QSG_0100_EN Rev. A

QUICK START GUIDE

FB EASY POSITIONING

OMRON QUICK START GUIDE

"Omron QSG" is a collection of interactive information, which allows a quick consultation of the main information you need to use the OMRON devices. "Omron QSG " does not want to replace the use of the manuals, but must be regarded as supplementing the manuals themselves.

INDEX

1 INTRODUCTION

1.1 DOCUMENT PURPOSE

This document explains the criteria of applicability and the method of operation, of the Function Block library named 'Easy Positioning'. This library includes a collection of function blocks created to perform open loop and closed loop axis control in an ‘easy’ way, through pulse train, analog etc. from simple electronic axis link to advanced positioning functions.

This guide refers to Function Block version 1.0_. To check the function block version, right click on the desired function block and select "Properties".

Note: Before using the FB, check the rules on use (Section 2.4)

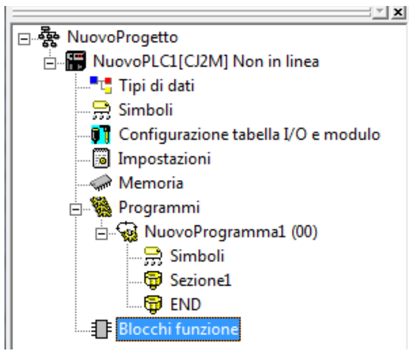
For detailed information on the use and general configuration of Omron products, always refer to the official manuals.

1.2 HARDWARE AND SOFTWARE REQUIRED

Supported CPU:	CJ1M/CJ2M series or, CP1H, CP1L (L-L or L-M).
	NOTE: Some models require specific CPU FBs and/or additional units (i.e. CJ2M-MD21_).
Software:	Use CX-Programmer version 5.0 or higher.

1.3 FB IN IMPORT OF CX-PROGRAMMER

In the workspace, right-click the icon "Function Blocks"



From the context menu, select the "Insert Block function", then "From File ..."

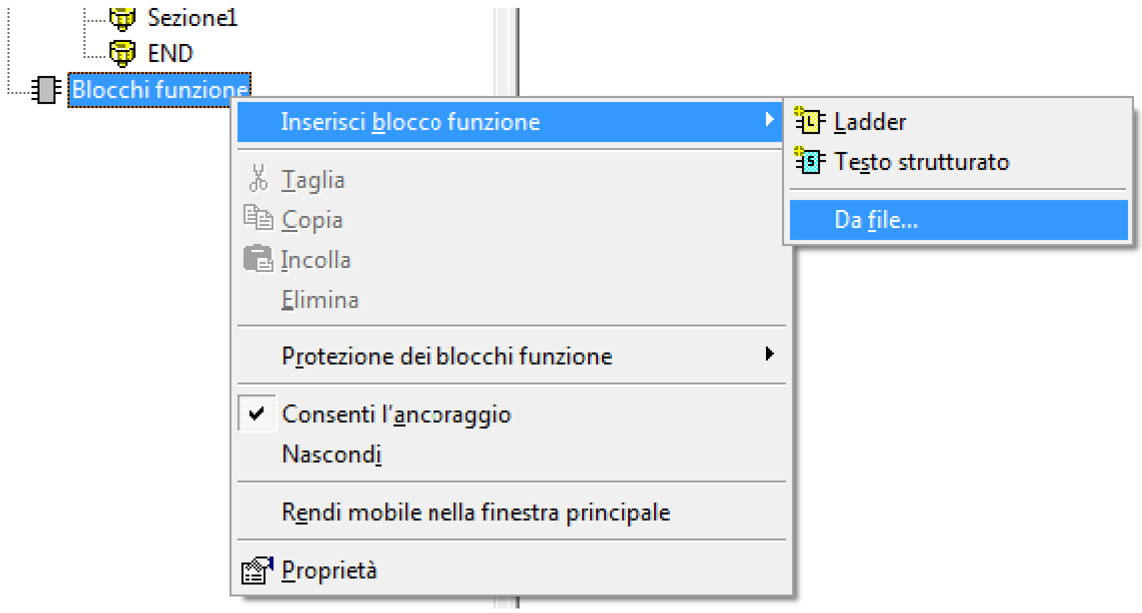
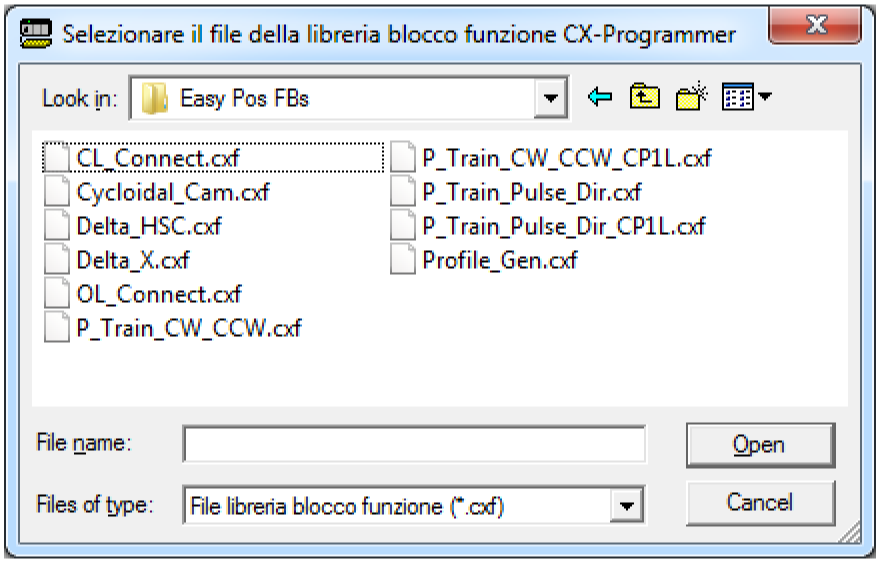


