0	1	integer
58	1	locked jog		R/W	TS	system	p43	0	1	integer
59	3	instrument name char 1		R/W	TS	system	p47	0	ASCII	integer
60	3	instrument name char 2		R/W	TS	system	p48	0	ASCII	integer

61	3	instrument name char 3		R/W	TS	system	p49	0	ASCII	integer
62	3	instrument name char 4		R/W	TS	system	p50	0	ASCII	integer
63	3	instrument name char 5		R/W	TS	system	p51	0	ASCII	integer
64	3	instrument name char 6		R/W	TS	system	p52	0	ASCII	integer
65	3	instrument name char 7		R/W	TS	system	p53	0	ASCII	integer
66	3	instrument name char 8		R/W	TS	system	p54	0	ASCII	integer
67	3	instrument name char 9		R/W	TS	system	p55	0	ASCII	integer
68	3	instrument name char 10		R/W	TS	system	p56	0	ASCII	integer
69	3	mcu name char 1		R/W	TS	system	p57	0	ASCII	integer
70	3	mcu name char 2		R/W	TS	system	p58	0	ASCII	integer
71	3	mcu name char 3		R/W	TS	system	p59	0	ASCII	integer
72	3	mcu name char 4		R/W	TS	system	p60	0	ASCII	integer
73	3	mcu name char 5		R/W	TS	system	p61	0	ASCII	integer
74	3	mcu number		R/W	TS	system	p62	0	ASCII	integer
75	3	Axis status		R	US	axis #	Pxx00	0	-6->14	integer
76	3	Error status		R	US	axis #	Pxx01	0	0->13	integer
77	3	Auto homing offset	controller UMAC offset in step	R/W	TS	axis #	Pxx02	0	float	integer
78	3	ACO definition	define air cushion address	R/W	TS	axis #	Pxx03	0	2^16	integer
79	3	homing type	define homing type	R/W	TS	axis #	Pxx04	0	8191	integer
80	1	synchro input activation		R/W	TS	axis #	Pxx05	0	2^16	integer
81	2	backlash value		R/W	TS	axis #	Pxx06	0	integer	integer
82	3	jog speed		R/W	TS	axis #	Pxx07	1	100	integer
83	3	after coma display		R/W	TS	axis #	Pxx08	2	13	integer
84	3	axis name char 1		R/W	AD	axis #	Pxx09	110	ASCII	integer
85	3	axis name char 2		R/W	AD	axis #	Pxx10	46	ASCII	integer
86	3	axis name char 3		R/W	AD	axis #	Pxx11	99	ASCII	integer
87	3	axis name char 4		R/W	AD	axis #	Pxx12	46	ASCII	integer
88	3	axis name char 5		R/W	AD	axis #	Pxx13	0	ASCII	integer
89	3	axis name char 6		R/W	AD	axis #	Pxx14	0	ASCII	integer
90	3	axis name char 7		R/W	AD	axis #	Pxx15	0	ASCII	integer
91	3	axis name char 8		R/W	AD	axis #	Pxx16	0	ASCII	integer
92	3	unit definition	m/dm/mm/um/deg/rad/cts	R/W	AD	axis #	Pxx17	0	0-7	integer
93	2	syi status		R	TS	axis #	Pxx21	0	0-1	integer
94	2	ACO delay		R	TS	axis #	Pxx22	0	100000	integer
95	2	Move Ack	move acknoledgment	R	AD	axis #	Pxx23	0	1	integer
96	3	driver-current	switch configuration	R/W	AD	axis #	Pxx24	1	16	integer
97	3	micro-stepping mode	switch configuration	R/W	AD	axis #	Pxx25	1	16	integer
98	3	current autoreduction	switch configuration	R/W	AD	axis #	Pxx26	0	1	integer
99	1	speed reduction by homing	after limit detect	R/W	AD	axis #	Pxx27	1	-	integer
100	1	FreeVariable 1	for sequence programming	R/W	TS	axis #	Pxx50	0	-	integer
101	1	FreeVariable 2	for sequence programming	R/W	TS	axis #	Pxx51	0	-	integer
102	1	FreeVariable 3	for sequence programming	R/W	TS	axis #	Pxx52	0	-	integer
103	1	FreeVariable 4	for sequence programming	R/W	TS	axis #	Pxx53	0	-	integer
104	1	FreeVariable 5	for sequence programming	R/W	TS	axis #	Pxx54	0	-	integer
105	1	FreeVariable 6	for sequence programming	R/W	TS	axis #	Pxx55	0	-	integer
106	1	FreeVariable 7	for sequence programming	R/W	TS	axis #	Pxx56	0	-	integer
107	1	FreeVariable 8	for sequence programming	R/W	TS	axis #	Pxx57	0	-	integer
108	1	FreeVariable 9	for sequence programming	R/W	TS	axis #	Pxx58	0	-	integer
109	1	FreeVariable 10	for sequence programming	R/W	TS	axis #	Pxx59	0	-	integer
110	1	FreeVariable 11	for sequence programming	R/W	TS	axis #	Qxx50	0	-	float
111	1	FreeVariable 12	for sequence programming	R/W	TS	axis #	Qxx51	0	-	float
112	1	FreeVariable 13	for sequence programming	R/W	TS	axis #	Qxx52	0	-	float
113	1	FreeVariable 14	for sequence programming	R/W	TS	axis #	Qxx53	0	-	float
114	1	FreeVariable 15	for sequence programming	R/W	TS	axis #	Qxx54	0	-	float
115	1	FreeVariable 16	for sequence programming	R/W	TS	axis #	Qxx55	0	-	float
116	1	FreeVariable 17	for sequence programming	R/W	TS	axis #	Qxx56	0	-	float
117	1	FreeVariable 18	for sequence programming	R/W	TS	axis #	Qxx57	0	-	float
118	1	FreeVariable 19	for sequence programming	R/W	TS	axis #	Qxx58	0	-	float
119	1	FreeVariable 20	for sequence programming	R/W	TS	axis #	Qxx59	0	-	float
120	2	Inhibit output	motor current inhibit output	R/W	AD	axis #	Mxx14	-	0-1	integer
121	2	Home input		R	AD	axis #	Mxx20	-	0-1	integer
122	2	Hardware limit + input		R	AD	axis #	Mxx21	-	0-1	integer
123	2	Hardware limit - input		R	AD	axis #	Mxx22	-	0-1	integer
124	2	SEI	SEI or emergency	R	AD	axis #	Mxx23	-	0-1	integer
125	2	Air cushion input		R	AD	axis #	Mxx95	-	0-1	integer
126	2	Air cushion output		R/W	AD	axis #	Mxx96	-	0-1	integer
127	2	Synchronisation input		R	AD	axis #	Mxx97	-	0-1	integer
128	2	Synchronisation output		R/W	AD	axis #	Mxx98	-	0-1	integer
129	2	Default/Error input	from driver error output	R	AD	axis #	Mxx15	-	0-1	integer
130	3	move command		R/W	US	system	Mxx	0	1-9	integer

Figure 3 MCU parameters overview

8. Parameters and commands

8.1.1.1. Mxx: Move command

Name	Write level	default	Unit	range
Mxx	USER	0	-	1..9

Mxx is a command that order a move of different types allowed by the MCU application. Absolute, relative, abort, homing are the standard move.

The jog mode is also possible but reserved so far for the remote control application.

- Mxx=1 -> Absolute move
- Mxx=2 -> Relative move
- Mxx=3 -> Start jog mode (reserved for the remote control)
- Mxx=4 -> Jog forward (reserved for the remote control)
- Mxx=5 -> Jog backward (reserved for the remote control)
- Mxx=6 -> Break jog (reserved for the remote control)
- Mxx=7 -> Stop jog mode (reserved for the remote control)
- Mxx=8 -> Abort move
- Mxx=9 -> Homing

Mxx=1: Start an absolute move with the goal value place in Qxx01

Mxx=2: Start a relative move with the goal value place in Qxx02

Mxx=8: Stop the move with a standard deceleration defines by Qxx06 and init the status and error to 0.

Mxx=9: Start homing procedure. The homing type is defined by the parameters Pxx02 and Pxx04 both are only once define (during the instrument installation). It means that you can call the Mxx=9 without any other condition if your system is already setup.

Mxx=3/4/5/6/7: Is a special mode use for a remote control. Manually it is not possible to use it because of the watchdog control. The commands must cyclically to be sent in order to detect a communication default. The parameter P37 is the watchdog value, time allowed between to sent commands.

When you enter in jog mode **Mxx=3** the watchdog is activated and the IO like air cushion are managed. Then you can send **Mxx=4 or Mxx=5 or Mxx=6** to move your axe forward or backward or to make a break (stop the move). The speed of the motor during this mode is defined by Pxx07 (default 100) that is the % value of the nominal speed Qxx04. The speed can be change during moving.

