REFERENCE HANDBOOK FOR

SIMPA AND SIMPA MICROSTEP MODULES

AFAQ ISO 9001
AFAQ N° 2000 / 14608

Date : 15.10.99

Reference : BLN48250.DOC

Revision : 15

Author : B.LOPEZ

SYNOPSIS

FOREWORD

This document describes functionalties that are common to all SIMPA products.
For each specific product, the user will have to associate the product "Product Instruction Manual" to it, dealing with the own characteristics of the module used, particularly its configuration and its specific connections.

This handbook describes functionalties of SIMPA modules in full step version, ½ step version, as well as in microstep version (SIMPA microstep). Except the limits of parameters and the inquiries of the modules, the differences remain tiny. They are specified when necessary.

I - INTRODUCTION

The products of SIMPA family designed by Midi Ingenierie are smart units intended for stepper motors control. These modules integrate the indexer function and, for most of them, stepper motors control specific power electronics. Some boards integrate up to 4 SIMPA modules.

Each SIMPA module includes: whole of the functions and components needed to control a motor axis (step by step), a logical unit with microprocessor and a switching current power unit which can deliver an intensity from 2A to 7A RMS or more by phase (3A to 10A peak) under a supply voltage of 15V to 100V depending on the type of board.

Each module has a serial RS232C link enabling it to communicate with a computer or a programmable controller. This one makes it possible to remotely define operating parameters of each module and to save preset movement sequences. Thanks to these sequences, SIMPA modules can work in an autonomous way and become genuine small controllers.

Various options can facilitate the implementation of SIMPA modules:

a) Interactive front panel

The front panel makes it possible to visualize on a 8 digits display, and to modify by means of pushbuttons, most parameters of the module.

It makes it possible to carry out simple movements and to carry out the preset sequences, to know the state of logical inputs.

b) Sequence selector

The sequence selector authorizes an autonomous exploitation of the module without a serial link making it possible to run any preprogrammed sequence. The number of sequence to be carried out is defined thanks to two rotary switches.

c) Multiaxis systems

SIMPA boards can be integrated in frames and dialogue with a computer by means of a single serial link, thus carrying out a multiaxis control device.

d) Wide family

Various back panel boards allow a simple interconnection between a SIMPA board (indexer alone) and the amplifier boards such as MI452, MI454, MI904, MI907, MIP806, MIP909…, then increasing SIMPA modules family.

I.1 - Implementation

The implementation of SIMPA modules can be done:

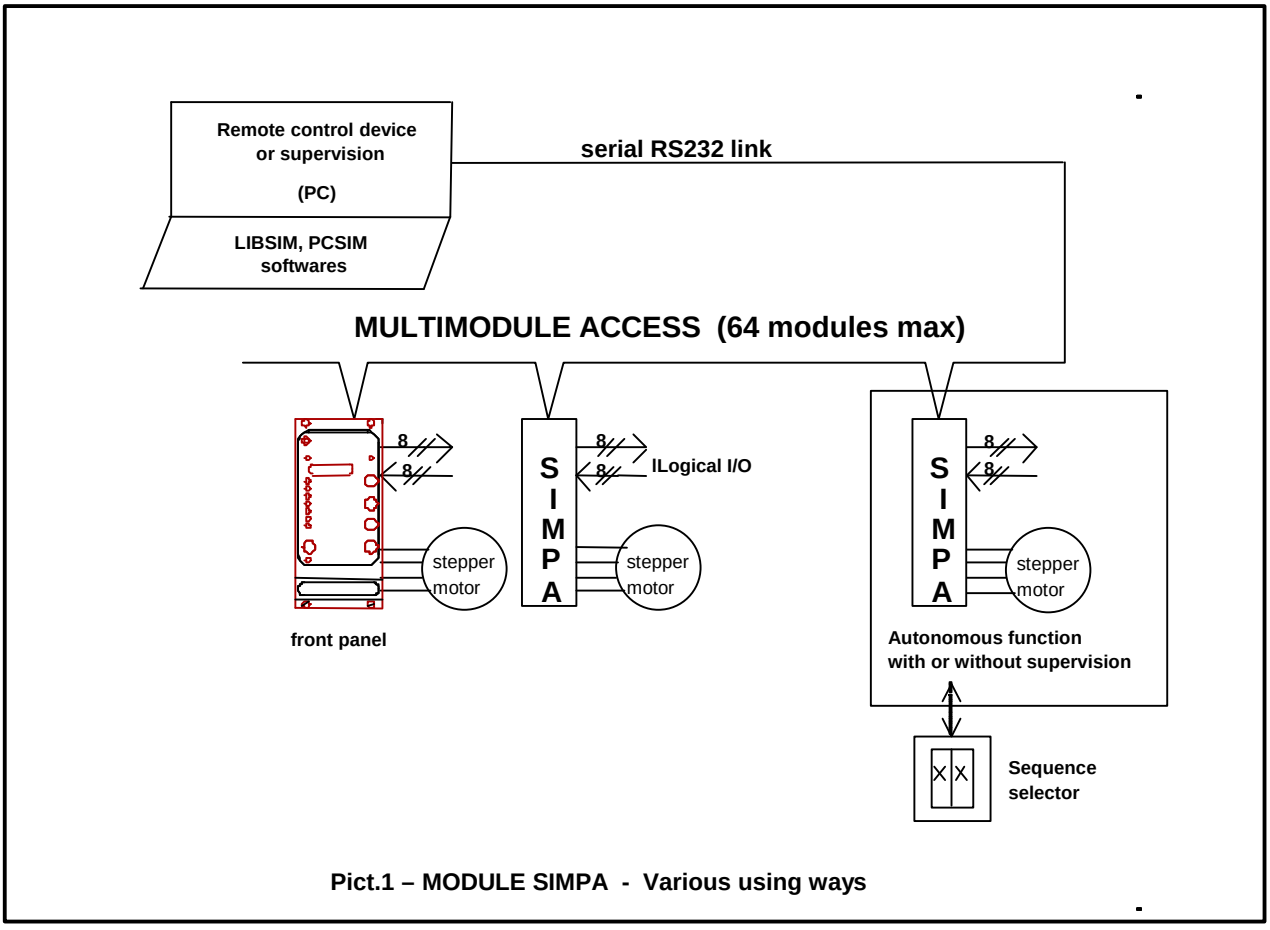
- Remotely, thanks to a standard RS232 serial link (only one for up to 64 modules). Software PCSIM and LIBSIM, DOS compatible, were designed to facilitate the dialogue with SIMPA modules.
  A Windows version of PCSIM as well as the DLL of management of the dialogue protocol between modules can be delivered on request.

I.2 – Control modes

Two control modes of the modules are possible:

- A direct mode accessible either via the front panel (local access), or by the serial link. It makes it possible to carry out the principal basic movements.

- A sequence mode, only accessible starting from the serial link, allows, after loading of the sequences, an autonomous operation of the modules. The synchronization of several axis can be carried out thanks to the 8 logical inputs and 8 outputs available on each module to communicate with the external environment (controllers, limit switches, sensors…).



Pict.1 – MODULE SIMPA - Various using ways

I.3 - Functions

In direct mode, the main accessible functions are:

- Definition of the movement parameters:
  * Start speed (V_{min}) in step/second (or half-step/s)
  * Set point speed or plateau speed (V_{max}) in step/second (or half-step/s)
  * Acceleration and deceleration slopes time in ms
  * Number of steps or microsteps to be carried out and motor rotation direction
  * Amplitude of the current in the motor.

- The powering of the motor and execution of the movements;

- The choice of operation with or without limit switch;

- The operation control and modules status.

In sequence mode, some additional functionalities are offered:

- Sequencing of several movements,

- Temporization,

- Real time control of 8 logical outputs,

- Real time modification of the course of the sequence in according to the state of 8 logical inputs,

- Call of sub-sequence, linking of sequence, conditional or not starting of sequences.

II - PRINCIPLES OF STEPPER MOTORS CONTROL

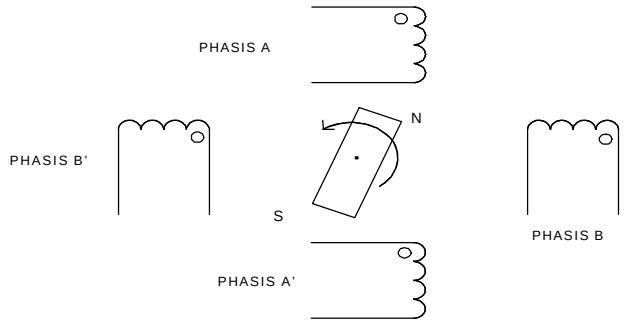
The purpose of this chapter is to point out fundamental principles used for stepper motors control.

It is not supposed to be a course on the subject, nor does it deal with every aspect of motor control.

On the other hand, it shows simply the principal rules to be respected so as to obtain the desired performances.

II.1 - Types of motors

The stepper motor, designed to develop an important torque, at low speed (even when stopped) is an information transmitter able to control position and speed by simply managing a frequency.

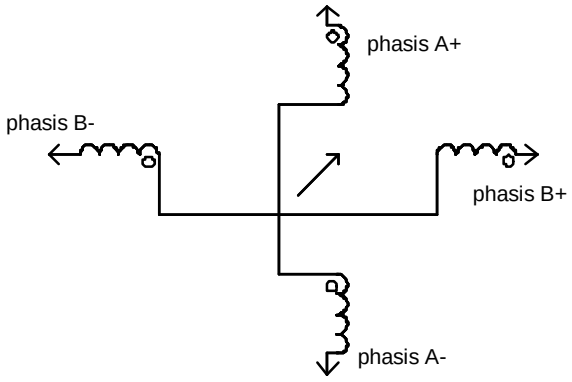


The motor consists of a stator made of a certain number of windings and of a rotor (magnetized or not).

When the rotor is passive (not magnetized) the motor is known as "variable reluctance". When it is magnetized, the motor is known as "permanent magnet". The combination of these two types leads to the "hybrid" motor.

The number of windings of the motor as well as its type of connection (4 wires, 6 wires or 8 wires) determines a number of phases which, supplied according to a given sequencing, create a rotating stator field.

For example: 4 wires connection, series windings:



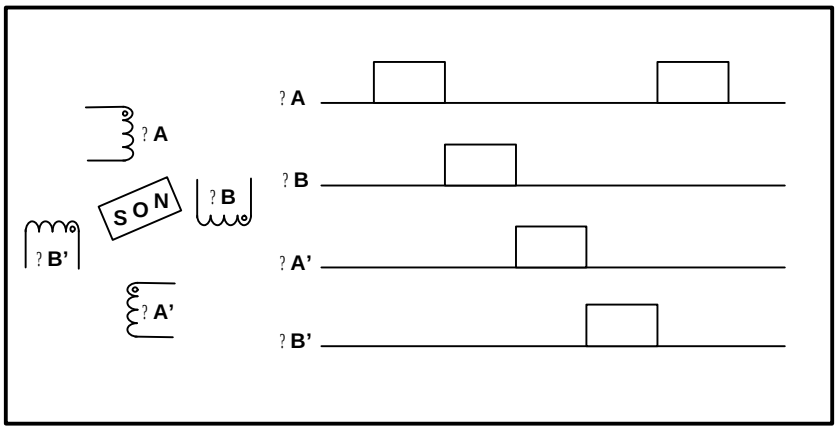
The rotor is directed in this field thanks to the torque developed between the two fields (magnetized rotor) or by the torque intended to decrease the reluctance (passive rotor).

The stepper motor is thus a synchronous motor, which consumes electric power due to the phase difference between rotor and stator magnetic fields. The higher the mechanical load (base load + frictions), the higher the torque is. Power consumption by Joule effects must be added to this phase-related power consumption. Beyond phase difference of ?/ 2, the motor stalls.

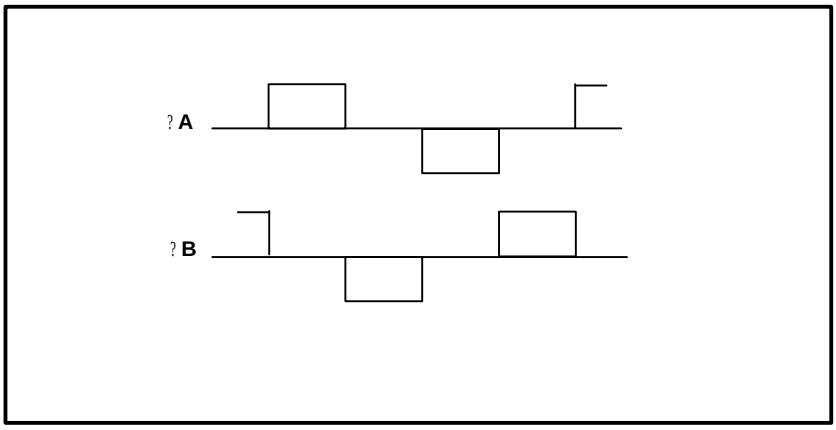
II.2 – Switching modes

II.2.1 – Switching one phase after the other (1Phase On)

Powering each phase of the motor one after the other, the rotor comes successively in front of each winding of phase.

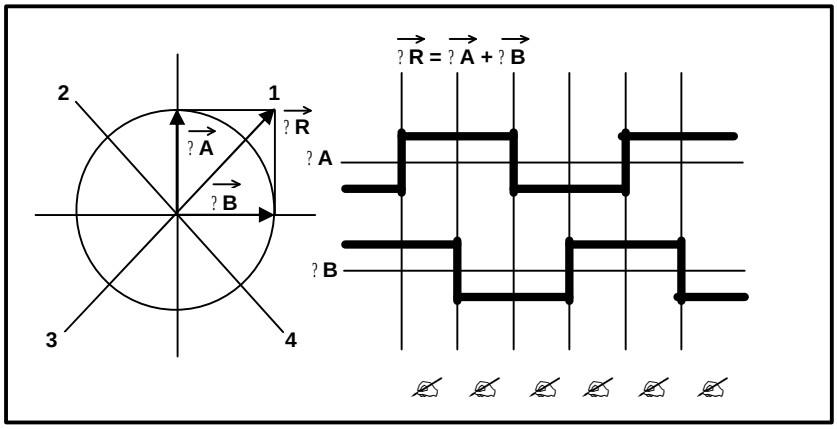


Placing the phases in parallel or series and exploiting the direction of the current in the windings, one obtains the bipolar command according to the following chronogram:



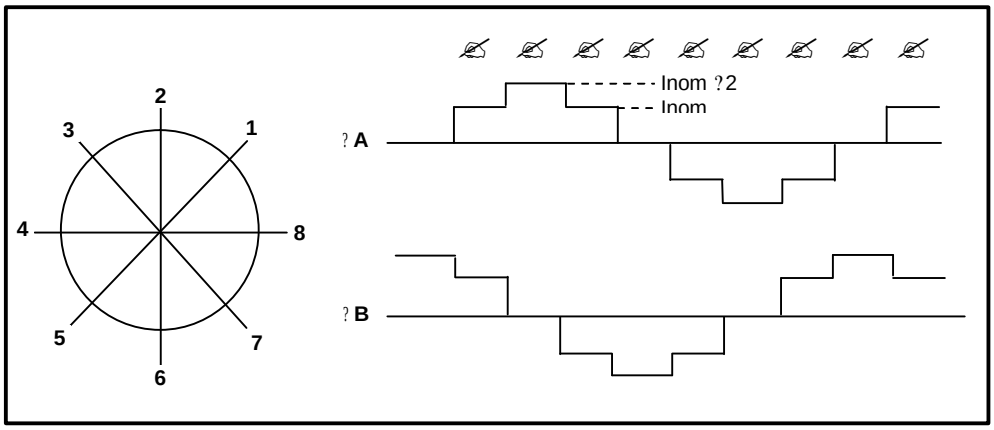
II.2.2 - Switching 2 phases at the same time (2 phases ON)

By simultaneously injecting the current in the 2 phases, the motor is aligned on the resulting field.



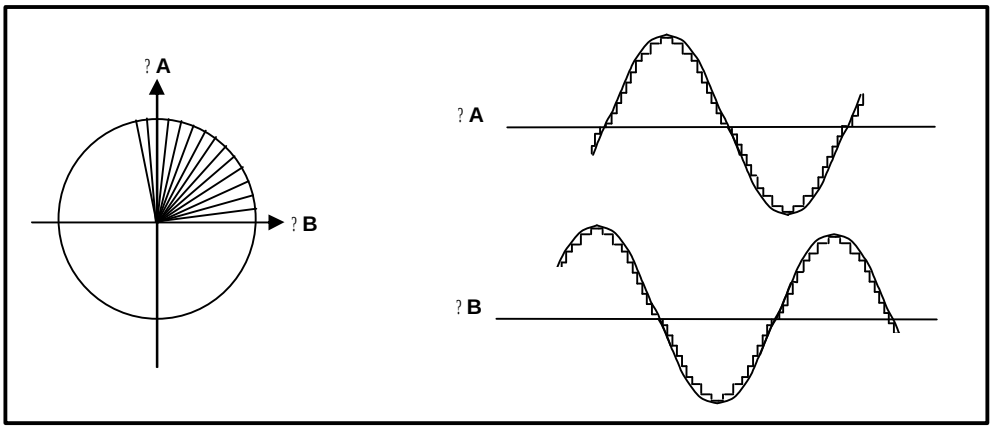
II.2.3 - Commutation in ½ step mode

By combining the 2 preceding modes, one obtains 8 angular positions per cycle. By modulating the range of the current by ?2, one maintains the amplitude of the resulting torque constant.



II.2.4 - Switching in microstep mode

Going further, one can create as many intermediate steps as desired between 2 "geometrical" steps of the motor. The fields ØA and ØB (and consequently currents which produce them) are respectively sine and cosine form.

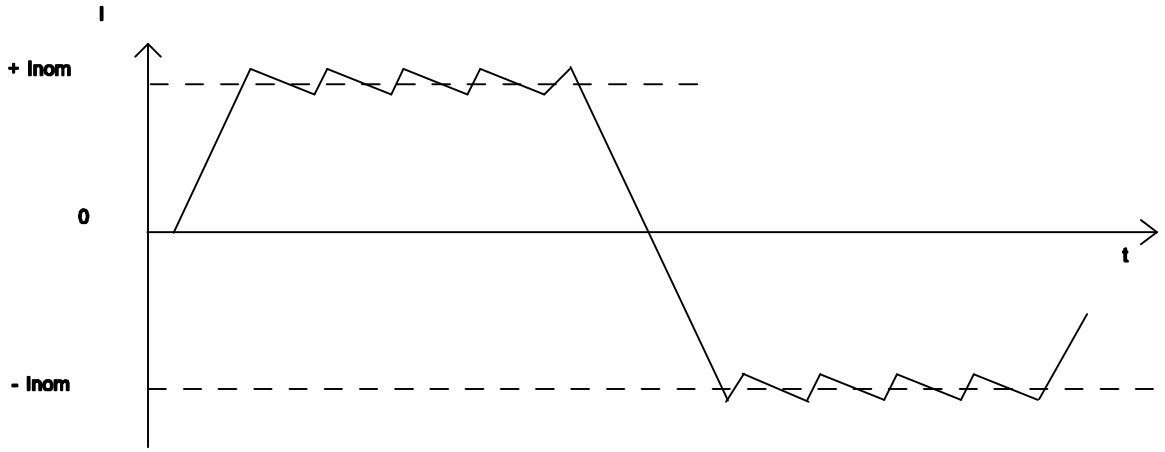


The higher the number of microsteps, all the more regular the movement is.

II.3 - Current control in the motor windings

The current control in each phase of the motor is obtained by supplying switching voltage to coil windings terminals.
The current variation is given by:

$$? \; V \; ? \; L \; ? \; \frac{dI}{dT}$$

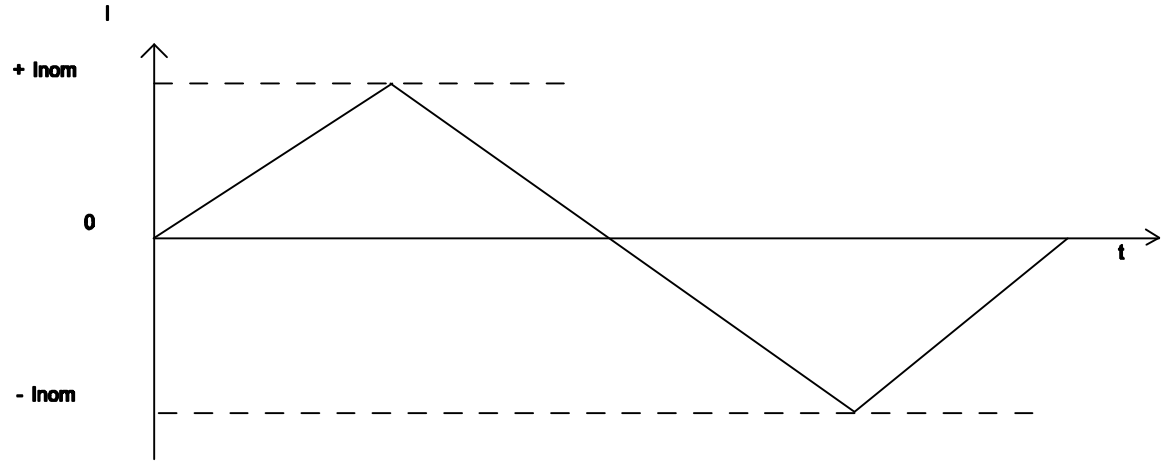


By alternatively imposing two voltages V1 and –V2 at the terminals of the winding, one can maintain the average current at the desired value simply by controlling the duty factor.

