

SFERA srl
MC6 axis controller
User guide

version 1.0.0

Table of contents

1	Introduction.....	4
1.1	Main characteristics.....	4
1.2	How to use the guide.....	4
2	Instructions for use	5
2.1	Electrical connections.....	5
2.1.1	Encoder.....	5
2.1.2	Drives	6
2.1.3	Zero sensors, overtravel sensors and other inputs/outputs	7
2.2	Configuration and calibration.....	8
2.2.1	Setting of DIP switches	8
2.2.2	Communication	10
2.2.3	Encoder interface	11
2.2.4	Drive interface	12
2.2.5	Protections	13
2.2.6	PID calibration.....	13
2.3	Independent movements.....	14
2.3.1	Introduction	14
2.3.2	Profile parameters.....	15
2.3.3	Positionings.....	15
2.3.4	Speed adjustment	16
2.3.5	Axis interrogation	16
2.3.6	Reset.....	17
2.4	Coordinated movements	18
2.4.1	General information.....	18
2.4.2	Linear segments.....	20
2.4.3	Circular movements	20
2.4.4	Continuous movements	20
2.5	Special functions	21
2.5.1	Electronic gearing	21
2.5.2	Electronic clutch	21
2.5.3	Electronic cam.....	22

Index of Tables

Table 1:	Encoder connection	5
Table 2:	Drive reference connection.....	6
Table 3:	Drive enabling connection	7
Table 4:	Connection of 24V inputs/outputs.....	8
Table 5:	Function DIP switch S1.....	8
Table 6:	Examples for setting of DIP switch S1	9
Table 7:	IRQ selection (JP6)	10
Table 8:	Encoder input termination networks	10

Index of Figures

1 Introduction

1.1 Main characteristics

- ⤴ Stand-alone axis controller to be assembled on the control cabinet.
- ⤴ Manages up to six servo axes with a control period of 122 us and generation of trajectories with period of 976 us.
- ⤴ Six encoder inputs with programmable digital filters, 32-Bit counters for 10 Mhz.
- ⤴ Six analog outputs for command of servo drives +/-10V, 14 bit resolution
- ⤴ Sixteen outputs and sixteen optocoupled inputs, one of them specialized in quick capture of position (200 ns max.).
- ⤴ Management of six further axes with stepper drives
- ⤴ Independent absolute and relative positionings.
- ⤴ Electronic CAM.
- ⤴ Electric shaft (gearing).
- ⤴ Electronic clutch.
- ⤴ Coordinated linear and circular movements.
- ⤴ Execution of on board resident programs

1.2 How to use the guide

The present user guide has been written in order to permit a simple and quick use of the MC6 axes controller card. In consideration of the fact that normally the interested persons are technicians which are already experienced in industrial automation, we tried to detail the specific aspects of the card rather than using a didactic approach on axes control for which you can refer to the existing literature.

Concerning the hardware aspects, such as for example pinout of contacts, please refer to the *hardware reference manual*.

2 Instructions for use

2.1 Electrical connections

The MC6 controller on its upper panel has got 6 flat connectors for the interface signals to servo and stepper drives:

- ^ encoder input
- ^ analog reference
- ^ stepper step and direction

Laterally there are clamp-type connectors for the other functions:

- ^ 24V power supply
- ^ digital inputs
- ^ digital outputs
- ^ analog inputs
- ^ CAN bus
- ^ RS232 serial connection
- ^ RS485 serial connection

2.1.1 Encoder

It is possible to use incremental encoders in quadrature with 5V power supply and with complementary outputs or equivalent “virtual” encoders rebuilt by the drive.

If possible, use encoders featuring “line drive RS422” outputs, which guarantee the best quality of signals, even at high frequencies and in presence of interferences; in any case, the encoder must be able to support the load of the termination resistance (150 Ohm) present on the card.

For the connection use a shielded cable, preferably a twisted pair, for data transmission.

Use internal conductors, too, to connect the power supply (GND and +5V) and weld the shield directly onto the connector body. Use metal shells. In case of cable made up of a twisted pair of wires, use them with complementary signals, for example A1+ and A1- on one pair, B1+ and B1- on the other pair and so on.