Call **Mxx=7** to stop the mode. **During a jog mode no other moves commands are allowed by the MCU.**

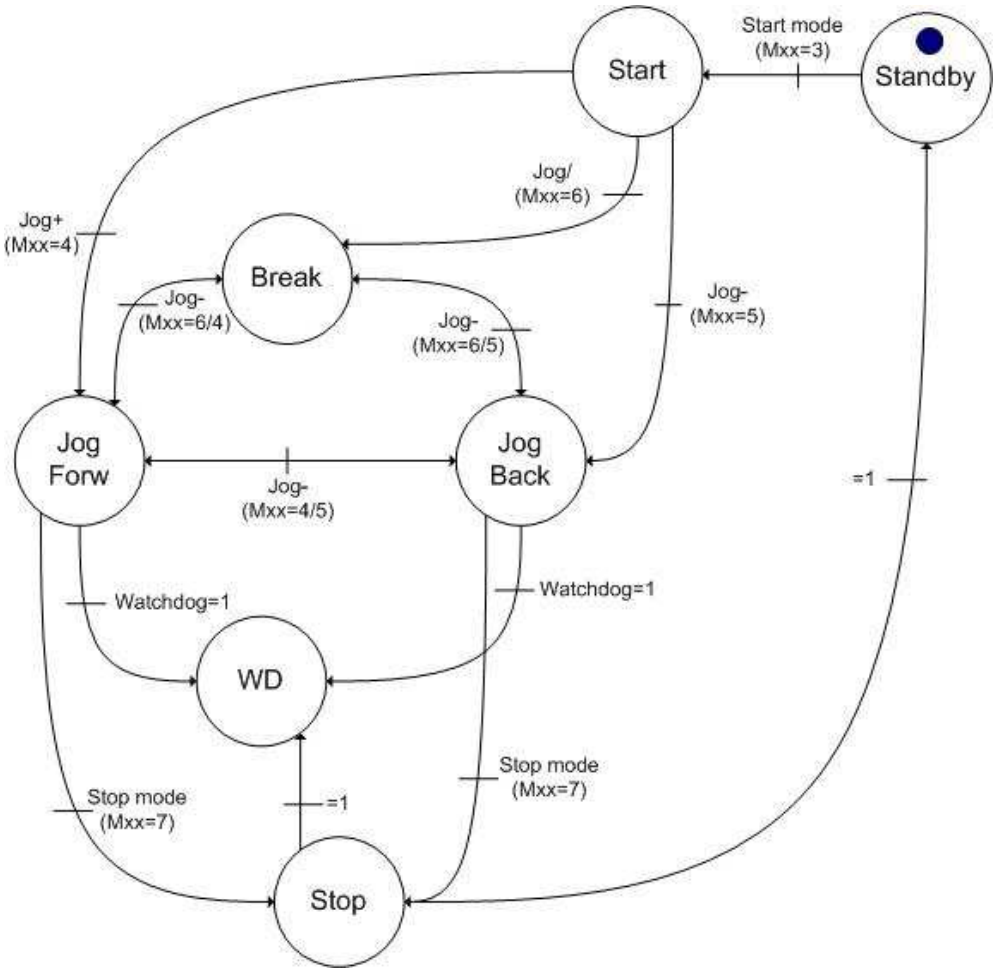


Figure 4 State diagram of the jog mode

8.2. Axis parameters

8.2.1.1. Qxx00: scale factor

Name	Write level	default	Unit	range
Qxx00	AD	1	-	float

This parameter defines the scale of the axis in order to work in user unit (degree,rad,mm,etc..). Qxx00=Distance(unit)/Resolution(steps).

The affected parameters by the scale are Qxx01 to Qxx09, If you make any change of the scale factor you must also change this parameters with this new factor.

Example:

Motor stepper with 200 steps/rev.

Driver micro-step 1/20

Gear ratio 1:500

*Qxx00= 360/ (200*20*500)=0.00018*

The axis works now in degree.

Note: When you use a feedback the scale calculation should take the encoder resolution than the motor resolution.

8.2.1.2. Qxx01: Absolute goal position

Name	Write level	default	Unit	range
Qxx01	US	0	User unit	float

Define the goal position the value is not an order you have to call Mxx=1 to start your move. This parameter works in user unit. The value stays in memory until the next write.

8.2.1.3. Qxx02: Relative goal position

Name	Write level	default	Unit	range
Qxx02	US	0	User unit	float

Define the goal position for a move relative to the current position, the value is not an order you have to call Mxx=2 to start your move. This parameter works in user unit. The value stays in memory until the next write.

8.2.1.4. Qxx03: Maximal speed

Name	Write level	default	Unit	range
Qxx03	TS	500	unit/s	float

This represents the maximum speed reachable by the motor. The default value represent 500Hz in open loop mode and if the scale Qxx00=1. This value is in **unit/s.**

8.2.1.5. Qxx04: Nominal speed

Name	Write level	default	Unit	range
Qxx04	AD	500	unit/s	float

Nominal speed < or = Maximal speed. This is the actual speed the motor use for a move, it is also in unit/s. Default value is set to the maximum (500).

8.2.1.6. Qxx05: Maximal acceleration

Name	Write level	default	Unit	range
Qxx05	TS	10	unit/s^2	float

This represents the maximum acceleration allowed by the motor. The default value represent 500Hz in open loop mode if the scale Qxx00=1. This value is in **unit/s^2.** If you write the parameter higher than the maximum then it is Nominal=Max.

8.2.1.7. Qxx06: Nominal acceleration

Name	Write level	default	Unit	range
Qxx06	AD	10	unit/s^2	float

Nominal acceleration < or = Maximal acceleration. This is the actual acceleration the motor use; it is also in **unit/s^2**. Default value is set to the maximum (10). If you write higher than the maximum then Nominal=Max.

8.2.1.8. Qxx07: User Offset

Name	Write level	default	Unit	range
Qxx07	AD	10	User unit	float

This parameter defines the offset between the 0 position of the “hardware position reference”. This can not be change during a move and you must also care to adapt your User software limits Qxx08/Qxx09.

8.2.1.9. Qxx08: User software limit +

Name	Write level	default	Unit	range
Qxx08	AD	0	User unit	float

This limit can be change at any time and it allows controlling the validity of a move before starting. If the goal position will over run the limit the move is cancelled and the error status Pxx01=1. If the value is 0 as well as the limit– then the function is not activated. (The hardware limit and controller SW limit still working).

8.2.1.10. Qxx09: User software limit -

Name	Write level	default	Unit	range
Qxx09	AD	0	User unit	float

This limit can be change at any time and it allows controlling the validity of a move before a start. If the goal position will over run the limit the move is cancelled and the error status Pxx01=1. If the value is 0 as well as the limit+ then the function is not activated. (The hardware limit and controller SW limit still working).

8.2.1.11. Pxx00: Move status

Name	Write level	default	Unit	range
Pxx00	Read only	0	-	-6..11

Read only parameter that give information about the motor state.

Pxx00=-6: **ABORT** status, the controller is cancelling the respective move, it decelerates the speed to 0, this will occur after a Mxx=8 command.

Pxx00=-5: **DEACTIVATED** status, the axis is not active, it means that no motor are plug in the corresponding slot. No move command will be interpreted.

Pxx00=-4: **EMERGENCY** status, no commands are allowed during this status. Release the emergency switch to go out from this state. All axis are concerned by this status.

Pxx00=-3: **INHIBIT** status, the motor current is 0 Amperes and no commands are allowed, this mode is triggered from the front display, the output address Mxx14 or the remote control.

Pxx00=-2: **MOVE LOCKED** status, the controller will not interpret any move commands. This state is activated with the P42 parameter.

Pxx00=-1: **JOG LOCKED** status, the controller will not interpret any move commands but the jog command still allowed. This state is activated with the P43 parameter.

Pxx00=0: **STANDBY** status, motor ready to move

Pxx00=1: **ORDER ACK** status, the move command is allowed (no Emergeny, no Inhibit, no locked, etc…)

Pxx00=2: **ORDER VALID** status, the move is possible, limits are checked.

Pxx00=3: **ACO** status, the **A**ir **C**ushion **O**utput status manages outputs if the Pxx03 is set.

Pxx00=4: **ACI** status, **A**ir **C**ushion **I**nput, check the feedback with a define delay Pxx22 before starting the motor.

Pxx00=5: **MOVING** status the motor is started, axis is running

Pxx00=6: **JOG START** status, jog mode started (reserved for remote control)

Pxx00=7: **JOG FORWARD** status, (reserved for remote control)

Pxx00=8: **JOG BACKWARD** status, (reserved for remote control)

Pxx00=9: **JOG BREAK** status, (reserved for remote control)

Pxx00=10: **JOG STOP** status, jog mode stopped (reserved for remote control)

Pxx00=11: BACKLASH status, the axis is doing the backlash move according to the Pxx06. If Pxx06=0 no backlash will occur.