Figure 1: Inserting the definitions of Function Block

Select the file/s you need:


Selezionare il file della libreria blocco funzione CX-Programmer

Look in: Easy Pos FBs

CL_Connect.cxf
Cycloidal_Cam.cxf
Delta_HSC.cxf
Delta_X.cxf
OL_Connect.cxf
P_Train_CW_CCW.cxf
P_Train_CW_CCW_CP1L.cxf
P_Train_Pulse_Dir.cxf
P_Train_Pulse_Dir_CP1L.cxf
Profile_Gen.cxf

File name:

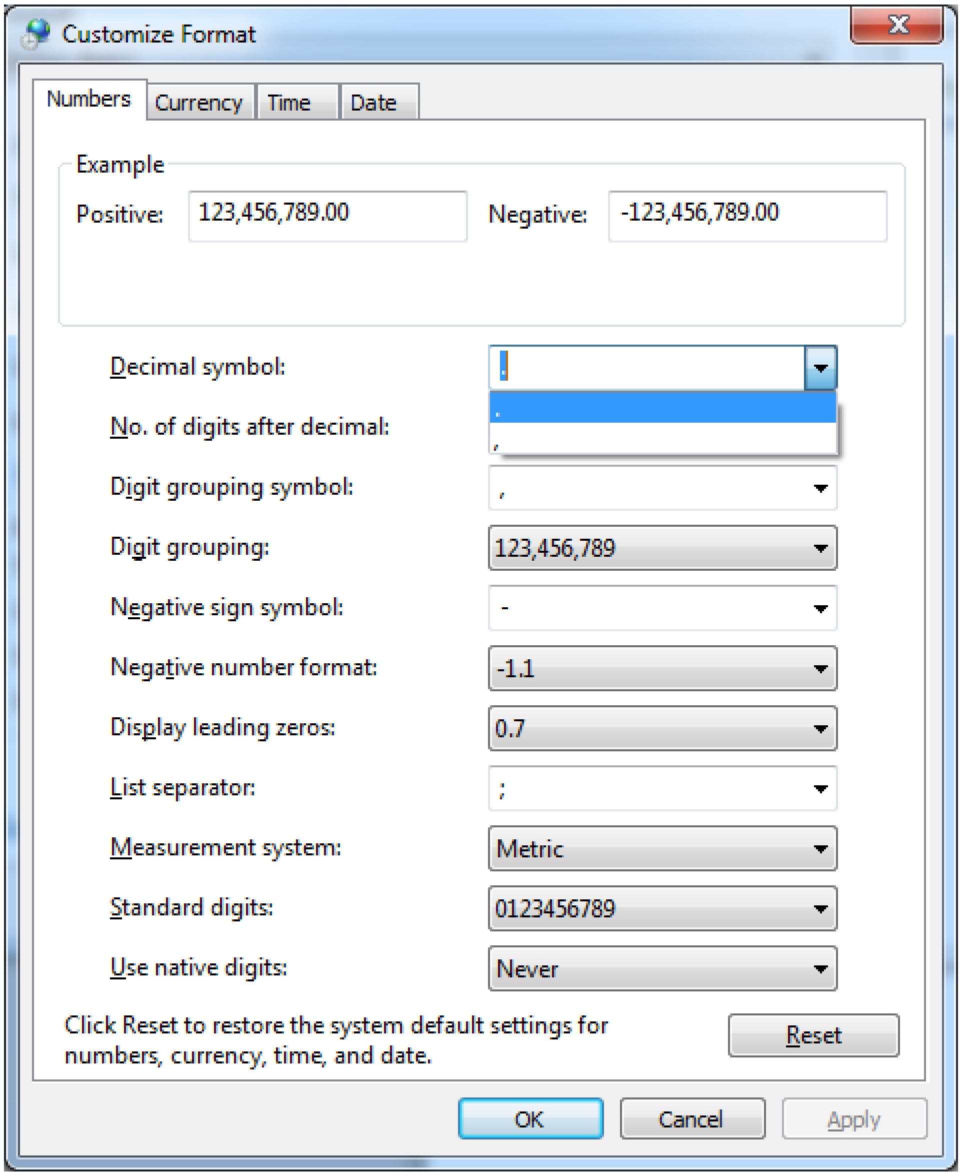
Files of type: File libreria blocco funzione (*.cxf)

Figure 2: Window selection of function blocks

1.4 PC SETTINGS

Most of the function blocks use in their code REAL type constants.

To avoid compilation errors, check that the dot character (and not the comma) is specified as Decimal Symbol in your local PC settings:



Example
Positive: 123,456,789.00
Negative: -123,456,789.00

Decimal symbol: .
No. of digits after decimal:
Digit grouping symbol: ,
Digit grouping: 123,456,789
Negative sign symbol: -
Negative number format: -1.1
Display leading zeros: 0.7
List separator: ;
Measurement system: Metric
Standard digits: 0123456789
Use native digits: Never

Click Reset to restore the system default settings for numbers, currency, time, and date.

Figure 3: Settings window (Windows 7)