Obviously the +5V power supply has to be connected in case of physical encoders only, if the encoder is rebuilt by the drive, don't connect it. Make sure that the total absorption of the six encoders does not exceed 900 mA, otherwise an external power supply must be used; in this case, too, do not connect together the two power supplies.

In case you should need to invert the direction of the encoder in order to adapt it to the conventional direction of the axis, do not act on the signal connection: you can do it easily from software.

The encoders of axes 1-6 must be connected respectively to the connectors P1-P6 of MC6.

2.1.2 Drives

Drives must have an analog reference input +/-10V, preferably of the differential type. The enabling input must be connected to a digital output of MC6. If the electric

interface is not compatible, an external relay must be used.

Sometimes drives permit to choose between working in torque mode (the supplied torque is proportional to the reference input) or in speed mode (the motor speed is proportional to the reference input); we recommend to choose the torque mode only if the application needs to adjust or to limit the force (for example placing, inserting or catching operations), otherwise it is preferable to use the speed mode.

For the reference use a shielded cable with the shielding connected to the driver earth. If you have to invert the command, don't act on the wiring, especially if the drive hasn't got a differential input, because a short circuit would be caused, but use the dedicated software command.

2.1.3 Zero sensors, overtravel sensors and other inputs/outputs

Sixteen inputs and sixteen 24V outputs are foreseen. Zero sensors and overtravel sensors may be connected to any input since it is possible to map such functions from software. The remaining four inputs and the outputs are available for the application. For example, they can control an actuator in a very tight relation with the axes movement or manage simple synchronizations with external devices.

If you need to acquire positions on the basis of an external signal, connect it to the number 1 input which permits an extremely short time of reaction.

Inputs are optocoupled and activated by applying a voltage between 15V and 30V with respect to GNDH (P5.30). The outputs are optocoupled of the source type (PNP), protected from short-circuit and from overheating and suitable to pilot loads up to 700 mA (for example relays, solenoid valves and lamps). The 24V dc nominal power supply (min. 15V max 30V) must be supplied externally to the point VCCH (P5.21) and it must be referred to GNDH.

The optocouplers separate the MC6 power supply from the 24V machine power supply only in order to increase immunity to interferences; for the correct working of the card and for safety reasons both power supplies must be referred to earth.

2.2 Configuration and calibration

2.2.1 Setting of DIP switches

DIP switch S1

The dip switch has got 8 positions, only the last position is used by the system (if activated at turning on, it puts the controller into programming mode); the remaining 7 positions are available for the application.

2.2.2 Communication

Normally the MC6 controller is connected to operator panels on which a graphical user interface is implemented, permitting axes configuration and calibration. Otherwise it is possible to connect, over a RS232 serial connection, a dumb terminal through which it is possible to give commands by means of a textual shell.

From now on, in the present user guide we will introduce some commands which,

SFERA srl – MC6 axis controller

used in an interactive way, will be useful to verify and to calibrate the system. Afterwards, such commands can be used by your software to realize the application. Command syntax is as shown here below:

- <Comando> [*Parametro1*] ... [*ParametroN*] [; *Commento*]

The hyphen at the beginning of the command is the program prompt and you don't need to type it in. Comment is facultative and does not effect working. The shell permits you to use indifferently in capital and in small letters, but remember that with the application software this will not be possible. The answer to the command, if foreseen, is displayed on the following lines.

2.2.3 Encoder interface

Turn off the controller and connect the encoders. Turn on the controller and give the commands:

- GetActPos 1 ; reading of the actual position of axis 1
-2

GetActPos returns the actual axis position, e.g. the one read by the encoder counter, which, when turning on, has got a value of 0 or a small value if the axis slightly moved for some vibration. Manually move the axis in positive direction (according to your application), so that the encoder carries out a turn and repeat position reading:

- GetActPos 1
4085

The new position should approximately correspond to the number of the encoder pulses multiplied by four, for example an encoder featuring 1024 pulses/turn must give about 4096 counts.