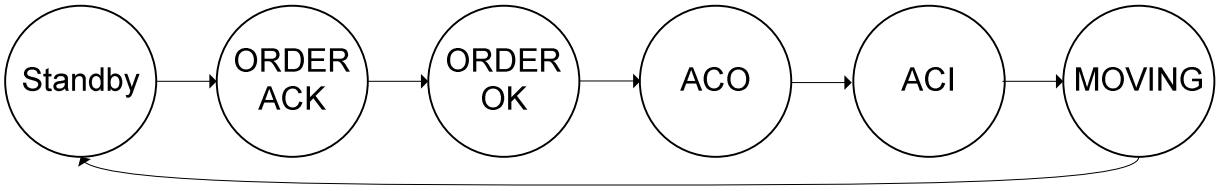


Figure 5 A typical move status sequence

8.2.1.12. Pxx01: Move error status

Name	Write level	default	Unit	range
Pxx01	Read only	0	-	0..13

This status gives the raison when a move do not start or finished properly. Several types of errors are detected by the controller and they are listed as below. The error status is automatically reset after an acknowledged order. The controller just logs the last occurred error.

Pxx01=0: NO ERROR

Pxx01=1: USER SOFTWARE LIMIT, the move order exceeds the user sw. limits Qxx08 Qxx09.

Pxx01=2: JOG1, a jog order Mxx=7,8,9,10 is asked before to start the jog mode.

Pxx01=3: JOG2, a jog start mode orders Mxx=6 but another axis is already running in jog mode. Only 1 axis in Jog mode is allowed.

Pxx01=4: JOG3, a jog order Mxx=1,2 is asked but 1 axis is in jog mode.

Pxx01=5: ORDER, Mxx value is out of range.

Pxx01=6: WATCHDOG, the Jog mode watchdog is triggered.

Pxx01=7: CURRENT, Output current over the limit allowed, too much IO are driven in parallel.

Pxx01=8: ACI1, Air cushion feedback stop during the move.

Pxx01=9: ACI2, no Air cushion feedback before to start the move.

Pxx01=10: SOFTWARE LIMITS, the move reach the software limit of the controller, it could be the hardware switch Mxx32/Mxx31 as well as the controller software limit Ixx13/Ixx14.

Pxx01=11: FOLLOWING, problem with the close loop. Motor or encoder problem, the feedback value overruns the limit Ixx11 between the actual position and the commanded position.

Pxx01=12: SEI, Security input is triggered, this input stop (kill) the axis, the address off the Input is Mxx23.

Pxx01=13: DEFAULT DRIVER, The default signal from the motor driver is on. Refer to the driver manual in order to identify the problem. It could be a temperature limit reach, an electrical short-cut, voltage or current over limits, etc…

8.2.1.13. Pxx02: Homing offset

Name	Write level	default	Unit	range
Pxx02	TS	0	1/16 counts	Signed integer

If this parameter is different from is default value the motor will make an offset at this end of the homing procedure. If the value is positive it will move the motor forward if negative backward.

8.2.1.14. Pxx03: ACO Air Cushion Output definition

Name	Write level	default	Unit	range
Pxx03	TS	0	-	Unsigned integer

1 output / axis is dedicated as Air cushion output functionality. The ACO address is Mxx96. They are for the compact version the MCU supply 8 outputs and for the control version 16 outputs.

Each axis can manage 1 until 8/16 Outputs. The Action of a move start the Air cushion when the move is finished the controller stop each output **only if no other running axis use the same output**. Each output could be affect to several axis.

- Pxx03-bit0 -> ACO 1
- Pxx03-bit1 -> ACO 2
- Pxx03-bit2 -> ACO 3
- Pxx03-bit3 -> ACO 4
- Etc….

Example of setup:

- Axis 2 manage ACO 2 and ACO 3 -> ***P203****=2^1 + 2^2=****6***

- Axis 5 manage ACO 2 and ACO 3 and ACO 5 -> ***P503****=2^1 + 2^2 + 2^4=****22***

- Axis 8 manage ACO 3 and ACO 8 -> ***P803****=2^2 + 2^7=****132***

8.2.1.15. Pxx04: Homing type definition

Name	Write level	default	Unit	range
Pxx04	TS	0	-	Unsigned integer

- Pxx04=0 -> Set position to zero immediately (manual homing)
- Bit0 -> Homing on HOME input
- Bit1 -> Homing on LIMIT+ input
- Bit2 -> Homing on LIMIT- input
- Bit3 -> Homing with Encoder index
- Bit4 -> Homing with HOME&DIR input
- Bit5 -> HOME&DIR logic definition
- Bit6 -> Index search direction definition: 0 backward 1 forward
- Bit7 -> Homing input search direction definition: 0 backward 1 forward
- Bit8 -> Index level sensitive 0-low 1-high
- Bit9 -> Flag level sensitive 0-low 1-high
- Bit10 -> HOME&DIR Direction setup
- Bit11 -> no homing needed (for absolute encoder)

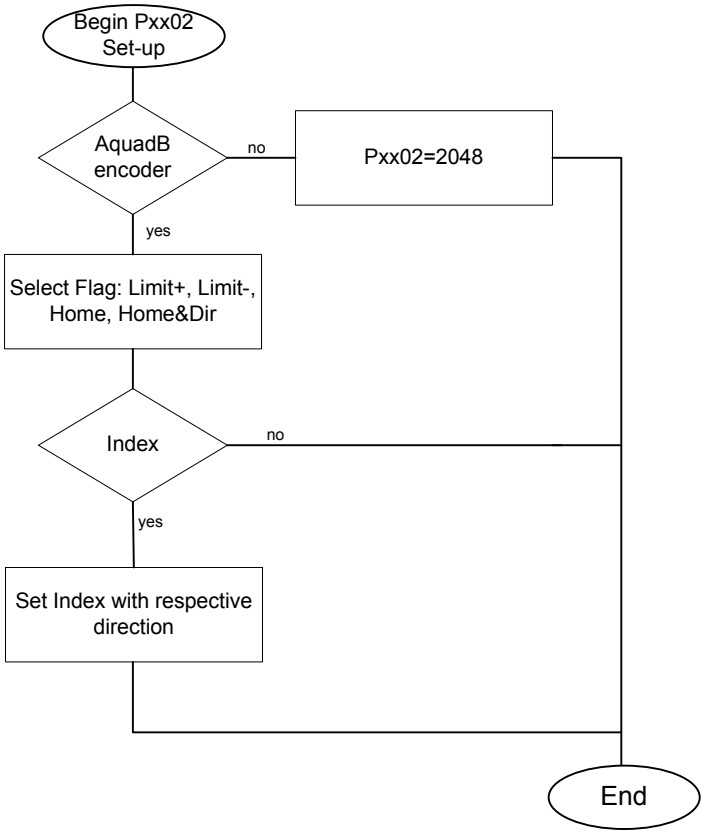


Figure 6 Homing configuration flow

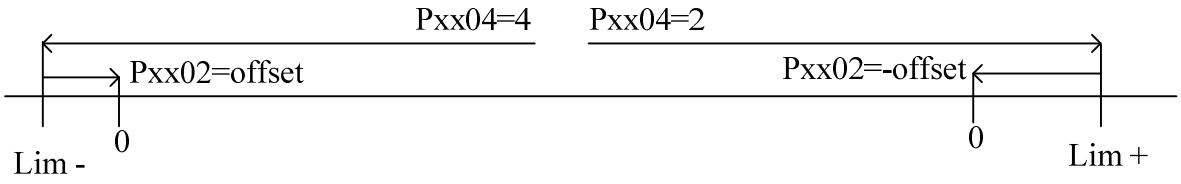


Figure 7 Homing on limit switch

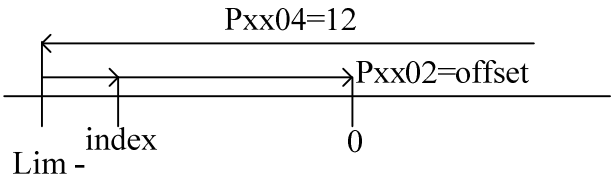


Figure 8 Homing example with index

#	Configuration possibilty (b:bit)	Comment
1	=0	immediat zero writing
2	b0.b7	Home flag and direction start selection
3	b0.b7.b3.b6.b8	#1 + index detection with index direction selection and index sensitive level
4	(b1+b2).b9	Limite selection flag with level sensitive selection
5	(b1+b2).b9.b3.b6.b8	#4 inclusive index definition
6	b4.b5.b9	home&direction flag with level and direction logic definition
7	b4.b5.b9.b3.b6.b8	#6 inclusive index definition
8	b1.b9.b12	limit- then home flag search, Pxx27 divide the speed for home flag research
9	b11	nothing is done

