目录

DEVICE CAMMING2 3
1. Introduction 3
1.1 Installation 4
1.1.1 Device declaration in the configuration file (.CNF) 4
1.2 Operation 5
1.2.1 Calculation of the resolution 5
1.2.2 Decimal point 6
1.2.3 Speed 6
1.2.4 Main controls 6
1.2.5 Change speed and moving ramp time 7
1.2.6 Description of epicycloidal motion 8
1.2.7 Profile reduction 9
1.2.8 Type of stop during acceleration ramp 9
1.2.9 Analogue output calibration 10
1.2.10 Movement 11
1.2.11 PID+FF calibration 12
1.2.12 Movement application 14
1.2.13 The sector structure 15
1.3 The sectors 15
1.3.1 The sector of acceleration 16
1.3.2 The deceleration sector 19
1.3.3 Speed change sector 21
1.3.4 The Start sector synchronized to the Master 24
1.3.5 The end cam sector 25
1.3.6 Absolute jump sector 25
1.3.7 Conditional jump sector 25
1.3.8 Loop cam sector 25
1.3.9 Not operative sector 26
1.3.10 Definition of area sampled zero sectors 26
1.3.11 Update count sectors 26
1.3.12 Cam sectors description 26
1.3.13 Basics for building a cam to wire-guides 28
1.3.14 Basics for building a cam for fly cut with extra speed 29
1.4 Device errors management 30
1.5 Device warning management 30
1.6 Simulated master management 30
1.6.1 Programming example 31
1.7 M/S Transducer frequency ratio limitation 31
1.8 Inputs configuration table 32
1.9 Outputs configuration table 32
1.10 Table commands, states and parameters: Symbols used 32
1.10.1 Conditions 33
1.10.2 Parameters 33
1.10.3 Axis variables 36
1.10.4 Program variables 37
1.10.5 Commands 38
1.10.6 States 40
1.10.7 Device limitations 42

DEVICE CAMMING2

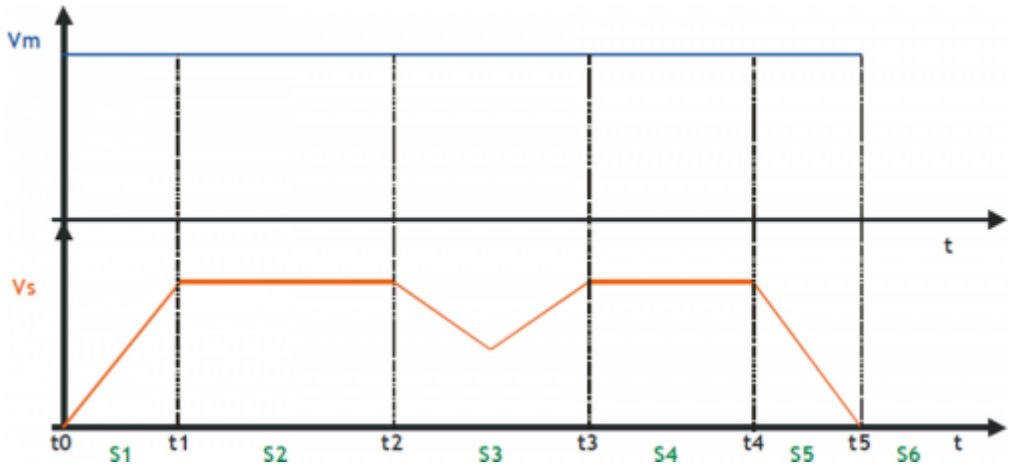
1. Introduction

The camming, is a technique applicable to motion control servo axes and allows you to solve applications where one or more slave axes have spaces, uneven, too staying in sync with respect to the position of a reference axis called “master”. The master axis can be a real or virtual axis (the master simulated).
Typical applications are:

- Cuts and processes on the fly, both linear and circular, on plastic, sheet metal, cardboard.
- Packaging in place of mechanical cams.
- In the winding of cable, wire, etc. with wiring-guide.
- In textiles and in feeding the “grinding machine” for layering of fabrics or pasta.
- In screen printing or flexo printing plates.
- In the lines of “transport product” for spacing and/or synchronization of the materials movement.

The absolute position that must assume the slave axis is always expressed as a function of position absolute master axis and this Association is placed in a specific table called “cam table”.
The “cam table” consists of 40 sectors; each sector consists of:
CodeG = *operating instruction* of the sector in use.
CodeQm = *incremental* position of the master, in units; are accepted only positive increments.
CodeQs = *incremental* position of the slave, in units; both positive and negative increments are accepted.
CodeM = general numeric code that can be used by the PLC logic.
CodeQma = special operating instructions auxiliary master quota used.
CodeQsa = special operating instructions auxiliary slave used quota.

The codeG operative instructions associated with each sector of the cam, you can be defined how motion law (acceleration, deceleration, constant speed…) the slave axis must move along the space established in **codeQs** at the same time as the master runs through the space defined as **codeQm**.
Until the master is moving at constant velocity, the space covered by the master axis is directly proportional to the time spent and the **codeQs** e **codeQm** spaces being always defined in the same amount of time even the law of motion applied to the slave axis, within the sector, is applicable in direct proportion to the space covered by the master in the sector; the master and the slave are therefore linked in space between them.
If the speed constant chosen for the master is the maximum you can evaluate immediately what the maximum accelerations, decelerations and speeds that will be the slave axis.
This procedure allows you to formulate the law of motion of the slave axis as a function of time to evaluate the dynamic performance required by your application and then apply the the same law of motion as a function of space covered by the master during the execution of the cam.
To simplify the calculation of the absolute positions of the master and the slave It is assumed that the master is moving at a constant speed whereby the axes positions can be represented in a Cartesian diagram Speed / Time. Below is a simple example of compiling the “cam table”.

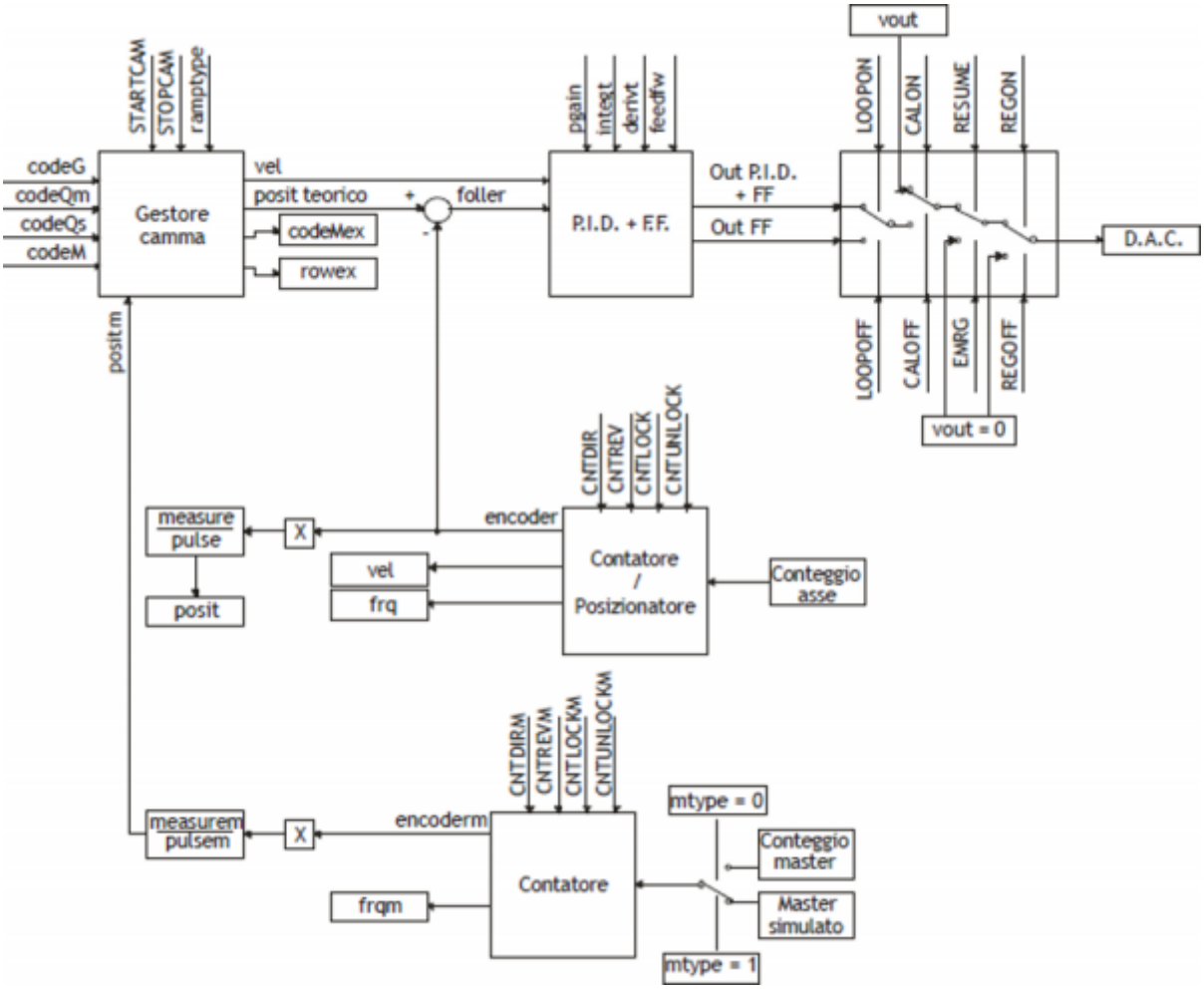


In order to can run a cam as in the example, you have to fill in the “cam table” as follows:

Sector	CodeG	CodeQm	CodeQs	
S1	132	100	50	Acceleration sector with Vs = Vm at the end of the sector
S2	133	200	200	Intermediate sector at constant speed
S3	134	160	120	Compensation sector with initial velocity = final velocity
S4	133	150	150	Intermediate sector at constant speed
S5	135	90	45	Deceleration sector with Vs = 0 at the end of the sector

Sector	CodeG	CodeQm	CodeQs	
S6	136	-	-	Finish command cam

QEM is available to help customers in the of the “cam table” compilation.
The device can be divided into two main parts:
A slave axis positioner with trapezoidal or selectable planetary ramps. Analog cam generator.
Basic block diagram is as follows:



1.1 Installation

1.1.1 Device declaration in the configuration file (.CNF)

In the configuration file (.CNF), the BUS section must be declared so that they are present the hardware resources required for the implementation of the device CAMMING2. There must be at least two bidirectional counter and a 16-bit resolution analog output. In the INTDEVICE section of the .CNF file must be to add the following definition:

```
;-------------------------------
; Internal device declaration
;-------------------------------
  INTDEVICE
        <DEVICE_NAME> CAMMING2 TCAMP COUNTS COUNTM INTL IAZERO INTLM
                      IAZEROM ING INGINT IOUTA OUT
```

Where:

INTDEVICE	Is a keyword indicating the beginning of the definition of internal device.
device_name	Is the device name.
CAMMING2	Is the keyword that identifies the device described in this document.
TCamp	Sample time device (1÷250 ms).
CountS	Bidirectional counter Slave address
CountM	Bidirectional counter Master address
IntL	Number of the interrupt line dedicated to the encoder zero pulse Slave during the activity phase of presets. Allowed values: 1÷8 (to prevent the device uses this resource, Enter the X character).
IAZero	Slave zero pulse enable input (to prevent the device uses this resource, Enter the X.X character)

IntLM	Number of the interrupt line dedicated to the Master encoder zero pulse during the research phase of presets. Allowed values: 1÷8 (to prevent the device uses this resource, Enter the X character).
IAZeroM	Enable input zero-pulse master (to prevent the device uses this resource, Enter the X.X character)
InG	Entry for generic function as described in the section of the table input configuration (to prevent the device uses this resource, Enter the X.X character)
InGInt	Number of the interrupt line dedicated to a generic function as described in the section of the table input configuration. Allowed values: 1÷8 (to prevent the device uses this resource, Enter the X character).
IoutA	Hardware address of the DAC from analog output Slave.
Out	Output for generic function as described in section output configuration table (to prevent the device uses this resource, Enter the X.X character)

1.1.1.1 Application example

You take as an example a CAMMING2 device configured as in the START-UP and with parameterisation of the axis (set-up) already written.
The task is first initialized the device and then run a interrupt input which shows its status on an exit.
The task will be carried out:

```
;-------------------------------------
; CAMMING2 device management
;-------------------------------------
INIT AxisX                      : Initializes the axis
WAIT AxisX:st_init              ; Wait until the axle is initialized
LOOPON AxisX                    ; Hooks the control loops
WAIT AxisX:st_loopon            ; Wait for the axis has hooked the
                                ; control loops
CALOFF AxisX                    ; Exit from calibration
                                ; of the axis
WAIT NOT AxisX:st_cal           ; Wait until the device is not in
                                ; calibration
CNTUNLOCK AxisX                 ; Unlocks the master counter
WAIT NOT AxisX:st_cntlock       ; Wait until the counter master is
                                ; unlocked
CNTDIR AxisX                    ; Sets the right direction of increase
                                ; slave counter
WAIT NOT AxisX:st_cntrev        ; Wait until the slave counter is
                                ; set in the sense of increase
CNTUNLOCKM AxisX                ; Unlocks the master counter
WAIT NOT AxisX:st_cntlockm      ; Wait until the counter master is
                                ; unlocked
CNTDIRM AxisX                   ; Sets the right direction master
                                ; counter increment
WAIT NOT AxisX:st_cntrevm       ; Wait until the counter master is
                                ; set in the sense of increase
REGON AxisX                     ; Unlock the adjusting
WAIT NOT AxisX:st_regoff        ; Wait for the regulation unlocking

MAIN:

IF AxisX:st_int                 ; If the interrupt line is active
  AxisX:funOut = 2              ; activates output
ELSE
  AxisX:funOut = 1              ; disables the output
ENDIF

ENDIF                           ; END

WAIT 1
JUMP MAIN
END
```

1.2 Operation

1.2.1 Calculation of the resolution

The CAMMING2 device did not has within it the *cntratio* parameter, but let the installer ability to work with unfinished encoder resolutions by setting the data as space covered in a round encoder (*measure*) and number of pulses/round encoder (*pulse*). The relationship between *measure* and *pulse* is the encoder resolution and must have values between 1 and 0.000935.

1.2.1.1 Definitions:

- The *measure* paramenter is placed in units without decimal points (for example 100.0 millimeters is inserted 1000 tenths of a millimeter).
- The *pulse* parameter is inserted 4 encoder bitrate (for example if connected an encoder with 1024 pulses round, is inserted 4096, If the measure is calculated on a encoder round).

1.2.1.2 Example:

You have to control a rotating table that have the accuracy of 0,1° with the 1024 pulses round encoder mounted directly to the motor; you will set the following values:
measure = 3600
pulse = 4096

1.2.2 Decimal point

If for the selected unit of measure is also provided for the presence of a decimal point positions must be represented as an integer and represent space on the drive measurement without decimal point. The resolution must be calculated using the same method and in the parameter *measure* the magnitude without decimal point. The decimal point will then be inserted into the representation of the value-time viewers (ex. as properties in the HMI). This parameter can take 0÷3 values.

1.2.3 Speed

The speeds are always expressed in whole units of measure in the unit of time choice. From this it emerges that the device must know the location of the decimal point of the unit of measure and this is done with the *decpt* parameter.

1.2.4 Main controls

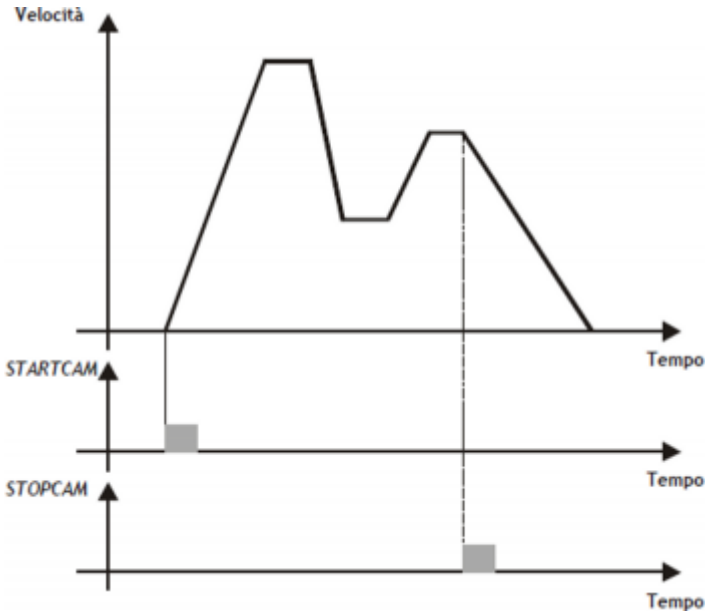
This section describes only the use of a number of some commands; for descriptions of all the set commands, refer to the following chapters.
The two main controls are what give start and break execution of cam: *STARTCAM* and *STOPCAM*. There are also a series of commands dedicated to emergency response, the loop control, START and STOP the axis.