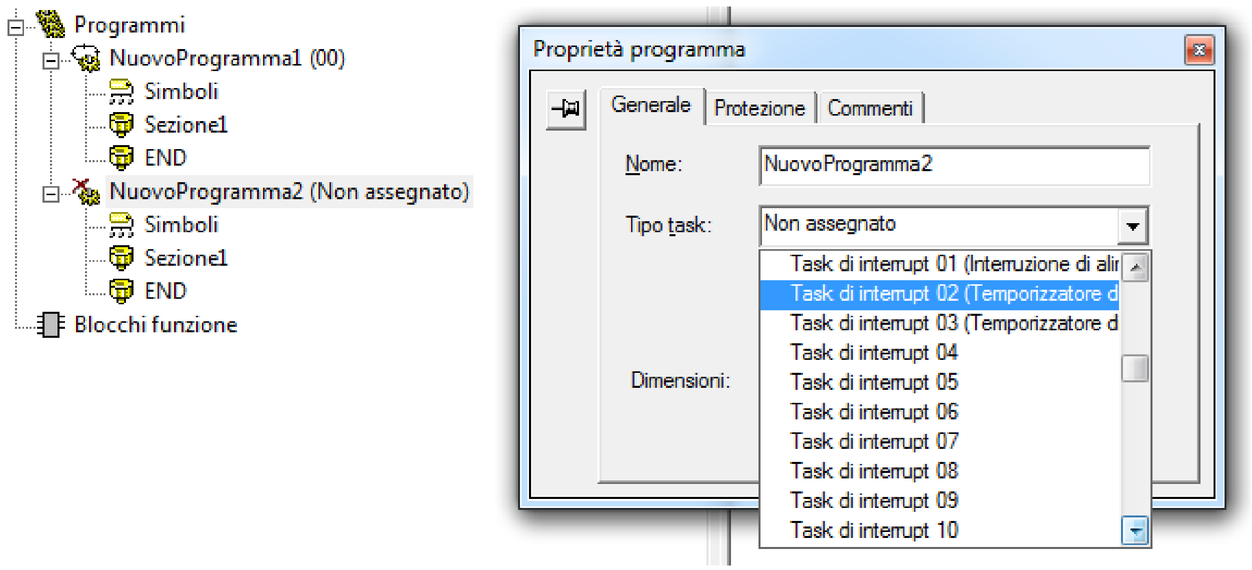
2 GETTING STARTED

All function blocks must be used within a scheduled interrupt task (a program, which execution is 'forced' by the CPU, at pre-fixed interval of time, through a software interrupt). The interval time defines the frequency of the positioning control.

Note: Although the interval time range should be evaluated from time to time, based on the real application requirements, it is possible to state that an interval times of 10 milliseconds (or more) is enough for many applications.

2.1 SCHEDULED TASK

To define a Scheduled Interrupt task, insert a new program, then right click on it; select 'Properties', and assign the task as described below:

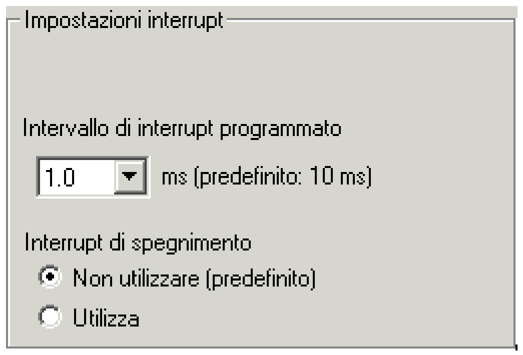

Programmi
NuovoProgramma1 (00)
Simboli
Sezione1
END
NuovoProgramma2 (Non assegnato)
Simboli
Sezione1
END
Blocchi funzione

Proprietà programma
Generale | Protezione | Commenti
Nome: NuovoProgramma2
Tipo task: Non assegnato
Task di interrupt 01 (Interruzione di alir
Task di interrupt 02 (Temporizzatore d
Task di interrupt 03 (Temporizzatore d
Task di interrupt 04
Task di interrupt 05
Task di interrupt 06
Task di interrupt 07
Task di interrupt 08
Task di interrupt 09
Task di interrupt 10
Dimensioni:

Figure 4: Configuring the task "NuovoProgramma2" as scheduled task

Note: The PLCs to which these function blocks are applicable support at least one scheduled interrupt tasks. The interrupt task must be enabled with an instruction named "MSKS"; the interval time value, specified as the second operand, depends on the base value selected in the PLC settings.

For example, to run the scheduled task to interrupt every 5 milliseconds, the combination of PLC settings and MSKS operand, should be as follows:



Impostazioni interrupt

Intervallo di interrupt programmato

1.0 ms (predefinito: 10 ms)

Interrupt di spegnimento

[✓ Non utilizzare (predefinito)] Utilizza

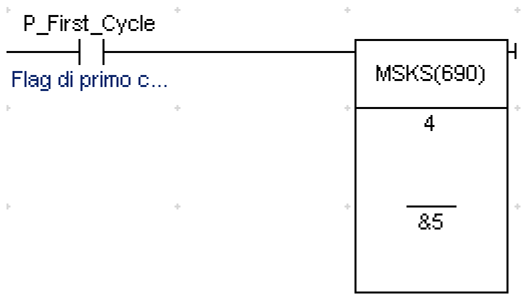


Figure 5: Enabling a scheduled task every 5 milliseconds

For more details, refer to the instruction manual of the PLC used.