Figure 9 Homing allowed configuration

8.2.1.16. Pxx04: Synchronisation configuration

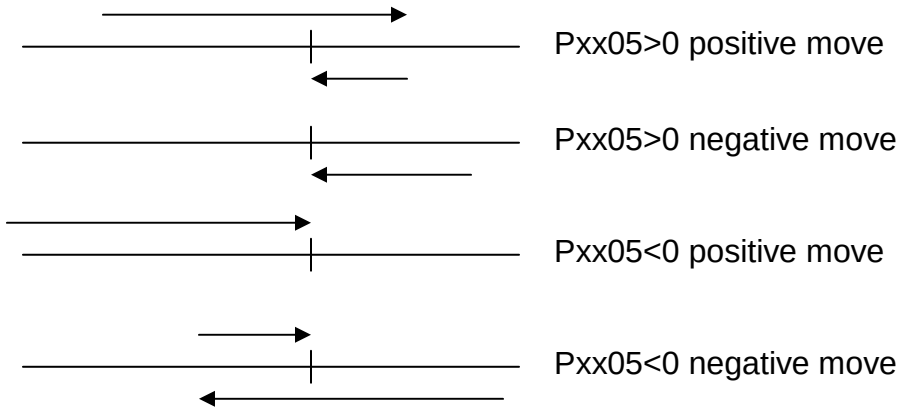
Name	Write level	default	Unit	range
Pxx05	-	-	-	-

Not implemented.

8.2.1.17. Pxx06: Backlash value

Name	Write level	default	Unit	range
Pxx06	TS	0	counts	Signed integer

If the value is not equal to 0 a move will always arrive to its position with the same direction. The value represents the numbers of count add to the goal position in order to move back to the goal.



8.2.1.18. Pxx07: JOG speed

Name	Write level	default	Unit	range
Pxx07	TS	0	%	Integer 0..100

In jog mode this parameter defines the speed the motor should take, 100 corresponding to 100% of the nominal speed Qxx04. The JOG mode will not change the value of Qxx04 at the end of a JOG the motor will work with its nominal value. This parameter is useful if you drive motor with an analogue command for example.

8.2.1.19. Pxx08: After coma digits

Name	Write level	default	Unit	range
Pxx08	AD	2	-	Integer 0..13

Define the number of digits to display after the coma. This parameter could be read by any host in order to know its display format.

8.2.1.20. Pxx09 to Pxx16: Motor name

Name	Write level	default	Unit	range
Pxx09..16	AD	0	-	ASCII

This 8 characters define the motor name, its an ASCII value, each parameter corresponding to one character.

8.2.1.21. Pxx17: Motor name

Name	Write level	default	Unit	range
Pxx17	AD	7	-	Integer 0..7

This value define the unit of the axis, the default value is 7 and means that the system work in count but you can define another unit with the following definition

- Pxx17=0 -> no unit
- Pxx17=1 -> m (meter)
- Pxx17=2 -> dm (decimetre)
- Pxx17=3 -> mm (millimetre)
- Pxx17=4 -> um (micrometer)
- Pxx17=5 -> deg (degree)
- Pxx17=6 -> rad (radian)
- Pxx17=7 -> counts (meter)

8.2.1.22. Pxx18 to Pxx21: Reserved

Name	Write level	default	Unit	range
Pxx18-20	-	-	-	-

8.2.1.23. Pxx22: ACO delay

Name	Write level	default	Unit	range
Pxx22	TS	0	ms	integer

Define a delay between ACO activation and move starting. If Pxx22=200 it will set a delay of 200ms.

8.2.1.24. Pxx23: Moving acknowledgement

Name	Write level	default	Unit	range
Pxx23	US	0	-	0-1

The controller only writes this parameter when motor is starting a move. It should be use to control that the motor is gone. This parameter has to be reset by the host. This will allows an asynchronous control of a move.

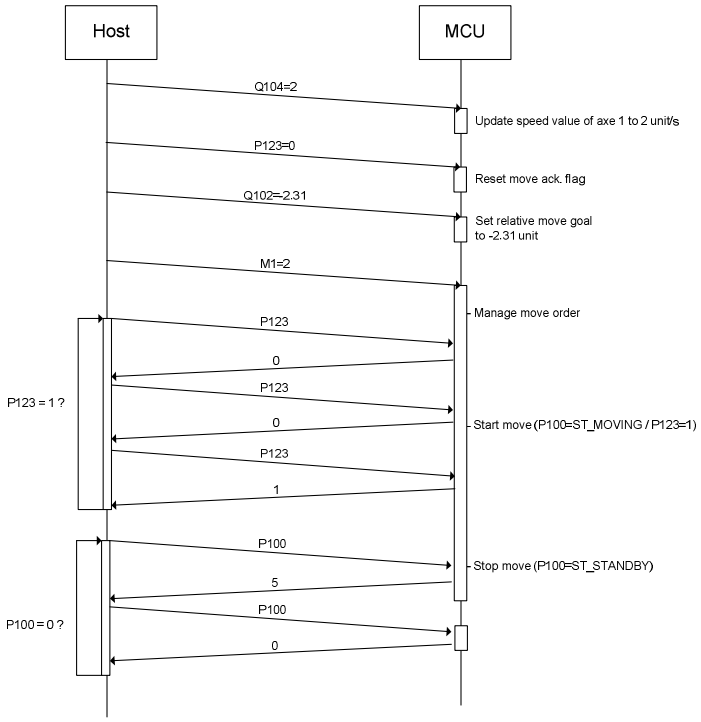


Figure 10 Asynchronous host control of a move

8.2.1.25. Pxx24: Driver current

Name	Write level	default	Unit	range
Pxx24	TS	0	no	0-16

Parameter only here for information, its should indicate the setting current in the driver interface

8.2.1.26. Pxx25: Microstepping

Name	Write level	default	Unit	range
Pxx25	TS	0	no	0-16

Parameter only here for information, its should indicate the setting micro-stepping mode in the driver interface

8.2.1.27. Pxx26: Auto current reduction

Name	Write level	default	Unit	range
Pxx25	TS	0	-	0-1

Parameter only here for information, it should indicate if the automatic current reduction from the motor driver is activated or not.

8.2.1.28. Pxx27: Homing speed divider

Name	Write level	default	Unit	range
Pxx27	TS	0	-	integer

This parameter is only used for the homing bit 12 mode (see Pxx04), when the procedure is looking for the Home flag the nominal Ixx23 speed will be divided by Pxx27 in order to enhance (depend of the sensor) the repeatability of the homing. It permit to find limit flag with an High and the home flag with a low speed.

8.2.1.29. Pxx50 to Pxx59: Free variable for sequences

Name	Write level	default	Unit	range
Pxx18-20	AD	0	-	Integer

They are 10 integer variables per axis reserved to write programming sequence. Each Axis has a reserved memory program that work in parallel to the other. The program space is a PLC xx program. A PLC program is automatically started after a Reset. To get more info look at the chapter MCU sequence programming.

8.2.1.30. Qxx50 to Pxx59: Free variable for sequences

Name	Write level	default	Unit	range
Pxx18-20	AD	0	-	Float

They are 10 also floating point variables per axis reserved to write programming sequence.

8.3. Inputs / Outputs address definition

8.3.1.1. Mxx14, Mxx20 to Mxx23, Mxx95 to Mxx96: IO definition

You can read an I or Read/Write an Input with the following Address

- Mxx14 -> IHO: Inhibit output
- Mxx20 -> H: Home input
- Mxx21 -> L+: Hardware limit + input
- Mxx22 -> L-: Hardware limit - input
- Mxx23 -> SEI: Security input - input 1 define as Emergency (default configuration)
- Mxx95 -> ACI: Air Cushion Input
- Mxx96 -> ACO: Air Cushion Output
- Mxx97 -> SYI: Synchronisation Input
- Mxx98 -> SYO: Synchronisation Output
- Mxx15 -> ER: Driver error input

Example:

Terminal: M521<CR> *//Ask for hardware limit switch + value*
Terminal: 0 *//Switch not active*

Terminal: M296<CR> *//Ask for ACO*
Terminal: 0 *//ACO not active*
Terminal: M296=1<CR> *//Turn ACO on*
Terminal: M296<CR> *//Read back*
Terminal: 1 *//ACO activated*

8.4. Application relative parameter

8.4.1.1. P37: Watchdog delay

Name	Write level	default	Unit	range
P37	TS	120	ms	integer

It represents the maximum time limit between two commands sent. If the order come to late the jog mode is stopped, the move is also stopped and the Error status Pxx02 is written. This parameter is a security relevant parameter; bigger is the value longer will be the reaction time after a lost of communication between the host and controller.