1.2.4.1 STARTCAM

The*STARTCAM*command, the slave axis attaches to master and will follow the trend described in the cam always starting from the first sector. You cannot take a *STARTCAM* during the execution of the (*st_camex* = 1) cam; ihis control is left to the programmer.
The cam will pop out automatically if it encounters a *END* instruction or you can stop it in flight by using the *STOPCAM*.

1.2.4.2 STOPCAM

If the cam is running (*st_camex* = 1), Once received the *STOPCAM* command the slave axis is released immediately by the master, brings his speed to zero following the deceleration ramp set (*tdec* parameter) and remaining in reaction to space. The deceleration ramp is asynchronous with respect to the master.

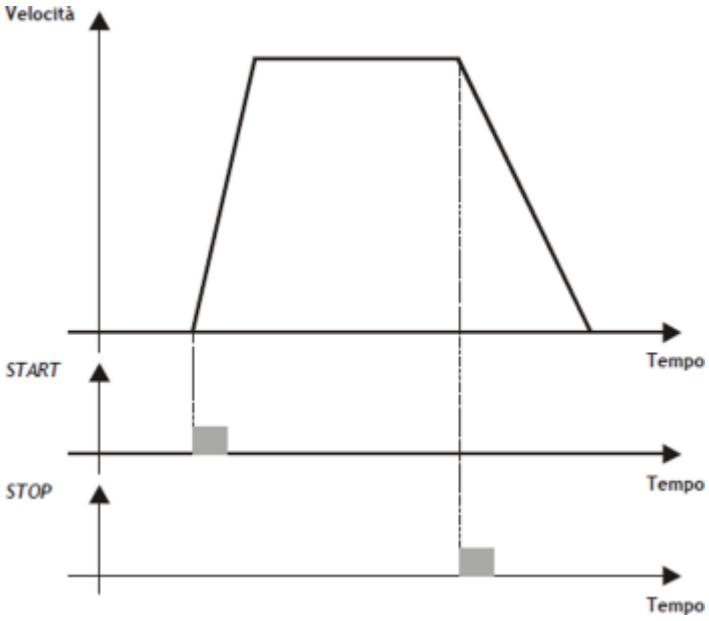


1.2.4.3 START

At the *START* command, the slave axis is positioned to the dimension declared in the *setpos* variable with the speed set in *setvel*; the placement will run using the acceleration ramp set in *tacc* parameter and the deceleration ramp set in parameter *tdec*. The type of ramp used (trapezoidal or epicycloidal) is inserted in the *ramptype* parameter.

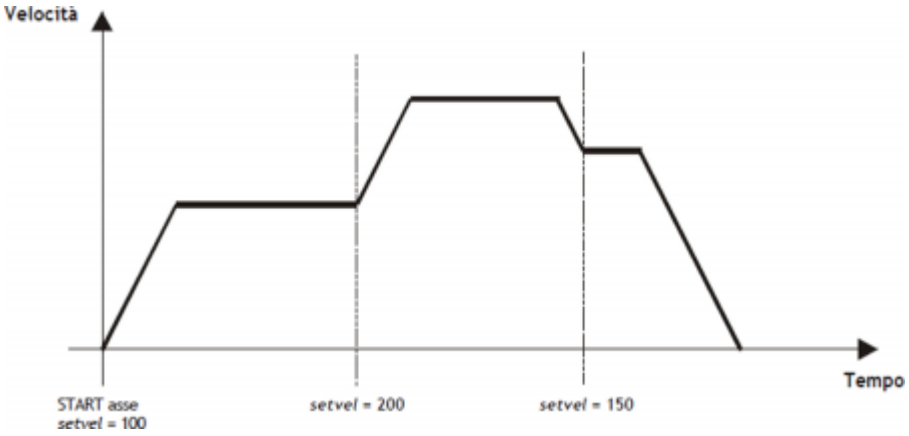
1.2.4.4 STOP

If during the placement (not during the execution of a cam) you must stop the axis with a deceleration ramp, It will simply give the*STOP* command the axis decelerates to a stop with the ramp in the *tdec* parameter.

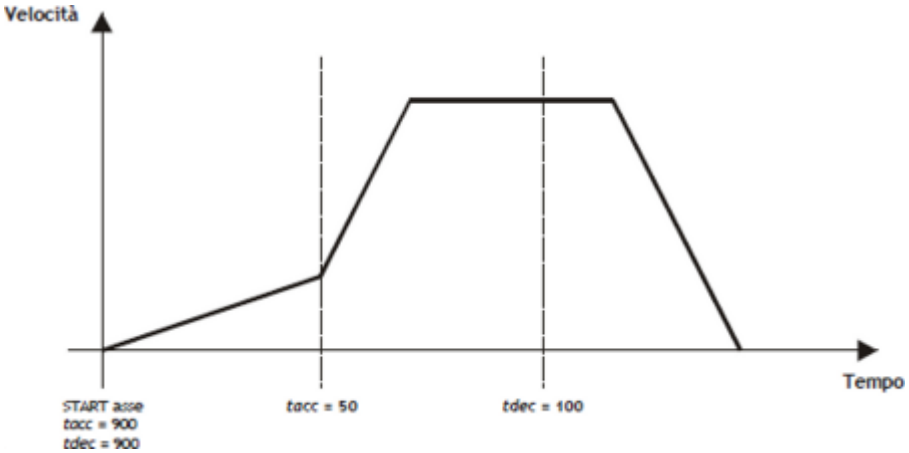


1.2.5 Change speed and moving ramp time

When positioning it is possible to vary the speed of the axis without affecting the location to get to. This can lead to an increase or a decrease in velocity, even in several places of the same positioning. This is accomplished with new writing in the *setvel* parameter. No gear is always available except during the deceleration ramp and is reported (*st_chvel* = 1)



When positioning can be varied even acceleration/deceleration times. For example, the device can start a placement at a rate very short and once they reach the speed set, varied *tacc* parameter and executed a change of speed with a very long ramp. For special applications and trapezoidal ramps, ramp time can be varied even during a change of speed, in this case the new time is put into execution immediately.



1.2.5.1 EMRG

This command puts the axis in emergency conditions; the *st_emrg* state is set to one. If the emergency command is sent to the axis during a placement, movement is stopped without deceleration ramp, the analog output will be set to zero volts and you dropped the reaction of space. If the cam is active (*st_camex* = 1), the movement is interrupted without deceleration ramp, the analog output will be set to zero volts, space reaction is released and the cam (*st_camex* = 0).
With *st_emrg* = 1 (emergency condition), you cannot move the axis.

1.2.5.2 RESUME

This command will reset the emergency condition; the axis comes in reaction to space and waits for a command to be able to move (does not automatically resume interrupted positioning).

1.2.5.3 LOOPOFF

The LOOPOFF command removes the reaction of space without stopping the axis. With *st_loopon* = 0 the axis handling axis commands but accepts all placements will be performed without reaction of space.
A placement made reaction loop-free is comparable to a positioning run without proportional gain (is not guaranteed to arrive in position).

1.2.5.4 LOOPON

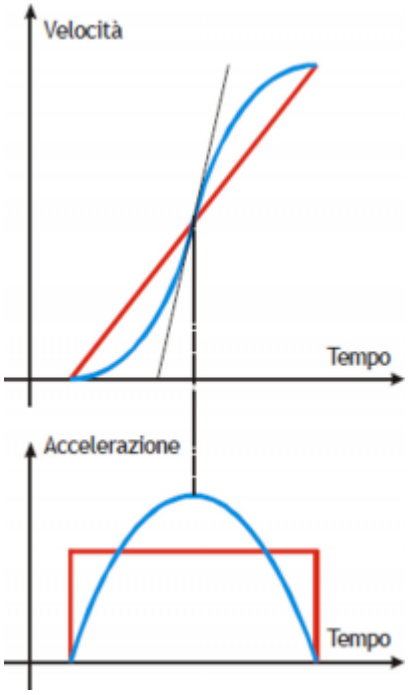
The *LOOPON* command closes the ring of space without stopping the axis. With *st_loopon*= 1 the axis is moved using the P.I.D. control features.

Following is a table summarizing the conditions necessary for the axis in reaction to space and to perform placements.

Loopon	Emrg	Space reaction	Possibility of movement
YES	NO	YES	YES
YES	YES	NO	NO
NO	NO	NO	YES
NO	YES	NO	NO

1.2.6 Description of epicycloidal motion

The epicycloidal motion is used to move the axes without sudden variations of speed. The time of positioning of an axis moved by trapezoidal ramps is the same compared to the same axis moved by epicycloidal ramps, but ramps vary the shear rate (acceleration) with up to half of the ramp itself.
For comparison shows the difference of the development of the acceleration in the two cases: with linear ramp (trapezoidal) and with ramp reducers.

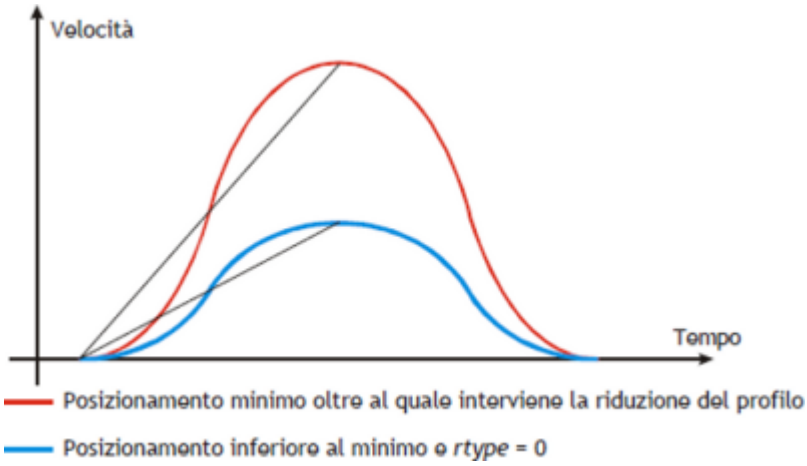


The same goes for the deceleration ramp.
Epicycloidal movement has the ability to behave in different ways in the event of a reduction in profile (*rtype*) and in the case of stop during acceleration (stopt) If the cam is not running (*st_camex* = 0).

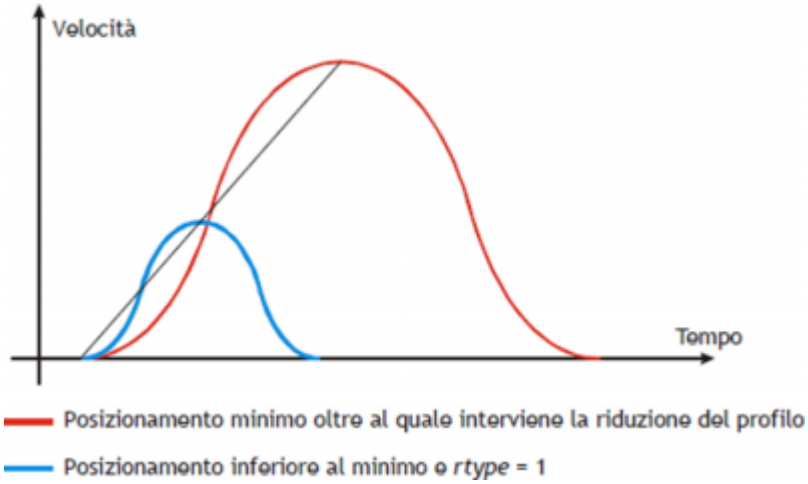
1.2.7 Profile reduction

	The profile reduction is used only if you are doing a placement and not if you're running a cam (*st_camex* = 0).

If the cam is not running (*st_camex* = 0) and space to go is less than that which allows to reach the speed set by the acceleration and deceleration ramps, you pass in the phase called “profile reduction”.
You can keep fixed the time of ramps, decreasing gradients of the ramps and the speed in proportion (*rtype* parameter set to 0).



You can also decrease the time of the ramps while maintaining the gradient of constant acceleration and decrease speed in proportion (*rtype* parameter set to 1).

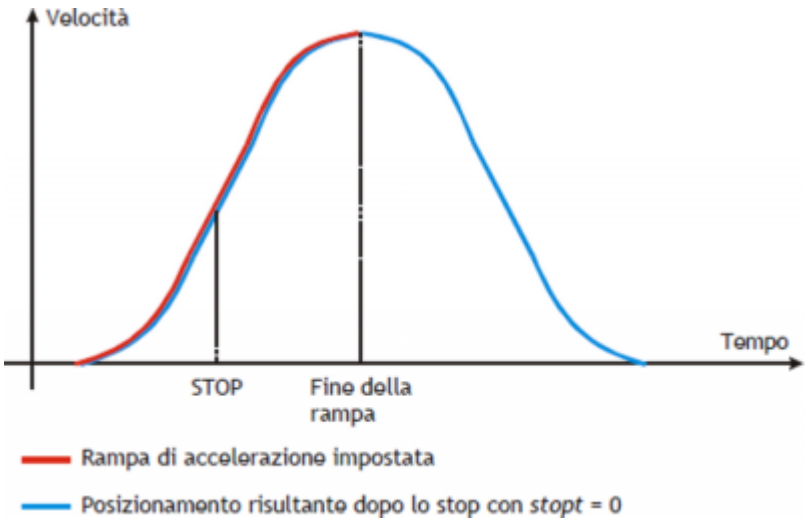


With the rtype parameter set to 0 xtend considerably the time needed to placements with loss of productivity of the machine, instead of setting it to 1 in case of short placements shorter, but keeping the constant gradient you lose the beneficial effect of the epicycloidal action.

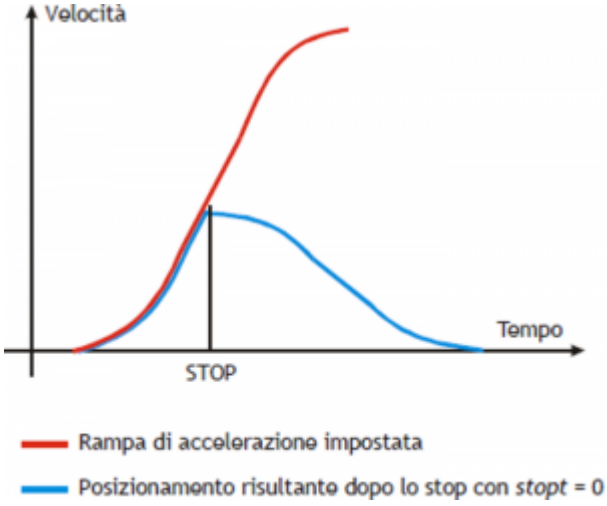
1.2.8 Type of stop during acceleration ramp

	The type of stop while the ramps is only used if you are doing a placement and not if you're running a cam (*st_camex* = 0).

In the event that the cam is not running (st_camex = 0) and we should curb the axis during acceleration with *STOP* commnand You must choose whether to complete the flight, or if you want to abort the flight and consequently change the epicycloidal action.
In case you set the *stopt* parameter to 0 is first completed the acceleration ramp and then performed the deceleration ramp.



In case you set the parameter stopt to 1 stops the acceleration ramp and started immediately the deceleration ramp set.



You immediately notice that there is a substantial difference between the setting of *stopt* to 0 or to 1. To make the choice of what type of stop use, one must remember that in case of emergency stop the emergency command exists that instantly locks and without ramp positioning.

1.2.9 Analogue output calibration

	**Before starting actual placements you must make sure that electrical connections and mechanical appliances have not cause malfunctions.**

For the management of the axis, the device uses a ± 10 V analogue output range and 16-bit signed resolution; this calibration function with analog output can be driven with a constant value in order to test links and functionality.

1.2.9.1 Preliminary motion

- Remove the emergency condition with the *RESUME* command.
- The state *st_emrg* = 0
- Enable calibration axis status with the *CALON* command; the st_cal state must therefore be 1.
- You can now set the analog voltage with the *vout* parameter; the value is expressed in tenths of volt (-100 ÷ 100 = -10 ÷ 10 V). It is recommended to introduce low values (5, 10, 15... equal to 0.5, 1, 1,5 V).
- When the axis is moving the *frq* parameter indicates the frequency in Hz of the sphases of the transducer.
- The *posit* parameter that displays the position varies indicating the space covered by the axis. If setting a positive voltage will count decrements, it is necessary to invert the phases of the transducer or reverse the direction in driving.
- You can reverse the direction of the count using the *CNTREV* command.
- If with output voltage to zero the axis is not stationary, adjust the *offset* parameter to correct the voltage

until movement does not stop. The input value (each bit corresponds to 0.3 mV), will be added algebraically to the value of the analogue output; This operation allows to compensate for any drift of electronic component, both in QMOVE outgoing and to driver input. The value is expressed in bits with sign.
For an optimal result of calibration this operation must be performed with the system to temperature capacity.