It may happen that the sign is negative, for example -4107; in such case the axis must be configured by using the command *ConfigServo* so as to invert the encoder counter direction. Besides the direction of the *ConfigServo* command you also have to enter the band of the digital filter, useful to avoid problems due to noise generated by the drives. For example, if a drive has got a maximum speed of 4500 rpm, the encoder has got 1024 pulses/turn and since it executes 5 turns for each motor turn because of the transmission, we have got:

$$F_{max} = \frac{4500}{60} * 5 * (1024 * 4) = 1536000 \text{ counts / second (1.536 MHz)}$$

Setting the parameter *EncDir* of the command *ConfigServo* to -1, we invert the counter direction and with the parameter *EncFilter* set to 4, we filter frequencies of over 2 Mhz (for more details see the reference guide of the command *ConfigServo*).

- *ConfigServo* 1 1 -1 4; enter inversion of encoder 1, Fmax 2 Mhz

Now it is advisable to check the correctness of the direction by repeating the procedure. Then note down the chosen code (you must use *ConfigServo* with the same code in the application program) and repeat this procedure for all the other encoders.

2.2.4 Drive interface

Disconnect the motor from the mechanics, for example by disassembling the transmission belts, so that it can turn without any risk for things or for persons. Also make sure to disconnect power supply in a rapid way, for example by pressing the emergency stop push button.

Now set the PID to zero and the position error threshold to 10000, then put the axis into torque:

- Pid 1 0 0 0
- ErrorLimit 1 10000
- MotorOn 1

Make sure that the drive is enabled. The motor can stand still or it can move slowly because of the offsets. Enter a small positive offset until the motor rotation direction is clearly visible, for example:

- DacOffset 1 100; 100 mV

If the motor rotates in negative direction, the reference voltage must be inverted with parameter RefDir of the ConfigServo command:

- ConfigServo 1 -1 -1 4; +1= normal reference, -1= inverted reference

Now the motor must rotate in the correct direction. Now we only have to adjust the offset by:

- DacOffset 1 0

Note down the inversion or less and repeat the procedure for the other axes, too.

2.2.5 Protections

For each axis, the forward and backward overtravel inputs have been foreseen. In case a rising edge on one of these inputs should be detected, the axis is put into error condition and it is immediately disabled. Since this error is caused by the edge and not by the signal level, it is possible, as soon as the axis is enabled again, to take the axis in the working field under the control of the application software.

A different protection is constituted by the continuous monitoring of the position error, which, if its absolute value exceeds the programmed limit, generates the axis error.

Use the following command to enter the position error limit:

- ErrorLimit <asse> <valore>

However, please bear in mind that there are some types of faults which cannot be managed in an efficient way by the MC6 controller, for example the interruption of the encoder signals or internal faults, which may cause escape of the axis. Therefore it is necessary to foresee special protections (brake, shock absorbers, barriers etc.) for safeguard of the system and of the personnel working with such system.

2.2.6 PID calibration

For a long time, the PID regulator has been used in the industry as a control method for feed-back systems and still today it is widely diffused. For its setting use the following commands:

- Kp <asse> <valore> ; proportional factor
- Ki <asse> <valore> ; integral factor
- Kd <asse> <valore> ; derivative factor
- Kvel <asse> <valore> ; feed forward speed factor
- Kacc <asse> <valore> ; feed forward acceleration factor
- Daclimit <asse> <valore> ; drive command limitation

In literature we can find many procedures for the calibration of PID systems, but personal experience is still the most important resource. However, here below you will find some advice.

Speed mode drives

1. Set to zero Kp, Ki, Kd, Kvel and Kacc.
2. Set Daclimit to 1 V.
3. Enable the axis.
4. Execute some positionings with trapezoidal profile and with high deceleration values, increasing Kp up to the oscillation limit.
5. Continue to increase Kd in order to stabilize the system: in case of success, repeat step 4.
6. Reduce the previously obtained Kp and Kd values by half, then execute some more positionings increasing the Ki value to reduce the error.
7. Continue to increase Kd in order to stabilize the system: in case of success, repeat step 6.
8. Take Daclimit to 10 V. Except for particular cases, there should be no need to set Kv and Ka.