8.4.1.2. P41: MCU lock

Name	Write level	default	Unit	range
P41	AD	0	-	boolean

If equal to 1 this parameter lock the MCU, any move commands Mxx will not be taken inconsideration. It could be useful if there is a local manipulation and you want to be sure that nobody drive any axis.

8.4.1.3. P42: MCU jog lock

Name	Write level	default	Unit	range
P42	AD	0	-	boolean

If equal to 1 this parameter lock the MCU, any move commands Mxx will not be taken inconsideration but not the jog command. If you have a remote control, you can drive your motor and you are sure that anybody can order any other axis.

8.4.1.4. P47 to P56: Instrument name

Name	Write level	default	Unit	range
P47 to P56	AD	0	-	ASCII

Define the instrument name; it could be read by local display or remote control.

8.4.1.5. P57 to P61: MCU name

Name	Write level	default	Unit	range
P57 to P61	AD	0	-	ASCII

Define the MCU name; it could be read by local display or remote control.

8.4.1.6. P62: MCU number

Name	Write level	default	Unit	range
P62	AD	0	-	integer

Define the instrument number; it could be read by local display or remote control.

9. UMAC System variable: Ixx Parameter

Ixx parameters are standard UMAC (delta tau controller) defined parameter, the exhaustive documentation of it is in the SRM Software manual of the UMAC system. A lot of parameters are defined but just a small part (about 50) will be use by our application. This documentation will not describe all of them but just give the most important and also gives a complementary information compare to the SRM manual.

9.1.1.1. Ixx00: Axis activation

Name	Write level	default	Unit	range
Ixx00	TS	1	-	Boolean

Axis is activated as Default! Then it allows without any configuration to drive the motor with the remote control with the standard speed and acceleration. If some axes are not connected you should set the parameter to 0, it will update the motor status to DESACTIVATED. When an axis is not active the Order and IO are not interpreted or read.

9.1.1.2. Ixx03: Position address

Name	Write level	default	Unit	range
Ixx03	TS	$350xx	-	address

Define the position feedback address from the encoder conversion table. By use of incremental encoder Ixx03=$350xx. For special feedback you should find the respective address in the encoder conversion table. The encoder conversion table list all the feedback of the system with its respective format. Look at the dedicated chapter to get more information.

Example

- I503=$3505 //encoder channel 5 connected to motor 5
- I503=$3501 //encoder channel 1 connected to motor 5
- I303=$352A //encoder channel 42 connected to motor 3

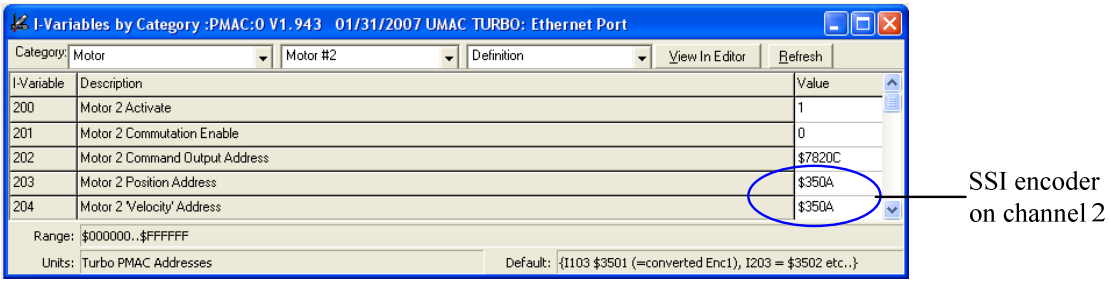


Figure 11 feedback configuration example (evaluation system)

9.1.1.3. Ixx04: Speed/velocity address

Name	Write level	default	Unit	range
Ixx04	TS	0	-	address

In most of the case Ixx04=Ixx03 but you can specify another address, if you use another type for feedback for the speed measurement.

9.1.1.4. Ixx10: Power on Address (for absolute feedback)

Name	Write level	default	Unit	range
Ixx10	TS	0	-	address

This parameter defines where the controller should read the position address on start of the system or after a reset. This parameter must be set for absolute feedback such as SSI or Litton mechanical encoder. The Address of each channel is specified by the respective feedback board documentation. For the Litton encoder looks at the respective chapter in this manual. If Ixx10 is set then Ixx95 must also be configure.

Encoder #	Processed Data
1	Y:$78C00
2	Y:$78C01
3	Y:$78C02
4	Y:$78C03
5	Y:$78C04
6	Y:$78C05
7	Y:$78C06
8	Y:$78C07

Figure 12 SSI position address board 1

9.1.1.5. Ixx11: Following error limit

Name	Write level	default	Unit	range
Ixx11	TS	0	1/16 counts	integer

This parameter is quite important it define the error tolerance between the targeted position and the feedback position of the positioning loop. If the axis over run the tolerance due to a problem then the motor is killed and the error status reports the respective error information.

This parameter has to be set in close loop mode as well as in open loop mode.

- In **open loop mode** when the nominal speed Qxx04 of the moving motor is higher than the Output command limit Ixx69, the following error will increase. The positioning loop limit the output pulse to Ixx69 but the wished speed Qxx04 is higher then this difference will increment the following error register. In this case you have to set-up your speed Qxx04 under the output limit or increase the limit until reach a stable value of the following error depending of your application limits.

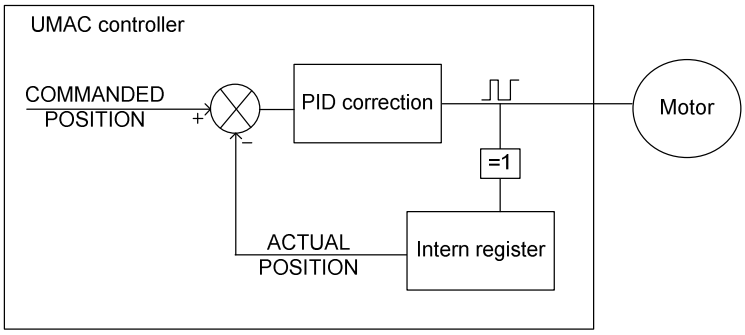


Figure 13 open loop basic model

- In **close loop mode** set this parameter when your positioning loop is already setup (PID parameter). First put the following error relative high that you can tune your motor without error and when its finished note the current following error by max speed and set the limit about 20% over the current value. The remark in open loop is also to consider for close loop mode.

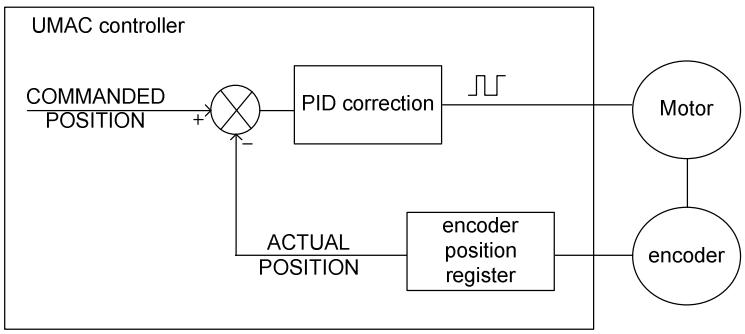


Figure 14 Close loop model

After tune in close loop mode if a following error occurs, it means that the feedback position do not accord the targeted position. It could be the following problem:

- Encoder connection not ok
- Encoder default
- Motor Default
- Mechanical problem (something bloc the motor)
- Driver Default

9.1.1.6. Ixx13: UMAC Controller software limit+

Name	Write level	default	Unit	range
Ixx13	TS	0	counts	Signed integer

Define the axis lower software limit, this limit could be placed close before the mechanical switch (you can then economize the switch activation) and from the application point of view it is the same behaviour and same error as the mechanical switch. If 0 the function is not active.

9.1.1.7. Ixx14: UMAC Controller software limit+

Name	Write level	default	Unit	range
Ixx14	TS	0	counts	Signed integer

Define the axis upper software limit, this limit is placed before the mechanical switch and from the application point of view it is the same behaviour and same error as the mechanical switch. If 0 the function is not active

9.1.1.8. Ixx15: Abort deceleration

Name	Write level	default	Unit	range
Ixx15	TS	0	Counts/ms2	Unsigned Float

Deceleration value call when a hardware or software limits is reached. It is not a good idea to set this value over or too close the capability of the motor, because (in close loop mode) doing so increases the likelihood of exceeding the following error limit, which stops the braking action, and could allow the axis to coast into a hard stop(Kill). But the parameter must also be set high enough for security raison.

9.1.1.9. Ixx23: Homing speed

Name	Write level	default	Unit	range
Ixx23	TS	2	Counts/ms	float

Define the homing procedure speed. Do not care about sign (write absolute value), the application will managed it respectively from the homing type and configuration Pxx04. The homing acceleration could be the same as the nominal move. But It guarantee in case of speed change of Qxx03 that the homing speed is a tested and define speed.