- To disable calibration status send the *CALOFF* command.
- The state *st_cal* = 0

1.2.9.2 Output setting

The device generates the voltage value of the analogue output based on a ratio between the maximum velocity of the axis and the maximum output voltage. Proportionality is obtained with the *maxvel* parameter, that represents the speed of the axis on the analog voltage (10 V). Obviously the axle must behave symmetrical analog voltage to zero, therefore the speed must be the same on both the positive and negative voltage at maximum.
Prior to determining the value of maximum velocity,we must establish the unit of time to use for the representation of the speed in the device; the *unitvel* parameter defines the unit of time of speed (Um/min or Um/s).

1.2.9.3 Theoretical method for the maximum speed determination

The theoretical method is a calculation that was performed on the basis of maximum motor speed. Once established the maximum revolutions per minute that are declared in the motor, We get the maximum velocity is expressed in the unit of measure the unit of time chosen.
Enter the maximum velocity value calculated in the *maxvel* parameter.

1.2.9.4 Practical method for determining the maximum speed

The practical approach is based on the reading of the speed detected by the device in the *vel* parameter, giving the drive a known voltage. To provide the voltage to drive the device should be placed in a position of calibration as described in the previous paragraph. If the system permits, give the drive a voltage of 10 V and read the speed value in the *vel* parameter. If, on the other hand, provides a part of the output voltage (1, 2, ... 5 V), calculate the maximum velocity with a proportion.
Enter the value found by maximum velocity in the *maxvel* parameter.

1.2.10 Movement

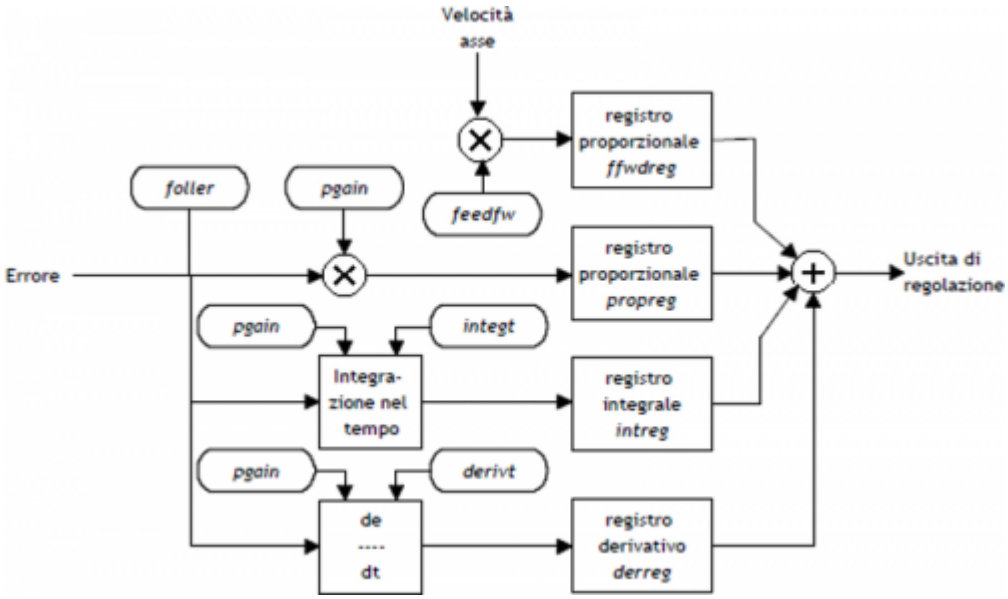
The procedures described here have allowed us to complete the first phase of parameterizing device. Now you can run simple movement of the axis.

- Move the axis in a position whereby it can fulfill a certain area without touching the maximum and minimum quota limits.
- Set the current position to zero axis, by setting the parameter *posit* = 0.
- Set up the parameters that define the position of the limit switches software: *minpos* = 0 and *maxpos* the value of the maximum stroke of the axis.
- Set the parameter that defines the axis time to reach the maximum speed *tacc* = 100. This parameter is expressed in hundredths of a second (100 = 1 sec.)
- Set the speed of positioning with the *setvel* parameter.
- Set the target quota with the *setpos* parameter.
- Set the parameter *feedfw* = 1000 (100%)
- If the device is in emergency state (st_emrg = 1) give the *RESUME* command.
- Start positioning with the *START* command. To stop the movement give the *EMRG* command.

This first movement was done without speed feedback space. The placement may have been executed with some error introduced by the non-linearity of the components or an imperfection in the calibration of the maximum speed. Enabling space feedback this error goes away.

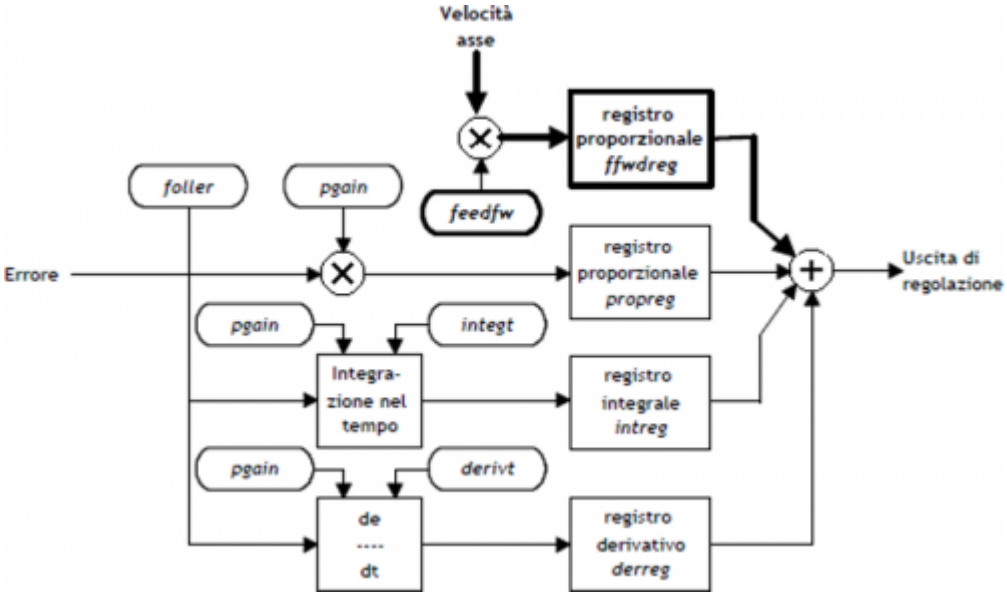
1.2.11 PID+FF calibration

The placement runs in the preceding paragraph has been made without considering any position errors.
To check the correct position of the axis in continuously and automatically, You must have the position feed-back; for this reason introduces the control algorithm PID + FF including proportional shares, integral, derivative and feed-forward; the value of the analog output is given by the summation of feed forward, proportional, derivative and integrative actions.
This section describes a series of actions to adjust the parameters that affect this control.
In order to achieve a satisfactory adjustment is sufficient to use only the actions feedforward and proportional; integral and derivative actions are used only for adjustments under special conditions.



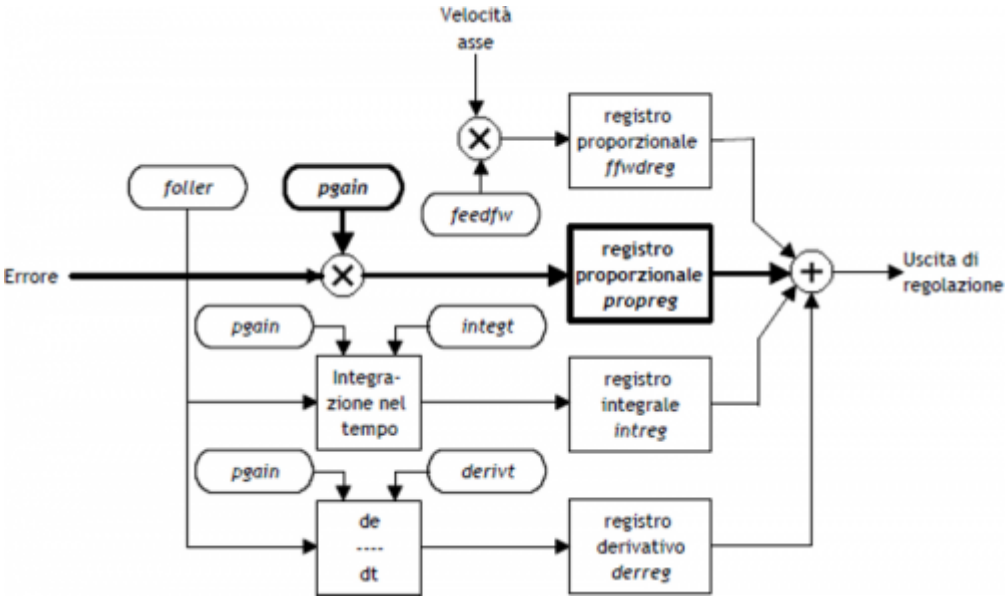
1.2.11.1 Feed forward action

The feed-forward helps make the system more ready on placements, providing the analogue output voltage proportional to the theoretical speed of positioning. In practice it is the component which you performed the placements of the previous chapter.
The contribution of this action can be adjusted with the parameter *feedfw*; This parameter is expressed as 1/1000 speed theoretical portion; so, to introduce such as 98.5% you must set 985 (thousandths).



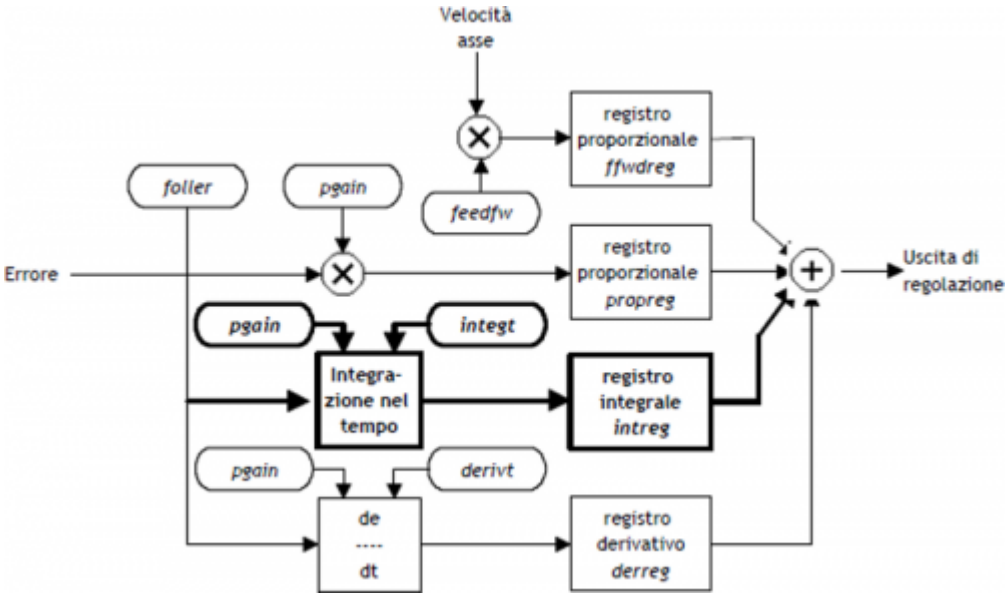
1.2.11.2 Proportional action

This action provides an output proportional to the instantaneous axis position error. The extent of the proportional action is defined by the *pgain* parameter that defines the sensitivity of the system.
The pgain parameter is introduced in thousandths; the unit value of the gain (1000) provides an analog output to maximum value (10 V) concerning the maximum speed error. For maximum speed error means the space taken by axis - at the maximum speed - for the duration of the sampling time of the device.



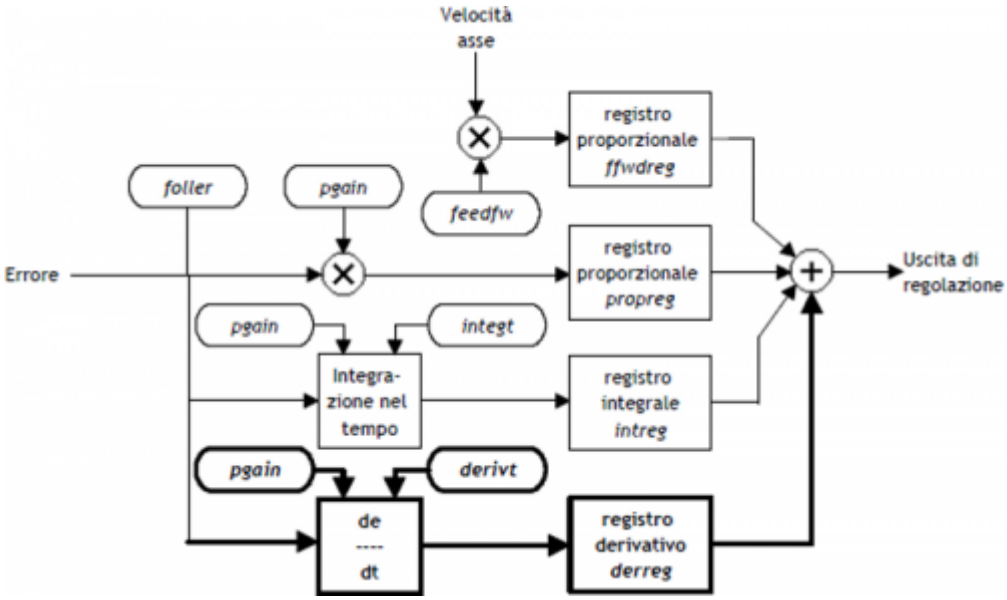
1.2.11.3 Integral action

Integrates the position error of the system over time set in *integt* parameter updating the release until the error is not canceled.
If it drops the integration time of the error, the system retrieves the error faster, but can become unstable, tending to swing.



1.2.11.4 Derivative action

Anticipates the change of the motion of the system by delete the overshoot positioning. The size of change is calculated over time set in *derivt* parameter.
When the time of derivation of error is higher, more faster is the transient error recovery system, but if you enter a value that is too high the system becomes unstable, tending to fluctuate.



1.2.12 Movement application

In order to move the slave axis must first declare the parameterization of axis. Once this stage is thought to want to move the slave axis with jog manuals using Inp01 input for moving forward and the Inp02 input to move it back.
As an example, consider a device configured as in START UP. The task is first initialized the device and then run the manual jog.

```
;-----------------------------------------------------
; Manual jog management
;-----------------------------------------------------

INIT AxisX                          : Initializes the axis
WAIT AxisX:st_init                  ; Wait until the axle is initialized
LOOPON AxisX                        ; Hooks the control loops
WAIT AxisX:st_loopon                ; Wait for the axis has hooked the
                                    ; control loops
CALOFF AxisX                        ; Exit from calibration
                                    ; of the axis
WAIT NOT AxisX:st_cal               ; Wait until the device is not in
                                    ; calibration
CNTUNLOCK AxisX                     ; Unlocks the master counter
WAIT NOT Axis:st_cntlock            ; Wait until the counter master is
                                    ; unlocked
CNTDIR AxisX                        ; Sets the right direction of increase of
                                    ; slave counter
WAIT NOT AxisX:st_cntrev            ; Wait until the counter is slave
                                    ; set in the sense of increase
CNTUNLOCKM AxisX                    ; Unlocks the master counter
WAIT NOT AxisX:st_cntlockm          ; Wait until the counter master is
                                    ; unlocked
CNTDIRM AxisX                       ; Sets the right direction of increase of
                                    ; master counter
WAIT NOT AxisX:st_cntrevm           ; Wait until the counter master is
                                    ; set in the sense of increase
REGON AxisX                         ; Unlock the adjusting
WAIT NOT AxisX:st_regoff            ; Wait the unlocked of the regulation
MAIN:

IF Inp01 AND Inp02                  ; If the Inp01 and the
                                    ; Inp02 inputs are active
  IF NOT AxisX:st_still             ; If the axis is not stationary
    STOP AxisX                      ; Stop the axis
  ENDIF
ENDIF

IF Inp01 AND NOT Inp02              ; I the Inp01 input is
                                    ; active and the
                                    ; Inp02 input are deactivate
  IF AxisX:st_still                 ; If the axis is stopped
    AxisX:setvel=AxisX:maxvel/10    ; I set the speed of
                                    ; manual movement
    MANFW AxisX                     ; Forward manual
  ENDIF
ELSE
  IF NOT Inp02                      ; If the Inp02 input
                                    ; is deactive
    IF NOT AxisX:st_still           ; If the axis is not stationary
      STOP AxisX                    ; Stop the axis
    ENDIF
  ENDIF
ENDIF

IF Inp02 AND NOT Inp01              ; If the Inp02 input
                                    ; is active and the Inp01
                                    ; input is deactivate
  IF AxisX:st_still                 ; If the axis is stopped
    AxisX:setvel=AxisX:maxvel/10    ; I set the speed of
                                    ; manual movement
    MANBW AxisX                     ; Backward manual
  ENDIF
ELSE                                ; Otherwise
  IF NOT Inp01                      ; If the Inp01 input is
                                    ; deactive
  IF NOT AxisX:st_still             ; If the axis is not stationary
     STOP AxisX                     ; Stop the axis
  ENDIF
ENDIF                               ; END

WAIT 1
JUMP MAIN
```

```
END
```

1.2.13 The sector structure

The device does not have inside datagroup or array data where you can contain various types of cams, so, if you need to manage different cams according to the type of work, you must place the CPU tools and download data on the device whenever there is a need.