Torque mode drives

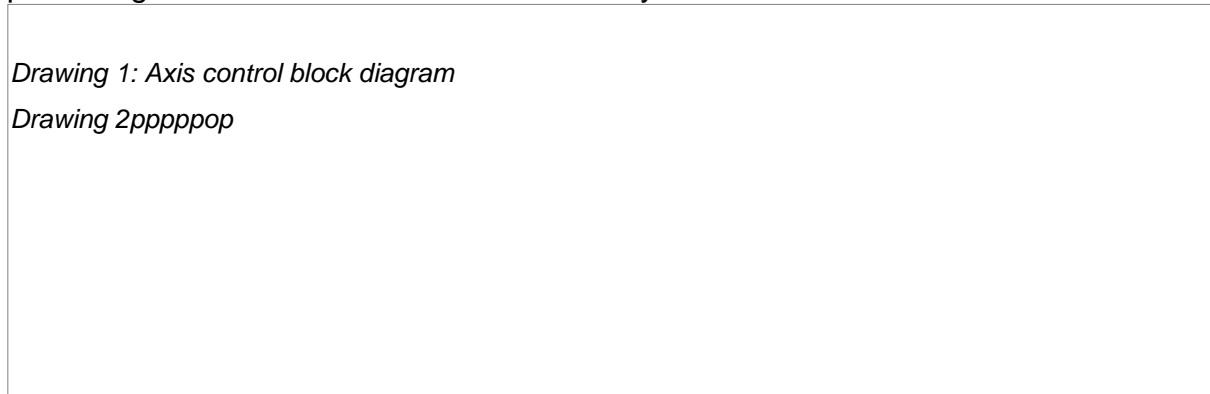
1. Set to zero Kp, Ki, Kd, Kvel and Kacc.
2. Set Daclimit to 1 V.
3. Enable the axis.
4. Execute some positionings with trapezoidal profile and with high deceleration values, increasing Kp up to the oscillation limit.
5. Continue to increase Kd in order to stabilize the system: in case of success, repeat step 4.
6. Reduce the previously obtained Kp e Kd values by half.
7. Take Daclimit to 10 V. Except for particular cases, there should be no need to set Ki, Kv and Ka.

2.3 Independent movements

2.3.1 Introduction

The block diagram here below shows the hardware and software structure

permitting to move an axis in a controlled way:



The blocks represent the software components (application program), firmware (profile generator, servo control) and hardware (MC6 controller, drive, motor, encoder).

We already saw how to execute the hardware connections and how to do PID calibration, thus obtaining a *position-controlled* system, e.g. a system where the *actual* axis position (*ActPos*) more or less faithfully follows the *reference position* (*RefPos*). The difference between reference position and actual position is the so-called *position error* (*PosErr*) and, in case of a good PID calibration, it should be very restrained.

Now we have to examine working of the generator of *profiles*, the block which determines, instant per instant, the reference position value *RefPos*. In practice, the application program must set the parameters of acceleration (*NomAcc*), deceleration (*NomDec*), speed (*NomVel*) and position to be reached (*Target*); as soon as the start command is given, the profile generator calculates a series of intermediate positions which will take the axis from the start position to the target position in the desired way.

Thanks to the power of the RISC processor, the MC6 controller has got the capacity to recalculate the profile of the movement at any moment. This means that it is possible to modify speed, acceleration, deceleration and target position even during the movement; for example, we can start a movement in high speed towards a nominal target position, reduce speed when arriving at a certain distance and then slightly modify the target position according to the value read by a transducer.

The block diagrams shows only one axis, but actually the MC6 controller contains a profile generator for each of the six axes which can be moved in an independent way one from the other. For such reason they are called *generators of independent profiles* and the deriving type of movement is called the *independent movement* of the axis.

2.3.2 Profile parameters

For setting of the profile parameters act as described here below:

- Profile <asse> <velocità> <accelerazione> <decelerazione>

For example:

- Profile 1 5000 300000 20000

on the axis 1 set a speed of 5000 counts/second, an acceleration of 300000 counts/second², a deceleration of 200000 counts/second².

As an alternative it is possible to set the single parameters by using the following commands:

- Vel <asse> <velocità>
- Acc <asse> <accelerazione>
- Dec <asse> <decelerazione>
- Smoothing <asse> <smoothing>

These commands are useful also if we wish to modify some of the parameters during the movement (except for the smoothing value which can be modified only if the axis is stopped).