9.1.1.10. Ixx24: Flag mode control

Name	Write level	default	Unit	range
Ixx24	TS	0	$C10001	interger

This parameter is used principally to activate or not the security input, emergency, and error input.

List of the most important configuration point:

- Bit 16 -> IHO (AENAn) should be set to 1 (Enable function not active), the control of IHO is done manual via the Display or the remote control
- Bit 17 -> Limits activation, for debugging or setup you can inhibit the limits by writing 1 in the bit 17
- Bit 20 -> SEI: Delta tau Fault input activation is set to 0 in order to enable the function
- Bit 21&22 -> Define if SEI flag kill all motor (Emergency) or just the respective motor, by default configuration the axis 1 is define as Emergency and the other inputs as normal SEI input.
- Bit 23 define the polarity of the default signal, default value is 1.

9.1.1.11. Ixx28: In position band

Name	Write level	default	Unit	range
Pxx28	TS	16	1/16 counts	integer

Define the tolerance the controller needs to apply to know if it is arrived in position. If there is no feedback (open loop) the value is 1 counts tolerance, if you have a feedback it is the ration between the motor resolution and the feedback resolution.

The factor is define as follow:L

- Fb: Feedback resolution
- Mr: Motor resolution
- Ds: Driver resolution (micro-stepping mode)
- Gr: Gear box(es) Resolution

Ratio= Fb / (Mr*Ds*Gr)

If Pxx28 < 1 then Pxx28=15 (close to 1 count)

Else Pxx28=Ratio*16

Example:

Encoder 200 pulse/rev.

Motor 200 step/rev.

Motor driver ustep ½

*Encoder decoding *4 (I7mn0=3 or 7)*

*Ratio=(200*4)/(200*2)=2*

*Ratio>1 then Pxx28=2*16=32*

9.1.1.12. Ixx30: Proportional Gain (PID)

Name	Write level	default	Unit	range
Pxx30	TS	5000	-	integer

P Gain factor of the PID correction loop.

9.1.1.13. Ixx31: Derivate Gain (PID)

Name	Write level	default	Unit	range
P62	TS	0	-	integer

D Gain factor of the PID correction loop. For stepper motor application this could stay equal to 0.

9.1.1.14. Ixx33: Integral Gain (PID)

Name	Write level	default	Unit	range
Pxx33	TS	0	-	integer

I Gain factor of the PID correction loop. For stepper motor application this could stay equal to 0.

9.1.1.15. Ixx64: Dead band Gain

Name	Write level	default	Unit	range
Pxx64	TS	-16	-	integer

Define the gain when the motor is in position.

With stepper motor application is could to -16.

9.1.1.16. Ixx65: Dead band size

Name	Write level	default	Unit	range
Pxx65	TS	8	-	integer

Define the dead band. This parameter should be equal to Ixx28 for stepper motor.

9.1.1.17. Ixx69: Output command limit

Name	Write level	default	Unit	range
P62	TS	200	F*Hz	0-32767

This parameter defines the maximal outputs frequency allowed by the system. If this parameter is under the commanded speed Qxx04 then the following error will increase until the limit and stop. If the following error limit is equal to 0 then the motor move at the max output frequency but the following error magnitude still growing. This could create hazardous behaviour when the motor reach the position, it will stop abruptly (The PID loop try to compensate, the error is big then the Gain is also big).

*Ixx69= (Flim/Fmax)*65536*

With

Flim: Output limits Hz,Khz,etc.

Fmax: Maximum configured frequence of the board default 307.2Khz

9.1.1.18. Ixx95: Power on position format (Absolute feedback)

Name	Write level	default	Unit	range
Ixx95	TS	$0	-	address

This parameter should be set for at least for absolute encoder, it define the type of read value and the size.

"Ixx95 specifies how the absolute power-on servo-position data, if any, for Motor xx is interpreted. Ixx10 specifies the address of the register containing this position data; Ixx95 controls how that data is read. This permits the use of a wide variety of absolute position sensors with the Turbo PMAC."

In the case of absolute encoder (Delta tau : parallel data read).
2 For a signed value: Ixx95= ($80 + Data length) *10000 (from 8 to 48)
For an unsigned signed value: Ixx95= ($00 + Data length)*10000 (from 8 to 48))

Example for a 24bit multi-turn signed : Ixx95=$980000.

9.1.1.19. I7mn0: Encoder decode control

Name	Write level	default	Unit	range
I7mn0	TS	8	-	integer

Define the format the standard encoder connected to the motor board. This configuration does not meet the absolute encoder. Default configuration is open loop.

- I7mn0 = 0: Pulse and direction CW
- I7mn0 = 1: x1 quadrature decode CW
- I7mn0 = 2: x2 quadrature decode CW
- I7mn0 = 3: x4 quadrature decode CW
- I7mn0 = 4: Pulse and direction CCW
- I7mn0 = 5: x1 quadrature decode CCW
- I7mn0 = 6: x2 quadrature decode CCW
- I7mn0 = 7: x4 quadrature decode CCW
- I7mn0 = 8: Internal pulse and direction
- I7mn0 = 9: Not used
- I7mn0 = 10: Not used
- I7mn0 = 11: x6 hall-format decode CW*
- I7mn0 = 12: MLDT pulse timer control
- (internal pulse resets timer; external pulse latches timer)
- I7mn0 = 13: Not used
- I7mn0 = 14: Not used
- I7mn0 = 15: x6 hall-format decode CCW*

Example:

I7210=8 //motor 1 board 1 open loop
*I7410=3 //motor 3 board 1 clockwise encoder *1 quad*
*I7430=5 //motor 3 board 3 counter clockwise encoder *4 quad*

Note:

**4 quadrature means 1 pulse of chA and chB equal 4 internal pulses.*
**1 quadrature means 1 pulse of chA and chB equal 1 internal pulses.*

9.1.1.20. I7mn8: PFM Pulse Frequency Modulation inversion

Name	Write level	default	Unit	range
I7mn8	TS	0	-	integer

I7mn8 controls the polarity of the direction output signal in the pulse-and-direction format for Channel n.

It allows the following possible settings:

I7mn8 = 0: Do not invert direction signal (+ = low; - = high)
I7mn8 = 1: Invert direction signal (- = low; + = high)

If I7mn8 is set to the default value of 0, a positive direction command provides a low output; if I7mn8 is set to 1, a positive direction command provides a high output.

10. Motor Setup method

This is an overview of the axis setup procedure the details of each step is written below. For each state of the configuration it is advice to test the motor/axis with moves commands.