For a given motor the choice of I_{nom} determines the maximum torque that the motor can provide according to the law of the amps turn applied to winding.

$$? \quad C = k_1 * I_{nom} * N = number\ of\ winding\ turns$$

The motor torque being proportional to the current in the windings, this current must have enough time to be established (v = Ldi/dt).
As soon as current regulation plateau cannot be reached any more, the torque provided by the motor decreases quickly. The maximum speed that could thus be obtained corresponds in first approximation to the speed for which the current waveform becomes triangular.



II.4 - Choosing a motor

Knowing the minimum torque required by the application and the maximum operating voltage imposed, the choice of the motor depends on the following:

For motors having similar torque but different electrical characteristics, Joules losses related to the engine ($R * I_{nom}^2$) are more or less constant whereas Joules losses related to the electronic control increase proportionally with the current delivered. It is thus advisable to choose the motor that requires the lowest rated current.

Consequently, for a current going from $-I_{nom}$ to $+I_{nom}$ (2 phases ON) at maximum speed, the relation ? becomes:

$$? \quad V_A \; ? \; 2 ? \frac{L ? I_{nom}}{?T} \qquad \text{with } ?T ? \text{ time for 2 motor steps at the maximum considered speed Wmax}$$

$$?T ? \frac{2}{Wmax}$$

$$V_A \; ? \; L ? I_{nom} ? Wmax$$

In addition, the inductance of a winding is proportional to the square of the number of turns:

$$? \quad L ? k_2 ? N^2$$

from ?, ?, ?, it comes:

$$? \quad I_{nom} \; ? \; \frac{k ? W \max}{V_A} \qquad \text{with} \qquad k ? k_2 ? \left(\frac{(C)}{k_1} \right)^2 ? L_0 I_0^2$$

L_0 and I_0 are the inductance and rated current of the selected motor.

Thus, at imposed torque, voltage and maximum speed, it is advisable to choose the motor which has the lowest possible rated current and which checks the equation ?.

Caution! The demonstration above is simplified to highlight the importance of the choice of the motor. The result obtained can give only a first idea of the motor to be selected.

III - FUNCTIONALITIES

III.1 - Organization of SIMPA modules

The modules are organized in order to:

- Control the movements of a stepper axis

- Operate in an autonomous way,

- Inter-react with each other to constitute multiaxis systems,

- Accept an external supervision and, if needed, to be integrated in a more extensive system,

- Accept a local control directly through the front panel

Each module then includes:

- A unit with microprocessor and memory to save:
  - Movements parameters
  - Sequences for autonomous operation

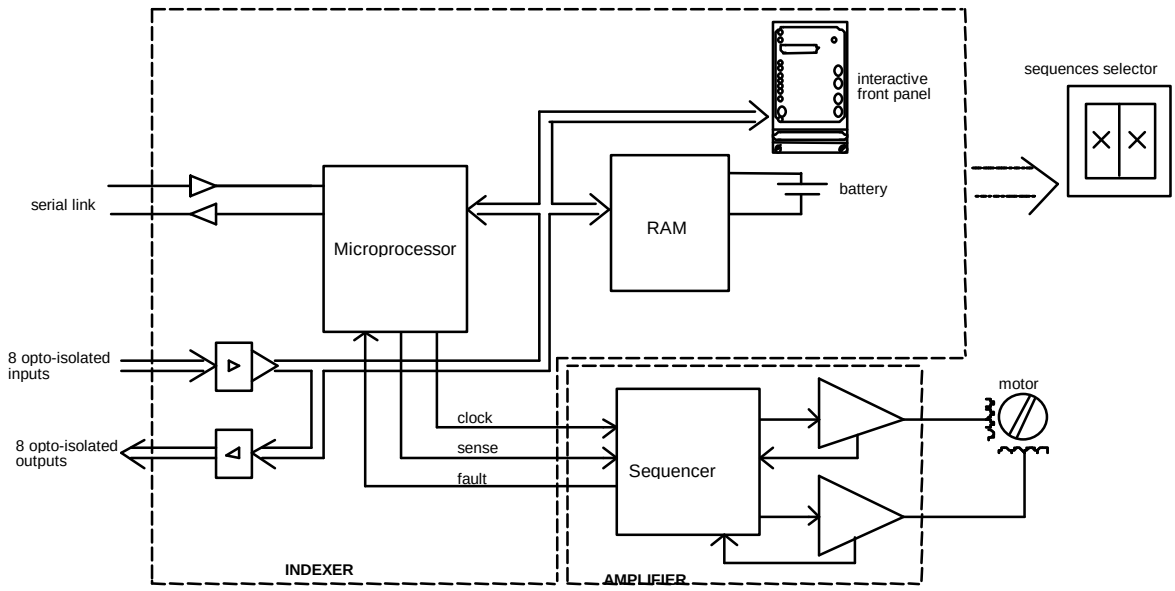
- A power stage for direct connection to the motor in 4, 6 or 8 wires for modules with integrated the power amplifier;

- Digital inputs/outputs to communicate with the environment (limit switches, synchronizations, status and various commands);

- A serial interface for remote control and multi-module supervision.

- An interface for local control: front panel, sequence selector…

Synoptic of a SIMPA module



III.2 - Indexer, amplifying indexer

The SIMPA modules family includes both modules with or without power electronic stage.

Modules without integrated amplifier (indexer alone) can control most standard amplifiers and particularly MI and MIP modules designed by MIDI INGENIERIE.

MI modules are microstep amplifiers, MIP modules are full step or ½ step amplifiers. Separating the indexer from the amplifier allows better adaptation of the performances to a particular need. They also allow a separated localization between low power control stage (indexer) and the motor power stage (amplifier).

CAUTION! Most of the commands further defined can be used on all the amplifying indexers modules, BUT in the case of 2 independent products, the use of certain functions depends on the interconnection between the SIMPA indexer and the amplifier. One should pay particularly attention to the control of the "standby" and the switching-on or switching-off of the motor.

III.3 - Absolute step counter

Each SIMPA module has an absolute counter of the number of steps or microsteps carried out. According to the direction of rotation, this counter counts or deducts the number of steps (or microsteps in microstep mode) generated. The absolute step counter authorizes returns to the origin 0 starting from any position. It can be reset at any time (provided the motor is stopped).

The range of the absolute step counter is:

Basic version	-8 388 608 ? CPa ? +8 388 607 (modulo 2^{24})
Microstep version	-2 147 483 647 ? CPa ? +2 147 483 647 (modulo 2^{32})

III.4 – Operating Parameters

III.4.1 - Parameters defining the motor movements

Each movement is defined by five parameters:

- The starting spee: V_{min} from 20 to 20 000 step/s (or ½ step/s) defines the initial speed of all the movements.
Taking into account inertias, the speed of the motor cannot take instantaneously any value, it is appropriate to define a speed from which the movement of the motor is accelerated until desired speed.

- The plateau speed (V_{max}) from 20 to 20 000 step/s (or ½ step/s) defines the nominal speed of execution of movements.

- Duration (Ta and Td) of the acceleration and deceleration slopes allowing to go from V_{min} to V_{max} or inversely (from 1 to 65 000 ms).

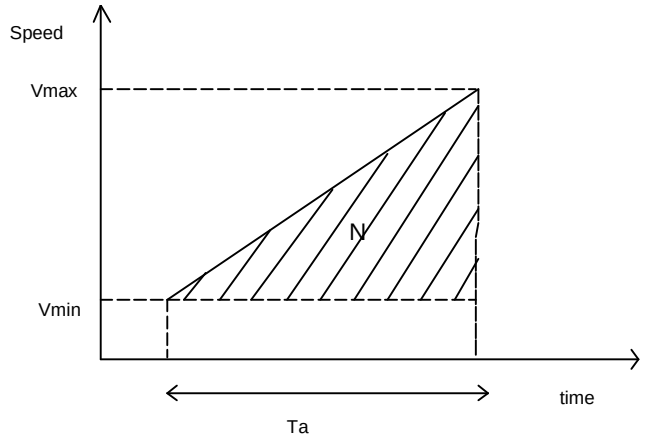
- The resolution ? in microsteps per step for the boards working in microstep mode;

- The number of steps or microstep N to carry out and the direction of rotation of the motor.

III.4.2 - Law of acceleration / deceleration

Most of the time, the main effort to be overcome is related to the inertia of the system. A variation of linear velocity (constant acceleration) causes a constant reaction torque of the moving system.
SIMPA modules automatically determine speeds to generate during the phases of acceleration or deceleration in order to maintain acceleration constant during assigned times Ta and Td to go from Vmin to Vmax or respectively from Vmax to Vmin. These "speed profiles" are respectively called law of acceleration and law of deceleration.
For full step or half-step modules, times Ta and Td must be equal equal to the only time of definable slope Tr.
For microstep modules, Ta and Td are equal by default, but can be differentiated (only via the serial link).

The theoretical law of acceleration being a law with constant acceleration, it is represented by a line on a graph speed versus time (the same explanations applies to the law of deceleration).

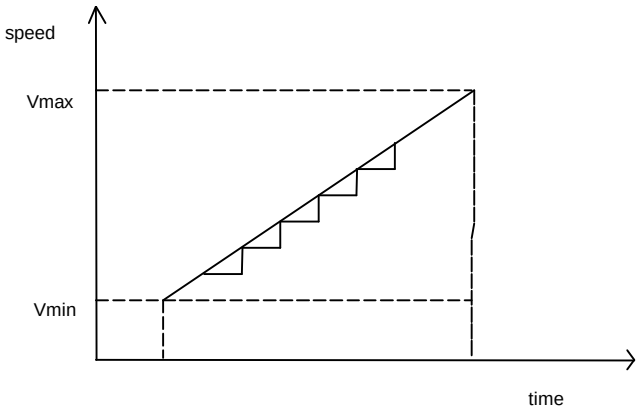


Parameters of the law being:

V_{min} = start speed
V_{max} = plateau speed
T = slope time
µ = resolution in microstep/step

This straight line has an acceleration slope of: $\frac{V_{max} - V_{min}}{T_a}$, the total number of microsteps executed during this acceleration phase is the integral of the speed : $N_a = \frac{(V_{max} - V_{min})}{2} \times T_a \times \mu$

The actual slope of acceleration is the decomposition of the theoretical slope in up to 250 stages of nearly identical durations (this is not completely checked for fast accelerations and particularly on low speed stage).



Speed in a point of the slope at time Ti is given by:

$$v_i = \frac{1}{p_i} = \frac{t_i}{T_r}(V_{max} - V_{min}) + V_{min} \quad (\text{in step/s})$$

Taking into account errors due to the discretization of the calculations and the microstep generator's resolution: 0,5 ?S in microstep version, 1 µs in full step or ½ step version.

At least 1 step or microstep is carried out on each stage. The iterative algorithm stops as soon as the speed on the stage being calculated is equal to the maximum speed.

The calculation of the law is only possible if the movement parameters are compatible with the size of the memory reserved for the storage of the law of acceleration (see imposed conditions on paragraph III.4.6).

III.4.3 - Full step, half step and microstep mode: units

The internal software of SIMPA modules can handle the modules according to 3 different modes:

- Full step mode,

- ½ step mode,

- microstep mode.

On the other hand the module's own electronics is specific and cannot handle all of those modes.
According to the type of module selected, it handles:

- Either microstep mode for the modules microstep,

- Or full step mode / half-step mode for full step / half-step modules because then, clock generation which controls movements is the same.

NB: Of course, microstep versions can work in full step or ½ step mode, defining a resolution of 1 or 2 microsteps per step.

1) Defining the resolution

- On full step / ½ step modules, a jumper or a microswitch allows to choose the desired resolution between the 2 only possible ones.

- For microstep modules, the resolution in microsteps per step must be defined by the user and corresponds to the resolution selected on the power amplifier if this one is independent of the SIMPA unit (including 1 microstep per step or 2 microstep per step).

2) Defining the speeds

- In full step mode, frequencies are defined in step par second (step/s)

- In half-step mode, speeds are in half-step per second (½ step/s) there is thus a ratio of 2 in the motor speed for identical values in comparison with full step mode.

- In microstep mode, speeds are always defined in full step, so that a modification of the resolution does not involve any modification of the motor speeds (provided that the resolutions are identical for SIMPA module and power amplifier, if this one is external).

Note: The values returned by the module correspond to the speeds actually generated.
They can differ from the programmed values, taking into account the resolution in time of SIMPA module: 1 ?s version full step, 0,5 ?s microstep version.

3) Definition of displacements

Displacements are always defined in the resolution of the module such as it is used:

- In full steps for modules in full step mode,

- In ½ steps for modules in ½ step mode,

- In microsteps for microstep modules.

For example it will be necessary to define a displacement of 13 to obtain a movement of 1 full step with a microstep module which resolution is 13 microsteps per step.

III.4.4 - Motor current control

The maximum motor current that a module can deliver is one of the elements that distinguishes various SIMPA family modules.

The current really applied to the motor can however be modulated, independently of hardware configurations:

- By programming motor current parameter in SIMPA when available (see User's manual of the module used)

- By using the "standby" mode (or wait): forcing, when the motor is stopped, of a holding current lower than the selected rated current.

- By using the "Boost" mode (overcurrent): forcing of a higher driving when accelerating or decelerating the motor. This function is not available on all SIMPA modules.

* In full step / ½ step version

The standby current is managed by logical output 1.
The "wait" state (logic 1) corresponds to the quiescent current (standby). Management is automatic only for direct movements. In sequence, the user must manage the logical output 1 according to the current that he wishes to apply to the motor. The "boost" mode is never accessible.

* In microstep version

standby and boost currents management is completely transparent and independent of the management of the logical outputs, even during a sequence.
However, the user can separately prevent the boost modes and standby if he wishes (using the MS command later described).

III.4.5 - Tolerated slip

In sequence, the absolute step counter can be compared with a given value at any time. If the differnece is higher than a fixed value the sequence can be diverted towards a particular treatment.
This functionality, associated to the detection of limit switches or position encoders, makes it possible to be sure that the motor did not stall.

III.4.6 - Limits and coherence of movement parameters

III.4.6.1 –Full step / ½ step version

III.4.6.1.1 - General limits

* 20 ? Vmin < Vmax ? 20000 (step/s or ½ step/s)
* ? = 1 only (not programmable)
* 1 ? TR ? 63535 ms
  -999998 ? Nrelative ? +999998 (step or ½ step)
-8388608 ? Nabsolute ? +8388607 (step or ½ step)

III.4 6.1.2 - Coherence between parameters

V_{max} ? Tr ? 65 10^{6}

III.4.6.2 - Microstep version

III.4.6.2.1 - General limits

* 62 ? Vmin < Vmax ? 20000 (step/s)
?

* 1 ??? 256 for microstep modules without integrated power
(SIMPA microstep or SIMPA microstep Front Panel…)

µ: 1, 2, 4, 8, 16, 32 or 64 for microstep modules with integrated power
(SIMPA microstep 1 axis, SIMPA microstep 1 axis Front Panel, SIMPA microstep 4 wire or SIMPA microstep 4 wire Front Panel…)

* 1 ? TR ? 65535 (ms)

* -2147483647 ? $N_{relative}$? +2147483647 (microstep)

* -2147483647 ? $N_{absolute}$? +2147483647 (microstep)

III.4.6.2.2 - Cohérence entre paramètres

III.4.6.2.2 - Coherence between parameters

$$? \quad \frac{10^{?3}}{\mu ? Tr\ (ms)} \quad ? \ V\min\ (p/s) \quad ? \quad \frac{63{,}75\,10^{6}}{\mu ? Tr\ (ms)}$$

$$? \quad \mu ? V\min \quad ? \quad 20\,000\ (p/s)$$

$$? \quad \mu ? V\max \quad ? \quad 1{,}28\,10^{6}\ (p/s)$$

?	X	Tr (ms)		
		? Vmax ? 16 KHz	16 KHz < ? Vmax ? X	X < ? Vmax
1		$Tr < \frac{63{,}75\,10^{6}}{VMAX}$		
2 . 3	32 KHz	Tr ? $\frac{63{,}75\,10^{6}}{? * Vmax}$	Tr ? 3984 ms	Tr ? $\frac{127{,}50\,10^{6}}{? * Vmax}$
4 .. 7	64 KHz			Tr ? $\frac{255\,10^{6}}{? * Vmax}$
8 .. 15	128 KHz			Tr ? $\frac{510\,10^{6}}{? * Vmax}$
16 .. 31	256 KHz			Tr ? $\frac{1{,}02\,10^{9}}{? * Vmax}$
32 .. 63	512 KHz			Tr ? $\frac{2{,}04\,10^{9}}{? * Vmax}$
64 .. 256	1024 KHz			Tr ? $\frac{4{,}08\,10^{9}}{? * Vmax}$

III.5 - Direct movements

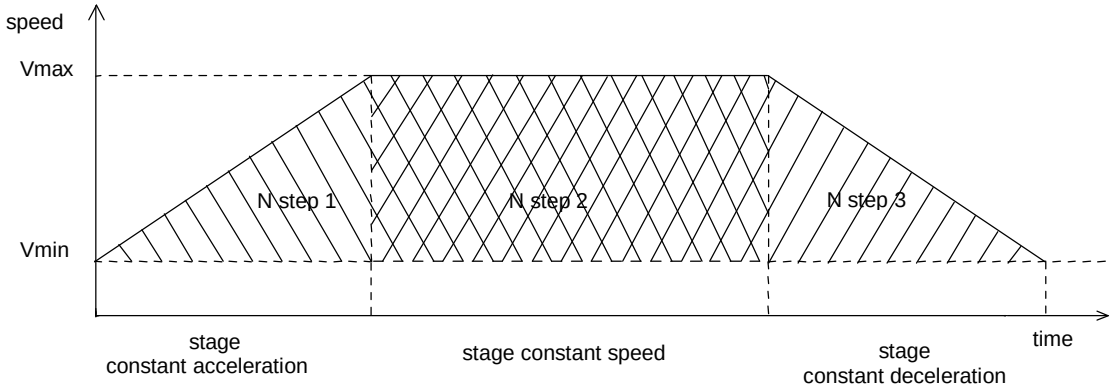
They are simple movements carried out immediately when the command is received by the module.

III.5.1 - Description of the basic movement

Direct movements can always be divideed into three stages:

- Constant acceleration stage with boost current (if available),
- Constant speed stage with rated current,
- Constant deceleration stage with boost current (if available).

At the end of the movement the current gets back to the "wait" level (standby).



N: number of steps or microstep to be carried out.

The module automatically calculates the number of steps or microsteps to be carried out during each stage of the movement according to the parameters of movement.

N = Npas1 + Npas2 + Npas3

If the total number of steps or microsteps imposed is not sufficient for the module to carry out complete stages of acceleration and deceleration, those stages are truncated and there is no constant speed stage.

NB: Generally, Npas3 = Npas1 (except for microstep modules if acceleration times (Ta) and deceleration (Td) programmed are different).

III.5.2 - Types of movements

SIMPA modules can manage three types of movements:

- Relative movements: the given set point defines the number of steps or microsteps to be carried out starting from the current position of the motor. The direction of rotation is defined by the sign of the set point.

- Absolute movements: the instruction defines the new position to be given to the motor.
The module automatically calculates the number of steps or microsteps to be carried out and determines their direction. A return to the origin can be done directly, without any set point. A new origin position can be set at any time, either by replacing it by the current position of the motor (which is equivalent to resetting the absolute steps), or by giving a new value to the current position.

- Infinite movements in one direction or the other: after an acceleration stage, the motor keeps going at plateau speed, waiting for a stop command.
In microstep versions, the speed stage of an "infinite" movement can be forced with a value lower than the preset maximum speed parameter, and changed during the movement without having to stop the movement first.

All these movements can be stopped by command before their complete execution, either instantaneously (care with motor stall), or with deceleration.

The activation of the "Limit Switch Mode" allows he user to stop any movement by means of 2 external switches connected to logical inputs E7 and E8 respectively associated to **+** and **-** directions. The activation of a limit switch in its direction causes the motor to stop immediately.

Note: In microstep mode, the deceleration stage can be followed of a certain number of microsteps carried out at minimum speed. This number cannot exceed 1 motor full step.

III 6 - Status of the module

- Status register:

This register allows the user to know the status of the module, and particularly:

.	An execution of phase 255	:	S
.	Some power supply defaults	:	W
.	Hardware failures of the module (short-circuit…)	:	X
.	Logical inputs failures	:	E
.	Memory loss	:	M

- Parameters:

Most operation parameters of the module can be checked through the serial link.

- Sequences:

Phases of the sequences can be read one by one.

III.7 - Sequences

In order to carry out more complex operations, SIMPA modules can memorize successive actions. This succession of elementary actions is called "sequence". Its progress can be conditioned by external events. This operating mode turns SIMPA modules into genuine automats.

III.7.1 - Sequences organization

Sequences consist in elementary actions called phases (relative movement, absolute movement, wait…). Each sequence is called with a number defined by the operator at its creation. Phases of a sequence are also defined by a specific number. These definitions are bi-univocal.

A certain number of commands make it possible to prepare, select, carry out and read sequences.

Several sequences can be automatically chained with or without condition.

III.7.2 – Preparing a sequence

The preparation of a sequence is composed of two stages:

- The creation of the sequence which reserves memory space. The number of the sequence and the number of phases to be done are to be specified in this stage

- The writing of the phases itself.