Then there are the commands which return the current value of the parameters:

- GetVel <asse> ; return to the nominal speed
- GetAcc <asse> ; return to the nominal acceleration
- GetDec <asse> ; return to the nominal deceleration
- GetSmoothing <asse> ; return to smoothing

2.3.3 Positionings

An absolute positioning is a movement that makes the axis stop in a specified position, called target position. To execute such movement, enable the axis, enter the profile parameters and give the command:

- MoveAbs <asse> <target> ; moves the axis into the target position

For example, if we wish to move the axis 2 at 10000 encoder counts:

- MoveAbs 2 10000

The movement can be interrupted at any moment by using the command:

- Stop <asse> ; interrupts the movement

causing braking controlled by the nominal deceleration. The movement can be completed by a subsequent start command. The target position, like all other positions, is expressed in encoder counts and it must be in a range between -2147483648 and 2147483647 (32 bits with sign).

A relative positioning is a movement for which we specify *for how much* we wish to move the axis with respect to the present position. The command is:

- MoveRel <asse> <delta> ; relative movement

For example, if we wish to move the axis 1 backward for 1000 encoder counts:

- MoveRel 1 -1000

For this movement, too, the allowed limits are between -2147483648 and 2147483647 encoder counts.

It is also possible to give the command for a relative movement whilst an absolute

movement is in progress and vice versa: the axis will move according to the last command given.

2.3.4 Speed adjustment

Some applications need to move the axis indefinitely at a pre-established speed. In this case we use the command:

```
- jog <asse> <velocità>
```

Speed must be specified in encoder counts per second. To terminate the movement we can use the *stop* command which causes a deceleration up to the stop of the axis or we can give a positioning command if we wish to stop the axis in a precise position.

2.3.5 Axis interrogation

There are several situations in which it is necessary to interrogate the axis, for example if we wish to know when the movement is finished or which is the actual position of the axis. For such purpose, several commands are available:

- GetPos <asse> ; returns the nominal position
- GetActPos <asse> ; returns the actual position
- GetPosErr <asse> ; returns the position error
- GetRefVel <asse> ; returns the instantaneous speed of the reference
- Status <asse> ; returns the status bits

The Status command returns a word the bits of which are status flags which are very important for synchronization of the application program flow with the axis movement; in particular, the flag *Done* is very useful.

The example shows the typical evolution of the status flags during a movement:

- ```
- MoveRel 1 1000000 ; moves the axis 1 for 1 million of counts
- Status 1 ; axis enabled and moving
 6
- Status 1
 6
- Status 1 ; nominal speed has been reached
 22
- Status 1
 22
- Status 1
 22
- Status 1
 6
- Status 1
 6
- Status 1
 10
- Status 1
 42
```

- Status 1  
42

The following example shows how to give a sequence of movements:

- MoveAbs 1 10000 ; move axis 1 to 10000 counts
- WaitDone ; wait for arrival in target position 10000
- Jog 1 1000 ; move axis 1 at 1000 counts per second
- WaitDone 1 ; wait for end of acceleration

### 2.3.6 Reset

Here above we saw how to execute positionings by specifying the target as an absolute position.

When turning the card on, we cannot know the actual position of the axis since the incremental encoders do not supply this type of information, therefore we arbitrarily adopt as zero point the point in which the axis is situated in this very moment.

For some applications this may be acceptable, for example if we only wish to execute relative positionings or for speed adjustments. In other cases it is absolutely necessary that the axis zero point, also called *origin*, corresponds to a precise physical position. To reach such point we execute an operation called *reset sequence*, corresponding to the command:

- Home <asse> <modo> <velocità1> <velocità2>

Where the parameter *modo* selects one of the following sequences:

#### **Modo (Mode) = 1**

1. Movement with direction and speed defined by parameter *velocità1*, terminated by controlled braking as soon as the axis home input is activated.
2. Movement with direction and speed defined by parameter *velocità2*, terminated by controlled braking at the first encoder index met after that the home input has been de-activated.
3. Positioning with speed defined by parameter *velocità2* on the mentioned encoder index.
4. Adoption of the reached position as new origin of the axis.