Example:
This example handles the cam programming with data entered in the second program of a datagroup. The device is configured as described in the startup.

```
;-------------------------------------------------
; Configuration file
;-------------------------------------------------

;-------------------------------------------------
; Global Variables
;-------------------------------------------------

GLOBAL
gfProgram F ;Enabling cam programming

;-------------------------------------------------
; System Variables
;-------------------------------------------------

SYSTEM
sbPuntProg B ;Program number that you want to put into execution

;-------------------------------------------------
; Datagroup Variables
;-------------------------------------------------

DATAGROUP
dgCamma

DATAPROGRAM 10 ;10 available programs
ddlCode L ;program code

STEP 128 ;128 available program step
ddbCodeG B ;G code
ddlCodeQs L ;Qs code
ddlCodeQs L ;Qm code
ddlCodeM L ;M code
ddlCodeQma L ;auxiliary Qm code
ddlCodeQsa L ;auxiliary Qs code

;-------------------------------------------------
; Cam programming tasks
;-------------------------------------------------
MAIN:
   .
   .
   sbPuntProg = 2 ;Setting the program pointer
   .
   .

;-------------------------------------------------
; Programming of the CAMMING2 device
;-------------------------------------------------
IF gfProgram
   AxisX:codeG1     = ddbCodeG [sbPuntProg , 1] ;Sector 1
   AxisX:codeQm1    = ddlCodeQm [sbPuntProg , 1] ;Sector 1
   AxisX:codeQs1    = ddlCodeQs [sbPuntProg , 1] ;Sector 1
   AxisX:codeQma1   = ddlCodeQma [sbPuntProg , 1] ;Sector 1
   AxisX:codeQsa1   = ddlCodeQsa [sbPuntProg , 1] ;Sector 1
   AxisX:codeM1     = ddlCodeM [sbPuntProg , 1] ;Sector 1
   AxisX:codeG2     = ddbCodeG [sbPuntProg , 2] ;Sector 2
   AxisX:codeQm2    = ddlCodeQm [sbPuntProg , 2] ;Sector 2
   AxisX:codeQs2    = ddlCodeQs [sbPuntProg , 2] ;Sector 2
   AxisX:codeQma2   = ddlCodeQma [sbPuntProg , 2] ;Sector 2
   AxisX:codeQsa2   = ddlCodeQsa [sbPuntProg , 2] ;Sector 2
   AxisX:codeM2     = ddlCodeM [sbPuntProg , 2] ;Sector 2

   .
   .

   AxisX:codeG128 = ddbCodeG [sbPuntProg , 128] ;Sector 128
   AxisX:codeQm128 = ddlCodeQm [sbPuntProg , 128] ;Sector 128
   AxisX:codeQs128 = ddlCodeQs [sbPuntProg , 128] ;Sector 128
   AxisX:codeQma128 = ddlCodeQma [sbPuntProg , 128] ;Sector 128
   AxisX:codeQsa128 = ddlCodeQsa [sbPuntProg , 128] ;Sector 128
   AxisX:codeM128 = ddlCodeM [sbPuntProg , 128] ;Sector 128

   gfProgram = 0
ENDIF
```

1.3 The sectors

The CAMMING2 device manages cam sectors scheduled incremental, within which shows the space ahead from the master and the space to take the slave. A cam is composed of several sectors which may be accelerating, decelerating, or dedicated to operations such as speed change, for example, the power factor correction counter or loop cam.
Each sector of the cam must contain information about:

- codeG sector type
- codeQm master quota (Attention: insert only positive values)
- codeQs slave quota
- codeQma auxiliary master quota (Attention: insert only positive values)
- codeQsa auxiliary slave quota

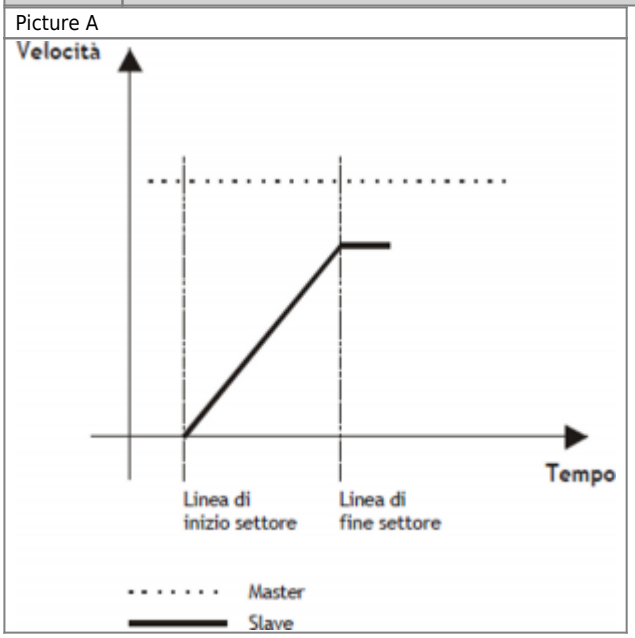
- codeM Code of general use, which is indicated by the codeMex variable. Typically contains the tools the state, cam special states, etc.

1.3.1 The sector of acceleration

The sector of acceleration is used with slave axis stopped (slave zero speed, regardless of the speed of the master); at the end of the sector the speed of the slave is the same as that of the master.
Typical cases of acceleration are shown in A, B, C and D pictures.
In the example of A picture, at the end of the sector to be equal to that of the speed of the slave master; the law that binds the space master and slave is:
Slave space = 1/2 Master space
More smaller will be the space master considered more greater will be the acceleration gradient of the slave, which we can derive from the formula:
Time acc. slave = Master space in the sector of acc. / Maximum master speed

In case you are facing this kind of acceleration we recommend using codeG = 132 code.

Picture A



Programming example:

- codeG 132
- codeQm Master space
- codeQs Slave space
- codeQma Not used
- codeQsa Nont used
- codeM generic code

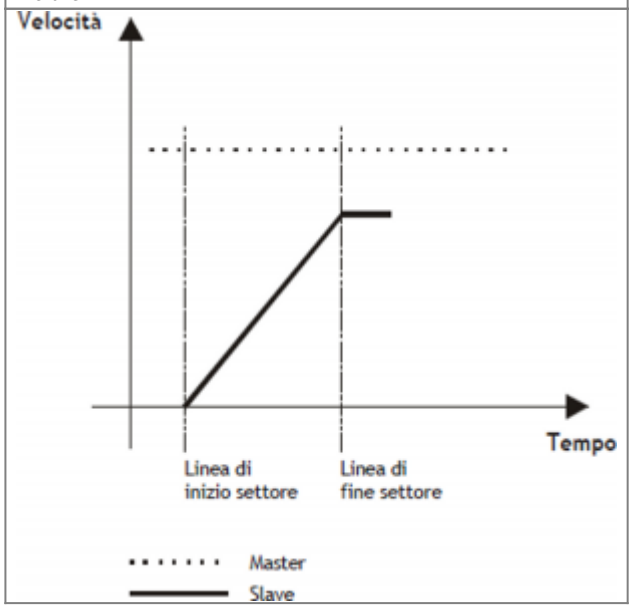
If you wish to use the epicycloidal ramps, we recommend the use of the codeG = 232 code.

In case you want to use the epicycloidal ramps to accelerate with the same functions described for 132 sector, simply program the sector as described above and planning *codeG* = 232.

In the example of B picture, at the end of the field the speed of the slave is in proportion to the speed of the master (the proportion will be called K), the law that binds the master and slave space space is:
Slave space = K/2 Master space
More smaller will be the space master who considers, more greater will be the acceleration gradient of the slave, that we can derive from the formula:
Time of slave acc. = Master space in the acc. sector / Maximum speed master

In case you are faced with this type of acceleration is required using codeG = 131 code.

Picture B



Programming example:

- codeG 131
- codeQm Master space
- codeQs Slave space
- codeQma Not used
- codeQsa Not used
- codeM generic code

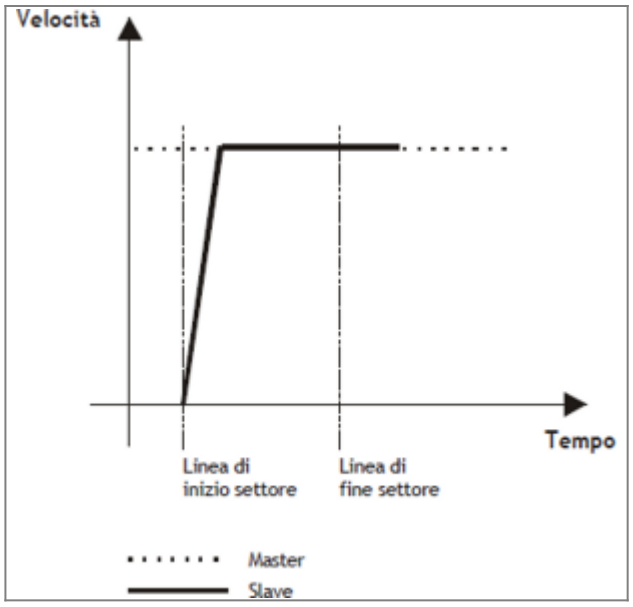
If you wish to use the epicycloidal ramps, we recommend using codeG = 231 code.

In case you want to use the epicloidal ramps to accelerate with the same functions described for 131, simply program the field as described above and planning *codeG* = 231.
In the picture C example, more acceleration are needed, and you cannot set Master/Slave quotes of finished value. 150 sector is basically the sum of two sectors: 131 and 133. This sector is used when you know the spaces next to the field of acceleration and you want one slave very small accelerative space, even less than the unit of measurement.
The sector 150 uses the following parameters:

- codeG : sector code (150)
- codeQma : master space within which the slave must be at a certain speed, which we call the synchronization.
- codeQm and codeQs : where the division indicates the relationship between the slave and master (synchronization report). These spaces will be made after accelerative section.
- **codeQsa : indicates the space into encoder pulses that must take the slave in the acceleration phase to reach the sync speed.**
  More smaller will be the space master who considers himself, more greater will be the acceleration gradient of the slave, who do we get from the formula:
  Time of Slave acc. = Master space on the acc. sector / Maximum master speed

In case you are faced with this type of acceleration is required using code codeG = 150.

Picture C



Programming example:

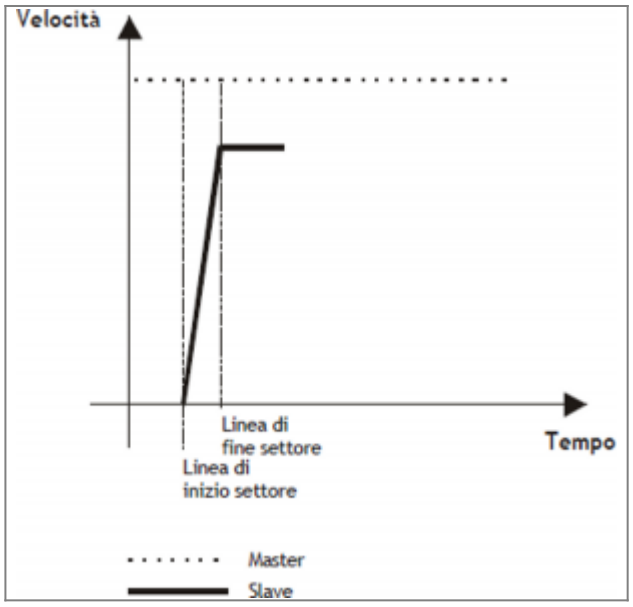
- codeG 150
- codeQm Master Space
- codeQs Slave Space
- codeQma Master Space in acceleration
- codeQsa Slave SPace in acceleration (bit * 4)
- codeM generic code

In the picture D example, more acceleration are needed, and you cannot set Master/Slave quotes of finished value. 152 sector is basically the same the sector 131. This sector is used when you know the synchronization report and you want one slave very small, even smaller than the space inertial measurement unit.
152 sector takes advantage of the following parameters:

- codeG : sector code (152)
- codeQma : master space within which the slave must be at a certain speed, which we call the synchronization.
- codeQm and codeQs : where the division indicates the relationship between the slave and master (synchronization report).
- **codeQsa : indicates the space into encoder pulses that must take the slave in the acceleration phase to reach the sync speed.**
  More smaller will be the space master who considers himself, more greater will be the acceleration gradient of the slave, who do we get from the formula:
  Time of Slave acc. = Master space on the acc. sector / Maximum master speed

	In case you are faced with this type of acceleration is required using code codeG = 152.

Picture D



Programming example:

- codeG 152
- codeQm Master Coefficient
- codeQs Slave Coefficient
- codeQma Master Space in acceleration
- codeQsa Slave Space in acceleration (bit * 4)
- codeM generic code

	If you wish to use the epicycloidal ramps, we recommend the use of the code codeG = 252.

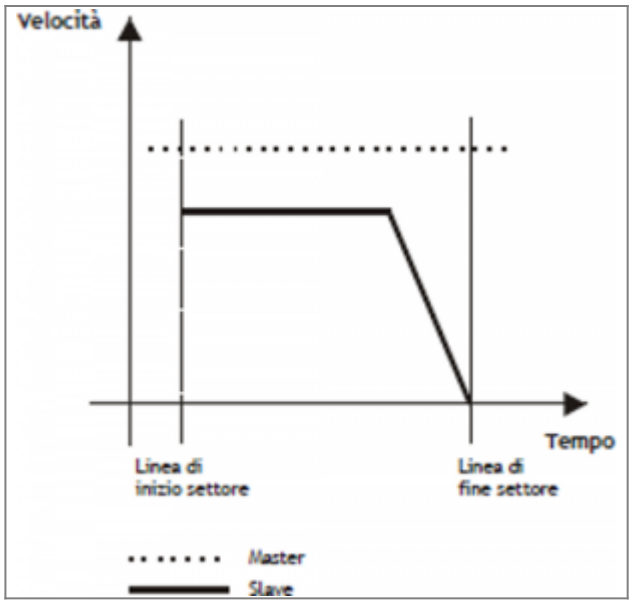
In case you want to use the epicycloidal ramps to accelerate with the same functions described for 152 sector, simply program the sector as described above and planning the code *codeG* = 252.

1.3.2 The deceleration sector

In case you need to stop the slave axis (regardless of its speed), remaining engaged with the cam (zero speed regardless of the speed of the master), it can be used the deceleration sector.
In the picture E example, at the end of the sector, the speed of the slave will be zero; la law that binds the space master and slave (the proportion between the master and the slave speed will be called K) is:
Slave space = K/2 Master space
More smaller is the space master who considers himself, more greater will be the degree of deceleration of the slave, which you can obtain from:
Time of Slave dec. = Master space in the dec. sector / Maximum master speed

	In case you find yourself in front of a deceleration is obligatory the use of the code codeG = 135.

Picture E



Programming example

- codeG 135
- codeQm Master space
- codeQs Slave space
- codeQma Not used
- codeQsa Not used
- codeM generic code

If you wish to use the epicycloidal ramps, we recommend the use of the code codeG = 235.

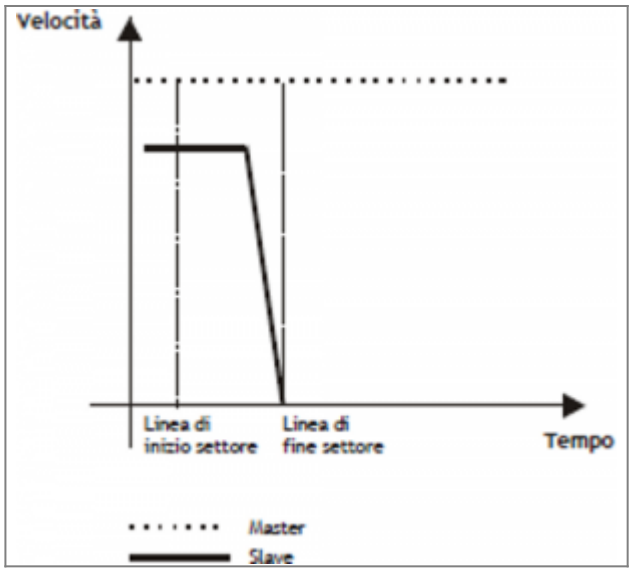
In case you want to use the epicycloidal ramps to accelerate with the same functions described for the sector 135, simply program the field as described above and planning *codeG* = 235.

In the picture F example, syou need strong deceleration, and you cannot set Master/Slave dimensions of finished value. The 151 sector is basically the sum of two sectors: 133 and 135. This sector is used when you know the spaces before the deceleration area and want a space decelerativo slave very small, even smaller than the unit of measurement.
151 sector makes use of the following parameters:

- codeG : sector code (151)
- codeQma : master space within which the slave must be from a certain speed, which we call the zero speed synchronization.
- codeQm and codeQs : where the division indicates the relationship between the slave and master (synchronization report). These spaces are made before decelerative section.
- codeQsa : indicates the space into encoder pulses that must cover the slave during deceleration.