The "delete" command of a sequence makes it possible to release the place reserved for a sequence not used.

Of course, each sequence can be checked with reading again its phases one by one.
But this does not constitute an "editor". The description of a sequence will have to be written in a file using PCSIM or any text editor, and then to be downloaded into the module.

The writing of the sequences and their phases can be done in any order, to rewrite a phase with the same number erases the contents of the already written phase and replaces it by the new content.

III.7. 3 - Selection and execution of the sequences

Sequences can be run by several ways with SIMPA modules:

- Immediate execution of the selected sequence with the host computer (possibly via PCSIM or Libsim) or by means of the front panel for modules that are equipped with it or of the option selector of sequence

- Delayed execution of a chosen sequence at Power up (or Reset)

Of course, the operator is free to stop any running sequence or to disable the automatic execution.

III.7.4 - Sequences progress

Any sequence begins with phase N ? 1 and finishes normally with the last defined phase.

Phases of a sequence can be chained, starting from phase 1, in two distinct ways:

- Natural branching,
- Conditional branching.

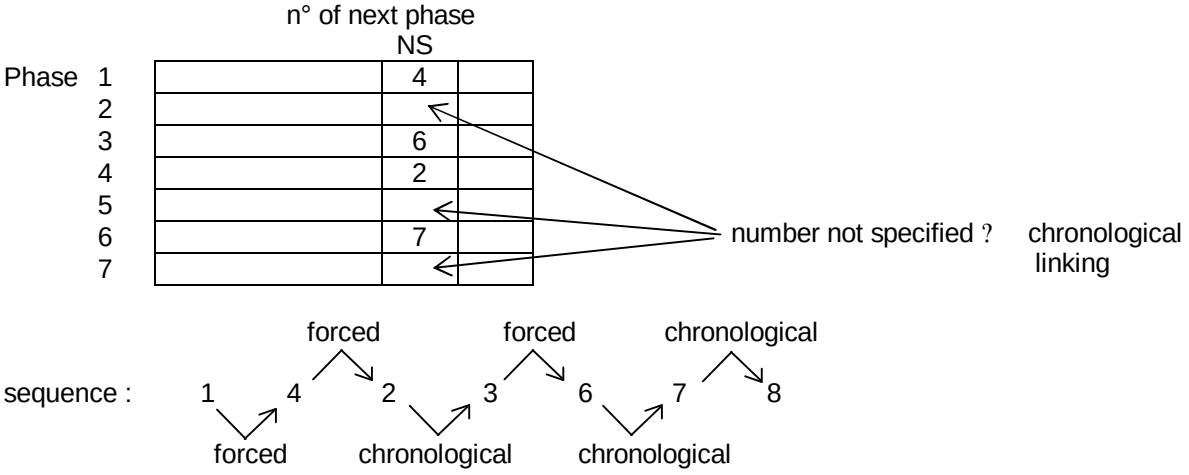
III.7.4.1 - Natural branching

By default, if nothing if specified, the phases are linked chronologically from the first one to the last one, in natural order 1, 2, 3... N.

This order can be forced by specifying in the definition of the phase the number of the desired next phase (NS parameter of the phases definition command).

If the next phase is not defined or does not exist, the sequence stops on this phase when running, even if other correctly defined phases come next.

Example:



III.7.4.2 - Conditional banching

Logical inputs make it possible to modify the chronological sequencing of the phases.

These phase shifts can apply:

- At the end of the phase in progress,

- During the phase in progress.

a) Conditional branching at the end of the phase in progress (NE)

Jumping to a particular phase, different from the natural next phase, is related to the state of a particular logical input. The 8 logical inputs can be related with 8 different phases. If several inputs are selected and active at the end of the phase in progress, the following phase will be the one associated the first one (in the chronological order 1, 2, 3.... 8).

For a given phase, each input can be associated only with only one phase, but the same number of phase can be associated with several inputs, making it possible to carry out "or" combinations. "and" combinations can be implemented by wiring or by using conditional phases branching (or by managing the inputs' polarity, for microstep versions only).

Conditional branching allows to realize very easily:

- Repetitions of one or several phases until an input toggles.

- Selective execution of phases.

By associating phases conditional branching to the sequences linking described further, one can also build repetition of sequences and conditional sequences.

b) Conditional branching during the phase in progress (NF)

This branching differs from the precedent only by the fact that inputs are not treated any more on their state at the end of the phase but at the time of their activation. The phase in progress is then stopped immediately and the connection to the phase associated with the input that switches is carried out.

If the input is already positioned at the beginning of the execution of the phase, the conditional jump immediately takes place without execution of the phase (Caution with logical outputs, see further).

c) Inactive input, active input

Logical inputs of SIMPA modules are either opto-isolated, or pulled-up to 5 V.
The inactive state of the inputs corresponds in their idle state: no current or pull-up to 5 V. It is associated to **logical** state '1'.
Conditional jumps are thus associated with **logical** state '0' corresponding normally either to the activation of the input's opto-coupler, or to the reset (0V) of a non isolated input.

* In microstep versions, it is possible to modify the polarity of the logical inputs.

- **Globally** with limit switches commands MB and MN
  - The **L** parameter by default associates the electric active state to **logical** level 0
  - The **H** parameter associates the electric active state to **logical** level 1

- **Individually** , specifying a ' - ' (minus) sign before the address of the jump makes the branching on **the inactive logical** state (1).

III.7.5 - Reserved phases and sequences

III.7.5.1 - Stop Phase: 54 (254)

Accessing this phase causes the sequence to stop. The sequence is then finished. If a chained sequence is defined, it will then be run.

In the execution of a sequence, accessing a phase that is not defined automatically connects to the stop phase. The user can use this phase to normally finish a sequence by explicitly naming it as the following phase.

- In full step / ½ step version, the number of the Stop Phase is 54 and logical outputs are reset (default state).

- In microstep version, the number of the Stop Phase is 254, logical outputs are unchanged.

III.7.5.2 - Interrupt Phase: 55 (255)

Accessing the reserved interrupt phase enables the user to finish the sequence as with a stop phase and to make the module to generate an interrupt signal ("BREAK" on the serial link).
Only sequence 99 can be executed next to an interrupt. If sequence 99 is not defined, no other sequence will be run.

- In full step / ½ step version, the number of the interrupt phase is 55

- In microstep version, the number of the interrupt phase is 255

III.7.5.3 – Sequence 99

The end of interrupt phase (255) automatically branches to reserved sequence N ? 99.

This sequence can be used to carry out safety movements, as for example return to an absolute position, search for limit switch, alarms…

It is not necessary to define this sequence if it is not going to be used, or if no action is wished.

III.7.6 - Linking Sequences and Sub-sequences

III.7.6.1 - Linking

It is possible to chain several sequences by indicating in each sequence the number of the desired following sequence. If no number is indicated, no sequence is chained.

The number of the following sequence can be specified in any defined phase of the sequence (it does not have to be in the last phase), provided the phase which defines the next sequence is run, which is not always the case with conditional branching. This also allows the user to choose the next sequence to be run according to logical inputs.

The number of the following sequence is given by optional parameter NL of phases definition command.

When several phases give different numbers of following sequence, the sequence pointed by the last phase executed that will be will be carried out.

Caution! Between each sequence, SIMPA module gets back to its idle state: the motor is stopped in standby position and in full step / ½ step version, the logical outputs are forced to their idle state. There is no continuity in the movement.

III.7.6.2 - Sub-sequences

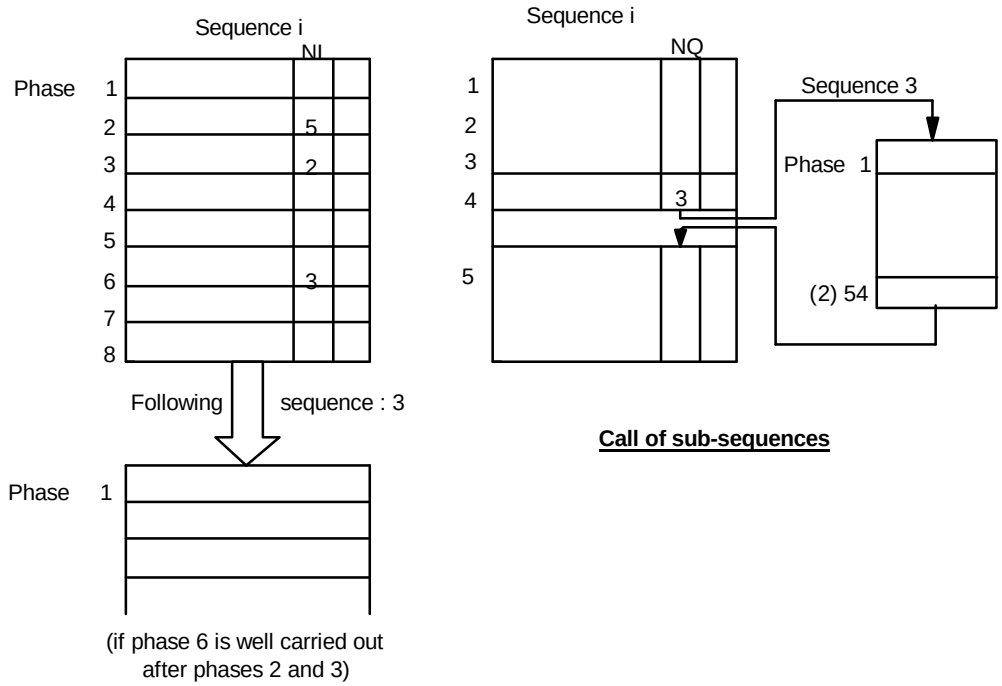
The execution of a sequence can be temporarily stopped to carry out a sub-sequence thanks to the NQ command. Contrary to the chaining, the module does not go back to its idle state, the logical outputs are unchanged, the control of the motor is entirely done by the sequences.
Caution with possible discontinuities of the movements due to the transfer time between sequences.

The execution of sub-sequence is run at the phase that calls it.

The end of execution of sub-sequence gets back to the phase chronologically following (Number of phase +1) the one which called sub-sequence. Several phases of the same sequence or different sequences can call the same sub-sequence. Sub-sequences are identical to standard sequences, except that the natural end of a sub-sequence brings back to the calling sequence. Sub-sequences can perfectly be carried out alone like any ordinary sequence without being called by another sequence.

Caution! The access to the Interrupt phase (255) clears the return address since it chains to sequence 99. Moreover only one level of sub-sequence is authorized: sub-sequences must not call other ones!

III.7.6.3 – Examples



Linking of sequences

Call of sub-sequences

III.7.7 - Sequences description

Each phase of a sequence is described by:

- Its sequence number: ns.
  Any phase is related to one (and only one) sequence.

- A phase number: np.
Phase 1 will be run first. Then phases are linked chronologically with possible unconditional or conditional branching.

- The type of the phase: Na

NA	:	acceleration
ND	:	deceleration
NV	:	constant speed
NP	:	relative movement (identical to direct relative movement)
NX	:	absolute movement (identical to direct absolute movement)
NH	:	return to origin ("HOME")
NZ *	:	absolute step counter reset. Determines new reference for absolute movements
NT *	:	power motor ON
NU *	:	power motor OFF
NW	:	wait
NC	:	modification of plateau speed V_{max}
NG	:	stall detection
NY	:	wait for with serial link synchronization using TIME OUT (PG command)
PV	:	variable programming
PI	:	modification of the to the initialization value of a variable
PR	:	modification of the current value of a variable
PC	:	transfer of the absolute step counter in a variable
PA	:	variable addition
PT	:	variable test

- The set point: Cns (number of steps, position, plateau speed, time, n° of parameter…
  NV, NP, NX, NC, NW). For a relative movements, the direction of the movement is defined by the sign of the instruction.

- Optional following parameters:

NL-NQ: Number of the sequence to be carried out at the end of the sequence in progress: NL sc, or number of the sub-sequence to be called at the end of the execution of the phase: NQ ss

NE-NF: relation between logical inputs and sequences to be branched to, allowing conditional branching. The branching will be made according to transitions (NE) or to logical states (NF).
8 addresses X1 to X8 can be defined associated with the 8 inputs (0 for unused inputs).

NS: number NS of the phase to be run if no conditional branching occurs. By default, the following phase is the chronological following phase (phase in progress number + 1).

NO: state of logical outputs during the phase. By default, the state of the outputs does not change from the previous phase. At the end of the sequence the outputs are put at idle state (logical state "1"), only in full step / ½ step version.

The structure of the command is then:

SP ns np Na [Cns] [NL sc] [X1… X8] [Ns ps] [NO out]
[NQ sq] [NF]

Note: hook specifies optional parameters.

Particularity of full step / ½ step version:

- The direction of the movement must be defined by a parameter: S+ or S- (for phases where direction is not important, one can specify either S+ or S-).

- A set point is obligatory (for example 0), even for phase like NH, NZ, NT and NU.

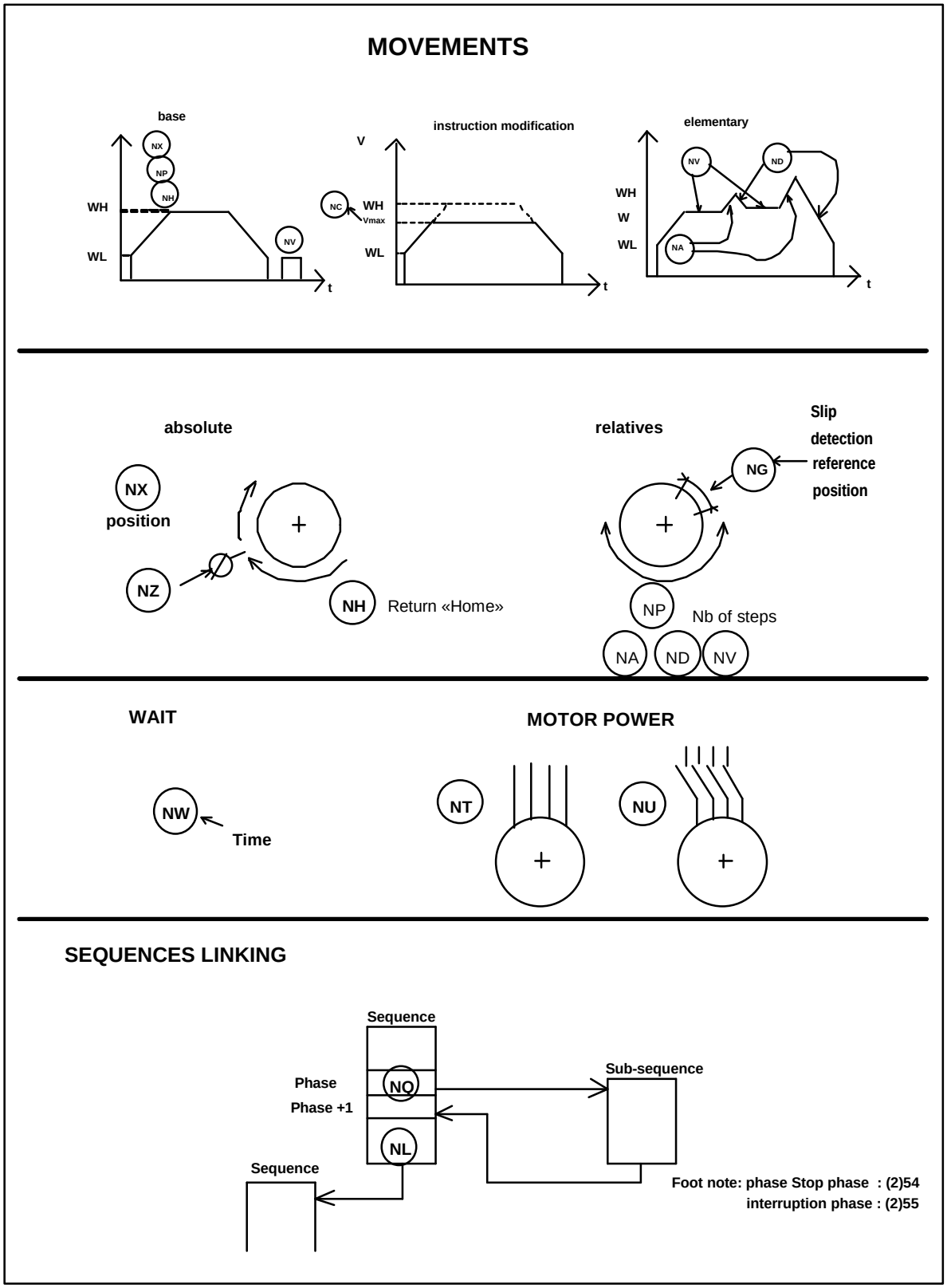
the structure of a phase is then:

SP ns np S+ Na Cns [NL sc] [X1… X8] [Ns ps] [NO out]
S- [NQ sq] [NF]

III.7.8 - Limits of functionality of the sequences

	Full step / ½ step version	Microstep version
Sequence Number	1 ? ns ? 99	1 ? ns ? 120
Total number of phases available	413	384
phase number in a sequence	1 ? np ? 50	1 ? np ? 200
Variables number	none	32
Relative displacement set point	-999998 ? dr. ? +999998	-2147483648 ? dr. ? +2147483647
Absolute	-8388606 ? da ? +8388606	-2147483648 ? da ? +2147483647
temporisation	1 ? W ? 65535 ms	

III.7.9 - Summary of different phases



III.8 - Logical inputs Utilization

III.8.1 – General information

Operation in sequence mode of SIMPA modules allows the management of the 8 logical inputs.

As presented before, logical inputs make it possible to interfere on phases chronological course of a sequence, either in an immediate way (transition), or at the end of the phase (level).

The normal active state of an input is logical level 0.

For microstep versions, the polarity of the inputs can be globally modified thanks to MB and MN commands.

By wiring inputs/outputs of several modules one can synchronize the movements of various axes (which is done, on 4 axis SIMPA boards).

III.8.2 - Time needed to take into account a logical input

Normally the logical inputs are managed in real time with the resolution of step or microstep. However, the higher the speed of the motor, the smaller the time available to manage inputs. the response time can then take several motor steps or microsteps.

Consequently, time needed by the module to take into account an input can go up to 18 ms. As long as there is no dialogue on the serial link, this time does not exceed 10 ms, it is appropriate to maintain the inputs in a determined state for more than 20 ms so as to be sure not to miss the active transition of the input.

IMPORTANT REMARK: So as to manage conditional branching according to logical states at the end of a phase, the logical state of an input should be held for the time of the phase.

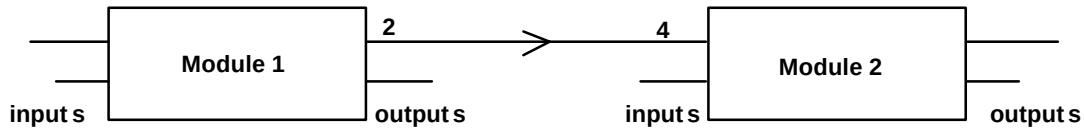
Example:

For a movement of 20 steps run at constant speed of 200 step/s, it lasts $\frac{20}{200}$? 0,1 second

In this case it is advisable to hold an input active during at least 100 ms if one wants it to be always taken into account at the end of a phase.

III.8.3 - Synchronization of modules

By connecting the output of a module to the input of another module, it is possible to link the action of the second module to an event on the first module.



Example:

Generating a Synchro Phase on module 1:

SP ns_1 np_1 + NW $_{T1}$ NO FD

Synchro waiting phase on module 2:

SP ns_2 np_2 + NW t_2 NE 00 00 00 05 00 00 00 00 NS np_2

Module 2 loops on the phase np2 waiting for the activation of its input 4 to carry out phase 5.
Module 1 activates its output 2 for a time T_1 that must be higher than the duration T_2 of the waiting phase on module 2.

III.8.4 - Logical inputs defaults

Too much activity of logical inputs, for example because of noise, could destroy the performances of SIMPA modules (particularly the serial link). Also, beyond a certain frequency (? 10 Khz) logical inputs are not taken into account any more, the motor is put in idle state and any sequence in progress is stopped, module state returns E and inputs state is then materialized by XX.

Only a module reset or a power off/on can go back to normal operation.

III.9 –Logical outputs control

III.9.1 – General information

SIMPA modules have 8 independent logical outputs fully controllable by the user. These outputs can be controlled via the serial link thanks to the "GL" command or in sequences thanks to the NO command. These actions are generally done immediately (or at the beginning of a phase in sequences).
For SIMPA microstep modules it is however possible to condition the change of logical outputs to the passage to a definite value of the absolute step counter (CPA) using the GP command.

III.9.2 – Immediate control

The new states of the 8 logical outputs are defined by a byte (2 hexadecimal characters (0… 9, A… F))
For the modules microstep, it is possible to modify each output independently using a mask defining which output should be modified:

- 0: not modifiable logical output.
- 1:logical output to modify

The commands or directives are written:

GL OUT	or	NO OUT	for all modules
GL OUT [: MSK]	or	NO OUT [: MSK]	for the microstep versions only

OUT is the desired state of the 8 outputs and MSK, the possible mask to apply. The new state of the logical outputs after execution is:

$$S_t \; ? \; (OUT \; ? \; MSQ) \; ? \; (S_{t?1} \; ? \; \overline{MSQ})$$

When the mask is not specified (GL OUT or NO OUT) all outputs are modified as for MSK = 0FFh

III.9.3 – Delayed control: only for microstep modules

Delayed control is implemented by the command:
GL OUT: MSK typ

In this case, the GL command does not occur immediately but when the absolute step counter gets to the value defined by the GP command.