#### **Modo (Mode) = 2**

1. Movement with direction and speed defined by parameter *velocità1*, terminated by controlled braking as soon as the axis home input is activated.
2. Movement with direction and speed defined by parameter *velocità2*, terminated by controlled braking as soon as the home input is de-activated.
3. Positioning with speed defined by parameter *velocità2* on the commutation point of the home input.
4. Adoption of the reached position as new origin of the axis.

#### **Modo (Mode) = 3**

1. Movement with direction and speed defined by parameter *velocità1*, terminated by

- controlled braking as soon as the first encoder index is met.
2. Positioning with speed defined by parameter *velocità1* on the mentioned encoder index.
  3. Adoption of the reached position as new origin of the axis.
- Besides the Home command, commands ArmIndex, ArmHome and SetOrigin are available, permitting to define customized reset sequences. For more information, please refer to the software reference guide.

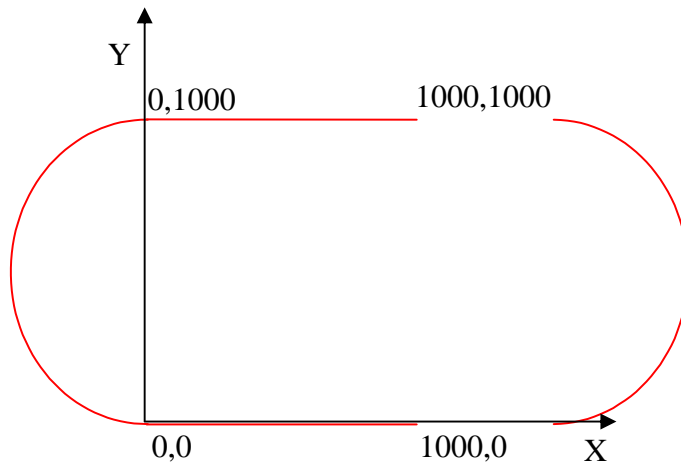
## 2.4 Coordinated movements

### 2.4.1 General information

Coordinated movements permit to move more axes at the same time along a pre-established path in space. The MC6 controller permits to have two systems of coordinated axes at the same time, one for each virtual axis. Each system foresees three Cartesian axes X, Y, Z, plus a rotation axis W, which can be associated to any servo axis or stepper axis of the card by means of the command *ConfigCoord*:

- ConfigCoord 13 2 3 0 8 ; virtual axis 13 coordinates axis 2 (X), axis 3(Y) ; axis 8 (W). Z-axis not present.

Differently from the independent movements, for the coordinated movements it is necessary to define a *path*, that means an assembly of points which must be passed through in succession. The path is defined by a series of *segments* (line segments and circle arcs) each of which starts at the end of the previous segment. Here below an example of a path and the commands used to define it:



- BeginCoord 13 ; start of the path associated to virtual axis 13
- LineAbs 13 1000 0 0 0 ; linear segment at position x=1000, y=0  
1000
- ArcAbs 13 1000 500 180000 0 ; 180° arc with centre x=1000,y=500  
1570
- LineAbs 13 0 1000 0 0 ; linear segment at position x=0, y=1000  
1000
- ArcAbs 13 0 500 0 180000 0 ; 180° arc with centre x=0,y=500  
1570

## SFERA srl – MC6 axis controller

- Profile 13 1000 100000 200000 ; sets speed and accelerations
- MoveAbs 13 5140 ; positioning at the end of the path
- WaitEnd 13 ; waiting for completion

Each segment has got a *starting position* (coordinates of the starting point), a *final position* (coordinates of the end point) and a *length* (space passed through along the segment). The entire path, too, has got its starting position, its final position and its length, the latter corresponding to the sum of the lengths of the segments it is made up of.

Once the path has been defined, the axes are bound to move along such path, like a train which is bound to move along a track. As it happens for the railway lines, any point along the path is identified by the distance from the starting station (ex. line-inspector's house at km 127), as if the path was "straightened" and aligned with a reference axis. Therefore it is natural to consider a coordinated movement as the assembly of the path and of a virtual axis used to move along the curvilinear coordinate by using the commands already available for the independent axes.