More smaller is the space master who considers himself, more greater will be the degree of deceleration of the slave, which we can derive from the formula:
Time of Slave dec. = Master space in dec. sector / Maximum master speed

Picture F



Programming example

- codeG 151
- codeQm Master space
- codeQs Slave space
- codeQma Master space decelerating
- codeQsa Slave space in decelerating (bit * 4)
- codeM generic code

1.3.3 Speed change sector

In order to do this operation there are two types of codes (codeG = 133 and codeG = 134) which differ only in the choice of the speed that you want to give to the slave at the end of the speed change sector.

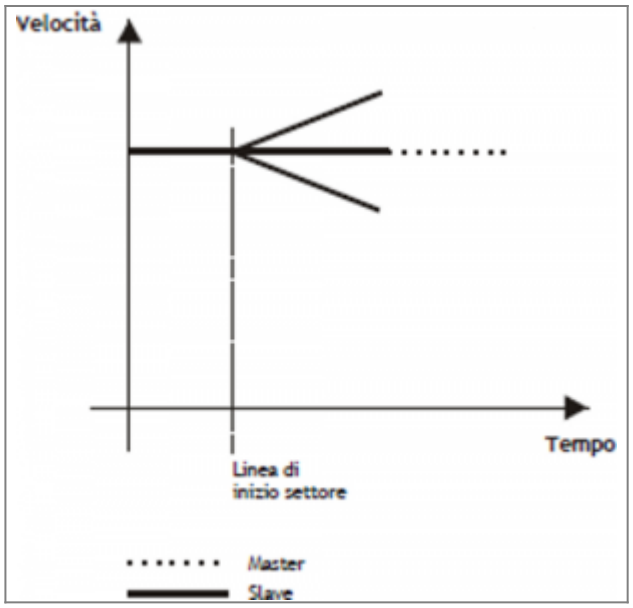
Speed change sector can be used:

- Whenever the slave axis must reach speeds (nonzero), starting at a different speed (also non-zero).
- Whenever the slave axis must maintain a constant speed.

In the example the speed of the slave is the same as that of the master (at the beginning of speed change sector). In case the speed is different you need to consider, in the follow formulas, the master and slave relationship of constant speed at the beginning of the sector.
The codeG = 133 provides that the speed of the slave at the end of the sector can be different from the initial and final speed of the slave (of end time sector), will depend exclusively on the master/slave relationship of spaces (see picture G).
We have three cases:
1)Master/slave relationship < 1 Speed of the slave at the end of the sector > of the master speed
2)Master/slave relationship = 1 Speed of the slave at the end of the sector = of the master speed
3)Master/slave relationship > 1 Speed of the slave at the end of the sector < of the master speed

The speed at the end of the sector is given by the formula:
Slave Speed = Master Speed + { [2 (Slave Space - Master Space) / Master Space] x 100 } %

Picture G



Programming example:

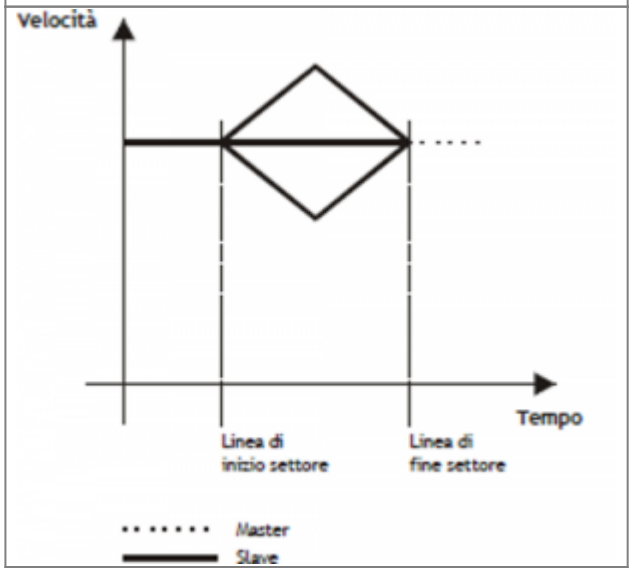
- codeG 133
- codeQm Master Space
- codeQs Slave Space
- codeQma Not used
- codeQsa Not used
- codeM generic code

If you wish to use the epicycloidal ramps, we recommend the use the code codeG = 233.

In case you want to use the epicycloidal ramps to accelerate with the same functions described for the sector 133, simply program the sector as described above and planning *codeG* = 233.
The codeG = 134 provides that the speed of the slave at the end of the sector is the same as the speed at half the slave sector will depend exclusively on the relationship of the master/slave spaces (see picture H). We have three cases:
1 Master/slave relationship < 1 Speed of the slave in the middle of the sector > of the master speed
2 Master/slave relationship = 1 Speed of the slave in the middle of the sector = of the master speed
3 Master/slave relationship > 1 Speed of the slave in the middle of the sector < of the master speed

The speed in the middle of the sector will be given by the formula:
Slave Speed = Master Speed + { [2 (Slave Space - Master Space) / Master Space] x 100 } % x (Master Speed)

Picture H



Programming example:

- codeG 134
- codeQm Master Space
- codeQs Slave Space
- codeQma Not used
- codeQsa Not used
- codeM generic code

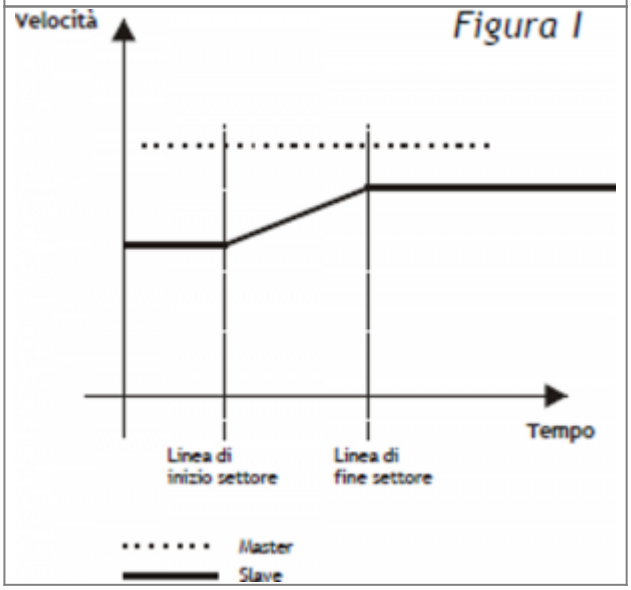
If you wish to use the epicycloidal ramps, we recommend the use the code codeG = 234.

In case you want to use the epicycloidal ramps to accelerate with the same functions described for the sector 134, simply program the sector as described above and planning *codeG* = 234.
If you are programming a sector 133, 134, 233 or 234 with master and slave space to 0, it is considered as a non-operating sector (codeG = 130).
In the picture I example, wants to change speed to the slave, and it is not possible to set a Master/Slave relationship of finite value. The sector 153 is the same as sector 133. This sector is used when you know the synchronization report and you want one slave very small inertial space, sometimes even lower than the unit.
The sector 153 uses the following parameters:

- codeG : sector code (153)
- codeQma : master space within which the slave must be at a certain speed, which we call the synchronization.
- codeQm and codeQs : whose division indicates the ratio between the slave and master (rapporto di sincronizzazione).
- codeQsa : the device indicates the space in which the slave route encoder pulses to reach the sync speed after acceleration phase.

In case you are using this type of transmission speed, we recommend the use of the code codeG = 153.

Picture I



Programming example:

- codeG 153
- codeQm Master Coefficient
- codeQs Slave Coefficient
- codeQma Space Master in acceleration
- codeQsa Space Slave in acceleration (bit * 4)
- codeM generic code

If you wish to use the epicycloidal ramps, we recommend the use the code codeG = 253.

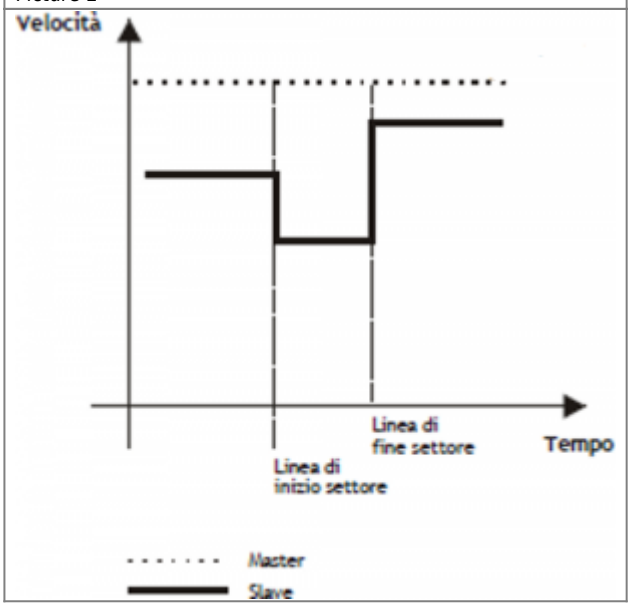
In case you want to use the epicycloidal ramps to accelerate with the same functions described for the sector 153, simply program the sector as described above and planning *codeG* = 253.

In the example of L picture, you want to bring the slave at a speed without having to run a connection ramp. The sector 154 unlike all other, forces the initial velocity equal to the final speed while maintaining a constant speed between two points. This sector can be used as the starting sector of the cam (starting without acceleration), as intermediate sector or final sector (stop without ramp).
The sector 154 uses the following parameters:

- codeG : sector code (154)
- codeQma : Softening sector type
- codeQm and codeQs : whose division indicates the relationship between the slave and master (synchronization relationship). These spaces are performed during the sector.
- codeQsa : if set to 0 indicates that the next sector is an area of movement, if set to 1 indicates that the next sector does not provide for movement (deceleration with zero ramp).

	In case you have to use this type of movement is obligatory the use of the code codeG = 154.

Picture L



Programming example:

- codeG 154
- codeQm Master Space
- codeQs Slave Space
- codeQma Softening sector type
- codeQsa
  0 = next sector of movement
  1 = stationary axis in the next sector
  2 = Gearing axis
- codeM generic code

1.3.4 The Start sector synchronized to the Master

Many times there is a need to start the slave on the master known, but there is the possibility to connect to a proximity sensor. The only constraint is that the sector containing the codeG 160 must be the first sector of cam motion and cannot be placed in the loop. At the STARTCAM command, the st_camex state goes to 1 and the the movement of the Slave axis begins only at Master quota (expressed in units of measurement) set in the sector 160 and from there will follow the trend described in the following sectors.
If the STARTCAM command is given with the above-quota master count, is set the warning 9; under these conditions the Master counting must become less than the quota set for it to be in the right situation of the system.
You can't get into a sector with codeG = 160 coming from a jump or a loop cam (error 7).

Programming example:

- codeG 160
- codeQm STARTCAMMA quota expressed in units of measurement
- codeQs Not used
- codeQma Not used
- codeQsa Not used
- codeM Not used

1.3.5 The end cam sector

The exchange end cam sector (codeG = 136), is used whenever one has to conclude the cam (disengage the cam) stopping the slave axis in reaction to space on the last point of the cam. The slave axis must be stopped at the time of the release of the cam, pwhereby it is assumed that the previous sector contains the code of the deceleration (codeG = 135).
After this area the cam is released and, to reattach it, you have to send the *STARTCAM* command.

Programming example:

- codeG 136
- codeQm Not used
- codeQs Not used
- codeQma Not used
- codeQsa Not used
- codeM Not used

1.3.6 Absolute jump sector

The absolute jump sector (codeG = 137), is used to jump to a sector (defined in the codeQm) in order to change the fly cam performance according to the conditions established by the programmer.
The most common situation for using this feature is that a portion of the cam that needs to be repeated several times.
Keep in mind that the counts are not updated, and then in the long run can go into overflow. Therefore, use the update count fields in the field that precedes the one containing the codeG = 137.

Programming example:

- codeG 137
- codeQm Sector number to jump
- codeQs Not used
- codeQma Not used
- codeQsa Not used
- codeM Not used

1.3.7 Conditional jump sector

The conditional jump sector (codeG = 190), is used to make a jump to a sector (defined in the codeQm) for a number of times (defined in the codeQs) after that you move on to the next sector. The count of the number of the jumps performed is available in codeQma.
Care must be taken that the counts are not updated, and then in the long run can go into overflow. Therefore, use the update count sector in the field that precedes the one containing the codeG = 190.

Programming example:

- codeG 190
- codeQm Sector number to jump
- codeQs Number of times
- codeQma Jump number viewing
- codeQsa Not used
- codeM Not used

1.3.8 Loop cam sector

The loop sector cam (codeG = 138), is used to repeat the cam running from the sector number one, eliminating by removing both master and slave counts.

We recommend using this code in repeated endlessly cams that have no problems with subtraction of counts.

Programming example:

- codeG 138
- codeQm Not used
- codeQs Not used
- codeQma Not used
- codeQsa Not used
- codeM Not used

1.3.9 Not operative sector

The not operative sector (codeG = 130), is used to reserve areas to perform functions only under special conditions defined by the programmer.
For example you may consider a cam for the flying saw, in which you have to reserve the sectors to be used in case, mechanically, you cannot make the cut master space reserved for this operation.

Programming example:

- codeG 130
- codeQm Not used
- codeQs Not used
- codeQma Not used
- codeQsa Not used
- codeM Not used

1.3.10 Definition of area sampled zero sectors

All sectors that require no space master to be processed are called “zero-sampling”; specifically, are all NOP, JUMP, LOOP and END sectors.
A sector zero sampling is also considered the codeG = 133 se programmato come:

- codeG = 133
- codeQm = 0
- codeQs = 0

It is not possible to sequence more than 9 zero sampling areas.

1.3.11 Update count sectors

The update count is used to make an exchange of the count, to values that may indicate the actual physical location of the axis. The most typical case is the circular axis (from 0° to 360°): whenever you reach 360° you must subtract a circle. To make an update count there are multiple codes of subtraction or bit encoder count setting, whether that unit of measure. For how it is structured the device, it is not possible to sequence more than 4 sectors count update. The following table containing the description of what happens during the update count based on the code used.

codeG Execute operations
139 Subtraction by count master value contained in the codeQm (expressed in units of measurement).
Subtraction from the slave count value contained in codeQs (expressed in units of measurement).
140 Forcing master count to the value that is contained in codeQm (expressed in units of measurement).
141 Forcing the slave to the count value in codeQs (expressed in units of measurement).
142 Forcing master count to the value that is contained in codeQm (expressed in units of measurement).
Forcing the slave to the count value in codeQs (expressed in units of measurement).
143 Subtracting the value contained in the master count codeQm (expressed in bit encoder multiplied by 4).
Subtracting the slave count value contained in codeQs (expressed in bit encoder multiplied by 4).
144 Forcing master count to the value that is contained in codeQm (expressed in bit encoder multiplied by 4).
145 Forcing the slave to the count value in codeQs (expressed in bit encoder multiplied by 4).
146 Forcing master count to the value that is contained in codeQm (expressed in bit encoder multiplied by 4).
Forcing the slave to the count value in codeQs (expressed in bit encoder multiplied by 4).