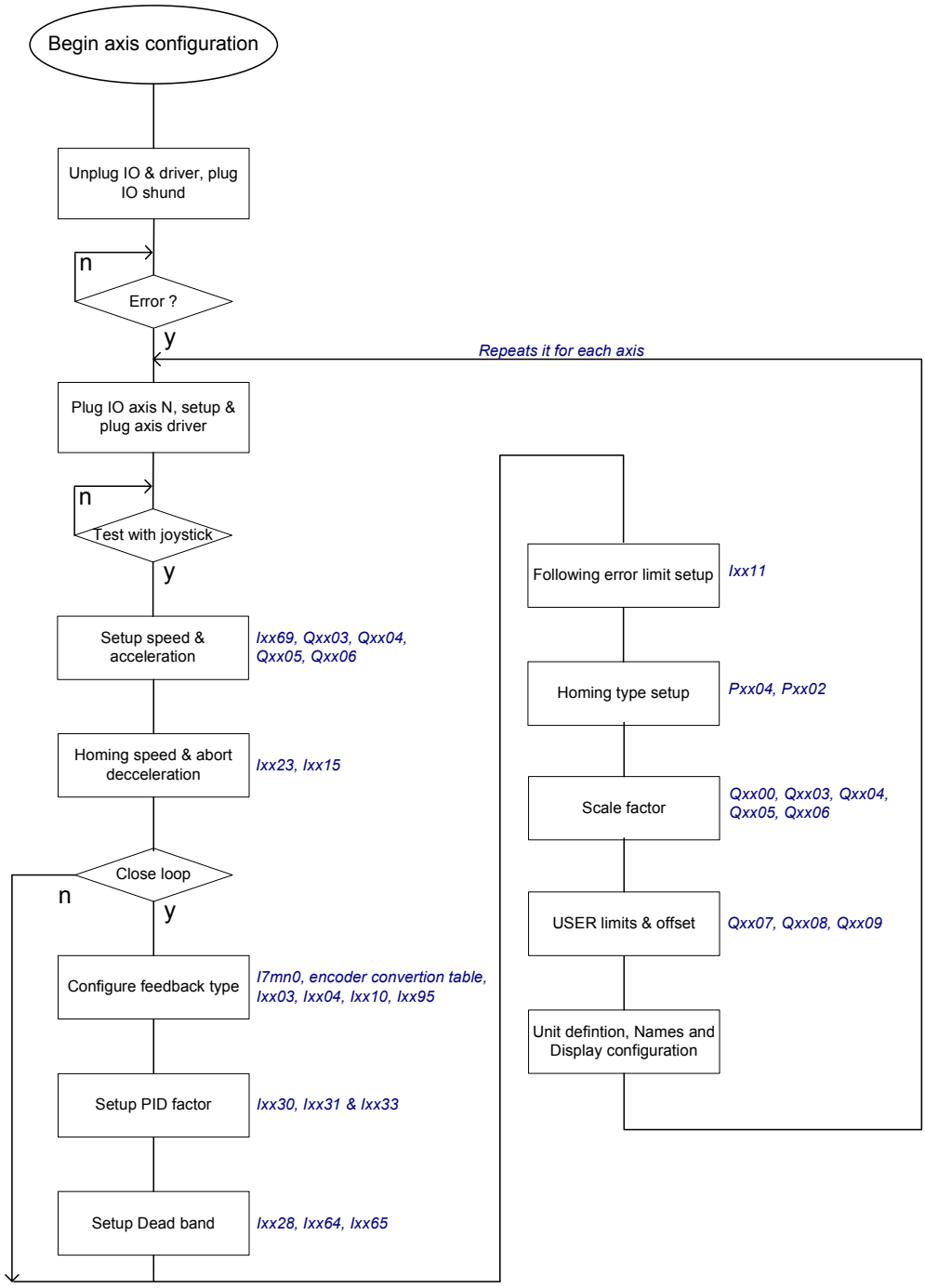


Figure 15 motor setup flow chart overview

10.1.1.1. Step 1: Hardware configuration

Before start the MCU you need to configure hardware as follow in order to not have any errors by starting:

- Connect the "IO shunt" connectors on each channels (will delete all limits inputs error if existing)
- Disconnected drivers and encoders connectors
- Motors, encoder and IO are not connected, you can start the MCU you should not have any inputs errors on (Limits, SEI, driver error, emergency, etc…).

-> The display on the front panel is not red.

Note: it is strong advice to let connected the emergency input (default : SEI1) in order to have it available during the setup tests.

10.1.1.2. Step 2: Axis IO connection

Connect the IO connector and test the limits, the display should stay green if not some IO are wrong connected.

10.1.2. Open loop configuration

10.1.2.1. Step 3: Motor driver configuration

- Do the set-up of coil current, micro-stepping mode and signals logic configuration. For the signals logic configuration you can look at the hardware manual on driver interfaces configuration.
- It is strong advice to work in micro-stepping mode with Delta tau controller, the controller does not manage start/stop frequency and the resonance frequency must be minimizing to avoid blocking behavior. It is not problem for the controller to work in micro-stepping mode the controller is able to work with relative high frequency for stepper. The standard configuration allows going up to 300 Khz stepping frequency.
- Plug the driver and test the Inhibit signal and the error signal should not occurs.

10.1.2.2. Step 4: First moves test of the motors

You can execute the first move with the remote control or with a terminal. Actions to do:

- Check if directions are correct
- Check if good reaction with limits
- Check if no mechanical collisions

10.1.2.3. Step 5: Speed and acceleration set-up

You can increase the motor acceleration Qxx04 and speed Qxx06 parameter to the nominal values you want. Do not forget also to increase the limits parameters like Ixx69, Qxx03 and Qxx05. If Ixx69 is too small the following error value will increase indefinitely and If Qxx03 and Qxx05 too small it will generate one error.

You should find parameters that generate small vibrations and makes several moves. It is also advice to test higher values than the nominal values in order to be sure that you are not on the motor limits.

10.1.2.4. Step 6: Homing speed and abort deceleration

- Set the homing speed Ixx23 (counts/ms). It is advice to take the same value as the nominal (Ixx23=Qxx04/1000). The homing speed is then independent of the other move, it ensure that when a homing sequence is call the speed is a tested speed.
- Set the abort deceleration Ixx15 in counts/ms2. This deceleration is called when axis reach limits.

10.1.3. Close loop configuration

10.1.3.1. Step 7: Encoder test and ratio (close loop)

In order to know if your encoders are good connected we will connected it to a different channel and test the feedback value in open loop, this channel will be named the “phantom channel”. From this test we will define 2 data: the rotation direction of the encoder and the ratio resolution between the motor and the encoder. The test also allows testing the encoder hardware connection without any close loop to do.

- Configure an free axis has “phantom feedback” with I7mn0 (for example axis 8 I7340=3 or 7)
- Configure speed address Ixx04 and position address Ixx03 of the “phantom axis” if you use an absolute feedback.
- Move the axis with the joystick for example and define the rotation parameter value and the ratio between the speed of your axis and the speed of the “phantom axis” Rme (Ratio motor encoder).
- Repeat the operation for all axis a list the values in a table as follow.
- Rewrite the “phantom axis” in the normal state I7mn0=8 (open loop)

Axis	Rotation value	Ratio
1	I7210=	Rme=
2	I7220=	Rme=
3	I7230=	Rme=
4	I7240=	Rme=
5	I7310=	Rme=
Etc.	Etc.	Etc.

10.1.3.2. Step 8: Speed and acceleration adaptation (close loop)

The encoder feedback is now the position and speed reference, then the Qxx03..06 have to be adapted with the Rme factor:

- Qxx03=Qxx03*Rme
- Qxx04=Qxx04*Rme
- Qxx05=Qxx05*Rme
- Qxx06=Qxx06*Rme
- Ixx23=Ixx15*Rme
- Ixx15=Ixx15*Rme

10.1.3.3. Step 9: Dead band, tolerance setup and closing loop (close loop)

- Define the tolerance or dead band of the axis with parameter Ixx28 & Ixx65
- Close the loop with the I7mn0 parameter (value in the table see before)
- Set the following error limit quite big (example 1000 step Ixx11=16000), in order to not have following error axis stops.

Now you can test if the motor working properly, if not check the direction of the encoder and/or the speed and acceleration value.

10.1.3.4. Step 10: PID parameters set-up (close loop)

Case of stepper motor:

The case of stepper motor is relative simple, you should actually just need to setup Ixx30 (P factor) parameter in order to limit the following error during a move. The default value 5000 is a middle value that is why the following error could be quite high some time (depending of your Rme), you can increase the factor in order to decrease the following error. Smaller is your following error better will be the reaction time in case of feedback failure. If the P factor is to small (about 100 for example) the loop will take time to react of the speed command change, the loop is too soft. To high is also no so good but you can go until 20000 with some stepper configuration without problem (motor can block if too high).

Case of Servo motor:

In this case it could be interesting to use auto-tune tools from Delta-tau.

10.1.4. User environment configuration

10.1.4.1. Step 11: Following error limit setup

Move your axis at the maximum speed allowed and note the following error and reach a stable state (following error do not increase any more). The error must be stable. Write the following error limit Ixx11 about 20% higher that the read values. **Ixx11=(Error*(1+0.2))*16.** If the error is not stable check Ixx69 and the close loop parameters.

If your axis is in close loop mode you can test the reaction after failure when you are moving your axis and unplug the feedback connector.

10.1.4.2. Step 12: Homing configuration

- Setup the Pxx02 parameter in order to define with switch is sensitive, if you want to look for the index, etc…
- Setup the offset with the Pxx04 parameter, value in counts
- Test the home procedure with the Mxx=9 command

10.1.4.3. Step 13: Controller software limits configuration

Setup Limit + Ixx13 and limit- Ixx14 (counts), move the axis to your limit position read the position value and write the limits parameters with this value. Do not forget to execute a homing before this operation, if not the value will not have any sense.

10.1.4.4. Step 14: Scale factor setup

The scale factor define the ratio between the axis resolution and you unit. This ratio should be < or = to 1.This factor should be calculated with the mechanical inputs of your system. After that write the following parameter as following:

1: Qxx00=scale
2: Qxx03=Qxx03*scale
3: Qxx04=Qxx04*scale
4: Qxx05=Qxx05*scale
5: Qxx06=Qxx06*scale

Now your are finished with the most important parameters, your axis is working in user unit Qxx10 and Qxx11 values are in unit and you can make some moves to check the behavior.

10.1.5. Optional parameters

10.1.5.1. Step 15: User limit

Qxx08 and Qxx09 are Admin level parameter, it means that you can let it to 0, in this case the function not active.

10.1.5.2. Step 16: Unit, Names and after coma display

- Define the user unit Pxx17, six unit are available so far (useful for joystick and display so far)
- Define names (useful for joystick and display so far)
- Define after coma display Pxx08 (useful for joystick and display so far)

10.1.5.3. Step 17: Axis deactivation

You can deactivation motors that are not used with the Ixx00 parameter. Then you are ensuring that the moving commands will not be interpreted.