The origin of the virtual axis is automatically redefined in the current position by the command *BeginCoord*, so that the 0 position corresponds to the starting point of the path. When moving the virtual axis in positive direction, the coordinated axes are automatically moved along the path in such a way that the curvilinear coordinate (the covered distance is calculated by using the common mathematical formulae) coincides with the current position of the virtual axis. Scalar speed, acceleration and deceleration (i.e. the component of speed, acceleration and deceleration in path direction) exactly correspond to those of the virtual axis.

The virtual axis may be controlled by using all the commands to be used for the independent axes, for example *MoveRel*, *MoveAbs*, *Jog*, *Vel*, *Acc*, *Dec*, *Smoothing*, *Profile*, *Start*, *Stop*, *GetPos*, but exclusively in positive direction, otherwise an error on the coordinated axes will be generated. Once the end of the path has been reached, the coordinated axes stop, even if the virtual axis proceeds further.

In order to execute the path of the previous example move the virtual axis 13 into position 5140 (the linear segments have got length 1000, the half-circles have got length  $500 \cdot 3.14 = 1570$ , therefore  $D = 1000 + 1570 + 1000 + 1570 = 5140$ ). All commands defining a segment return its length, which can be used in the application program for synchronization of the events (for example the activation of an output), as soon as particular positions are reached.

The movement can be stopped before it has come to the end by giving the *Stop* command to the virtual axis. Afterwards it can be restarted by using the *Start* command or, as an alternative, it is possible to cancel the remaining part of the movement by giving a further command *BeginCoord*.

### 2.4.2 Linear segments

To define a linear segment it is sufficient to specify the coordinates of the final position of the segment in absolute or relative terms at the final position of the previous segment. Please note: the W-axis is considered as a fourth Cartesian axis and therefore it is significant for the calculation of the segment length.

The commands are:

- LineAbs *<handle>* *<xf>* *<yf>* *<zf>* *<wf>*
- LineRel *<handle>* *<dx>* *<dy>* *<dz>* *<dw>*

### 2.4.3 Circular movements

To define a circular movement it is sufficient to specify the coordinates of the centre in absolute or in relative terms at the final position of the previous segment, the rotation angle and the W-axis rotation.

The commands are:

- ArcAbs *< handle >* *<center\_x>* *<center\_y>* *<angle>* *<dw>*
- ArcRel *< handle >* *<center\_dx>* *<center\_dy>* *<angle>* *<dw>*

The W-axis rotation is not taken into consideration when calculating the segment length; it may be useful to keep the W-axis tangent to the circle during the movement.

### 2.4.4 Continuous movements

The MC6 controller permits to define the path in real time, adding segments as other segments are completed. Such characteristic permits to execute paths made up of an arbitrary number of segments and therefore to approximate an arbitrary curve with a series of small linear segments. For such purpose use command *Segments* which returns the number of free segments (inside the card it is possible to memorize up to 500 per virtual axis).

When using this operating mode proceed with caution because it is the responsibility of the application program to define the new segments within the time-limit. Otherwise the coordinated axes will be stopped immediately in the final point of the last segment.

## 2.5 Special functions

### 2.5.1 Electronic gearing

The function *electronic gearing*, also called *electric shaft* or *gearing*, essentially lies in moving a *slave* axis with a speed which is proportional to the speed of a *master* axis. Once the gear ratio has been defined and the *gearing* has been enabled, the *slave* axis will move as if there was a mechanical transmission connecting it to the *master*.

Use the following command:

- Gearing *<Slave>* *<Master>* *<Num>* *<Den>*

where *Slave* and *Master* make out the interested axes, whilst *Num* and *Den* define the gear ratio.

For example:

- Gearing 4 1 2000 3000

defines the gearing between the axis 4 (slave) and the axis 1 (master). The axis 4 will move for 2000 encoder counts for every 3000 counts passed through by the axis 1.

The MC6 controller permits to define a gearing for every axis (as slave) that can use as master the nominal or the actual position of any other axis. Each gearing has got its own gear ratio which may also be negative: in this case the slave will move in direction opposite to the master.

The gearing can be disabled by indicating as master the 0 axis.