1.3.12 Cam sectors description

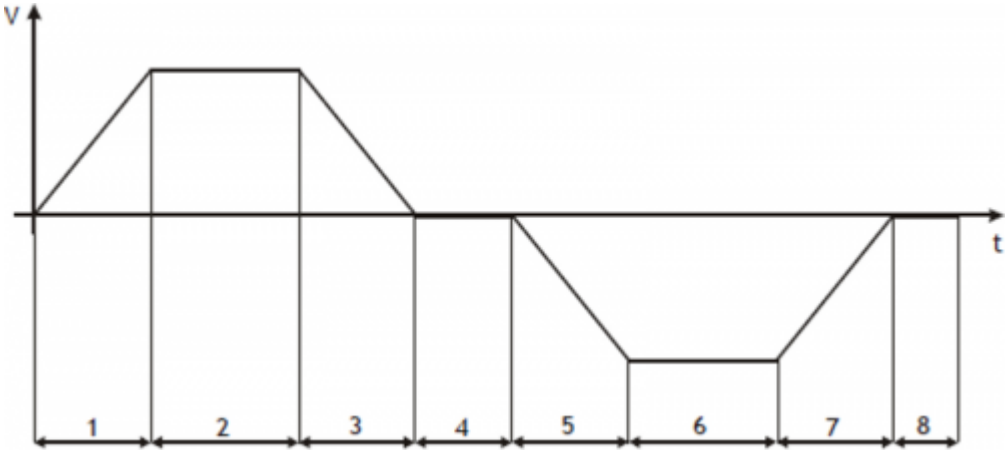
CodeG	codeQm	codeQs	codeQma	codeQsa	codeM	Description
130	n.u.	n.u.	n.u.	n.u.	n.u.	NOP: Disabled sector (not operative)
131	Increase Master (Um)	Increase Slave (Um)	n.u.	n.u.	c.u.	AZL: Sector of acceleration with zero initial speed and final speed calculated according to slave space to go. Final slave speed = f (slave space).
132	Increase Master (Um)	Increase Slave (Um)	n.u.	n.u.	c.u.	AZM: Sector of acceleration with zero initial speed and final velocity equal to that of the master (final slave speed = master speed), by varying the degree of acceleration.
133	Increase Master (Um)	Increase Slave (Um)	n.u.	n.u.	c.u.	RSC: Intermediate sector (fitting without compensation) with speed velocity equal to the final speed of the previous sector and final speed calculated according to slave space to go.
134	Increase Master (Um)	Increase Slave (Um)	n.u.	n.u.	c.u.	RCC: Intermediate sector (fitting with compensation) with initial and final speed equal to the final speed of the previous sector: this is obtained by a slave space clearing, dividing into two phases (acceleration and deceleration) the execution of the sector.
135	Increase Master (Um)	Increase Slave (Um)	n.u.	n.u.	c.u.	DZC: Sector of deceleration with initial speed equal to the final speed of the previous sector and final speed equal to zero: this is obtained by a slave space clearing, dividing into two phases the execution of the sector.
136	n.u.	n.u.	n.u.	n.u.	n.u.	END: End cam. The system releases the running cam and remains in the reaction of space with the slave on the last position elaborated by the previous sector.
137	Sector number to which jump	n.u.	n.u.	n.u.	n.u.	ABJ: Absolute jump. The system maintains the position and speed of the last sector used. The counters do not vary. The number of the cam that I miss all indicated in codeQm and must be between 1 and 128.
138	n.u.	n.u.	n.u.	n.u.	n.u.	LOOP: Loop cam. When this instruction is encountered resumes processing industries from the first, keeping them as speed than the last sector tried and subtracting the count of the amount of space done up to that point.
139	subtraction Master count value (Um)	Subtraction Slave count value (Um)	n.u.	n.u.	n.u.	SMS: Subtract the counts in units. Is subtracted from the count of the Master the value contained in the codeQm and the Slave count the value contained in the codeQs (subtraction Master and Slave counting in units of measurement)
140	New Master counter (Um)	n.u.	n.u.	n.u.	n.u.	NCM: Change master counter. Writes the value contained in the codeQm in the Master counter. The updating is done by subtraction (updates the Master count unit of measure)
141	n.u.	New Slave counter (Um)	n.u.	n.u.	n.u.	NCS: Change Slave count. Writes the value in codeQs in the counting of the Slave. The updating is done by subtraction (updates the slave counter in unit of measure)
142	New Master counter (Um)	New Slave counter (Um)	n.u.	n.u.	n.u.	NMS: Change counters. Master and Slave counts are written out with the values found respectively at codeQm and codeQs (updates the master and slave counters in unit of measure).
143	The subtraction master counter value (bit*4)	The subtraction slave counter value (bit*4)	n.u.	n.u.	n.u.	SBMS: Subtract the Master and Slave bit counts. Subtracts the value in Master codeQm counter and the counter of the Slave the value in codeQs (master and slave bit count x 4 subtraction4).
144	New Master counter (bit*4)	n.u.	n.u.	n.u.	n.u.	NBM: Change Master counter in bit. This statement writes the value in codeQm in the counting of the Master. The updating is done by subtraction (updates the master counter in bit x 4).
145	n.u.	New Slave counter (bit*4)	n.u.	n.u.	n.u.	NBS: Change Slave counter in bit. This statement writes the value in codeQs in the counter of the Slave. The updating is done by subtraction (update the slave counter in bit x 4).
146	New Master counter (bit*4)	New Slave Counter (bit*4)	n.u.	n.u.	n.u.	NBMS: Change Master and Slave counters in bit. This statement updates the Master and Slave counters with the values found respectively at codeQm and codeQs (update the master and slave counter in bit x 4)
150	Master Increase (Um)	Slave increase (Um)	Master Space in acceleration (Um)	Slave Space in deceleration (bit*4)	c.u.	AZMC: Acceleration sector with zero initial speed and final speed calculated according to Master and Slave space indicated in codeQm and codeQs. The acceleration is performed in the space indicated in codeQma and codeQsa. Run the spaces indicated in codeQm and codeQs with the law described in the codeG 133.
151	Master Increase (Um)	Slave Increase (Um)	Master Space in deceleration (Um)	Slave Space in deceleration (bit*4)	c.u.	DZMC: Deceleration sector with initial speed equal to the final speed of the previous sector and final speed equal to zero. The deceleration is performed in the space indicated in codeQma and codeQsa. Run the spaces indicated in codeQm and codeQs with the law described in the codeG 133.

CodeG	codeQm	codeQs	codeQma	codeQsa	codeM	Description
152	Master Coefficent	Slave Coefficent	Master Space in acceleration (Um)	Slave Space in acceleration (bit*4)	c.u.	AZMS: Acceleration sector with zero initial speed and final speed calculated according to Master and Slave coefficients indicated in codeQm and codeQs. The acceleration is performed in the space indicated in codeQma and codeQsa. Do not run the spaces indicated in codeQm and codeQs
153	Master Coefficent	Slave Coefficent	Master Space in speed change (Um)	Slave Space in speed change (bit*4)	c.u.	NVSR: Change speed on ramps: the Slave axis moves from the current speed at the speed calculated according to Master and Slave coefficients indicated in codeQm and codeQs. The speed change uns in space indicated in codeQma and codeQsa. Do not run the spaces indicated in codeQm and codeQs
154	Master Inrease (Um)	Slave Increased (Um)	Type of softening	Sector type	c.u.	NVS: Change speed without ramp. The Slave axis moves from the current speed at the speed calculated according to master and slave spaces listed in codeQm and codeQs without ramp (run the step). In the codeQsa indicates whether this is the last sector (by setting 1 indicates that the next time the slave axis is stopped) or if the movement continues (setting 0 indicates that the next time the slave axis is of movement) setting to 2 on codiceQsa you can use the axis as GEARING. Once you have set the codeQm and codeQs so you get the MASTER/SLAVE speed ratio. The new R.V. is obtained without ramp so if you want a smooth change you have to change the codeQs. Changing the codeQs (back to 0 or 1) you move on to the next sector (If you haven't planned any sector later to stop the device simply give one Stopcam otherwise you run into an error). N.B. During this last feature parameters posit and positm lose their meaning given that remain fixed at a value corresponding to half of the spaces planned in codeQm and codeQs.
160	Master Quota (Um)	n.u.	n.u.	n.u.	c.u.	STS: Synchronized Start to the STARTCAM expects the Master axis exceeds the height indicated in codeQm to move to the next sector. Sectors earlier than this should not be about movement and this code cannot be placed in the loop cam.

1.3.13 Basics for building a cam to wire-guides

As an example, consider a simple wire-guides:

- Starting with acceleration ramp.
- Achieving a speed proportional to that of the master.
- Maintenance of speed for the entire journey.
- Stop with deceleration ramp.
- Stop the axis for some space of the master.
- Return to the starting point with the same stroke mode.



Sector 1 Acceleration, from zero speed and positive shift slave (codeG = 131). It is important to calculate the ratio of space master/slave of this section so that the output speed is the one that will be maintained by the slave axis in the stretch at constant speed.

Sector 2 Intermediate with constant speed and positive shift slave (codeG = 133).

Sector 3 Deceleration with final zero speed, with a possible braking speed compensation in the first half of the stroke and the slave positive displacement (codeG = 135). May have the same values set in the field 1.

Sector 4 Stop working with slave zero shift (codeG = 133). Programming master space while the slave is set to 0.

Sector 5 Acceleration, from zero speed and negative shift slave (codeG = 131). It is important to calculate the ratio of space master/slave of this section so that the output speed is the one that will be maintained by the slave axis in the stretch at constant speed. Ideally, you can use the same values entered in the sector 1 changing the quota sign slave.

Sector 6 Intermediate with constant speed and negative shift slave (codeG = 133).

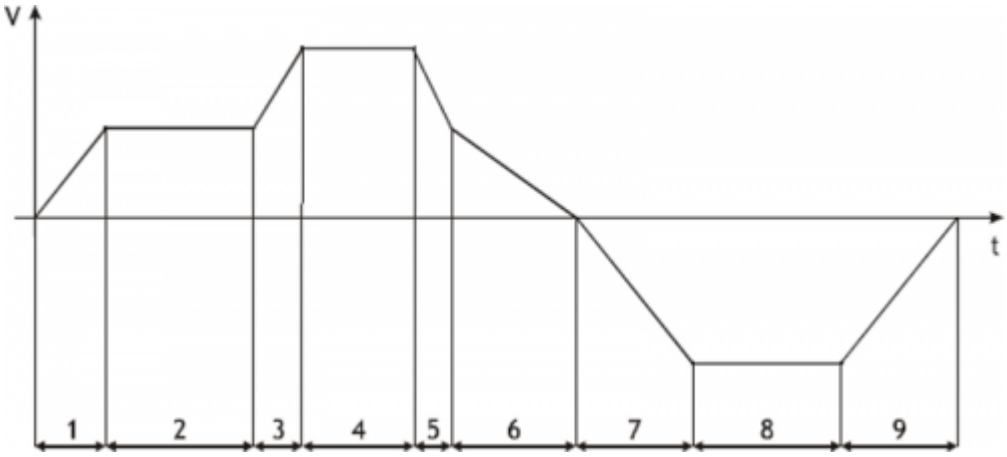
Sector 7 Deceleration with final zero speed, with a possible braking speed compensation in the first half of the stroke and negative shift slave (codeG = 135). May have the same values set in the sector 5.

Sector 8 Stop working with slave zero shift (codeG = 133). Programming master space while the slave is set to 0. After you run the sector 8, there must be some functions that perform power factor correction of Master and Slave counters by subtracting the space covered until the end of the sector; then you will have to have the automatic replay same cam from sector 1 (JUMP or loop cam).

1.3.14 Basics for building a cam for fly cut with extra speed

As an example, consider a simple fly cut:

- Start slave axis with acceleration ramp.
- Achievement of the master speeed.
- Maintaining the speed reached throughout the cut.
- Finished cutting, the slave axis must accelerate to go to go to an extra speed, keeping it for a certain space.
- Stop slave axis with deceleration ramp.
- Return of the slave axis to the starting point (home), without inversion time and following the acceleration and deceleration ramps.



Sector 1 Acceleration, with starting from zero speed and positive shift slave (codeG = 132). At the end of this sector the slave will have the same speed of the master.

Sector 2 Intermediate with constant speed and positive shift slave (codeG = 133). In this area the space covered by the master is the same as the path from the slave.

Sector 3 Positive acceleration and displacement slave (codeG = 133). The code set rule slave acceleration than the master, it sets a space more than the master.

Sector 4 Intermediate with constant speed and positive shift slave (codeG = 133). In this area the space covered by the slave will be proportion to that route from the master.

Sector 5 Deceleration and positive shift slave (codeG = 133). In this sector takes the slave at the same speed as the master.

Sector 6 Deceleration with final zero speed, with a possible braking speed compensation in the first half of the stroke and the slave positive displacement (codeG = 135).

Sector 7 Acceleration, with starting from zero speed and negative shift slave (codeG = 131). In this area the output speed of the slave may differ from that of the master.

Sector 8 Intermediate with constant speed and negative shift slave (codeG = 133).

Sector 9 Deceleration with final zero speed, with a possible braking speed compensation in the first half of the stroke and negative shift slave (codeG = 135).

After you run the sector 9, there must be a function that performs the power factor of the counter of the Master, subtracting the space covered until the end of the field and, subsequently, automatically resubmitting the same cam (JUMP or loop cam).

1.4 Device errors management

A bug in the system camming is reported by the *st_error* state.
Being caused by a serious event and not being guaranteed in this situation the slave axis management, It was decided arbitrarily to block the axis without ramps as had taken place an emergency.
When *st_error* is equal to 1, are present on the *errcode* variable the type of error occurred (see the table) and in *errvalue* variable an indication on the cause of the error.

Code	Priority	Description
1	0	Too many consecutive invalid sampling areas
2	0	JUMP from one sector with final nonzero speed on a sector with zero initial speed (acceleration).
3	0	G code of the invalid sector.
4	0	Master space of the cam sector too small, so the sector is not calculated.
5	0	Attempted to write in the sector running.
6	0	Into the JUMP code, was required to go to a row between 1 and 128.
7	0	Sector with codeG = 160 don't run at the beginning of the cam.

If the device goes in error, in order to start cut you have to clear the *st_error* status through *RSERR* command and then the usual routine of emergency recovery (*RESUME* axis).

NOTE: Error 4 is due to the fact that the sector is run in a time less than the sampling time of the device, so it cannot be tried. If you are in this situation you have to increase the share of the master in the sector, or lower the speed of the master.

1.5 Device warning management

The presence of a warning system camming is signaled by the *st_warning* state.
Being caused by a minor event and being guaranteed in this situation the slave axis management, the slave axis continues his work.
When *st_warning* is equal to 1, are present on the *wrncode* variable the type of warning intervened (see the table) and in the *wrnvalue* variable the sector number of the cam that caused the warning.

Code	Priority	Description
1	6	Constant acceleration sector greater than that programmed.
2	7	Constant deceleration sector greater than that programmed.
3	4	Slave to + 10V analog saturation (with latching)
4	5	Saturation of the slave - 10V analog (with latching)
5	9	Final speed of opposite sign to the initial speed.
6	2	Met a new sector of acceleration when the cam comes from an area with non-zero final speed.
7	8	Medium speed of opposite sign to the initial speed.
8	0	Event captured from the interrupt input but not processed immediately due to overload in the calculations of the device.
9	1	Starting quota Slave axis with codeG = 160 already outdated

The highest priority is marked from 0, the lowest with 8.
To clear the st_warning status must send the RSWRN command.
NOTE: In case of warning 8, the function will be delayed long enough to allow the CPU to terminate the internal calculations. In the case of cam start from the interrupt input, the startup location cam can not be that of the moment of interrupt, but that after the end of the calculations. The execution time of calculations (expressed as sampling time of the device), is shown in the following table:

Parameters involving recalculations	N.r samples that are distributed the consequential recalculations
codeG, codeQs, codeQm,codeQsa, codeQma, maxpos, minpos,prspos, prsposm,toll, tacc, tdec, taccmax, tdecmax, syncrange, pgain, feedfw, integt, derivt	1
tbfm	2
tbf	3
maxvel	5
decpt, unitvel	6
pulsem, measurem	42
pulse, measure	51

1.6 Simulated master management

The encoder device master of CAMMING2 is in no way linked to the encoder EANPOS device.

The device CAMMING2 can handle two types of master:

- Both may be coming from an encoder mechanically connected to the master system and electrically connected to the system QMOVE or simulated encoder. It also accepted the mixed solution (one connected electrically and one simulated). The exchange between the two encoder is done through the parameter mtype without any constraint, so that even in the execution of a cam, you can make the exchange between devices. On your system using the device CAMMING2 can be declared a simulated encoder using a device of movement (for example the EANPOS) declared with the counter on slot 1 (normally reserved to the CPU of the system) and any other disable devices:

```
;-----------------------------
; Internal Device declaration
;-----------------------------

INTDEVICE
<device_name>   EANPOS   TCamp   ICont   IntL   IAZero   IOutA
Master          EANPOS   2       1.CNT01 X      X.X      X.X
```

where:

<device name>	The name assigned to the device.
EANPOS	Keyword that identifies the device analog positioner.
TCamp	Sample time device (1÷255 ms).
ICont	Bidirectional counter input.
IntL	Number of the interrupt line dedicated to the encoder zero pulse during the research phase of presets.
IAZero	Enable input to acquire the transducer's zero pulse during the quest activity phase of presets.
IOutA	Hardware address of the DAC from analog output (must be stated as X. X).

The device thus configured as a simulated master and is parameterized and used like a normal device bearing in mind that the control loops must be open (*st_loopon* = 0) and therefore don't need to parameterize the P.I.D. but just set the feedforward to 100% (*feedfw* = 1000).

1.6.1 Programming example

We use the EANPOS device configured as in the previous example, and you want to give the set of speed (setvel) expressed in Hz. It is assumed that the master simulated should continue its movement ad infinitum.
The sf01 flag for the start and stop simulated device.

```
;-----------------------------------------------------
; Simulated master management
;-----------------------------------------------------

Master:measure = 1000
Master:pulse = 4000
Master:decpt = 0
Master:unitvel = 1
Master:maxvel = 1000
Master:taccdec = 100
Master:maxpos = 999999
Master:minpos = -999999
INIT Master
WAIT Master:st_init
LOOPOFF Master
WAIT NOT Master:st_loopon
RESUME Master
WAIT NOT Master:st_emrg

MAIN:
IF sf01
  IF Master: st_still
    Master:posit = 0
    Master:setvel = 500
    Master:setpos = 999999
    START Master
  ENDIF
  IF Master:posit GE 500000
    Master:posit = 0
  ENDIF
ELSE
  IF NOT Master:st_still
    STOP Master
  ENDIF
ENDIF

WAIT 1
JUMP MAIN
END
```

1.7 M/S Transducer frequency ratio limitation

To have a proper functioning during synchronisation, It is required that the pulses in the time (frequency) generated from ther

Master transducer are greater or equal to those of the Slave axis. In any case you are required to comply with the condition

Slave frequency = 1,5 × Master frequency

In the case of a failure to comply with this condition will cause trouble in the calibration of the Slave axis in synch because of a roughness in the movement.