III.9. 3.1 – Absolute, relative

Two kinds of operation are possible according to the type (typ) given in the GL command:

- Absolute: the modification position is given directly in the GP command

- Relative: the modification position relates to the current value of the absolute step counter when the GP command loads the function. The position of modification is given by the value defined by GP added to the current value of the absolute step counter.

Anyway, it is possible to choose not only the position of activation but also to condition activation with the meaning of the movement by writing the right sign before the parameter. If no sign is specified, the direction is indifferent.

III.9.3.2 – Pulse generation

GP command allows the definition of a pulse width by defining a second position (in the same way) for which logical outputs will be complemented (always through the mask if necessary).

By using together commands: GL OUT: MSQ typ and GL POS: DEL

One obtains when the value of the ASC (absolute Step Counter) is POS $S_t \,?\, (OUT \,?\, MSQ) \,?\, (S_{t?1} \,?\, \overline{MSQ})$

Then when the ASC is DEL $S_{t'} \,?\, (\overline{OUT} \,?\, MSQ) \,?\, (S_{t'?1} \,?\, \overline{MSQ})$

When absolute:

t is defined by $ASC_T = POS$ and
- $ASC_{t-1} < ASC_T$ if typ = +A
- $ASC_{t-1} > ASC_T$ if typ = - A
- or typ = A

And t' is defined the same way: $ASC_{t'} = DEL$ and
- $ASC\ t'_{-1} < ASC_t$ if typ = +A
- $ASC_{t'-1} > ASC_t$ if typ = - A
- or typ = A

When relative, if the position of the absolute step counter when GP command is ASC_O

t is defined by $ASC_T = POS + ASC_O$ and
- $ASC_{t-1} < ASC_t$ if typ = +R
- $ASC_{t-1} > ASC_t$ if typ = - R
- or typ = R

And t' is defined the same way: $ASC_{t'} = DEL + ASC_t$

Caution! The transition direction of the absolute step counter taken into account is independent from the direction of the movement during GP command, in particular for relative operation.

III.9.3.3 – Repetitive operations

GP command can be used without parameter to restart the very same "GP" command as the last carried out.

Moreover, it is possible to activate a repetitive mode by adding the "M" type to the GP command that causes automatic reload of the output function differed at the end each execution.

In absolute mode, logical outputs will be activated (through a mask) every time the ASC position is POS (possibly replaced by DEL position) as long as the delayed management function is not stopped by GPA command.

In relative mode, outputs are activated every X step (x = POS) provided the movement direction is not modified, nor is the value of the ASC by a reset to 0 (DI command) because switching time management is still absolute. SIMPA module determines the future switching position by adding current position to the desired relative value (there is no count of independent step).

III.9.3.4 – Permanent operation

Usually, delayed logical outputs operation is erased (unload) by a Hardware or Software Reset (power up or MR command). The P mode (permanent) defined in GP command makes it possible to reload the delayed logical output function at Reset, whatever the defined types or other modes.
It is then possible, for example, to generate a synchro for each motor turn without having to have it redefined at every powering up.
Permanent operation is erased by any GP command except if this one contains the 'P' parameter.

III.9.3.5 – Interaction with sequences

Delayed outputs management is completely independent from immediate outputs management.

It is then possible to modify directly the state of the outputs by a GL command (without any absolute or relative mode) or a NO command in a phase, even when a delayed logical outputs activation has been run before by a GP command.

There can be three cases:

- The outputs activation by the delayed command already occurred, then the immediate command modifies the state of the outputs from the obtained state,

- Delayed activation did not occur yet, the immediate command imposes its state, it is this state that will be considered as present state (S_{t-1}) by delayed activation,

- Delayed activation is quasi simultaneous from the immediate command, it is then possible that the immediate command is not really carried out, if it stops the process of delayed modification in particular for an immediate command requested by a sequence. It is thus recommended to avoid a "simultaneous" modification of logical outputs and to insert time or distance for non-overlayso as to avoid any risk.

III.10 – Variables (only the microstep versions)

Microstep version SIMPA modules have variables that can be useful to increase the performances of automatisms: cycles count, calibration, parameters setting are examples of possible use of these variables.

III.10.1 – Variables definition

A variable is a data that can be modified at any time by serial communication or in sequences.
These data are stored in 4 signed bytes, they can represent digital values of set points or binary (or hexa) data like logical output and its associated mask.

III.10.2 – Writing variables

SIMPA software defines up to 32 variables, they are used like any data and are represented by numbers from 1 to 32 preceded by character #

The values of the variables can be modified or imposed by immediate commands:

PG v_1, [v_2 …, [v_{10}]..]]
PD # n : v_N (decimal)
PH # n : v_m (hexadecimal)

Or in sequences by phases natures
PV # n : v_N , PC # n, PI # n, PR # n, PA #: v_m

Their values can be read again by QN # command n or be tested in sequence by a PT # n phase.

III.10.3 – Use of variables

Variables make it possible to modify movements set points during sequences. This functionality is particularly useful to use with sub-sequences.

Furthermore, the value of the logical outputs can also be stored in variable (No command).

The value of a variable can also be used like cycle counter. It is then possible to carry out a given number of cycles: PT phase.

III.10.4 – Variables Initialization, current values

The value of a variable defined by immediate commands PG, PD or PH is preserved when powering up or after a reset (MR.).

On the other hand, the value taken by a variable during a sequence is not preserved at initialization. Only the value defined by the last immediate command is restored, which makes it possible to perfectly define initial conditions.

When current value differs from initial value, the read command of variable gives the two values. By default, the init value of a variable is 0 (factory configuration or command MRZ).

III.11 - Status when powering

Movements parameters are stored in a memory supplied by a battery.
They are then preserved even after powering off of the modules.

However integrity of information stored in memory is checked every powering.

If a default is reported (status = M), then the following default values are imposed (factory values):

V_{min} : 500 step/s (or ½ step/s) (full step / ½ step version) 75 step/s (version microstep)

V_{max} : 2 000 step/s (or ½ step/s) (full step / ½ step version) 1000step/s (version microstep)

T_R: 1 000 ms (duration of acceleration and deceleration slope). 200 step/s (version microstep)

N: 0 (number of steps to be carried out).

?: 1 (full step mode).

Movement direction: +

Tolerated stall (Number of steps): 10

Motor current: 0

Polarity of Logical inputs: active with electric 0

Limit switches: inactive

Outputs: FF (forces "standby" current in full step version)

Value of absolute step counter: 0

The two last values are imposed on each powering or Reset.

In microstep version, the factory configuration can be forced by MRZ command.

IV - OPERATOR INTERFACES

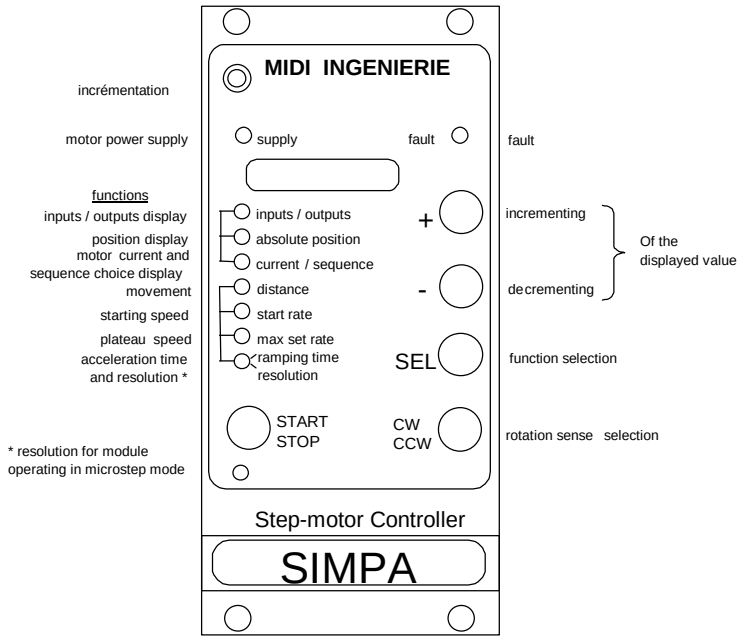
IV.1 - Front panel

Some SIMPA family modules can be optionally equipped with a front panel, real operator interface.

This front panel offers to the operator a simple access to module functions for quick implementation of simple motor movements.

All the functions remain accessible by the serial link, in particular sequences operations.

IV.1.1 - Front panel – Description



The front panel includes five parts:

- In top of the module, a power supply control light and a default visualization ,

- A 7 segments display that visualizes essential parameters of the module,

- Selection of the environment and state parameters of the module:
  * State of logical inputs/outputs of the module,
  * The value of the absolute step counter,
  * The list of existing sequences in memory,
  * The value of rated current of the motor.

- Acquisition of module operation parameters:
  * The number of steps to be carried out,
  * The start speed,
  * The set point speed,
  * Duration of the acceleration slope,
  * The resolution in microsteps per step.

- The lower part is reserved for direct control of the motor:
  * ON / OFF,
  * Movement direction (CW/CCW),
  * Motor activity LED.

IV.1.2 - Selection of the parameters to be visualized or modify

A led is associated to each parameter. It is light when the parameter is actually displayed

For microstep mode modules, "slope time" and "resolution" parameters are both visualized by the same led. The character on the left of the display specifies the nature of the parameters.

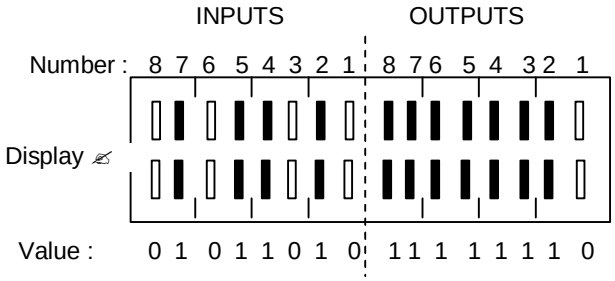
A: Time of Slope

U: Resolution

The selection of the parameter to be visualized or modified is made by successive push on the "SEL" button.

Parameters are displayed with the units defined in the chapter "functionalities", real generated speeds can slightly differ from those defined taking into account time resolution of SIMPA modules.

For inputs/outputs information, each input or output is represented by a specific segment on the display: output are on the right side, input on the left side and 1 to 8 from right-hand towards left hand. A segment is present when the corresponding output or input is active.



On the example:

Input display: 5Bh = 0 1 0 1 1 0 1 0
Output display: FEh = 1 1 1 1 1 1 1 0

For current/sequence parameters, the left side displays "i" the proportion of the rated current of the board applied to the motor, if adjustable by software $\left| \mathrm{Im} \right| \le \frac{Inom}{255} \cdot i$

The right side displays the list of sequences (listed by means of the + or - pushbuttons). A non-existent sequence is preceded by a minus sign.

Note: when the motor is moving, (the led located under the START/STOP pushbutton is ON), display is restricted to environments parameters.

Status of the logical inputs and outputs
Absolute step counter
Number the current sequence
And on the left side, motor current

IV.1.3 - Modification of parameters

Only operation parameters can be modified.

Table 1 for microstep mode modules

Numbers of programmable steps	: 1 to 9 999 999
Start speed	: (62/resolution) to plateau speed (p/s) *
Plateau speed	: start speed to 20 000 p/s *
Slope time	: 1 ms to 65535 ms *
Resolution	: 1 ?step/step to 256 ?step/step

Table 2 for full step and ½ step modules

Number of programmable step	:	1 to 2^{23}-1 (8388607)
Start speed	:	20 to (speed of plateau -1) p/s (or ½ step/s)
Speed of stage	:	(start speed + 1) to 20 000 (p/s or ½ step /s)
Time of slope	:	1 ms to 65535 ms *

* Speed parameters, Slope times of and Resolution must also respect relations of paragraph III.4.6

Principle of modification

The modifications of parameters are done using the "+" and "-" pushbuttons (Bp+ and Bp -).

A short press on Bp+ increments by one unit the selected parameter, and a short press on the Bp- causes a decrement of one unit.

In order to give quick access to the desired value, a long press on the considered BP (+ or -) accelerates the incrementing (Bp+) or the decrementing (BP -) as follows:

- The evolution will be 1 by 1 until the displayed value is multiple of 10, then 10 by 10, then 100 by 100…

Self-incrementing (or decrementing) stops as soon as the value of the parameter reaches the maximum (or minimal) authorized limit.

The release of the pushbutton (+ or -) during a short time (? 1 S.) preserves the row of the incrementing (decrementing) (1, 10, 100, 1000…).

A prolonged release will get back to the 1 by 1 incrementation.

IV.1.4 - Motor control

Rotation direction (or the next motor movement)

The pushbutton CW/CCW selects, alternatively at each press, one of the two directions of rotation of the motor.

The direction of the motor is symbolized by the presence or not of the "-" sign before the "numbers of steps" parameter (CW: clockwise direction " ", CCW: counter-clockwise direction "-").

Note: the press on CW/CCW pushbutton automatically causes selection and visualization of "number of steps" parameter.

ON / OFF

A press on the Start/Stop pushbutton powers on the motor if the corresponding led is off, it will stop otherwise.

The Start/Stop command makes the motor carry out the number of step defined by the distance parameter by respecting speed and acceleration parameters.

Motors stopped by start/stop pushbutton are immediately stopped without deceleration phas.

Caution! if an existing sequence is selected and displayed when the Start/Stop button is pressed, then the module is going to run that sequence instead of a simple movement.

Law calculation

Each modification of speed and slope time parameters involves a law calculation before the first movement is carried out with these new parameters.

If the law calculation is not finished at the time a movement is request, this one will only be carried out at the end of the calculation.

The law calculation of is symbolized by the following pattern on the display: "--------"

Maximum time necessary to calculation is about 4 seconds.

The flashing of the "Absolute position" led announces an overflow of the absolute position compared to the possibility of the display (7 digits). The value is not significant any more, but the internal value of the counter is still correct.

IV.1.5 – Extended functions

Three additional functions are accessible via the front panel:

- Execution of an infinite movement (GF)

- Reset of the of the absolute step counter (DI)

- Complete reset of the module, similar to the MR command.

These three functions are caused by a combined press of + pushbutton and– pushbutton when the following parameters are selected:

Logical inputs / outputs: Infinite movement	(GF)
Absolute Position: Reset of the step counter	(DI)
Sequence / current: Reset	(MR.)

IV.1.6 –Motor current setting

For boards which motor current is programmable, command GI Im (0 ? Im ? 255) allows the definition of its value

The current really applied to the motor is then:

$$\text{Imotor} \quad ? \ \text{Inom} \quad ? \ \frac{\text{Im}}{255} \quad (Aeff)$$

Caution! The Inom value of the rated current depends on the board model.

Im value can also be directly modified on the board, if this one has the optional front panel.

This adjustment of the current is only possible at powering according to the following switches combinations:

Switch

Sw1.4	Sw1.5	Sw1.1 to Sw1.3	Use/Mode
ON	ON	Address pattern	Computer Xon/Xoff protocol
ON	OFF	Address pattern	Computer Ack/Nack protocol
OFF	ON	Address pattern	Terminal
OFF	OFF	Current pattern *	Front panel, Current Adjustment * Computer Xon/Xoff protocol imposed on the serial link * Address forced to 0

Current pattern

Only 8 values of current are accessible:

Sw1.3	Sw1.2	Sw1.1	Im Value	% of the rated current
ON	ON	ON	31	12 %
ON	ON	OFF	63	24 %
ON	OFF	ON	95	37 %
ON	OFF	OFF	127	50 %
OFF	ON	ON	159	62 %
OFF	ON	OFF	191	74 %
OFF	OFF	ON	223	87 %
OFF	OFF	OFF	255	100 %

IV.2 - Sequence Selector

The " sequence selector" interface makes it possible to run the execution of pre-recorded sequences without requiring connection to a computer for modules without front panel.

This option makes it possible to select, thanks to two rotary switches, a sequence memorized in SIMPA module (its number must be between 1 and 99).

The execution of the selected sequence is run as soon as logical input E2 is active. Two operating modes are possible:

- By holding the E2 input active permanently, the sequence called by the rotary switches is carried out immediately. By modifying the position of the rotary switches during the execution of the sequence, one can select the next sequence to be run *(Caution, do not to modify this number close to the end of the execution, the module may then take into account a wrong sequence number).*

- By activating the E2 input only when the sequence is to be run, the module can then be synchronized on an external event (controller, foot switch…)

V - COMPUTER CONNECTIONS AND COMMANDS

The serial link present on each SIMPA board allows a host computer to control the modules.
The same serial link makes it possible to control up to 64 modules of a multiaxis device.

Dialogue with SIMPA modules can be facilitated by the use of PCSIM software, true operator interface for SIMPA modules, or thanks to LIBSIM Handler to interface a program written in an advanced language.

These two PCs compatible software remove all protocol aspect. The user only has to manage parameters and commands.

A Windows version of PCSIM can be delivered on request with the dialogue protocol management DLL.

V.1 - Line and communication protocols

The RS232 communication interface (current loop or V24) is configured as follows:

- Speed: 4 800 bauds (or 9600 bauds only for version microstep)
- Data: 8 bits
- Stop Bit: 1
- Parity: no parity
- Lines used: Rx and Tx

SIMPA modules can be configured in one of the two following dialogue mode: computer mode or terminal mode.

All the modules placed on the same communication line must use the same mode, transmission speed and protocol.

V.1.1 - Computer mode

It ensures a global control of the message (number of characters and "check sum").

Any received (or emitted) message involves an acknowledgment (ACK character: 06h) or non-acknowledgment (NACK character: 15h) according to whether the structure of the message is correct or incorrect (number of characters and "check sum", but no syntax check).

SIMPA modules having emitted a message on the serial link consider the absence of an acknowledgment in a 70 ms delay. Any other character than ACK is considered a non-acknowledgment.

Non-acknowledgments cause two additional emission attempts before positioning an error status.

Interpretation mistakes of a command are announced only during the next addressing of the module by the return, instead of a ACK character, of a BEL character (07h). If the current command was to be returned a NACK, then the BEL will be return by the following command.

ACK/NACK protocol can be completed with an additional protocol allowing each module:

- To avoid receiving other commands as long as those in progress are not completely interpreted,
- To provide an execution report.

By transmitting after ACK or BEL character, a XOFF (13h) character, the concerned module requires the suppression of emissions on the serial link, including commands intended for other modules. The concerned module, after complete interpretation of the received command, authorizes the restart of the dialogue on the line by emitting a XON (1Ah) character if the received commands has been correctly interpreted or a XONERREUR (17h) character if the command could not be executed (syntax error). In the case of a message including several commands, if a command is detected incorrect, the module will generate a XONERREUR, preceding commands have been carried out, following commands are lost.

Errors Management according to the selected protocol:

ACK/NACK protocol: the acknowledgment character returned by the addressed module (module 0 for global command) depends on:

- Syntax of the previous message,
- Structure of the received message.

		Structure of the received message	
		Incorrect	Correct
Syntax of previous message	incorrect	NACK (15 H)	BEL (07 H)
	correct	NACK (15 H)	ACK (06 H)

Protocol XON/XOFF

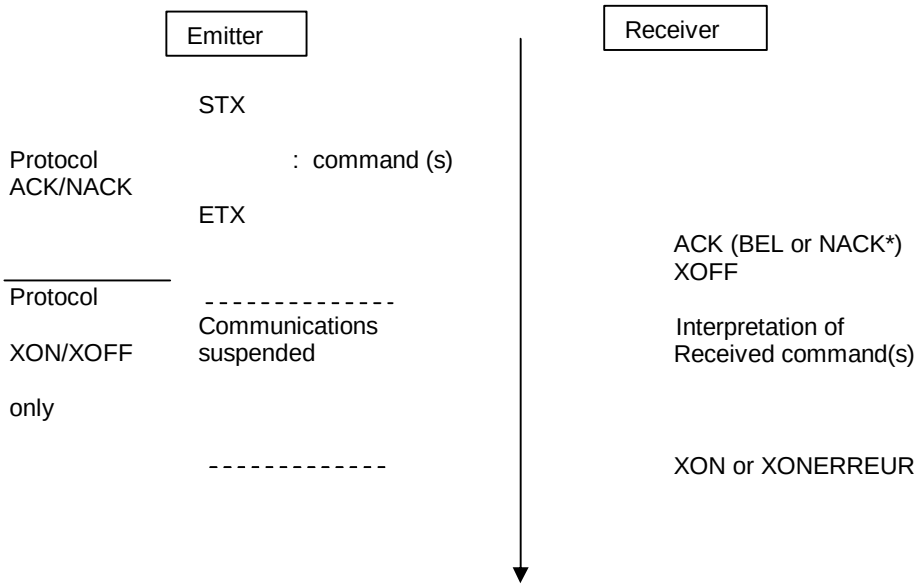
The character of acknowledgment returned by the addressed module depends only on the structure of the received message, the line release character depends on its syntax

		acknowledgment	Release
Structure of received message	correct	ACK (06 H)	
	incorrect	NACK (15 H)	
Syntax of received message	correct		XON (1A h)
	incorrect	BEL (07 H)*	XON ERROR (17 h)

NOTE:
In case of global commands, only module 0 (obligatory) ensures the return of ACK, NACK, XOFF, XON and XONERREUR characters. Other modules memorize the possible errors and will only transmit the BEL character during their next individual addressing.