11. FAQ: Frequently Asked Question or issues

When I make a move the following error does not stop to increase?

- feedback is not read
- The motor is blocked
- The Ixx69 maximum output frequency limit the speed

I get a following error when I drive my motor too fast?

- The following error limit is too small
- Mechanical limit is reached

12. Absolute encoder software documentation

12.1. Encoder conversion table

The Delta Tau controller store the feedback register values from the each type of interface board into the conversion table. In other word, the feedback register are not directly read from the IO interface but trough a controller memory interface called encoder conversion table. This will allow controlling the values and applying some filter for example. It is a quite powerful and flexible interface.

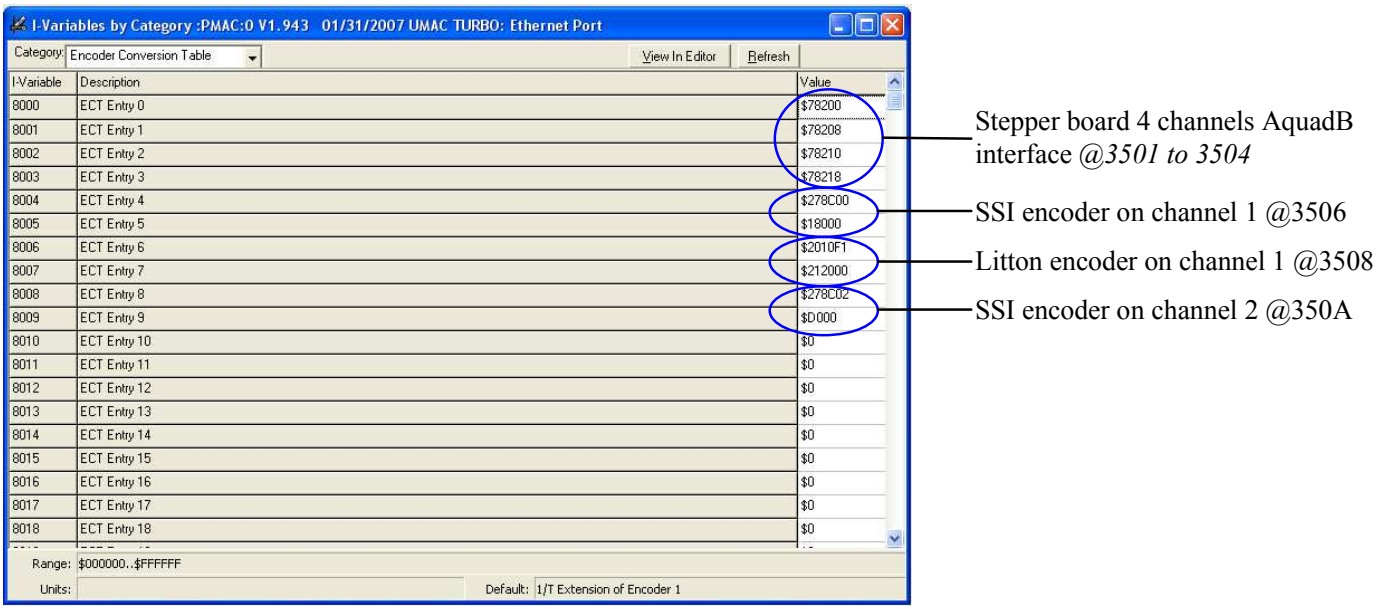


Figure 16 Encoder conversion table of the evaluation system

One feedback can be configured with 1 to 3 parameter, one is the interface register address and the two other are optional, it could be limit or filter.

Method Digit	# of lines	Process Defined	Mode Switch	First Additional Line	Second Additional Line
$0	1	1/T Extension of Incremental Encoder	None	-	-
$1	1	Acc-28 style A/D converter (high 16 bits, no rollover)	0 = signed data 1 = unsigned data	-	-
$2	2	Parallel Y-word data, no filtering	0 = normal shift 1 = unshifted	Width/Offset Word	-
$3	3	Parallel Y-word data, with filtering	0 = normal shift 1 = unshifted	Width/Offset Word	Max Change per Cycle
$4	2	"Time Base" scaled digital differentiation	None	Time Base Scale Factor	-
$5	2	Integrated Acc-28 style A/D converter	0 = signed data 1 = unsigned data	Input Bias	-
$6	2	Parallel Y/X-word data, no filtering	0 = normal shift 1 = unshifted	Width/Offset Word	-
$7	3	Parallel Y/X-word data, with filtering	0 = normal shift 1 = unshifted	Width/Offset Word	Max Change per Cycle
$8	1	Parallel Extension of Incremental Encoder	0 = PMAC IC 1 = PMAC2 IC	-	-
$9	2	Triggered Time Base, frozen	0 = PMAC IC 1 = PMAC2 IC	Time Base Scale Factor	-
$A	2	Triggered Time Base, running	0 = PMAC IC 1 = PMAC2 IC	Time Base Scale Factor	-
$B	2	Triggered Time Base, armed	0 = PMAC IC 1 = PMAC2 IC	Time Base Scale Factor	-
$C	1	Incremental Encoder, no extension	None	-	-
$D	3	Exponential filter of parallel data	None	Max Change per Cycle	Filter Gain (Inverse Time Constant)
$E	1	Sum or difference of entries	None	-	-
$F	-	(Extended entry – type determined by first digit of second line)	-	-	-

Figure 17 sum-up of the table configuration

12.2. Absolute SSI encoder

12.3. Mechanical Litton encoder

13. Sequences programming method

13.1. How to start and stop a program

The MCU offer the possibilities to write sequences in order to play series of orders automatically. They are two possibilities to Start and stop a programme.

The first one is with the SYI synchronisation input, if high the sequence is started and if low the sequence is stop. SYI is a sinking input; it means that you have to connect it to the ground to activate the sequence. A sequence is called a PLC program by Delta tau. SYI 1 starts the PLC 1, SYI2 starts the PLC2 and so one until the maximum axe number allowed by the software.

The second way to start a sequence is with an online command. The command is as follow ENABLE PLC xx or DISABLE PLC xx.

13.2. Sequence structure

The PLC should be written a editor, better is to use de PWIN PRO editor that allows load of program and debugging function but it is possible to use any editor. You can create 31 PLC. It is advice to organise PLC 1 for axis 1 PLC 2 for axis 2, etc... . Particularly if you wisch to use the SYIxx inputs. But in a sequence you can call any commands form any axis.

Structure:

OPEN PLC xx CLEAR

;program

CLOSE

CLEAR ensure that the program will be empty before writing new commands.

Important: When started the PLC did not stop at the end but loop without end. Several PLCs could be start in parallel in order to achieve multitasking sequence.

13.3. Sequence programming: variables, condition, comparator

Delta Tau PMAC controller does not allows working with local variables that is why some global variables are reserved for PLC programming. Each Axis has a block of variables that could be sufficient to write your sequence.

- Pxx50 to Pxx59 could be use as integer value
- Qxx50 to Qxx59 could be use as floating point value

Conditional structure:

```
If (condition1 AND/OR condition2)
;prog
Endif

While (condition2 AND/OR condition2)
;prog
Endwhile
```

Comparator:

```
 =
<
>
!=
```

If you need to insert commentaries the normal character to use is the ";", but it should be also possible to user the C commentary // and /*…*/.

For people accommodated with C programming the Delta tau program loader accepts a post-compiling operation; most of the C programming post command are interpreted (#define, #ifndef, #ifdef, etc…). You can find more detail in the Detlta Tau software manuals.

13.4. Example of sequence

Program 1: IF SYI active move axis 2 to 0 position

```
OPEN PLC 2 CLEAR
        If (P250=0)                 ; verify that the move is not already done
                Q201=0              ;set goal absolute position
                M2=1                ;order absolute move
                P250=1              ;avoid to recall the move
        Endif
CLOSE
```

Program 2: Each time Axis 1 will reach position 0 axis 2 will go to 10

```
OPEN PLC 2 CLEAR
        If (Q110=0)                 ;check if axis 1 position is 0
                Q201=10             ;set goal absolute position
                M2=1                ;order absolute move
        Endif
CLOSE
```

The following flow chart give a description of how to program a move in a move series, it will ensure that the move is done correctly.

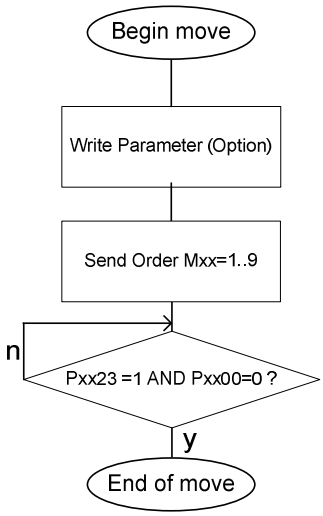


Figure 18 Move check sequence flow chart