### 2.5.2 Electronic clutch

The function of *electronic friction* or *clutch* is used together with the gearing and it permits, exactly as a mechanical clutch, to gradually engage or disengage the transmission with the moving master axis, avoiding to abruptly accelerate the slave axis.

An important characteristic of the command is that it realizes engagement/disengagement of the clutch according to the master position, guaranteeing that the transition will take place in the interval comprised between a starting position (of the master) and a specified final position.

A further important characteristic is that the space passed through by the slave in the engagement/disengagement phases is known a priori, thus permitting coupling to the master with a position relation or in a pre-established phase; this characteristic for example permits to realize in an extremely simple and precise way applications for flying cutting.

To engage the clutch use command:

- ClutchOn <Slave> <Mode> <BeginPos> <EndPos>

where *Slave* identifies the axis, *Mode* indicates if the positions are absolute or relative, *BeginPos* indicates the position of the master where clutch engagement will start, *EndPos* the position of the master where clutch engagement will end. Space passed through by the slave in acceleration is:

$$S = \frac{(EndPos - BeginPos)}{2} \square GearRatio$$

For example:

- ClutchOn 4 1 7000 10000

Engages the gearing between slave 4 and the corresponding master when latter passes through the segment comprised between the absolute positions 7000 and 10000. In practice, the slave remains stopped until the master has reached position 7000, then it accelerates in an adequate way in order to be in gearing with the gear ratio requested for the moment in which the master reaches position 10000. In case of gear ratio 2000/3000, the accelerating slave would cross the space:

SFERA srl – MC6 axis controller

$$S = \frac{(10000 - 7000)}{2} \square 2000/3000 = 1000$$

Therefore, in the moment in which the master reaches position 10000, the slave reaches position 1000; from this point onwards, the two axes are coupled with the gear ratio 2000/3000.

The command permitting clutch disengagement works in the same way:

- ClutchOff <Slave> <Mode> <BeginPos> <EndPos>

For example:

- ClutchOff 4 0 0 1000

Clutch disengagement is started immediately (relative 0 position), terminating between 1000 encoder counts of the master.

Please note: the two commands can be active at the same time, on condition that the clutch engagement and disengagement intervals are not overlapping.

The electronic clutch requires a movement of the master in positive direction.

When the axis controller is turned on, per default the electronic clutches of all axes are engaged.

### 2.5.3 Electronic cam

The electronic cam function lies in moving a *slave* axis according to the position of a *master* axis on the basis of a law defined by a table where the desired cam profile is memorized. This is the typical behaviour of a mechanical cam where the angular position of the shaft and the shape of the cam determine the position of the driven part.

Each axis (including the steppers and the virtual axes) can be coupled as *slave* of any other axis (*master*) of the card. It is also possible to automatically engage or disengage each cam as soon as the corresponding master axis reaches a pre-established position.

To use this characteristic, first of all it is necessary to define the cam shape under form of a table, for example by using a spreadsheet of the PC. The number of points contained in the table is arbitrary and depends on the desired degree of precision; normally some tenths of points are sufficient since the MC6 controller automatically generates all the intermediate values per linear interpolation, nevertheless it is possible to define tables containing thousands of points.

The above mentioned tables are situated in memory blocks of the controller on which it is possible to write in remote mode (via can bus or serial connection) for a maximum of flexibility. Obviously, in the application program these tables can also be calculated by algorithms.

The command to be used to define a cam is the following:

Cam <Handle> <Master> <Modulo> <Block>

SFERA srl – MC6 axis controller

where *Handle* identifies the slave axis, *Master* identifies the master axis, *Modulo* is the number of counts of the master axis corresponding to a turn of the cam, *Block* is the handle of the block containing the cam profile.

Then the cam must be inserted by using the command:

- Engage <*Handle*> <*Mode*> <*Pos*>

where *Handle* identifies the axis, *Mode* indicates if the position is absolute (1) or relative (0), *Pos* identifies the master position in which the cam must be engaged. For example, in order to immediately engage the cam in which axis 2 is the slave:

- Engage 2 0 0

In the same way we can disengage a cam by using the command:

- Disengage <*Handle*> <*Mode*> <*Pos*>

For the meaning of the various parameters see command *Engage*.