2.2 UNDERSTANDING FBs

All the control and generation of motion profiles must run <u>within the scheduled task</u>.

In principle, the calculations required to realize the function of electronic axis refer to the position variations (delta positions) of each axes, compared to the position detected at the previous execution of the scheduled task.

If, within the time interval, the Master axis physically moves to a specific distance (equivalent to a number of encoder pulses) the axis (or the axes) Slave will move accordingly.

In the case of open loop control, the information coming from the axis Master (delta Master) is enough.
In the case of closed loop control, the space covered by the axes Slave (delta Slave) needs to be known (feedback pos.) to perform the appropriate corrections.

Each of the axes, generate then a 'position delta'. Specific function blocks named 'Delta' calculate these position variations, needed to further FB (named 'Connect') to perform the position loop control returning a command value (this value is the command value for the Slave axes). Other FB (i.e. 'Pulse_Train' FBs) turn the command value into real pulse trains, or analog value.

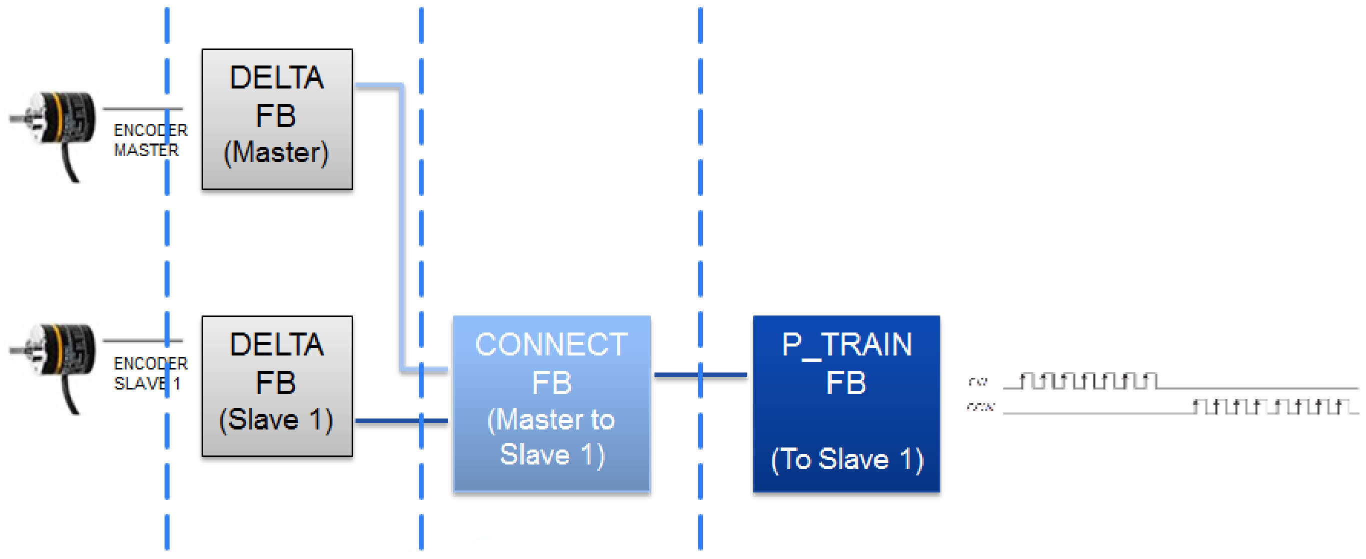
Basically you can then consider three types of basic function blocks:

1 - Blocks that calculate the position of the delta (delta FB)
2 - Blocks that create electronic (logic) axis (FB Connect)
3 - Blocks that generate pulse trains (or values for analog boards)

The following is a graphical representation of the three types of function blocks, for the creation of an electronic shaft in a closed loop control between a Master axis (encoder Master) and one (or two) Slave axis (each Slave axis has its own feedback encoder).