When a BEL character or XONERREUR is returned by a module, it is possible to query the module (reading status codes) to obtain more information on the cause of the error.

Summary of the protocols



* XOFF character is then immediately followed by XON character

V.1.2 - Terminal mode

This protocol was developed to facilitate dialogue from a simple terminal, without computer, or possibly from a programmable controller.

In this "echo" type dialogue mode, the user has to ensure that echo characters returned by the module are correct. The module returns characters identical to the received ones.

As soon as a CR character (return) is received, the addressed module returns

CR	
LF	line jump
>	next command prompt

If an error is detected by module 0 in the address field, characters CR, LF and "prompt" returned are preceded by:

"_:?" error in the address field of the message (returned by module 0).

If an error is detected on the other fields of the command by the addressed module, the string returned is CR LF preceded by:

"_:!" syntax error, parameter error or unauthorized command.

Note: the symbol "_" materializes a "space" and not an undercsore (20 H)

V.2 - General syntax of elementary commands

a) Computer mode: ACK/NACK protocol with or without XON/XOFF protocol

STX nc [@] cd1 [, cd2, cd3…] CS ETX

STX and ETX : respectively message start (02h) and message end (03h) character.

nc : total number (3 binary coded decimals) of the transmitted characters except STX, ETX, nc and CS.
the number of characters must be lower than 128 (nc ? 127)

[ ] : optional characters.

@ : module address (00 to 63) to which the message is intended.
If the address is omitted, the message is intended for all the modules on the line.
A module of address 00 is essential in all configurations.

Cdi : label of command i with its parameters.

, : separation of commands if several commands are put together in one message.

CS : validity check of the type "check sum". Coded on two ASCII characters, each one representing one of the two hexadecimal digits obtained by making the arithmetic sum (modulo 256) of all the characters of the message except STX, nc, CS and ETX.

The various fields of the commands can be separated by one or more spaces except between the address field (@) and the first command (cd 1) where no space is authorized.

b) Terminal mode: with echo

[@] cd1 [, cd2, cd3…]CR

CR : carriage return

Note: the number of transmitted characters between 2 CR should not exceed 127 characters!

V.3 - Status of the module and error codes

A status register is set during the interpretation of each command received by a module.

The register can be read again using the request: @RX (full step modules) or @QX (microstep modules).
The answer to this request is:

@EE_code

@: responding module address
EE: answer type: status of the module
Code: character specifying the status of the module

* In absence of any defect code N is returned.
* In case of anomaly, an error code is returned

There are two levels of errors:
Level 1: errors related to the status of the module or general errors.
Level 2: command specific errors. These errors are detailed more particularly with each command.

Errors of level 1: the returned character is alphabetical

A: unauthorized command because of the state of the module (motor moving, sequence mode …).
B: immediate stop of the movement on limit switch detection.
C: unknown command (syntax error).
D: phase coordination error:
a phase command out of SP command
a non-phase command in SP command
E: default on logical inputs (switch frequency higher than 10 Khz).
I: in microstep version, prohibited GI command when the used module is an indexer alone (the current is then imposed by the associated amplifier).
M: memory reset (following a default of safeguard, or in microstep version following a forced request for memory reset (command MR0))
S: passage to phase 255.
W: power supply fault.
X: material defect.

Errors of level 2: the returned character is digital.

0: fault related to the command parameter(s):
- absence of parameter
- too much parameters
- syntax of the parameter
1: first parameter is out of limit (commands DG, GI, DR., GA, RS, SE, SS, SA, SC, SD, WH, WL and WT) or non-existent sequence (command SP)
2: second parameter is out of limit (commands RS or SN) or non-existent phase (command SP)
3: the specified sequence does not exist (command RS, SS, SA, SC, SD)
4: unknown phase (command RS)
sequence already created (command SN)
defect on NE or NF parameter (command SP)
5: the number of available phases is insufficient to create the sequence.

VI - DESCRIPTION OF THE COMMANDS

VI.1 - Standard used

In this chapter, each command is described using a standard to express:

- The type and size of parameters,
- The type of addressing (mono or multi-modules),
- The validity of the command according to the state or the type of the module.

An optional parameter is represented between square brackets ' [' and '] '

Example:
GO [ddddddddd]

a) Parameters Type and size

There are three different types:

- Digital : d
- Hexadecimal : h
- Alphanumeric : a .

The maximum size is represented by the number of characters (d, h or a) used to express the parameter.

Spaces are represented by the character _.

Examples:

- Parameter of absolute displacement

  Pa = ±ddddddd

  Means that the absolute position can be given by using a maximum of 7 decimal digits, plus a sign.

- Parameter defining the management of inputs in phases definitions:

  Ne : aa_dd_dd_dd_dd_dd_dd_dd_dd _

  Means that the Ne parameter should consist of two alphanumerics characters followed by 8 groups of three characters (a space and two decimal digits).

Note: For the digital values, the nonsignificant 0 as well as the "+" sign can be omitted or replaced by spaces.

Example: 00340 = __ 340 = 340

b) Type of addressing

The module address parameter is represented by:

- @ if the command can only be addressed to one module at the same time.

- [@] if the command can be addressed to one or the whole of the modules.

- nothing for commands intended for all the modules.

c) Validity of the command according to the status or the type of the module

If the character "*" precedes a command this one is usable whatever the status of the module, otherwise the module must be out of sequence and the motor stopped.

VI.2 - Commands list

The commands can be divided into five groups:

- Initializations parameters
- Reading of the module status,
- Movements (preparation and execution),
- Sequence (preparation and execution),
- Variable

The table below gives for each group commands mnemonic.

Initialization			Module status	Movements		Sequences		Variable
Parameters	Motor current	Logical outputs	Module status	Preparation	Execution	Preparation	Execution	
p µ	p µ	p µ	p µ	p µ	p µ	p µ	p µ	p µ
MB	GI	GL	RD	D+	GA	SA	SS	PG
MN	GM	GP	QD	D-	GE	SC		PD
MR.	GR			DR.	GF	SD		PH
					GH	SE		PC
MS			QG	DG		SN		
WH			RL	DI	GO	SP		
WL			QL	DP	GS	SR		
WT			RP					
			QP					
WN			RS					
			QS					
			RV					
			QV					
			RX					
			QX					

A list of SIMPA commands, provided at the end of the document gives, for each mnemonic, the associated parameters as well as a short definition of the action.

Full step/half-step modules specific commands are pointed out by the index p.

Microstep modules specific commands are pointed out by the index µ.

VI.3 - Glossary of the commands

(D+) $_p$: Clockwise (full step / ½ step modules only) **(D+)** $_p$

Syntax: [@]D+

Parameters: none.

Description: programming of the direction of next relative movements.

This command selects clockwise direction of rotation (according to the motor phases wiring).

No specific error code positioned.

(D -) $_p$: Counter-clockwise direction (full step / ½ step modules only) **(D -)** $_p$

Syntax: [@]D-

Parameters: none.

Description: programming of the direction of next relative movements.

This order selects the counter-clockwise direction of rotation (according to the motor phases wiring).

No the specific error code positioned.

DG: Stall tolerance **DG**

Syntax: [@] DG_np

Parameters: Np = ddd: number of step (or ½ steps or microsteps) defining the maximum tolerated slip.

Digital
Unsigned (absolute value)
Limit: Np ? 255

Description: set the slip threshold taken into account by the phases stall detection (NG).

By default, the threshold is initialized to 10.

Inside a sequence the value of the threshold is not modified. It can only be modified between the execution of two sequences.

The stall value be read using commands (RL) $_p$. or (QL) $_\mu$

Error code

0: absent parameter or incorrect syntax.

1: value is out of range.

DI: absolute position Initialization **DI**

Syntax: [@]DI

Parameters: none

Description: reset of the absolute step counter.

The current position of the motor then becomes the position of absolute reference for next movements: ORIGIN

NOTE: Powering or MR also causes this initialization (with current position of the motor).

No specific error code positioned.

Caution! This command is not accepted when the motor is moving or if a sequence is being run

DP: Reference position **DP**

Syntax: [@]DP_Pr

Parameters: Pr = ±ddddddddd: desired reference position

Digital,
Limit: -8388606 ? Pr ? +8388606 (full step / ½ step version)
-2147483647 ? Pr ? +2147483647 (microstep version)

Description: Programming of new reference position next absolute movement.
This command makes it possible to define without any movement a reference position different from the current position of the motor. Pr gives the absolute position (in the old reference mark) of the new reference.
The position is defined in step or ½ step for the full step / ½ step modules, and in microstep for microstep modules.

Error code

Code 0 in following cases:
- Command DP does not have a parameter,
- The provided set point does not correspond to a digital value.
- The absolute value of the provided parameter is higher than $2^{31}-1$ (or $2^{23}-1$)

$|Pr| > 2^{31}-1$ (microstep version)

$|Pr| > 2^{23}-1$ (full step / ½ step version)

Note: Command DP 0 is equivalent to command DI.

(DR.) $_p$: Relative movement (full step / ½ step modules only) **(DR.)** $_p$

Syntax: [@]DR_n

Parameters: N = ddddddd: number of steps or microsteps of the movement.

Digital,
Unsigned,
Unit: step, ½ step or microstep according to the resolution of the module
6 digits maximum,
Limit N ? 999998

Description: Programming of the number of steps or ½ steps of next relative motor movements (command GO).

This value is saved when powering off, the module thus preserves the last instruction when powering on.

Example: programming of the set point for relative movement of 1890 steps:

DR 1890

Error code

Error code = 0 if:
- The command DR does not have a parameter,
- The provided parameter does not correspond to a digital value,
- The provided parameter is signed.

Error code = 1 if n > 999998

GA: Absolute movement **GA**

Syntax: [@]GA_Pa

Parameters: Pa = ±ddddddddddd: desired absolute position

Digital,
Unit: step, ½ step or microstep according to the resolution of the module,
10 digits maximum,
Limits: -8388606 ? Pa ? +8388606 (full step / ½ step version)
-2147483647 ? Pa ? +2147483647 (microstep version)

Description: immediate execution of an absolute movement.

The module puts the motor to the absolute position defined by the command.

The position is defined in step or ½ step for full step / ½ step modules, and in microstep for microstep modules.

If it is the first movement after the modification of one of the speed parameters, the module possibly finishes the calculation of the law of acceleration before carrying out the movement.

If the module is already at the required position, there is no movement.

With a microstep module, the value given can be replaced by a variable: GA: # n

Example: execution of a movement to the absolute position + 14576:

GA 14576 (the sign + is optional).

Error code

Code 0 in the following cases:

- The command GA does not have a parameter,
- The provided set point does not correspond to a digital value.
- $|Pa| > 2^{31}-1$ (microstep version)

Code 1 in the following case:

$|Pa| > 2^{23}-1$ (full step / ½ step version)

Note

The command GA 0 is equivalent a GH command.

*** GE**: Stop with deceleration *** GE**

Syntax: [@] GE

Parameters: None

Description: the movement in progress (direct or sequence) is decelerated down to Vmin then stopped.

In full step / ½ step version only, logical outputs are disabled ("1" logic (FFh)) after the stop.

The next motor movements will be carried out starting from Vmin.

The command GE possibly finishes the sequence in progress.

No specific error code positioned.

Note:

- If the movement in progress is done at speed v = Vmin, the stop is immediate.

GF: Infinite movement **GF**

Syntax: [@]GF [±] [v]

Parameters: Full step / ½ step mode: None
microstep Mode: v = ddddd new speed value
Digital
Unit: p/s
Limits 0 < v < 20000
The direction of the movement is defined by the sign of the set point if this one is explicit: + or -

Description:
Runs a movement with appropriate acceleration (or deceleration) until specified speed v then keeps going at this speed until a stop command or an initialization (GS, GE, MR.) occurs.
The movement is carried out in the direction given by the sign if specified. Otherwise, the direction is the one of the last movement.
If the movement is in progress (microstep version only) the direction is maintained whatever the specified sign.
If speed is not specified (in particular for full step versions), the speed of the movement is Vmax (defined by WH) if the given value is 0, the movement is carried out at minimum speed Vmin (defined by WL).
If it is the first movement after modification of one of the speed parameters, the module finishes first, if necessary, the calculation of the law of acceleration before carrying out the movement.

Note: Only the movements initialized by GF can be modified by a new GF command in microstep version, in any other case, the command is refused.

Only GS or GE, as MR commands stop an infinite movement.

Error code: 1: parameter is incorrect

GH: Return to origin position: HOME **GH**

Syntax: [@]GH

Parameters: none.

Description: immediate execution of the movement ' Return to origin '. The module carries out a movement until the absolute step counter value is zero.

The origin position can be:

- The position of the motor at powering or after a RESET,
- The position of the motor when DI command is executed or that defined indirectly by DP command.

If it is a first movement after modification of one of the speed parameters, the module finishes initially, if necessary, the calculation of the law of acceleration before carrying out the movement.

If the module is already in origin position, there is no movement.

No specific error code positioned.

Note: Command GH is equivalent to command GA0

GI: Motor current **GI**

Syntax: [@]GI_Im

Parameters: Im = ddd: amplitude of the motor current

Digital,
Unsigned,
3 digits maximum,
Limit 0 ? Im ? 255

$$Imotor = \frac{Inom \times Im}{255}$$

Inom: rated current delivered by the module.

Description: Programming of the value of the current applied to the motor (command without action to certain SIMPA modules: refer to specific documentation).

Example: For a SIMPA module delivering a maximum current of 7A, order GI 128 makes it possible to position the value of current applied to the motor to 3,5 A.

Parameter GI is saved at powering off, the default value at first powering or at factory setting is however: 0 not any current delivered by the board

Error code

Error code = 0 if

- Command GI does not have a parameter,
- The provided parameter does not correspond to a digital value,
- The provided parameter is signed negatively.

Error code = 1 if Im > 255

In microstep version:

Error code = I (i) if command GI is addressed to a module indexer alone (not amplifier), not authorizing the programming of the current!

*** GL**: Positioning of the logical outputs *** GL**

Syntax: [@]GL_Out [: MSQ] [_ t]

Parameters:	Out = hh	hexadecimal pattern to apply on logical outputs
	MSQ = hh	Modification mask: only the bits corresponding to the bits set on the mask are really modified.
in µstep mode	t: type	- if not specified, immediate positioning only
	+A/- A/A	- Outputs will be updated when the ASC reaches the absolute position defined by the GP command (increasing (+), decreasing (-) or indifferently).
	+R/- R/R	- same for a displacement related to the current position of the absolute step counter

Description:

When the type is not specified, this command makes it possible to control overall or partially (using a mask) the logical outputs of the module and thus to activate for example some parallel electromechanical devices when running a sequence.
With a specified type, it is possible to condition the state of the logical outputs to the position of the motor via the absolute step counter. It is thus possible to carry out pulses synchronized with the instantaneous position of the motor.

See GP command for the definition of the positions and width of the pulses, start and stop of the function.

Caution:

The answer to this command, can be disturbed by the course of a sequence or inversely.

In full step / ½ step version, outputs are set back to FF at the end of the movement or sequence, and bit 1 of logical outputs controls the activation of the standby, it is then appropriate to set it to 0 when running a movements in order not to activate the standby function.

Pattern	Active output (to 0)	
FF	None:	Standby Current in the motor
FE	1:	Rated current (full step / ½ step version)
FD	2:	Standby
FB	3:	Standby
F7	4:	Standby
EF	5:	Standby
DF	6:	Standby
BF	7:	Standby
7F	8:	Standby
00	All:	Rated current (full step / ½ step version)

Error code:

Code 0 in the following cases:
- Command GL does not have a parameter,
- The provided pattern does not correspond to a hexadecimal coded digital value.

Code 1 in the following cases:
- The provided parameter is higher than 255 (0FFh)
- Defect on one of the optional parameters

*** GM**: Motor power *** GM**

Syntax: [@]GM

Parameters: none

Description: activation of the motor power: the current injected into the motor is:

- The standby current when no movement command is being executed.

- The rated current if a movement is in progress (possibly programmable by command GI if the module accepts current programming).

Note: in full step / ½ step version , the standby is controlled by logical output #1 which can modify the states previously described.

Recall:
Motor powering is implicit for each direct movement (GO GF GA GH) and when a sequence starts.

Caution! In the case of a SIMPA indexer module associated with a separated amplifier, this command can be without effect (see the explanations given for GR command).

GO: relative movement **GO**

Syntax: [@]GO_[±][n]

Parameters:

full step / ½ step mode: none

microstep mode: N = ddddddddd value of displacement
Digital
Unit: step, ½ step or microstep according to the resolution of the module
10 digits maximum
Limit |n| ? 2147483647
The direction of the movement is defined by the sign of the set point, if it is explicit: + or –
If the sign is not given, the direction of the movement is that of the last relative movement carried out.

Description: execution of a movement of n step or microstep in the direction explicitly provided in the command.

Displacement is defined in steps or half-steps for full step / ½ step modules, and in microsteps for microstep modules.

If there is no parameter: the movement is identical to the last relative movement carried out by the module.

If the parameter is a sign: the movement is carried out in the direction specified by the command with the same number of microstep than in the last relative movement carried out.

For full step / ½ step modules , displacement is defined by the commands DR, D+ or D -.

For modules microstep, the value can be replaced by a variable: GO # n.

The directions and instructions are saved at powering off, the module thus preserves the last parameters when powered on.

Error code:

0: - Parameter "n" is not a digital parameter

1: - The parameter is out of limit (> to 2^{31}-1)

(GP) $_{\mu}$: Definition of the position of commutation of the differed logical outputs and loading **(GP)** $_{\mu}$
Microstep modules only

Syntax: [@] GP [p]: [d] [M] [P] [A]

Parameters:

p = ± ddddddddd relative or absolute position of activation according to the type given in command GL

d = ± ddddddddd position of unloading (optional)
p and d: digital, unit: microstep

M: repetitive operation: modify the outputs at each passage to the position defined by p in the direction defined by the type given by command GL.

P: maintains operations and reloads the differed outputs command at each reset or powering.

A: stop of the differed logical outputs.

Description:

This command loads the differed logical outputs function by defining the relative or absolute motor position at which the modification must be carried out, and possibly the position of unloading (the direction of transition is specified in command GL, a type A or R must be defined beforehand). The parameter d must be given to specify pulse operation if it is desired. When no parameter is given, the function is reloaded with the last parameters used (p and possibly d) but the presence of the only parameter p stops pulse operation. If continuous operation is required, it must systematically be given in all GP commands.

Note: no equivalent command for the full step / ½ step modules.

*** GR.**: Motor power off *** GR.**

Syntax: [@]GR

Parameters: none.

Description: suppression of the motor power, the motor current is brought back to 0.

No the specific error code.

All the displacement commands (GO, GA, GF, GH) as well sequence start automatically cause the powering of the motor. It is possible to power off and on during a movement.

Caution!

- In the case of a SIMPA module indexer, microstep version, associated to a separated amplifier, this command activates simultaneously the boost and standby outputs. If those are correctly connected to the corresponding inputs of the MI or MIP modules, their simultaneous activation causes the powering off of the motor.

- In the case of other modules, it will probably be advisable to manage the motor power using a logical output. This is also valid when using a SIMPA indexer in full step / ½ step version .

*** GS**: Stop *** GS**

Syntax: [@]GS

Parameters: none.

Description: immediate stop of a movement or a sequence in progress.

The next movements will be carried out starting from Vmin.

Command GS finishes the sequence possibly in progress.

In full step / ½ step version , the logical outputs are positioned to "1" logic (FFh).

No the specific error code.

Note: the stop is carried out without deceleration, it can thus cause a slip or stall of the motor

MB: Authorization of the Limit Switch Mode, Polarity of the logical inputs **MB**

Syntax: [@] MB [L/H]

Parameters:

Full step / ½ step mode: none

Microstep mode: Optional coding of the polarity of the logical inputs
H: an excited input imposes logical state 1 (?)
L: an excited input imposes logical state 0

Description: This command authorizes the module to manage the Limit Switch Mode and can also define, for microstep modules, the polarity of the logical inputs.
If no parameter is given, the actual polarity of logical inputs is preserved.
An input is considered active when its logical state is 0.

In the limit switch operating mode, two logical inputs (E7 and E8) are used as a detection of limit switch.
The activation of the logical input E7 (or limit switch +) by a clockwise movement, causes the immediate stop of the movement or of the sequence currently carried out. This input has no effect on the movements in counter-clockwise direction. In the same way, the activation of the logical input E8 (or limit switch -) by a counter-clockwise movement will cause its immediate stop (movement or sequence).
The configuration of the Limit Switch Mode is saved during the powering off.

Error code:

0: The provided parameter is different from the single sign digit "H" or "L".

Note: The selection of the polarity can be done without implementing the Limit Switch Mode by the MN command.
The value of the coding of the logical inputs is by default: L (factory configuration)
The value returned by the reading commands is always the logical state.
The configurations Limit Switch Mode and Polarity can be read by command QL (µ)

MN: Inhibition of the Limit Switch Mode, Polarity of the logical inputs **MN**

Syntax: [@] MN [L/H]

Parameters:

Full step / ½ step mode: none

Microstep mode: Optional: coding of the polarity of the logical inputs
H: an excited input imposes logical state 1 (?)
L: an excited input imposes logical state 0

Description: This command inhibits the Limit Switch Mode (see command MB).
This inactive state of the Limit Switch Mode is the mode by default (factory configuration).
No logical input is assigned to a specific function.
An input is considered active when its logical state is 0.