1.8 Inputs configuration table

The device has the ability to manage a normal entrance and an interrupt input to run commands or perform actions. The address of the inputs can be setting in the configuration file (InG and InGInt). To perform a specific function at the entrance, simply assign at the funInp variable (If this is the normal input) or funInt variable (If this is the interrupt input) the code listed in the following table.

Code	Input function
00	Input disabled
01	STOPCAM
02	STARTCAM
03	Writes the value of the encoder in the delta1 variable
04	Writes the value of the encoderm in the delta2 variable
05	Increments of 1 the delta1 variable
06	Increments of 1 the delta2 variable
07	Writes the contents of the delta1 variable in encoder
08	Writes the contents of the delta2 variable in encoderm
09	Writes the value of the encoder in the delta1 variable + STARTCAM

All functions of the inputs can be handled either on normal inputs on interrupt inputs.
To have a correct operation of the inputs, they are enabled by respecting the conditions set out in the description of the command or action described.

1.9 Outputs configuration table

The device has the ability to handle an exit to signal certain states. The address of the output can be setting in the configuration file (Out). To perform a specific function on output, simply assign the variable funOut code shown in the following table.

Code	Output function
00	Output disable
01	Disabling output
02	Enable output
03	st_toll
04	st_tpos
05	st_sync
06	It activates output only if codeMex is equal to the value 1000
07	It activates output only if codeMex is equal to the value 1000 and st_sync is active
08	It activates output only if codeMex is equal to the value 1001
09	It activates output only if codeMex is equal to 1002

1.10 Table commands, states and parameters: Symbols used

The parameter name, condition or command is taken back to the left of the table

R

Indicates whether its parameter or state is retentive (upon initialization of the device maintains the previously defined), or the state assumes upon initialization of the device.
R = Retentive
0 = Upon initialization of the device the value is forced to zero.
1 = Upon initialization of the device the value is forced to one.

D

Indicates the size of the parameter.
F = Flag
B = Byte
W = Word

L = Long

1.10.1 Conditions

Describes all the conditions necessary so that the parameter is considered correct or because the command is accepted.
In some cases, limit values are specified for the acceptance of the parameter: If any values outside the limits set, the data is anyway accepted; so they should be provided appropriate internal controls to ensure the proper functioning. To run a command, all the conditions must be met; otherwise the command does not run.

A

Indicates the access mode.
R = Read.
W = Write.

1.10.2 Parameters

Name	D	Write conditions	R	A	Description
decpt	B	st_still = 1 st_camex = 0 st_prson = 0	R	RdWr	**Decimal point** (0÷3) Defines the accuracy with which you wish to set the presets and view the counts relating to the slave axis
measure	L	st_still = 1 st_camex = 0 st_prson = 0	R	RdWr	**Measure** (1÷999999) Indicates the space, in units, location from the slave axis to get the encoder pulses that are set in the *pulse* parameter. This parameter is used to calculate the resolution with the formula: Resolution = *measure** 4 / *pulse* The resolution must have a value between 0.00374 and 4.00000
pulse	L	st_still = 1 st_camex = 0 st_prson = 0	R	RdWr	**Pulse encoder** (1÷999999) Indicates the pulses supplied by the encoder slave to get 4 times the space set in *measure* parameter. This parameter is used to calculate the resolution with the formula: Resolution = *measure** 4 / *pulse* The resolution must have a value between 0.00374 and 4.00000
measurem	L	st_still = 1 st_camex = 0 st_prson = 0	R	RdWr	**Measure of master** (1÷999999) Indicates the space, in units, location from the master axis to get the encoder pulses set in *pulsem* parameter. This parameter is used to calculate the resolution with the formula: Resolution = *measurem* * 4 / *pulsem* The resolution must have a value between 0.00374 and 4.00000
pulsem	L	st_still = 1 st_camex = 0 st_prson = 0	R	RdWr	**Pulse encoder of master** (1÷999999) Indicates the pulses supplied by the encoder master to get 4 times the space set in *measurem* parameter. This parameter is used to calculate the resolution with the formula: Resolution = *measurem** 4 / *pulsem* The resolution must have a value between 0.00374 and 4.00000
unitvel	B	st_still = 1 st_camex = 0 st_prson = 0	R	RdWr	**Velocity unit** (0÷1) Defines if the unit of time of the speed of the slave is expressed in minutes or seconds. 0 = Um/min 1 = Um/sec
maxvel	L	st_still = 1 st_camex = 0 st_prson = 0	R	RdWr	**Max velocity** (0÷999999) Defines the maximum speed of the slave axis (analogue reference of +/-10V). The input value is per unit of time of the speed set in the unitvel parameter.
prsvel	L	st_prson = 0	R	RdWr	**Preset velocity** (0÷maxvel) Defines the speed of the slave axis during the search procedure of presets. The input value is per unit of time of the speed set in the unitvel parameter.
sprsvel	L	st_prson = 0	R	RdWr	**Preset velocity** (0÷maxvel) Defines the speed of the slave axis during the search procedure of presets. The input value is per unit of time of the speed set in the unitvel parameter.
taccmax	W	st_prson = 0	R	RdWr	**Search preset velocity** (0÷prsvel) In the slave preset search procedure, defines the speed of the axis in the acquisition phase of zero-pulse. The input value is per unit of time of the speed set in the unitvel parameter.

Name	D	Write conditions	R	A	Description
tdecmax	W	st_prson = 0	R	RdWr	**Max deceleration time** (0÷999) Used during execution of the cam to do the comparisons on gradient of maximum deceleration. Defines the minimum time of deceleration by which the slave axis can be taken from maximum speed to axis stopped (speed equal to 0). The input value is expressed in hundredths of a second.
tacc	W	No condition	R	RdWr	**Acceleration time** (0÷999) Defines the time that slave axis to move from standstill to maximum speed. The input value is expressed in hundredths of a second. If the axis is moving (st_still = 0) you can change the gradients of the ramp only if the new values to the quota set.
tdec	W	No condition	R	RdWr	**Deceleration time** (0÷999) Defines the time necessary for the slave axis to decelerate from maximum speed to zero (condition of axis stopped). The input value is expressed in hundredths of a second. If the axis is moving (st_still = 0) You can change the gradients of the ramp only if the new values to the quota set.
maxpos	L	st_still = 0	R	RdWr	**Max position** (-999999÷999999) Defines the maximum quota reached by the slave axis. This limit is not checked during the execution of the cam.
minpos	L	st_still = 0	R	RdWr	**Min position** (-999999÷999999) Defines the minimum quota to reach from the slave axis. This limit is not checked during the execution of the cam.
prspos	L	st_still = 0	R	RdWr	**Preset position** (minpos÷maxpos) Defines the value that is loaded on slave with the preset search procedure.
prsposm	L	st_prsonm = 0	R	RdWr	**Preset position of master** (-999999 ÷999999) Defines the value that is loaded on the counting master with the preset search procedure.
toll	L	st_still = 0	R	RdWr	**Tolerance** (0÷999999) Defines a count range around locating dimensions of the slave axis. If the placement (not the arrival in cam) ends within this range, it is considered correct and is reported through the st_toll status.
maxfollerr	L	No condition	R	RdWr	**Maximum following error** (0÷2 31-1) Defines the maximum acceptable deviation between the theoretical position and the actual position of the slave axis. The input value is expressed in bits 4 transducer.
syncrange	L	No condition	R	RdWr	**Synchronism range** (0÷999999) Is the value expressed in units in which is indicated the timing slave (st_sync = 1) compared to the master during the execution of the cam.
prsmode	B	st_prson = 0	R	RdWr	**Preset mode** (0÷2) Defines the type of the slave preset search: **0** = For research enabling zero-pulse, the axis movement begins in, meet the enable signal, reverses direction slowing and, on falling edge relative to the signal of the slave axis, load the preset quota. **1** = For research enabling zero-pulse, the axis movement begins in, meet the enable signal, reverses direction and slowly acquires the first zero-pulse (after disabling the enable input of the slave axis). **2** =Does not activate the search procedure presets (st_prson = 0). The count is updated to the presets to the activation of the enabling zero-pulse of the slave axis.
prsmodem	B	st_prsonm = 0	R	RdWr	**Preset mode of master** (0÷2) Defines the type of preset search master: **0** = If st_prsonm = 1, the count is updated to preset the deactivation of enabling zero-pulse of the master axis. **1** = If st_prsonm = 1, the count is updated to preset quota when the zero-pulse after disabling the enable zero-pulse of the master axis. **2** =Does not activate the presets search procedure (st_prsonm = 0). The count is updated to the presets to the activation of the enabling zero-pulse of the master axis.
prsdir	B	st_prson = 0	R	RdWr	**Preset search direction** (0÷1) Defines the direction of movement research axis of the slave axis zero pulse enable switches. **0** = the axis goes forward. **1** = the axis go back.

Name	D	Write conditions	R	A	Description
mtype	B	No condition	R	RdWr	**Master type** (0÷1) Defines the address of the master used: **0** = The master is the encoder with address “A”. **1** = The master is the encoder with address “B”. (See chapter “Master simulated manager”)
ramptype	B	st_still = 0	R	RdWr	**Ramp type of slave** (0÷1) Defines the type of slave ramps used in regular placements; in the execution of the cam fittings are always executed with trapezoidal ramps: **0** = trapezoidal ramps. **1** = epicycloidal ramps. (See chapter “Trapezoidal motion Description”)
rtype	B	No condition	R	RdWr	**Riduction profile type** (0÷1) Defines the type of positioning the slave axis profile reduction if you selected flights of epicycloidal type (ramptype = 1). **0** = Acceleration and deceleration times remain d and l those of the set speed and the speed is decreased proportionally. **1** = Acceleration and deceleration times are decreased (maintaining the gradient acceleration and deceleration set) and also the same speed. (See chapter “Description trapezoidal motion”)
stopt	B	No condition	R	RdWr	**Stop type** (0÷1) Defines the type of braking that is used if stop slave axis positioning if you have selected flights of epicycloidal type (ramptype = 1). **0** = When you run a braking ramp is first completed the acceleration ramp and then runs the deceleration ramp. **1** = When such a braking ramp is immediately executed the deceleration ramp. (see chapter “trapezoidal motion description”)
pgain	W	No condition	R	RdWr	**Proportional gain** (0÷32767) Is set the value of 1000, the coefficient is 1,000 Is the coefficient that is multiplied against the tracking error raises the proportion of the control output of the slave axis. (see chapter)
feedfw	W	No condition	R	RdWr	**Feed forward** (0÷32767) A value of 1000, the figure is 100%. Is the coefficient percentage multiplied by the instantaneous speed, generates the output feed-forward control part of the slave axis. (see chapter)
integt	W	No condition	R	RdWr	**Integral time** (0÷32767) Is the time, in milliseconds, that produces the lowest coefficient of integrating the tracking error. The integration of that error multiplied by that coefficient raises the integral part of the control output of the slave axis. (see chapter)
derivt	W	No condition	R	RdWr	**Derivation time** (0÷32767) Is the time, in milliseconds, that produces the derivative of the coefficient of following error. The derivation of that error multiplied by that coefficient raises the integral part of the control output of the slave axis. (see chapter)
offset	W	No condition	R	RdWr	**Offset output** (-32767÷32767) Offset DAC output slave axis in bits. defines the value in bits of the correction concerning the analogue output of the slave axis in order to compensate for any drift in the system
tbfm	W	No condition	R	RdWr	**Time base frequency-meter master** (0÷3) Defines the sampling time of the frequency counter on the master axis. **0** = 240 ms **1** = 480 ms **2** = 24 ms **3** = 120 ms N.B. The smaller the sampling time, the faster the acquisition of frequency, but the higher the error at low frequencies.
tbf	W	No condition	R	RdWr	**Time base frequency-meter slave** (0÷3) Defines the sampling time of the frequency counter on the slave axis. **0** = 240 ms **1** = 480 ms **2** = 24 ms **3** = 120 ms N.B. The smaller the sampling time, the faster the acquisition of frequency, but the higher the error at low frequencies.

1.10.3 Axis variables

Name	D	Write conditions	R	A	Description
frqm	L	No condition	0	RdWr	**Actual frequency of master** Indicates the frequency of the transducer relative to the master axis. To change the accuracy refer to the tbfm parameter. The value is expressed in Hz
positm	L	st_init = 1 st_camex = 0	R	RdWr	**Actual position of master** (-999999 ÷ +999999) Indicates the current position of the master axis. The value is expressed in units of measurement.
encoderm	L	st_init = 1 st_camex = 0	R	RdWr	**Encoder value of master** Indicates the current position of the master axis. The value is expressed in bits encoder x 4.
vout	B	st_init = 1 st_cal = 1	0	RdWr	**Output voltage** (-100÷100) A value of 100, the percentage is 100%. Used to set or view (in this case without any condition) of the output voltage on the analog output of the slave axis. The data is expressed in tenths of a Volt.
follerr	L	No condition	0	Rd	**Following error** Indicates the error between the theoretical position and the actual position of the slave axis in absolute value. The value is expressed in bits encoder x 4.
vel	L	No condition	0	Rd	**Actual velocity** Indicates the current speed of the slave axis. The value is expressed in the unit of time of the set speed (Velocity unit).
frq	L	No condition	0	Rd	**Actual frequency** Indicates the frequency of the transducer on the slave axis. The value is expressed in Hz
posit	L	st_init = 1 st_camex = 0	R	RdWr	**Actual position** (-999999 ÷ +999999) Indicates the current position of the slave axis. The input value or bed is expressed in units of measurement.
encoder	L	st_init = 1 st_camex = 0	R	RdWr	**Encoder value** ($-2^{31} \div 2^{31}-1$) Indicates the current position of the slave axis. the read value is expressed in bits transducer x 4.
delta1	L	No condition	R	RdWr	**Delta 1** ($-2^{31} \div 2^{31}-1$) General purpose variable. Used as a log for data exchange.
delta2	L	No condition	R	RdWr	**Delta 2** ($-2^{31} \div 2^{31}-1$) General purpose variable. Used as a log for data exchange.
setvel	L	No condition	R	RdWr	**Set velocity** (0÷maxvel) Defines the speed of the slave axis on placements. The input value is per unit of time of the set speed (Velocity unit). If the axis is moving (st_still = 0) You can change the speed setpoint only if the new value is used to reach the setting quota.
setpos	L	No condition	R	RdWr	**Set position** (minpos÷maxpos) Defines the placement quota reached from slave axis at the speed setvel.
rowex	W	No condition	0	Rd	**Row in use** (0÷40) Defines the number of the sector running.
ffwdreg	L	No condition	0	Rd	**Feed-forward register** ($-2^{31} \div 2^{31}-1$) Is the instantaneous value of the register of feed-forward, expressed in bits.
propreg	L	No condition	0	Rd	**Proportional register** ($-2^{31} \div 2^{31}-1$) Is the instantaneous value proportional registry expressed in bits.
intreg	L	No condition	0	Rd	**Integral register** ($-2^{31} \div 2^{31}-1$) Integral log slave axis.
derreg	L	No condition	0	Rd	**Derivate register** ($-2^{31} \div 2^{31}-1$) Derived log slave axis.
codeMex	L	No condition	0	Rd	**Code M in execution** ($-2^{31} \div 2^{31}-1$) Reads the sector's code M running.
funInp	B	No condition	R	RdWr	**Programmable function of input** (0÷99) Allows you to configure how the normal input see table inputs configuration. (See chapter)

Name	D	Write conditions	R	A	Description
funInt	B	No condition	R	RdWr	**Programmable function of interrupt input** (0÷99) Allows you to configure how the interrupt input see table input configuration. (See chapter)
funOut	B	No condition	R	RdWr	**Programmable function of output** (0÷99) Allows you to configure how the table output output configuration. (See chapter)
inpcapt	B	No condition	0	RdWr	**Capture mode** (0÷2) Defines how to capture input function for generic function (see configuration files). **0** = Disable. **1** = Single capture on falling edge. **2** = Single capture on the rising edge. The capture is enabled if the state st_enbl = 1.
intcapt	B	No condition	0	RdWr	**Interrupt capture mode** (0÷2) Defines how to capture the interrupt function (see configuration files). **0** = Disable. **1** = Single capture on falling edge. **2** = Single capture on the rising edge. he capture is enabled if the state vt_intenbl = 1.
errcode	B	No condition	0	Rd	**Error code** (0÷100) Indicates the type of failure intervened in the system. The code is valid only if st_error = 1 (See chapter)
errvalue	B	No condition	0	Rd	**Error value** (0÷100) Specifies the sector that caused the error in the system. The value is valid only if st_error = 1 (See chapter)
wrncode	B	No condition	0	Rd	**Warning code** (0÷100) Indicates the type of warning in the system. The code is valid only if st_warning = 1 (See c hapter)
wrnvalue	B	No condition	0	Rd	**Warning value** (0÷100) Specifies the sector that caused the warning in the system. The value is valid only if st_warning = 1 (See chapter)

1.10.4 Program variables

Name	D	Write conditions	R	A	Description	
codeG1	W	rowex ?1	R	RdWr	**Code G1** Indicates the value that assumes the G code in step 1. See G codes description.	
codeG2	W	rowex ?2	R	RdWr	**Code G2** Indicates the value that assumes the G code in step 2. See G codes description.	
codeG40	W	rowex ?40	R	RdWr	**Code G40** Indicates the value that assumes the G code in step 40. See G codes description.	
codeQm1	L	rowex ?1	R	RdWr	**Code Q1 master** (0÷999999) Indicates the incremental quota of the sector master 1. The input value is in units of measurement.	
codeQm2	L	rowex ?2	R	RdWr	**Code Q2 master** (0÷999999) Indicates the incremental quota of the sector master 2. The input value is in units of measurement.	
codeQm128	L	rowex ?128	R	RdWr	**Code Q128 master** (0÷999999) Indicates the incremental quota of the sector master 128. The input value is in units of measurement.	
codeQs1	L	rowex ?1	R	RdWr	**Code Q1 slave** (-999999÷999999) Indicates the incremental quota of the sector slave 1. The input value is in units of measurement.	
codeQs2	L	rowex ?2	R	RdWr	**Code Q2 slave** (-999999÷999999) Indicates the incremental quota of the sector slave 2. The input value is in units of measurement.	
codeQs128	L	rowex ?128	R	RdWr	**Code Q128 slave** (-999999÷999999) Indicates the incremental quota of the sector slave 128. The input value is in units of measurement.	