Error code:

0: The provided parameter is different from the single character "H" or "L".

*an opto-isolated input is excited in a conducting state, a non isolated input is excited for a null tension.

*** MR..** : Master Reset of the module *** MR..**

Syntax: [@] MR. [m]

Parameters: full step / ½ step: none
microstep m = a: restore factory configuration
(only allowed value m = Z)

Description: this command is equivalent to a new powering of the module:

- The movement in progress is stopped,
- The absolute step counter is reset to 0,
- The outputs are forced to inactive state: "1" logic (FFh),
- If a Start Sequence is defined it is carried out

When the parameter m is given (value Z), the stored configuration is erased and replaced by the factory configuration (see detail "Status at powering").
Consequently, all the sequences are erased and the concept of Start Sequence does not exist any more.

(MS) µ: Management of the quiescent current and overcurrent **(ms) µ**
(Microstep modules only)

Syntax: [@] MS m

Parameters: m = a mode of management
allowed values N,S or B

Description: Definition of the mode of management of the quiescent current and overcurrent

* m = B all 3 state is authorized
- Quiescent current (standby) when the motor is off
- Overcurrent (Boost) in acceleration and deceleration phases
- Nominal in the movements

* m = S the mode of overcurrent is not used any more, the phases of acceleration and of deceleration are done with the rated current.

* m = N whatever the movement or not of the motor, the current in this last is maintained at its rated value.

Caution!
- All SIMPA modules are not ready to manage the overcurrent mode. In this case command MS B is equivalent to command MS S
- For indexer modules driving a separated amplifier, this command is fully effective only if connections between the Boost and Standby outputs of the indexer board are correctly connected to the corresponding inputs of the amplifier.

Error code:

Code 0: MS command does not have a parameter
The provided code is different from characters N, S or B

(* PD) µ: Decimal value of a variable **(* PD) µ**
(Microstep modules only)

Syntax: [@] PD # n: v

Parameters: N = dd: number of the variable to be modified 1 ? N ? 32
v = ± ddddddddd: value to be given to the variable
Digital
10 digits maximum
-214783647 ? v ? +2147483647

Description: gives the value v to variable # n (current value and init value)

Error code:

Code 0: Wrong number of variable
Non digital value

Code 1: $|v| > 2^{31} - 1$

(* PG) µ: Global variables definition, synchro **(* PG) µ**
(microstep modules only)

Syntax: [@] PG [v_1: [v_2: [.....: [v_{10}]....]]]

Parameters: vi = ± ddddddddddd: value to be given to variable # i
Digital
10 digits maximum
-2147483647 ? VI ? +2147483647

Description: gives to variables 1 to 32 values v_1, v_2 to v_{32} .If less than 32 values are given, only first variables are modified.
This command is also used as synchro for the phases of type NY. In that case, it does not necessary have a parameter.

Error code:

Code 0: Non digital value

Code 1: $|v_U| > 2^{31} - 1$

(* PH) µ: Hexadecimal value of a variable **(* pH) µ**
(Microstep modules only)

Syntax: [@] PH # n: v

Parameters: n = dd: number of the variable to modify 1 ? N ? 32
v = hhhh: hexadecimal value to give to the variable
Digital
8 digits maximum
0 ? v ? oFFFFFFFFh

Description: gives value v to variable # n (common value and init value)

Note: the variables are coded on 31 bits + sign. Thus, the value 10000001h corresponds to – 1 into decimal and oFFFFFFFFh corresponds to $-2^{31}+1$

Error code:

Code 0: Wrong number of variable
Non digital Value

Code 1: v ? 100000000h

(* QD) µ: movements and sequences monitoring **(* QD) µ**
(Microstep modules only)

Syntax: @QD

Parameters: none

Format of the answer:

@ED_Sequence_Phase_Sense_Type_Position_Inputs_Outputs_Control_Next Sequence_Rotary switch
@ED_ddd_ddd_±_aa_±ddddddddd_hh_hh_aa_ddd_dd

That is to say a maximum of 45 characters

- Only the useful digits are returned

Description: reading of all the control parameters of movement / sequence.

When executing an immediate movement or sequence, it is possible to monitor the operation by seeking certain parameters such as:

@	: Address of the questioned module
Sequence	: Number of the sequence in the course of execution (or of the last sequence carried out). The n° 0 announce that the module carries out an immediate movement.
Phase	: Number of the phase currently executed (or n° of the last phase carried out).
Direction	: Movement sense (or that of the last movement carried out).
Nature	: Nature of the movement or the phase
Position	: Value of the absolute step counters.
Inputs	: State of the inputs, in the event defects on the logical inputs, the returned value is XX (cf III.8.4).
Outputs	: State of the outputs.
Control	: Lx: local or direct mode xF: power off Sx: sequence mode xO: power on Ux: sub-sequence mode
Chained sequence	: N ? of the next sequence (valid if a sequence is in progress). (0 for none)
Rotary switch (0 for none)	: N ? of sequence (? 99) read on the Sequence Selector of the interface.

Careful:
This command must be imperatively addressed to a single module.

Precision on the reading of the absolute step counter (ASC)

During the reading of the absolute step counter by command QD, the ASC can be modified by the normal course of the steps (particularly at high speed), which can introduce aberrant results. To avoid this, the module carries out up to 10 reading tests. If despite everything, it does not obtain reading without interruption, due to the movement, the value returned by the module will be preceded by the symbol "!" in order to specify that the given value could be wrong.

This phenomenon can occur whatever the nature of the phase in progress.

Note: Equivalent command for the full step / ½ step modules: RD.

(* QG) µ: Sequences and phases monitoring **(* QG) µ**
(Microstep modules only)

Syntax: @QG

Parameters: none

Format of the answer:

@EG_Sequence_Phase_Sense_Control

@EG_ddd_ddd_±_aa

That is to say a maximum of 17 characters, only the significant digits are emitted.

Description: reading of the N ? of sequence and phase in progress, as well as control information.

Sequence: n° of sequence in progress (or n° of the last sequence carried out). In the event of direct movement, the returned value is: 0

Phase: n° of phase in progress (or n° of the last phase carried out)
N ?254 is given for the stop phase
N ?255 is given for the interruption phase

Sense: movement sense (or sense of the last movement carried out)

Control: Two characters specifying:

1) Local mode: **L** or sequence mode: **S** or sequence mode: **U**

2) Power off: **F** or Power on the motor: **O**

Note: This command does not have an equivalent in full step / ½ step version .

(* QL) µ: Local parameters reading **(* QL) µ**
(Microstep modules only)

Syntax: @QL

Parameters: none

Format of the answer:

@EL WL:Vmin WH:Vmax WT:ta[:td] WN:resolution DR:± set point GI:current DG:stall MD:
Limit Switch Mode Polarity

@EL_WL:ddddd_WH:ddddd_WT:ddddd_[:ddddd]_WN:ddd
DR:±ddddddddddd_GI:ddd_DG:ddd_MD:da_aa_a

Description:

This command enables to the user to seek the following parameters:

@	: address of questioned module	
Vmin	: minimum speed (parameter of command WL) *	
Vmax	: maximum speed (parameter of command WH) *	
ta	: duration of acceleration	
td	: duration of deceleration (if different of ta)	
resolution	: programmed resolution (parameter of command WN) *	
set point	: relative movement set point, memorized during command GO ±n	
current	: set point for motor current (parameter of command GI)	
slip	: value of the tolerated maximum slip (parameter of command DG)	
mode	: 0: Microstep mode 1: Full step mode 2: Half-step mode	B: management of overrcurrent and standby N: forcing of the rated current S: management of the standby
Limit Switch	: MN or MB	
Polarity	: L or H	

Constraint:

This command must be imperatively addressed to a single module in a muti-axis configuration.

* Note:

Reading the parameters speed can give different values from those programmed.
The returned values are those really generated by the module taking into account the quantification due to the frequency generator and the resolution of the calculation of law of acceleration.

Note: Equivalent command for full step / ½ step modules: RL.

(* QN) µ: variable read **(* QN) µ**

(Microstep modules only)

Syntax: @QN # n

Parameters: n = dd number of the variable to read (1 ? N ? 32)

Format of the answer:

@EN _ # n: v [,(V)]

@EN _ # dd = ±ddddddddddd, (±ddddddddddd)

That is to say a maximum of 33 characters.

Description: reading of a variable

v = common value of the variable
V = value of init of the variable (if different from v)

Error code:

0: wrong n° of variable, absence of number of variable…

Note:

The addition of the suffix `-H' to the command allows reading again into hexadecimal format the variable, in this case the format of the answer is:
@EN_dd: Hhhhhhhhh, (Hhhhhhhh)
with hhhhhhhh: hexadecimal coding

(* QP) µ: Reading of position **(* QP) µ**
(Microstep modules only)

Syntax: @QP

Parameters: none

Format of the answer:

@EP Absolute_position Logical_inputs Logical_outputs

@EP_±ddddddddddd_hh_hh @EP!±ddddddddddd_hh_hh

That is to say a maximum of 22 characters (only the significant digits are emitted).

For high speeds motors it is possible that the software cannot read a stable position of the counter of steps. The returned value can then be aberrant. Character "!" allows to indicate this possible risk (cf QD commands).

Note: Equivalent command for full step / ½ step modules: RP.

(* QS) µ: Reading of phase **(* QS) µ**
(Microstep modules only)

Syntax: @QS_Sequence_Phase

Parameters:

Sequence and phase: ddd
Digital Unsigned
3 digits maximum
Limits 1 ? Phase ? 200
Limits 1 ? Sequence ? 100

Sequence: number of the sequence containing the phase to be read

Phase: number of the phase in the sequence

Format of the answer: (non significant zeros on the left, are omitted)

@ES_Sequence_Phase_Type_ set point _NS_Next_Phase _NO_Logical_Output
_NL_ Next_Sequence
_NQ_Sub-séquence
_NE_EL1_EL2_EL3_EL4_EL5_EL6_EL7_EL8
_NF_EL1_EL2_EL3_EL4_EL5_EL6_EL7_EL8

@ES_ddd_ddd_aa_±ddddddddddd_NS_ddd_NO_hh: hh
_NL_ddd
_NQ_ddd
_NE_ddd_ddd_ddd_ddd_ddd_ddd_ddd_ddd
_NF_ddd_ddd_ddd_ddd_ddd_ddd_ddd_ddd

Datas NE/NF NL/NQ and NO are returned in this order only if they were defined in the phase.
Datas NL and NQ as well as NF and NE are exclusive.

Error code:

0: Defect on the parameters of the command:
- There is not the exact number of parameters (2)
- The parameters are not digital or are signed *

1: N° of sequence out of limits: = 0 or > 100
2: N° of phase out of limits: = 0 or > 200
3: Non-existent sequence
4: Non-existent phase in the sequence (n° of the phase > ns of the SN command)

* Only the sign ' - ' causes a signed parameter erro.

Note:
- Reading a non defined phase gives:
@ES Sequence Phase XX NS_0

Note: Equivalent command for full step / ½ step modules: RS.

(* QV) µ: Reading of the version and index number of the software **(* QV) µ**
(Microstep modules only)

Syntax: @QV

Parameters: none

Format of the answer:

@EV VR codes

@EV_hh_ddddd

V: version number
R: index number
Code: information reserved for internal use to Midi Engineering

Note: Equivalent command for full step / ½ step modules: RV.

(* QX) µ: Interruptions Status / Acknowledgement query **(* QX) µ**
(Microstep modules only)

Syntax: @QX

Parameters: none

Description: when the module detects an anomaly on a command, it positions an error code.

This code is read by the QX command. The addressed module answers the following message:

@EE_Code_status

@ : module address on 2 digits
EE : header of the answer
Code_status: Status code of the module

Note:

The fact of reading this code by command QX causes a reset of the error code, a return at the nominal state of the module: Code_status = N

In the event of permanent anomaly (example: defect on the power supply: W), the error code can persist as long as the origin of the defect detected was not removed.

Example

00QX… 00EEN

00GI -10 ; wrong command because of signed value

00QX… 00EE_0; return the error code 0 and erases it

00QX… 00EE_N; `no error'

Constraint

This command must imperatively be addressed to a single module

Note: Equivalent command for the full step / ½ step modules: RX.

(* RD) p: Movements and sequences monitoring **(* RD) p**
(Full step / ½ step modules only)

Syntax: @RD

Parameters: none

Format of the answer:

@ED_Sequence_Phase_Sense_Type_Position_Inputs_Outputs_Control_Next sequence _Rotary switch

@ED_dd_dd_± aa_±ddddddd_hh_hh_aa_ddd_dd

39 characters are systematically returned.

Description: reading of all control parameters of movement/sequence.

During the execution of immediate movement or sequence, it is possible to control operation by seeking some parameters such as:

@ : Address of questioned module
Sequence : Number of the sequence in progress (or of the last sequence carried out).
n° 0 announce that the module carries out an immediate movement
Phase : Number of the phase in progress (or n° of the last phase carried out).
Direction : Movement sense (or sense of the last movement carried out).
Type : Type of the movement or the phase
Position : Value of the absolute step counters.
Inputs : Status of the inputs, in the event of detection of defects on the logical inputs, the returned value is XX (cf III.8.4).
Outputs : Status of the outputs.
Control : Lx:local or direct mode xF: power off
Sx: sequence mode xO: power on
Ux: sub-sequence mode
Chained sequence : n° of the next sequence (valid if sequence in progress).
(0 for None)
Rotary switch : n° of sequence (? 99) read on the Sequence Selector of thee interface.
(0 for None)

Constraint
This command must be imperatively addressed to a single module.

Precision on the reading of the absolute step counter (ASC)

During the reading of the absolute step counter by a RD command, this one can be modified by the normal progress of the steps (particularly at high speed), which can introduce aberrant results. To avoid this, the module carries out up to 10 tests of reading. Nevertheless, if it cannot read without interruption, due to the movement, the value returned by the module will be preceded by the symbol "!" in order to specify that the given value is doubtful.

This phenomenon can occur whatever the type of the phase in progress.

Note: Equivalent command for microstep modules: QD.

(* RL) p: Reading of local parameters **(* RL) p**
(Full step / ½ step modules only)

Syntax: @RL

Parameters: none

Format of the answer:

@EL Vmin Vmax Tr set point Current Mode gL

@EL_ddddd_ddddd_ddddd_±dddddd_ddd_d_ddd

That is to say a total of 41 characters.

Description: reading of local parameters full step ½ step

This command enables the user to seek the following parameters:

@:		Address of questioned module
Vmin	:	Minimum speed (parameter of command WL) *
Vmax	:	Set point Speed (parameter of command WH) *
Tr	:	Slope Time (parameter of command WT)
Direction	:	Relative movement sense(positioned by D or D+)
Set point:		Set point for relative movement (parameter of the command DR.)
Current :		Motor current set point (possibility of torque programming)
Mode	:	information on step or ½ step operation (2: ½ step, 1: full step, 0: microstep)
gL:		value of the maximum tolerated slip fixed by command DG (by default 10)

Constraint
This command must be imperatively addressed to a single module in a multiaxis configuration.

* Note

Reading again the speed parameters can give values different from those programmed. That is due to the resolution of calculation in the treatment carried out by the module.

Note: Equivalent command for microstep modules: QL.

(* RP) p: Reading of position **(* RP) p**
(Full step / ½ step modules only)

Syntax: @RP

Parameters: none

Format of the answer:

@EP Absolute_Position_ Logical_Inputs Logical_Outputs

@EP_±ddddddd_hh_hh @EP!±ddddddd_hh_hh

That is to say a maximum total of 19 characters

Description: reading of position inputs/outputs

For high motor speeds, it is possible that the software cannot read a stable position of the steps counter. The returned value can then be aberrant. Character "! " indicating this possible risk (cf command RD).

Note: Equivalent command for microstep modules: QP.

(* RS) p: Reading of phase **(* RS) p**
(Full step / ½ step modules only)

Syntax: @RS_ns_np

Parameters:

nS = dd number of the sequence containing the phase.
np = dd number of the phase in the sequence.

Description: reading of a phase.

This command enables the user to read a phase previously programmed.

The parameters returned by the module correspond exactly to the command Sp (definition of the phase).

Format of the answer:

		NE		NQxx
@ES_Ns_Np_Se_Na_set	point_NF_EI..	E8_NS_ps_NO_	xx_NLxx	

All those parameters have the same format and meaning than for a SP command.

Notice

If the phase has not been not initialized by a SP command, the answer is: @ES_X

When a parameter has not been defined by SP command it is positioned to XX in the answer to the RS command. Thus a return with RS… NOXX specifies that the parameter was not defined by SP.

Constraint

This command must be imperatively addressed to a single module in a multiaxis configuration.

This command is accepted by the module only when it is out of a sequence.

Error code positioned

0: Defect on the parameters of the command.
There is not the exact number of parameters (2).
The parameters are not numerical, or are signed.

1: n° of Sequence provided is out of limit: = 0 or > 99

2: The number of phases is out of limit: = 0 or > 50

3: The specified sequence does not exist.

Caution!

The control of the sign is carried out only on the first parameter of the command.

Note: Equivalent command for microstep modules: QS.

(* RV) p: Reading of the version and index number of the software **(* RV) p**
(Full step / ½ step modules only)

Syntax: @RV

Parameters: none

Format of the answer:

@EV VR code

@EV_hh_ddddd

V: version number
R: index number
Code: information reserved for Midi Engineering internal use

Note: Equivalent command for microstep modules: QV.

(* RX) p: Interruptions Status/Acknowledgment query **(* RX) p**
(Full step / ½ step modules only)

Syntax: @RX

Parameters: none

Description: when the module detects an anomaly in a command, it positions an error code.

This code is read by means of the RX command. The addressed module answers the following message:

@EE_Code_status

@ : module address on 2 digits
EE : header of the answer
Code_status: Status code of the module

Note:

The fact of reading this code by a RX command causes a reset of the error code, a return at the nominal status of the module: Code_status = N

In the event of permanent anomaly (example: defect on the power supply: W), the error code can persist as long as the origin of the detected defect was not removed.

Example

00RX… 00EEN

GI -10 ; wrong command because of signed value

00RX… 00EE_0; return the error code 0 and erases it

00RX… 00EE_N; ` no error'

Constraint

This command must imperatively be addressed to a single module

Note: Equivalent command for microstep modules: QX.

SD: Starting sequence **SD**

Syntax: [@]SD_ns

Parameters: ns = ddd: n° sequence to be started at initialization (powering or MR command.)

Digital,
Unsigned,

Limit of the parameter:	1 ? N ? of sequence ? 99	full step / ½ step version
	1 ? N ? of sequence ? 100	microstep version

Description: selection of the sequence to be automatically carried out at next powering or initialization (command: MR.).

The module can automatically carry out (without action of the user) a sequence at powering.

The choice of the Start Sequence is done by command SD.

Parameter "n° of sequence" must correspond to a known sequence of the module.

Example: so that the module carries out sequence n° 11at each powering, execute command: SD_11.

Error code positioned

The module positions the following codes:

0: Defect on the parameters.
Incorrect format.
Number of parameter is incorrect.

1: n° of sequence provided is out of limit.

3: n° of sequence does not correspond to a known sequence of the module.

Constraint

The sequence must be created before being selected.

SE: Sequence erasing **SE**

Syntax: [@]SE_ns

Parameters: ns = ddd: N ? sequence to be erased

Digital,
Unsigned,

Limit of the parameter:	1 ? N ? of sequence ? 99	full step / ½ step version
	1 ? N ? of sequence ? 100	microstep version

Description: erasing of the sequence which number is parameter "N ? sequence ".

All the memory capacity previously reserved to describe the phases of this sequence becomes available for the definition of other sequences.
Parameter "N ? sequence" must correspond to an already existing number of sequence.

Note: if parameter "N ? sequence " is 0, all sequences of the module are erased.

Example: Erasing of sequence 12: SE_12.
Erasing of all sequences: SE_0.

Error code positioned

0: Defect on the parameter.
Not digital parameter.
The number of parameter of the command is not respected.

1: Parameter is out of limit (> 200 or 99)

Note: The erasing of a non-existent sequence does not cause an error!

SN: Creation of a sequence **SN**

Syntax: [@]SN_ns_np

Parameters: ns = ddd: number of the sequence to be created
np = ddd: number of phases reserved for this sequence.

Numerical,
Unsigned,
3 digits maximum,
Limits: 1 ? n° of sequence ? 99 } full step / ½ step version
1 ? Phase number ? 50 }

1 ? n° of sequence ? 100 } microstep version
1 ? Phase number ? 200 }

Description: this command makes it possible to reserve the place necessary to the definition of a sequence in the memory of the module.

- The parameter Sequence Number makes it possible to identify the sequence to be created. Any later operation on this sequence will refer to this Sequence Number.

- The parameter Number of Phases enables the module to reserve only the necessary memory space to the sequence.

Example: Creation of sequence N ? 2 containing up to 30 phases: SN_2_30.

Error code positioned

0: Defect on the parameters of the command.
There is not the exact number of parameters (2).
The parameters are not digital, or are signed.

1: N ? of Sequence provided is out of limit

2: The number of phases is out of limit

4: The number of sequence provided in the command is already affected with a sequence of the module.

5: There are not enough available phases to create this sequence

This error can occursif one creates sequences having a significant number of phases.

Currently the total number of phases is limited to ***414 in full step / ½ step***
384 in microstep

Constraint

It is impossible to increase the number of phases of an already existing sequence.
It is necessary to erase it and recreate it with the good number of phases.
One cannot create an already existing sequence (error code 4).

Caution!!!
The control of the sign is carried out only on the first parameter of the command.

SP: Definition of a phase **SP**

Syntax:
Full step / ½ step version :
[@]SP_ns_np_Ss_Na_Cns[_NL_sc or _ NQ_sq][_Ne_X1 _… X8][_NS_ps][_NO_out])

Microstep version
[@]SP_ns_np_Na_Cns[_NL_sc_NQ_sq][_Ne_X1 _… X8][_NS_ps] NS@1[:@2][_NO_out][:msq]]

Description: definition of all descriptive parameters of a phase.

Parameters:

a) N° of sequence
* ns = ddd: number of the sequence to which the phase belongs

b) N° of phase
* np = ddd: number of the phase in the sequence

c) Sign full step / ½ step version only:

* Ss = S+ or S-: movement sense

This information is then obligatory, it specifies the direction of the movement of the motor.

- S+: clockwise (direction +)
- S-: counter-clockwise direction (direction -)

Note:

S+ or S- information is not always used by the module, in particular for special phases ' return to origin' and ' absolute movement': the module does not take into account this information, but defines the direction from the movement according to the current position of the counter and the set point of the movement.

d) Nature of the phase
* Na = aa: nature of the phase

Allows to specify the type of phase to be carried out:

NP:	Relative movement phase	
NH:	return to Origin phase	
NX:	Absolute movement phase	
NV:	constant speed phase	
NA:	acceleration phase	
ND:	deceleration phase	
NW:	waiting phase	
NZ:	reset of the absolute step counter phase	
NG:	slip detection phase	
NC:	plateau speed modification phase	
NT:	activation of Motor Power phase	
NU:	suppression of Motor Power phase	
NY:	synchronization	only for microstep modules
PV:	modification of a variable	
PC:	transfer of the CPA in a variable	
PT:	test of a variable	
PR:	init value transfer to common value	
PI:	common value transfer to init value	
PA:	addition on variables	

- NP, NX: Relative and absolute movements. The set point respectively sets the number of steps to be carried out or the new absolute position to reach.
As for the immediate movements, those are divided into 3 stages: acceleration, stage at constant speed, deceleration.

- NH: Movement from current position to the origin. As for NP and NX, the movement is also divided into three stages: acceleration, stage at constant speed, deceleration.

- NV: Elementary movement. The instruction sets the number of steps to be done during the phase.
The speed of the phase is Vmin, except when it is chained with phases NA or ND, in this case the speed corresponds to the speed at end of the preceding phase.

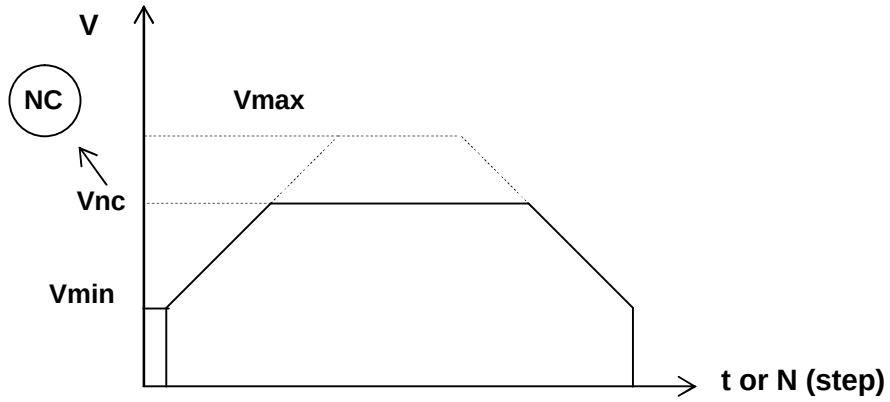
- NA, ND: Accelerated or decelerated relative movements. The instruction fixes the number of steps (or microsteps to be realized) while accelerating or decelerating according to the law speed in memory in the module. The variation speed is carried out starting from the final speed of the preceding phase. Speed remains limited in every cases to speeds defined by WL and WH.
When a speed limit is reached, the motion goes on at constant speed until the total realization of the number of microstep defined by the set point.
If a logical input detected on edge is activated, the motor speed is maintained constant during a few additional steps (microsteps) until total loading of the following phase, then the execution of this one is run. Furthermore, if the loading of the following phase is not finished before the number of microsteps defined by the set point is done, the motor movement is maintained at constant speed until the end of the loading.
Moreover, at high speed, the counting of microsteps is carried out by bursts. Thus, the number of microsteps really carried out could be slightly higher taking into account the resolution of the " bursts ". A block can never exceed a motor step, so if the instructions of the phases NA, NV, ND correspond to number of full steps (multiple of the programmed resolution) no positioning error is added by this management "by bursts ", in the contrary case additional microsteps are generated.

- NW: Waiting Phase allowing to generate a delay in millisecond.
This phase does not cause any motor movement but the logical inputs (NE/NF) continue to be managed, thus allowing to wait for the activation of a particular input.

- NZ: Reset of the absolute step counter, the current position becomes the position of origin of next absolute displacements.
For microstep versions, a set point different from zero can be given, then forcing a waiting period of the given value expressed in ms (1 ms by defect).

- NG: Comparison between the value of the absolute step counter and the set point. If the difference is higher than the threshold fixed by the DG command, the sequence is diverted on the phase specified by the NS subcommand (obligatory). Otherwise, connection on the naturally following phase.
This phase does not cause any motor displacement. There is no possible management of logical inputs.
For the microstep versions, the number of the phase to be connected can be different from the next natural phase by specifying a second address @ $_2$ in the NS command: NS @ [: @ $_2$]

- NC: Makes it possible to carry out the next "point to point" movements (in the sequence or sub-sequence) at limited speed by modification the plateau speed (V_{max}) while preserving the initial law of acceleration.



The new speed set point Vnc makes it possible to truncate temporarily the initial law of acceleration.

A set point = 0 makes it possible to restore the initial value V_{max} for next movements.
The initial speed V_{max} is automatically restored at the end of the sequence.

A Vnc value lower than Vmin imposes an operation only at minimum speed.
A Vnc value higher than Vmax restores operation with the initial law of acceleration.

The set point Vnc is preserved when calling of a sub-sequence and at the return to the called sequence.

- NT, NU: Switch on or off of the motor power. The motor power is implicitly established at the beginning of a sequence as with commands GA, GH, GM, GO. It is removed by commands GR. and MR.

  For the microstep versions, a set point different from 0 can be given then forcing the module to wait for a given value expressed in ms (by default 1 ms).

Phases natures that follow are specific to microstep version and do not authorize the management of logical inputs, sub-sequences and chained sequences.

- (NY) µ: Wait for. Serial link Synchro (only for microstep modules). This phase behaves as a NW waiting phase. It makes it possible to wait for the reception of a PG command, with or without parameter, before continuing the sequence. The value given as parameter makes it possible to limit the waiting and to jump to the phase defined by NS if the allocated time (in ms) is spent.

  Example: NY time out NS @ $_1$. [: @ $_2$.]

  At reception of command PG, the sequence continues with the naturally following phase (n°phase +1) or with the phase which address is @ $_2$ if it is specified. If command PG is not received before the expiry of the time time-out expressed in ms, the sequence is redirected on the phase which address is @ $_1$.

  Note: the test of the logical inputs is not taken into account in this type of phase.

- (PV) μ: Modification of variable (current value)

Form n°1: PV # n:v v = ± ddddddddd
The current value of variable n changes to v

Form n°2: PV # n:# m
The current value of variable n changes to value of the variable m

- (Pi) μ: Modification of the initialization value of a variable Pi # n
The 'current' value of variable n will be used with MR or Reset to initialize variable n.

- (PR) μ: Reassignment of the value of a variable PR # n
The initialization value of variable n becomes the current value.

- (PC) μ: Recopies absolute step counter in a variable PC # n
The instantaneous value of the absolute step counter is recopied in variable n

- (Pa) μ: Addition on variable (current value)

Form n°1: Pa # n: v v = ± ddddddddd
Value v is added to the current value of variable n: # n = # n + v

Form n°2: Pa # n: # m
The current value of the variable m is added to the current value of variable N: # n = # n + # m

- (Pt) μ: Test on variable

Pt makes it possible to condition the course of the sequence to the current value of a variable or to manage a " loop counter ".
In every case, it is advisable to associate PT to the directive NS in its wide form: NS @1 [: @2 [: @3]] which makes it possible to describe up to 3 numbers of following phase.

Form n°1: PT # n: v NS @1 [: @2 [: @3]] v = ± ddddddddd
Compare the current value of variable N with value v and selects consequently the following phase to carry out.

@1	if	# n = v (equality)
@2	if	# n < v (inferiority)
@3	if	# n > v (superiority)

Note: if the addresses @2 and @3 are not specified by the directive NS they are replaced by the number of the naturalyl following phase (phase +1).

Variable n is not modified

Form n°2: PT # n: # m NS @1 [: @2 [: @3]]
Compare the current value of variable n with the current value of the variable m and selects consequently the next phase to carry out.

@1	if	# n = # m	(equality)
@2	if	# n < # m	(inferiority)
@3	if	# n > # m	(superiority)

Note: if the addresses @2 and @3 are not specified by the directive NS they are replaced by the number of the naturally following phase (phase +1).

Variables n and m are not modified.

Form n°3: (counter management) Pt # n NS @1 [: @2]
The current value of variable N is decreased of 1: (# n = # n – 1)
It is then compared with value 0 and the following phase is selected consequently.

@1	if	# n = 0	(# n after decreasing)
@2	if	# n ? 0	

Note: the absence of the address @2 in the directive NS will involve the selection of the naturally following phase (phase +1)

! Variable # n is modified

e) Set point

* Cns = [±]ddddddddd: set point

This set point can be:

- A number of relative steps in the phases:

NV = constant speed on Cns steps, ½ steps or microsteps (according to the resolution of the module)
NP = movement of Cns steps, ½ steps or microsteps (according to the resolution of the module)

In full step / ½ step version , displacement must be between 1 and 999999.
The direction of the movement is defined by the S+/S- command

In microstep version, the direction of the movement is determined by the sign of the instruction its value is limited

-2147483647 ? Cns ? 2147483647

- An absolute position

The parameter represents the position that the motor must reach for NX phases or a reference position for NG phases (in step, ½ step or microstep, according to the resolution of the module).

Numerical format signed or not -8388606 ? Cns ? 8388605 full step / ½ step version
-2147483647 ? Cns ? 2147483647 microstep version

- A delay

For NW phase (wait) and for NZ, NU, NT, NY phases in microstep version; the time is given in ms (1 to 65535 ms).

- A speed

In step/s (or ½ step/s) for NC phases

- Forced to 000, for full step / ½ step version

In the NU, NT, NZ and NH phases.

NU, NT and NZ phases do not carry out any movement. For phase NH the module seeks the number of steps and the direction of the movement before carrying it out.

- Variable set point:

In microstep version, the set point can be given by the value of a variable by giving its number n preceded by #: # n.

Caution! A set point is obligatory for full step / ½ step version

f) Chained Sequence (Directive NL)
* Sc = ddd: number of the chained sequence (optional)

This information specifies the number of the new sequence to be carried out as soon as the sequence in progress is finished.
By default, the number of the sequence to be chained previously defined is not modified until specified by the phase in progress, consequently there is no chaining of sequence until the module meets this data in the description of a phase which it carries out.
To modify or define a number of sequence to be chained, the phase containing command NL must obligatorily be carried out (not only defined).

g) Sub-sequence (Directive NQ)
* sq = ddd: number of the sub-sequence (optional)

When the execution of the phase in progress is finished, the module stops the motor movement, seeks and loads the sub-sequence sq, entirely carries it out and continues the execution of the initial sequence starting from the natural phase according to the phase having caused the connection with the sub-sequence.

A sub-sequence differs from a sequence only by the fact:

- That it cannot call other sub-sequences (only one level of call),
- That it cannot connect sequence (NL).

The parameters NE/NF and NS are not taken into account during the execution of a phase of call of sub-sequence (no conditional call).

Directives NL and NQ are exclusive.

h) Test on logical inputs (Directives NE and NF)

*Ne: aa_ddd_ddd_ddd_ddd_ddd_ddd_ddd_ddd: selection of conditional inputs

(optional)

NE X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8 : management on status
or NF X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8: management in the course of the phase

By default, there is no management of logical inputs

The NE code implies the taking into account of logical inputs at the end of the phase only.
Only the state (active/inactive) of the inputs at the end of the phase counts, any variation (active < - - > inactive --> active) during the course of the phase is ignored.

NF code implies an immediate processing of the logical inputs at activation of an input in the course of a phase or at the beginning of a phase if the input is already active.

The Parameters X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8 specify the eight numbers of conditional phases attached to the eight logical inputs, respectively E1, E2, E3, E4, E5, E6, E7, and E8.
The user must specify all eight Xi parameters when calling NE or NF.

Xi specifies the number of the phase to be carried out if the corresponding input is active. If Xi is 0, the associated input is not taken into account.

For the microstep versions, it is possible to choose individually the active state of each input by writing the sign – before the address of the jump associated with an input. Logical state 1 then becomes active instead of the nominal activity on logical state 0.
This modification of the active logical level of the input is valid only for the phase where it is specified and it combines with the total polarity definite by commands MB and MN.

In case of simultaneous activation of several inputs, the scanning of the inputs is carried out in the ascending order.

Example

1 * Switching sequence 1 to phase n° 7 when logical input N ? 3 is active during the execution of phase 2:

(E1)(E2)(E3)(E4)(E5)(E6)(E7)(E8)

SP_1_2… NF_00__00__07__00__00__00__00__00

Value 00 indicates that the corresponding logical input is not taken into account.

2 * Switching sequence 3 to phase n° 5 if logical input n° 5 is present at the end of phase 4.

SP_3_4… NE_00_00_00_00_05_00_00_00_00.....

i) Next phase

* ps = dd: n° next phase	(optional)

Allows to force the order of succession of the phases by imposing next phase number, different from chronological order.
This information is optional: by default the next natural phase is the number of the phase in progress + 1.

Example: To cause a jump from phase n° 3 to phase n ? 5.

SP_1_3..... NS_05

Notice 1

By combining codes NE/NF and NS, it is possible to repeat a phase until a logical condition is active.

SP_1_2… NE_03_00_00_00_00_00_00_00_NS_02.

This phase loops to itself (NS02) as long as logical input E1 is inactive (at the end of the phase).

It passes then to phase N ? 3.

Notice 2

It is possible to control in the same phase several logical inputs, but if those are positioned simultaneously, the module will only take into account the input whose number is the lowest.

SP_1_2… NE_00_00_05_00_17_00_07_00_NS_02

If inputs 5 and 7 are active at the end of phase 2, the sequence will continue in phase 17. With a NF type command, the jump depends on the real " simultaneity" of the activation of two inputs. The user must have a good control of the timing of his/her inputs when he/she uses NF command if he/she does not want to be confronted with risks sometimes not easily controllable.

J) Logical outputs (NO Directives)

* out = hh: state to be placed to logical outputs msq = hh: modifiable outputs mask	} (optional)

This last part of the command makes it possible to configure the state of logical outputs of the module during the phase.
By default the logical outputs remain in the state given by the phase previously carried out.
Outputs takes the value $S ? OUT ? msq ? S_{?1} msq$ where S_{-1} represents the preliminary state of the outputs.
If the mask is not specified, all 8 bits of output are modified.
The state and the mask can be specified by means of only one variable.
The least significant byte gives the output value OUT, the most significant byte goves the value of the mask if different from 0.

n = H H H H

mask ⟵ ⟶ OUT

Caution! Neither the concept of mask, nor the concept of variable is usable in full step / ½ step modules.

Example

NO_A5: place logical inputs in the state:

1010 0101 = (A5H)
S8..... S1

Restrictions: ***if this information exists in the phase definition command, it must appear obligatorily at the end of SP command.***

Full step / ½ step boards do not recognize the concept of mask, the use of a mask causes a syntax error.

k) Error codes

The module carries out several controls on the coherence and on the syntax of command SP and positions the following codes:

0: defect on the parameters of the phase:
the parameters are not digital.

1: parameter n° of sequence incorrect.
The specified number does not correspond to an existing sequence or the number of sequence is out of limit

2: parameter n? of phase incorrect.
The number of the phase is higher than the number of phases reserved for the sequence (defined by the SN command).

4: incorrect logical inputs parameter.
The eight numbers of conditional phases are not all digital.

Caution!!!

The control of the sign is carried out only on the first parameter of the command.

SR: Suppression of the starting sequence **SR**

Syntax: [@]SR

Parameters: none

Description: suppression of the automatic starting of sequence selected by command SD.

This command erases only the information that made the module to find the sequence to be carried out at the powering.

The sequence itself is not erased.

No parameter.

Error code:

0: Presence of a parameter

SS: Running of a sequence **SS**

Syntax: [@]SS_ns

Parameters: NS = ddd: n° of sequence to run

Digital,
Unsigned,
Limit of the parameter: 1 ? n° of sequence ? 99 full step / ½ step version
1 ? n° of sequence ? 100 microstep version

Description: immediate execution of the requested sequence.

If it is the first execution of the sequence after modification of the speed parameters, the module finishes, if necessary, the calculation of the law of acceleration then carries out the sequence. Parameter n° of Sequence must correspond to a known sequence of the module.

Example: To carry out sequence 15 directly:
SS_15.

Error code
The module positions the following codes:
0: Defect on the parameter
Incorrect format
Numbers incorrect parameter

1: Parameter n° of sequence is out of limit

3: n° of sequence provided does not correspond to a known sequence of the module.

WH: speed set point **WH**

Syntax: [@]WH_w

Parameters: W = ddddd: speed in step/s or ½ step/s

Digital,
Unsigned,
5 digits maximum,
Absolute limits of the parameter: 1 < W ? 20 000

Description: programming of the speed set point for all motor movements even after reset.

This command gives to the Vmax parameter the value W.

Vmax corresponds to the maximum speed that the motor will be able to reach, it does not depend on the selected resolution in microsteps per step.

This value W is given in step/s for the full step or microstep modes, in ½ step/s for ½ step mode.

The absolute limit of this parameter are 1 step/s and 20000 step/s.

Example: the speed set point is 13500 step/s

WH 13500

Error code

The module sets the following error codes:

0: Command WH does not have a parameter.
The provided parameter does not correspond to a digital value.
The provided parameter is signed.

1: Coherence between parameters V_{min}, V_{max}, T_R and µ is not respected.
(cf § III.4.6.)

Note:

The parameter is memorized not in the frequency format (step/s) but in period format in 0,5 ?S ($2 * 10^6$ / frequency).

A Frequency_Period conversion is thus ensured before updating the parameter.

Reading the parameter by commands QL or RL carries out the opposite operation (Period_Frequency conversion).

Because of the calculation round-offs (truncation), the result of the reading can be different from the programming but corresponds to the frequency really generated.

WL: start speed **WL**

Syntax: [@]WL_w

Parameters: W = ddddd: start speed in step/s or ½ step/s

Digital,
Unsigned
5 digits maximum
Absolute limits of the parameter:1 ? W < 20 000

Description: programming of the start speed for all next motor movements, even after reset.

This command makes it possible to initialize the Vmin parameter with the value of W.

It does not depend on the selected resolution in microstep.

This value W is given in step/s for the full step or microstep modes, in ½ step/s for the ½ step mode.

Vmin also corresponds to the minimum speed of the motor.

Example: the start speed is 250 step/s: WL 250

Error code positioned

The module positions the following error codes:

0: Command WL does not have a parameter.
The provided parameter does not correspond to a digital value.
The provided parameter is signed.

1: Coherence between the parameters V_{min}, V_{max}, T_R and µ is not respected.
(cf § III.4.6.)

(WN) µ: Resolution in microstep **(WN) µ**
(Microstep modules only)

Syntax: [@]WN _ ?

Parameters: ? = ddd: resolution in microsteps per step

Digital,
Unsigned,
3 digits maxima
Limits: 1 ??? 256 for the microstep modules without integrate power
(SIMPA microstep or SIMPA microstep Front Panel…)
µ: 1, 2, 4, 8, 16, 32 or 64 for the microstep modules with integrated power
(SIMPA microstep 1 axis, SIMPA microstep 1 axis Front Panel, SIMPA microstep 4 wires or SIMPA microstep 4 wires Front Panel…)

Description: definition of the resolution of the driver in microsteps per step. Maximum and minimum speeds, as well as slope time of are not modified.

One can thus modify the resolution without modifying the final speed of the motor.

However the generation speed of the microstep is multiplied by ?.

Consequently, taking into account the quantification related to the resolution of the generator, speeds really generated (returned by command QL) can slightly evolve according to the selected resolution.

Example: the resolution of the driver is 32 microsteps per step: WN 32

Error code positioned

0: Command WN does not have a parameter.
The provided parameter does not correspond to a digital value.
The provided parameter is signed.

1: The value given of the parameter is out of limit:

Coherence between the parameters V_{min}, V_{max}, T_R and µ is not respected (cf § III.4.6.)

This command does not exist in full step / ½ step version

Note:
- In the case of a couple microstep indexer / amplifier, it is necessary that the resolution programmed by WN corresponds to the resolution selected (rotary switches, jumpers…) on the amplifier.