Name	D	Write conditions	R	A	Description	
codeQma1	L	rowex ?1	R	RdWr	**Code Q1 auxiliary master** (0÷999999) Indicates the master auxiliary incremental quota of the sector 1. The input value is in units of measurement.	
codeQma2	L	rowex ?2	R	RdWr	**Code Q2 auxiliary master** (0÷999999) Indicates the master auxiliary incremental quota of the sector 2. The input value is in units of measurement.	
codeQma40	L	rowex ?128	R	RdWr	**Code Q40 auxiliary master** (0÷999999) Indicates the master auxiliary incremental of the sector 40. The input value is in units of measurement.	
codeQsa1	L	rowex ?1	R	RdWr	**Code Q1 auxiliary slave** (-999999÷999999) Indicates the slave auxiliary incremental of the sector 1. The input value is in units of measurement.	
codeQsa2	L	rowex ?2	R	RdWr	**Code Q2 auxiliary slave** (-999999÷999999) Indicates the slave auxiliary incremental of the sector 2. The input value is in units of measurement.	
codeQsa40	L	rowex ?128	R	RdWr	**Code Q40 auxiliary slave** (-999999÷999999) Indicates the slave auxiliary incremental of the sector 128. The input value is in units of measurement.	
codeM1	L	rowex ?1	R	RdWr	**Code M1** Introduces a code not relating to positioning, but that identifies a variable that will be can then processed by the application program (tool code, machining type, number of pieces ...).	
codeM2	L	rowex ?2	R	RdWr	**Code M2** Introduces a code not relating to positioning, but that identifies a variable that will be can then processed by the application program (tool code, machining type, number of pieces ...).	
codeM40	L	rowex ?128	R	RdWr	**Code M40** Introduces a code not relating to positioning, but that identifies a variable that will be can then processed by the application program (tool code, machining type, number of pieces ...).	

1.10.5 Commands

Name	Conditions	Description
INIT	st_init = 0	**Init** Initializing command device. If the device is not initialized does not perform the calculations related to the axis and then sits idle. Enable the st_init state.
EMRG	st_init = 1	**Emergency** Slave axis in emergency stopping, without deceleration ramp, any movement in progress. It also disabled the reaction space axis.
RESUME	st_init = 1 st_emrg = 1	**Resume** Restoration of emergency condition of the slave axis; is reenabled the reaction of space. The acquisition of the start, the axis incorporates placement.
STOP	st_init = 1 st_regoff = 0 st_emrg = 0 st_cal = 0 st_still = 0 st_camex = 0	**Stop** Stops any ongoing slave axis positioning. The axis follows the deceleration ramp tdec parameter. The axis remains in reaction of space.
START	st_init = 1 st_regoff = 0 st_emrg = 0 st_cal = 0 st_still = 0 st_camex = 0 st_prson = 0	**Start** The slave axis start the placement to setpos quota with setvel speed. The slave axis start the placement to the setpos quota with setvel speed setting.
PRESET	st_init = 1 st_regoff = 0 st_emrg = 0 st_cal = 0 st_still = 0 st_camex = 0	**Preset** Start search presets slave axis. Preset search procedure is started set by the parameters prsmode and prsdir. If the preset search is already running, the command performs the reverse search.
RSPRSOK	st_init = 1 st_prson = 0	**Reset state st_prsok** Reset the st_prsok state
PRESETM	st_init = 1 st_camex = 0 st_prson = 0	**Master preset** Start preset search master axis. Start of the search procedure of presets in the manner set with prsmodem parameter.

Name	Conditions	Description
RSPRSM	st_init = 0 st_prson = 0	**Reset preset of master** Reset the st_prsokm state if the master preset is concluded. If the master preset is in progress (st_prsonm = 1) is blocked
RSERR	st_init = 1	**Reset status st_error** Reset st_error status and error code errcode and errvalue.
RSWRN	st_init = 1	**Reset status st_warning** Reset the st_warning status and wrncode and wrnvalue warning code.
LOOPON	st_init = 1 st_loopon = 1	**Loop on** Enable the slave axis space reaction. The analog output sharpen every external action that attempts to move the axis from the position reached (drift, operator, ...). This operation resets any tracking error follerr.
LOOPOFF	st_init = 1 st_loopon = 1	**Loop off** Disable the slave axis space reaction. The axis can be moved from its position that the analogue output contrasts the movement.
MANFW	st_init = 1 st_regoff = 0 st_prson = 0 st_camex = 0 st_cal = 0 st_still = 1 st_emrg = 0	**Forward** Forward manual movement of slave axis. Controls the forward manual movement of the axis at the speed set with setvel. The movement is stopped with the STOP command.
MANBW	st_init = 1 st_regoff = 0 st_prson = 0 st_camex = 0 st_cal = 0 st_still = 1 st_emrg = 0	**Backward** Backward manual movement of slave axis. Controls the backward manual movement of the axis at the speed set with setvel. The movement is stopped with the STOP command.
CALON	st_init = 1	**Volt generator on** The analog output of the slave axis is used as a voltage source; in this case you cannot use it to position the axis. The output value is settable at will through the vout variable.
CALOFF	st_init = 1 st_cal = 0	**Volt generator off** The analog output of the slave axis is not used as a voltage source, so it can be used for managing placements.
CNTLOCK	st_init = 1	**Lock counter** Block the acquisition of the counting of the slave axis even if the transducer continues to send signals. At this stage the possible displacement of the axis is not detected.
CNTUNLOCK	st_init = 1	**Unlock counter** Unlock the count of the slave axis. Resumes reading of the signals sent from the transducer and, therefore, the counter updating.
CNTREV	st_init = 1	**Reverse counter** Invert the phases of the slave transducer within the device. Reverses the direction of the counter (increase/decrease).
CNTDIR	st_init = 1	**Direct counter** Resets the counter direction of the transducer of the slave axis.
CNTLOCKM	st_init = 1	**Lock counter master** Block the acquisition of the counting of the master axis even if the transducer continues to send signals. At this stage the possible displacement of the axis is not detected.
CNTUNLOCKM	st_init = 1	**Unlock counter master** Unlock the count of the master axis. Resumes reading of the signals sent from the transducer and, therefore, the counter updating.
CNTREVM	st_init = 1	**Reverse counter master** Invert the phases of the master transducer within the device. Reverses the direction of the count (increase/decrease).
CNTDIRM	st_init = 1	**Direct counter master** Resets the counter direction of the transducer of the master axis.
STOPCAM	st_init = 1 st_camex = 1	**Stop cam** Stops the current cam. Axis stop following a deceleration ramp asynchronous, according the tdec parameter. The axis remains in reaction to space.
STARTCAM	st_init = 1 st_still = 1 st_camex = 1 st_prson = 0 st_emrg = 0 st_regoff = 0	**Start cam** Start the slave axis positioning enabling the processing of cam 1 sector introduced and running the code.
REGOFF	st_init = 1 st_still = 1 st_camex = 0 st_prson = 0	**Regulation OFF** Disable the adjustment and upgrading the slave axis DAC, together with all the movement commands.
REGON	st_init = 1 st_still = 1 st_regoff = 1 st_emrg = 0	**Regulation ON** Enable the adjustment and upgrading the slave axis DAC, together with all the movement commands.

Name	Conditions	Description
ENBL	st_init = 1	**Reverse counter** Invert the phases of the slave within the transducer device. Reverses the direction of the count (increase/decrease).
INTENBL	st_init = 1	**Direct counter** Resets the counter direction of the transducer of the slave axis.
DSBL	st_init = 1	**Lock counter master** Block the master axis counting acquisition even if the transducer continues to send signals. At this stage the axis shift is not detected.
INTDSBL	st_init = 1	**Interrupt disable** Disables the interrupt input function inserted in the funInt parameter. Disable the st_intenbl state.
RSCAPT	st_init = 1 st_capt = 1	**Reset status of capture input** Disable the st_capt state.
RSINTCAPT	st_init = 1 st_intcapt = 1	**Reset status of capture interrupt input** Disable the st_intcapt state.
DELCNT	st_init = 1 st_still = 1 st_camex = 0 st_prson = 0 st_cal = 0 st_regoff = 0	**Delta counter** The slave axis counter (axis position) is changed by adding algebraically the value specified in the delta1 parameter (posit = posit + delta1).
DELCNTM	st_init = 1 st_prsonm = 0 st_camex = 0	**Delta counter of master** The master axis counter (axis position) is changed by adding algebraically the value specified in the delta2 paramenter (positm = positm + delta2).

1.10.6 States

Name	Dim.	Write conditions	Access	Description	
st_init	F	No condition	Rd	**Init** Reporting of initialized device. **0** = not initialized device **1** = initialized device To power up by default loads the value zero.	
st_chvel	F	No condition	Rd	**Status of enable velocity change** Indicates that the device can accept a slave axis speed set other than running and place taking place by the speed change procedure. Speed change procedure is available only during placements (not while running the cam). To power up by default loads the value zero.	
st_emrg	F	No condition	Rd	**Emergency** (0÷1) Reporting of slave axis in emergency. **0** = axis not in emergency **1** = axis in emergency To power up by default loads the value zero.	
st_toll	F	No condition	Rd	**Tolerance** (0÷1) Reporting of slave axis in tolerance in relation to the quota put running from the START command. **0** = axis not in tolerance **1** = axis in tolerance To power up by default loads the value zero.	
st_tpos	F	No condition	Rd	**Tolerance of set position** (0÷1) Indicates that the counter of the slave axis is within the tolerance in relation to the quota in setpos variablethough was given a START or not. **0** = axis not in tolerance **1** = axis in tolerance To power up by default loads the value zero.	
st_prson	F	No condition	Rd	**Preset ON** (0÷1) Reporting of slave axis preset search correctly completed **0** = preset search not yet completed or not executed **1** = preset search was successfully executed To power up by default loads the value zero.	
st_prsok	F	No condition	Rd	**Preset ok** (0÷1) Reportin of slave axis preset search correctly completed. **0** = preset search not yet completed or not executed **1** = preset search was successfully executed To power up by default loads the value zero.	

Name	Dim.	Write conditions	Access	Description
st_prsonm	F	No condition	Rd	**Preset of master ON** (0÷1) Reporting ongoing master axis preset search. **0** = preset search not in progress **1** = preset search in progress To power up by default loads the value zero.
st_prsokm	F	No condition	Rd	**Preset ok of master** (0÷1) Reportin of master axis preset search correctly completed. **0** = reset search not yet completed or not executed **1** = preset search was successfully executed To power up by default loads the value zero.
st_still	F	No condition	Rd	**Rd Still** (0÷1) Reporting of slave axis stopped. During the execution of this state is equal to 1. **0** = axis moving **1** = axis stopped To power up by default loads the value 1.
st_camex	F	No condition	Rd	**Cam to execution** (0÷1) Cam running report. **0** = cam not running **1** = cam running To power up by default loads the value zero.
st_movdir	F	No condition	Rd	**Direction BW** (0÷1) Signaling the slave axis direction of movement only if you are performing a cam (st_camex = 0). **0** = forward **1** = backward To power up by default loads the value zero.
st_loopon	F	No condition	Rd	**Loop ON** (0÷1) Reporting of slave axis in reaction to space. **0** = axis not in reaction to space **1** = axis in reaction to space To power up by default loads the value zero.
st_foller	F	No condition	Rd	**Following error** (0÷1) Reporting the following error slave axis (withholding of 500 ms) **0** = axis not in following error **1** = axis in following error To power up by default loads the value zero.
st_sync	F	No condition	Rd	**Syncronism** (0÷1) Reporting of slave axis in synch during execution of the cam: **0** = axis does not in sync **1** = axis in synch To power up by default loads the value zero.
st_cal	F	No condition	Rd	**Calibration** (0÷1) Reporting of slave axis as voltage generator. **0** = voltage axis off **1** = voltage axis on To power up by default loads the value zero.
st_cntlock	F	No condition	Rd	**Rd Locked** (0÷1) Reporting of slave axis count blocked. **0** = Axis count unlocked **1** = Axis count blocked To power up is maintained a the same state of the power off.
st_cntrev	F	No condition	Rd	**Reversed** (0÷1) Reporting of slave axis count inverted. **0** = Axis count unlocked **1** = Axis count blocked To power up is maintained a the same state of the power off.
st_cntlockm	F	No condition	Rd	**Master locked** (0÷1) Reporting of master axis count blocked. **0** = Axis count unlocked **1** = Axis count blocked To power up is maintained a the same state of the power off.

Name	Dim.	Write conditions	Access	Description
st_cntrevm	F	No condition	Rd	**Master reversed** (0÷1) Reporting of master axis count inverted. **0** = Axis count not inverted **1** = Axis count inverted. To power up is maintained a the same state of the power off.
st_regoff	F	No condition	Rd	**Regulation OFF** (0÷1) Reporting of slave axis adjustment disable and and updated DAC not implemented. **0** = adjustment unlocked **1** = adjustment locked To power up by default loads the value zero.
st_enbl	F	No condition	Rd	**Normal input enabled** (0÷1) Reports enable normal input inserted into the funInp parameter. Is activate by the ENBL command and deactivated by DSBL command. Is automatically disabled to capture occurred **0** = Capture of the count is not enabled. **1** = Capture of count enabled. To power up by default loads the value zero.
st_intenbl	F	No condition	Rd	**Interrupt input enabled** (0÷1) Reports enabling interrupt input function inserted in the funInt parameter. Is activate by the INTENBL command and deactivated by INTDSBL command. Is automatically disabled to capture occurred. **0** = Capture of the count is not enabled. **1** = Capture of count enabled. To power up by default loads the value zero.
st_capt	F	No condition	Rd	**Capture of normal input** (0÷1) Activating the capture of the selected function in funInp; is reset from the RSCAPT command. **0** = Capture not executed. **1** = Capture executed. To power up by default loads the value zero.
st_intcapt	F	No condition	Rd	**Capture of interrupt input** (0÷1) Activating the capture of the selected function in funInt; is reset from the RSINTCAPT command. **0** = Capture not executed. **1** = Capture executed. To power up by default loads the value zero.
st_int	F	No condition	Rd	**Status of interrupt line** (0÷1) Indicates the status of the interrupt line of general purpose. **0** = Interrupt input deactivate. **1** = Interrupt input activate. To power up by default loads the value zero.
st_error	F	No condition	Rd	**Status of camming device error** (0÷1) CAMMING2 device error state. To decode the error you must refer to the errcode and errvalue variables. **0** = Error not present. **1** = Error present. To power up by default loads the value zero.
st_warning	F	No condition	Rd	**Status of camming device warning** (0÷1) CAMMING2 device warning state. To decode the warning you must refer to the wrncode and wrnvalue variables. **0** = Warning not present. **1** = Warning present. To power up by default loads the value zero.

1.10.7 Device limitations

1. It is not possible to sequence more than 7 zero-sampling sectors.
2. You cannot put in sequence over 3 sectors count update.
3. With the parameters:
   pulse = 999999
   measure = 934
   maxvel = 999999
   unitvel = 0
   decpt = 3

You lay down the conditions for creation of overflow in the calculations of the sectors 150, 151, 152 e 153.

4. During the of the cam execution (st_camex = 1), you cannot change the sector running and that later.
5. The device is designed to work with the master increase. Is it possible to run the cam with the master that decreases under the following conditions:
   The cam stops remains in reaction to space on previous sector if it meets the sectors 130, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 160. Can run the previous sector only if you have already run at least once (could not be committed due to a jump).