WT: slope time **WT**

Syntax [@]WT_ta [:td]

Parameters: ta = ddddd: duration of the slope of acceleration in ms.
td = ddddd: duration of the slope of deceleration (only for microstep version)
Digital,
Unsigned,
5 digits maximum
Limits: $1 < T_{ms} < 65535$ ms

Description: programming of the acceleration time ta and deceleration time td of next motor movements.

Ta = ta Td = td

Ta corresponds to the time for the motor speed to increase from Vmin to Vmax and conversely Td corresponds to the necessary time to return from Vmax to Vmin.

When the value td is omitted, the deceleration and acceleration times are forced with the same value

Ta = Td = ta

Full step / ½ step modules do not admit parameter td, the acceleration times and of deceleration are thus always equal.

Example: programming of the slope time t = 500 ms:

WT 500.

Error code positioned

The module positions the following error codes:

0: Command WT does not have a parameter.
The provided parameter does not correspond to a digital value.
The provided parameter is signed.

1: Coherence between the parameters V_{min}, V_{max}, T_R and ? is not respected.
(cf § III.4.6).

VII - CONTROL BY INTERRUPTIONS

SIMPA modules have a specific interruption protocol allowing the fast transmission of information towards the host computer. The computer does not have then to permanently scan the whole of the modules that it controls.

VII.1 – General information

Any module can emit constantly a request for service (interruption) towards the computer in the only condition that it was previously authorized by a global authorization or an authorization by group (see paragraph VII.4 hereafter).

Three events are likely to cause an interruption from a module:

- Absence of power, supply
- Motor defect (short-circuit, open circuit…),
- Access to phase 255 (55 in full step / ½ step version)

This last event enables the user to be immediately informed of a particular sequence of phases and thus to detect specific configurations (example: detection of limit switches, locating absence, etc.)

VII.2 - Hardware

The interruptions use the serial link RS232 as a transmission support. They are materialized by the generation of a " BREAK " on the serial link (reset to 0 of the connection during a time longer than the duration of emission of a character).

VII.3 - Protocol

The management of the interruptions obeys a protocol completely independent from the normal communication protocol with the modules. All addressings are made thanks to only one character in order to minimize the response time.

Reception of any other character than those of this protocol by a module, replaces it in the traditional mode of dialogue and stops the management of the interruption process.

The management of interruptions includes 3 phases:

- A phase of authorization of the interruptions that let the modules make possible requests,

- A phase of examination that makes it possible to locate the interrupter modules,

- A phase of acknowledgement that makes it possible to indicate to the interrupter module that its request was taken into account.

VII.4 - Authorization of the interruptions

Global authorization:

Character: 1Bh (00011011 b)

All the modules connected to the serial link are authorized to emit an interruption (BREAK).

Authorization by group

It is possible to authorize only certain modules to emit an interruption.

The command has the form:
Characters: C0h to FFh (11mmmmmm b)

Bits 0 to 6 (authorization mask) make it possible to specify the modules authorized to transmit an interruption.

Any module whose address is such as: (address) AND (mask) = mask is authorized to emit an interruption logical AND on 6 bits

The other modules cannot emit interruption.

Example: command 1 1 0 0 0 1 1 0 (C6h)

Mask	0 0 0 1 1 0	
Modules	X X X 0 X X	not authorized
	X X X X 0 X	not authorized
	X X X 1 1 X	authorized

VII.5 - Scan of interruptions

An interruption arriving to the computer announces that a module has emitted an interruption but does not allow to know the generating module. The authorization by group makes it possible to specify groups of at least two modules.

Example: mask 0 1 1 1 1 1
modules of group { 0 1 1 1 1 1 (31)
1 1 1 1 1 1 (63)

The reception of a BREAK indicates that one or more modules require a service.

After a global authorization, successive group authorizations make it possible to determine more quickly than by individual interrogations the concerned modules (dichotomic approach).

Only six group authorizations are necessary to determine an interrupter module whatever its number among 63.

To allow the dichotomic search, the state of interruptions is freezed during all the scan phase. Thus, the modules can emit an interruption towards the computer during group authorizations only if they are addressed and that if they presented their interruption at the time of the last global authorization.

This protocol makes it possible moreover to manage a certain form of chronology in the interruptions since it is necessary to authorize again the whole of the interruptions to be able to take into account new requests.

Note: the number of necessary scans falls with the number of inter-connected modules if they are affected with successive numbers starting from 0.

The number of connected modules: N

The number of scans necessary: n

$$2^{n-1} < N < 2^{n}$$

Example:

The modules 9(Ø9h), 45(2Bh) and 6(Ø6h) go successively in phase 55.

Evolution of the dialogue between the host computer and SIMPA modules

✍	Emission of global authorization (1Bh) towards all the modules	
?	←	module 9 goes in phase 55
?	←	module 45 goes in phase 55 break masked by the preceding break
✍	First scan (C1h). **Inhibit any sending of break** by any other module than 9 and 45	mask 000001
	✍ modules break 9 and 45	module 6 passes in phase 55 (break sending inhibited)
✍	Second scan (C3h) No break	mask 000011 and (001001 OR 101101) ? mask
✍	Third scan (C5h) ✍ module break 45 (45 = 101101)	mask 000101
✍	Fourth scan (CDh) ✍ module break 45	mask 001101
✍	Fifth scan (DDh) No break	mask 011101
✍	Sixth scan (EDh) ✍ module break 45 101101	mask 101101

Time

At this stage, one notices that, although module 9 is announced before module 45, the dichotomic search detects module 45 first.

The search of other modules is continued only after acknowledgement of module 45.

VII.6 - Acknowledgment of interruptions

Any module detected in interruption must be acknowledged. This acknowledgment can be done in two ways:

a) By command of acknowledgment: 80h to BFh (10aaaaaa b)

aaaaaa: binary address of the module to be discharged

Example: to acknowledge module n° 45 (101101) sending of command 10101101 is (ADh)

Moreover, this command forces the other interrupter modules before the first scan (module 9 in the example) to confirm their request for service.

It is a fast way:

- To acknowledge the interruption on a first module (45 in the example);

- To determine if other modules are in request for service simultaneously (module 9 in the example).

b) By interrogation of the status of the module

A normal interrogation makes it possible to find the cause of the interruption (see command RX or QX) while acknowledging it.

Example: for module 45

Command:	45RX
return:	45EE_S

Note: Anyway, the status of the module (S) is maintained until the reading by command RX or QX, unless it has been erased by another defect even if the acknowledgment is carried out by the direct acknowledgment comman.

The second method does not make it possible to freeze the state of the requests at a given moment, since it breaks the interruption protocol returning to normal dialogue.

VII.7 - Continuation of the scans

The first series of scans made it possible to isolate module 45.

The acknowledgment of this module by the acknowledgment command announces that there are other modules in interruption.

It is thus possible to start again a series of six scans to isolate a second module.

- Acknowledgment of module 45 (ADh)
  - there are other modules in fault (break presence)
- First scan: C1h
- break (of module 9)
- Second scan: C3h
- Third scan: C5h
- Fourth scan: C9h
  - break (of module 9)
- Fifth scan: D9h
- Sixth scan: E9h

Modulate 9 detected in fault

- Acknowledgment of module 9

The acknowledgment of module N ? 9 reveals that there is no more module in interruption (absence of break in answer).

At this stage, all the interrupter modules before the first scan were isolated and acknowledged (45 and 9).

The user can then operate on these only modules if he wishes it (search for error causes for example).

To locate module 6, (the request for interruption happened after the first scan) it is sufficient to authorize the modules again in a global way.

- global authorization
  - break (of module 6)
- Six scans to isolate module 6
- Acknowledgment of module 6

Note:

A series of six scans that does not involve any reception of break means that the request for service comes from module 0.

It is up to the application to ensure the management of interruptions (detection, search for concerned modules, acknowledgment).

These repetitive operations are entirely controlled with all other implementation functions of the modules by software LIBSIM developed by MIDI INGENIERIE to facilitate the teleprocessing of the modules from a PC.

Interruptions scan algorithm

Example :

interruption detection for module n° 45 (20h)

break general Authorization Cd : 1 Bh

scan level 1 Cd : 11xxxxx1b

scan level 2 Cd : 1100001xb

scan level 3 Cd : 110001xxxb

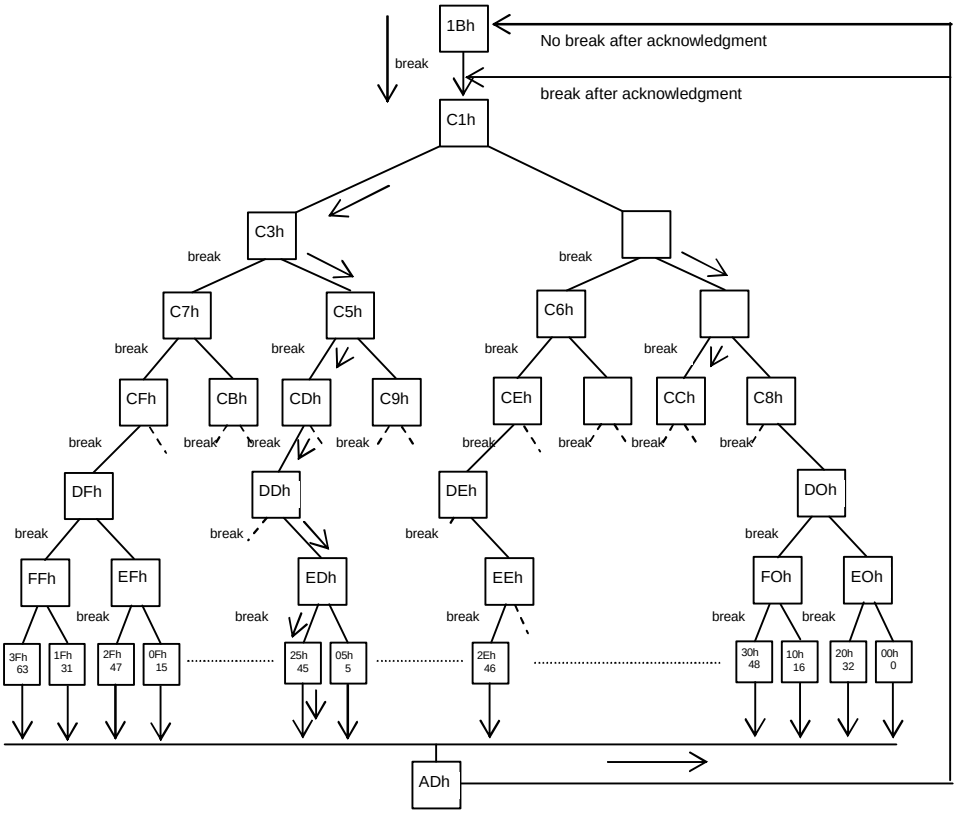
scan level 4 Cd : 11001xxxb

scan level 5 Cd : 1101xxxxb

scan level 6 Cd : 111xxxxxb

module detected

acknowledgment for detected module

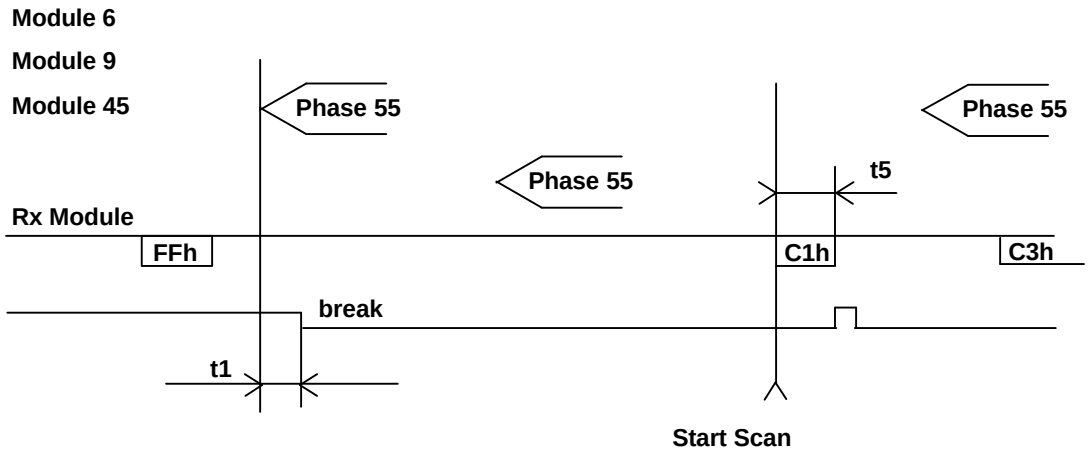


VII.8 - Timing of the interruptions

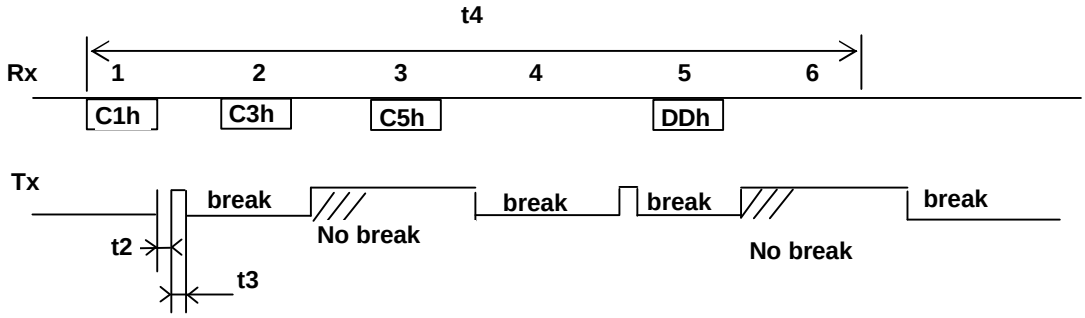
Time Considerations must be taken into account during the treatment of interruptions.

	At 4800 bauds minimum	At 9600 bauds	unit
Answer time for modules to a global authorization	3	1,5	ms
Time of recognition of the break for the UART of the PC	2,5	1,25	ms
Minimum duration of a cycle of 6 scans	31	17	ms
Duration of emission of a character at 4800 bauds	2,05	1,03	ms

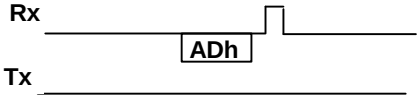
Chronogram:



First group of 6 scans



Module 45 acknowledgment



		At 4800 bauds	At 9600 bauds	unit
Time of emission	T1	< 20 µs		µs
Release time of the Tx line	t2	< 30 µs		µs
Duration of emission of the break in interrogation	T3	= 500 µs		µs
Scan time for 63 modules	t4	31	17	ms
Duration of a character	t5	2,05	1,3	ms

NOTICE

- The acknowledgment of the axis could have been obtained by sending the following commands (using the standard protocol).

Command	45RX	for axis 45
Answer	45EE_S	
Command	09RX	for axis 09
Answer	09EE_S	

- Although axis 9 required an interruption before axis 45 it is the latter that is recognized first by dichotomy.

On the other hand, if the beginning of the scan happens before the request of axis 6, to locate this axis it will be necessary to make global authorization again, followed by a new scan.

LIST OF SIMPA COMMANDS (Full step / ½ step version)

* [@]MR		Master Reset of the module, outputs at FF (implicit with powering)
* [@]MB		Selection of the limit switch operating mode
* [@]MN		Cancel of the limit switch operating mode
[@]WH	V_{max}	Set point speed (20 to 20000 step/s) stage
[@]WL	V_{min}	Start Speed (20 to 20000 step/s)
[@]WT		Acceleration slope time in ms (1 with $65x10^6$ / Vmax)
[@]D+		Clockwise direction for next relative displacements
[@]D-		Counter-clockwise direction for next relative displacements
[@]DG	n	n steps (or half steps) tolerated in stall (< 256 steps)
[@]DI		Reference 0 of next absolute displacements
[@]DP	Pr	Reference Pr for next absolute movements
[@]DR	n	Number of steps for next relative displacements (0 to 999998)
[@]GA	Pa	Execution of an absolute movement, direct mode (Pa) (-8388606 < Pa < 8388606)
* [@]GE		Deceleration and stop of a movement or a sequence
[@]GF		Execution of a continuous movement
[@]GH		Return to origin
[@]GI	Im	Imotor = Irated * Im/255
* [@]GL	Out	Positioning of logical outputs (Out: in hexa)
* [@]GM		Motor power (implicit with GO, GA, GH and sequence execution)
[@]GO		Execution of a relative movement in direct mode (D+/- n steps)
* [@]GR		Motor power off
* [@]GS		Stop (immediate stop of a movement or a sequence)
[@]SA	ns	Execution of sequence if condition E2 (edge)
[@]SC	ns	Selection of conditional Start Sequence (E2 after RESET)
[@]SD	ns	Selection of Start Sequence (carried out on RESET)
[@]SE	ns	Erasing of sequence (NS = 0: erasing of all the sequences)
[@]SN	ns np	Creation of sequence NS (1 to 99) of Np phases (1 to 50)
[@]SP	ns np S+/-/I Na Cns [NL sc or NQ ss] [Nex1x2… x8] [NS ps] [NO out]	

Definition of phase np sequence ns

Na: Nature

NA acceleration	NT motor power ON
NC plateau speed modification	NU motor power OFF
ND deceleration	NV constant speed
NG stall detection	NW wait
NH return to origin (HOME)	NX absolute movement
NP relative movement	NZ reset absolute position

Cns: set point (number of steps, position, plateau speed or time, depending on nature)

NL sc: number of next sequence

NQ ss: call sub-sequence (NL and NQ are exclusive)

Ne: NE or NF phase jump on state or edge of logical input

NS ps: next phase

NO out: positioning logical outputs

[@]SR		Suppression of the selection of start sequence
[@]SS	ns	Execution of the sequence ns
* @ RD		Sequences and movements monitoring
* @ RL		Reading of local parameters
* @ RP		Reading of step counter, I/O status
* @ RS	ns np	Reading of the np phase of the ns sequence
* @ RV		Reading of the version and index of the software
* @ X-ray		Reading of status code of the module

@ address of the module

[@] multimodules commands

* command usable whatever the module status

[Cmd] optional command

LIST SIMPA COMMANDS (Microstep version)

* [@]MR		Master Reset of the module, outputs at FF (implicit with powering)
* [@]MB	[H]/[L]	Selection of the limit switch operating mode
* [@]MN	[H]/[L]	Cancel of the limit switch operating mode
[@]MS	[N][S][B]	Current Management Mode N:rated/S:rated + Stb/B:rated + Stb + Boost
[@]PG	n:[v2…[v32]]	Variables Management, value to be given
[@]PD	#n:v	Variables Management, decimal initialization value
[@]PH	#n:v	Variables Management, hexadecimal initialization value
[@]WH	V_{max}	Plateau speed Set point (20 to 20000 step/s)
[@]WL	V_{min}	Start Speed (20 to 20000 step/s)
[@]WT	T_R	Acceleration slope time in ms (1 with $65x10^6$ / Vmax)
[@]WN	µ	Resolution in microsteps per step (only in microstep mode)
[@]DG	n	n microsteps tolerated in slip (< 256 steps)
[@]DI		Reference 0 of next absolute displacements
[@]DP	Pr	Reference Pr for the next absolute movements
[@]GA	Pa	Execution of an absolute movement ($-2^{31}-1 < Pa < 2^{31}-1$)
* [@]GE		Deceleration and stop of a movement or a sequence
[@]GF	[+]/[-]	Execution of a continuous movement
[@]GH		Return to origin
[@]GI	Im	Imotor = Irated * Im/255
* [@]GL	Out	Positioning of logical outputs (Out: in hexa)
* [@]GM		Motor Power ON (implicit with GO, GA, GH and execution of a sequence)
[@]GO	[±]/[n]	Execution of a relative movement (± n microstep < $2^{31}-1$)
* [@]GR		Motor power OFF
* [@]GS		Stop (immediate stop of a movement or a sequence)
* [@]GP	P[I]	Type: remote management for logical output
[@]SD	ns	Selection for Start Sequence (carried out on RESET)
[@]SE	ns	Erasing of sequence (ns = 0: erasing of all the sequences)
[@]SN	ns np	Creation of a sequence ns (1 to 200) of np phases (1 to 200)
[@]SP	ns np Na Cns [NL sc or NQ ss] [Nex1x2… x8] [NS ps] [NO out]	

Definition of phase np sequence ns

Na: Nature

NC modification of plateau speed

NG slip detection	NV constant speed
NH return to origin (HOME)	NW wait
NP relative movement	NX absolute movement
NT motor power ON	NZ absolute position reset
NU motor power OFF	PV modification of variable

PI modification of initialization value of a variable

PR current value of a variable as init value

PA addition on variable	PC variable transfer CP1

PT variable test

Cns: set point (number of microstep, position, plateau speed or time, depending on nature)

NL sc: number of the next sequence

NQ ss: call for sub-sequence (NL and NQ are exclusive)

Ne: NE or NF phase jump on status or edge of logical input

NS ps: next phase

NO out: positioning of logical outputs

[@]SR		Suppression of the selection of starting sequence
[@]SS	ns	Execution of the sequence ns
* @ QD		Monitoring of sequences and movements
* @ QL		Reading of local parameters
* @ QP		Reading of microsteps counter, I/O status
* @ QS	ns np	Reading of the phase np of the sequence nS
* @ QV		Reading of version and index of the software
* @ QX		Reading of status code of the module
* @ QC		Reading a parameter remote logical output(GL/GP)
* @ N	# n[H]	Reading a variable

@ module address * command usable whatever the status of the module

[@] multimodules commands [Cmde] optional command