The function block input order, in the ladder, goes from the left side to the right side (the ones on the left are the first to be used).

Examples:
a) Electronic shaft (closed loop) between a Master encoder and a Slave axis (pulse train command):



b) Electronic shaft (closed loop) between a Master encoder and two Slave axes (pulse train command):

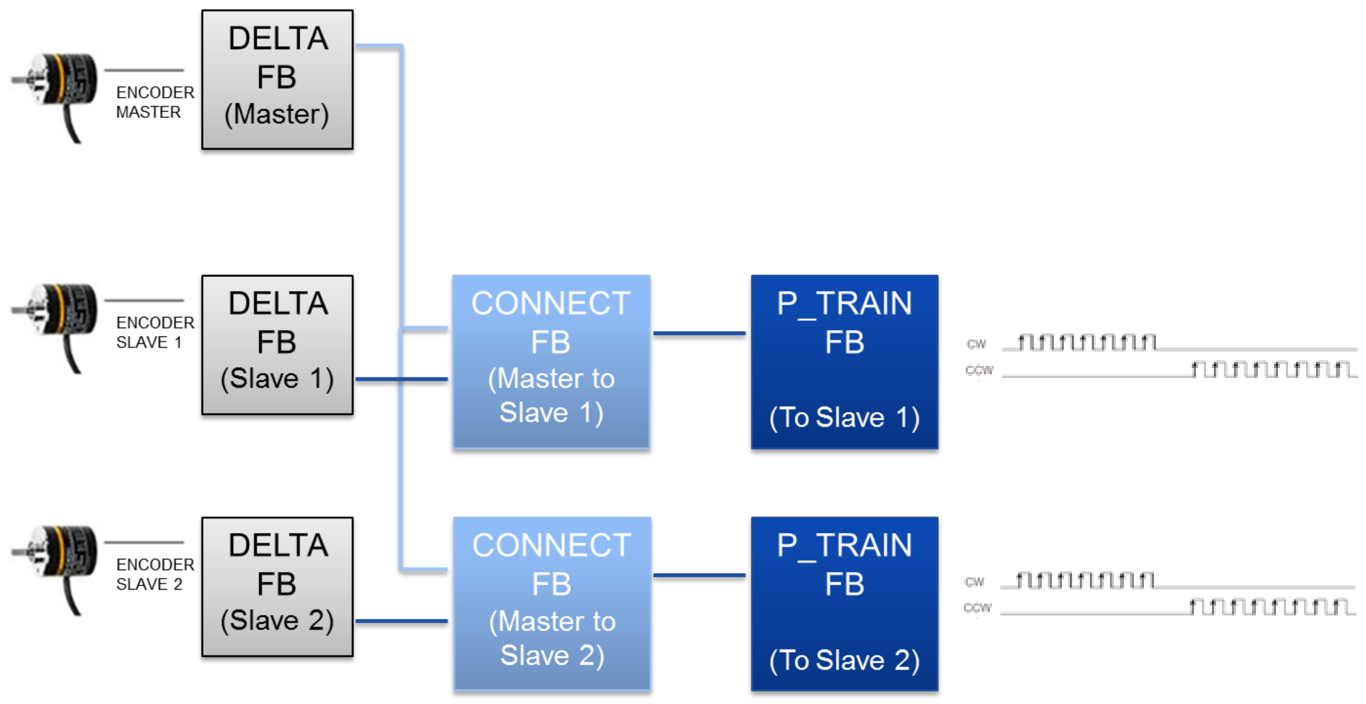


Figure 6: Example of use of three types of FB to control Slave axes

2.3 PLC SETTINGS

The only settings required (in addition to those related to the interval range of interrupt tasks, in Figure 5) concern the high-speed counters.

Each high-speed counter wired to an encoder (no matter whether related to a Master axis or to a Slave axis) must be set in circular mode with a maximum value of count equal to or greater than 9,999 (in the case of less than 100 milliseconds scheduled task). The bigger is this value the better is for the program calculation.

Note: a maximum count pulses value of 9,999 corresponds to 10,000 total pulses (since the count starts from 0).

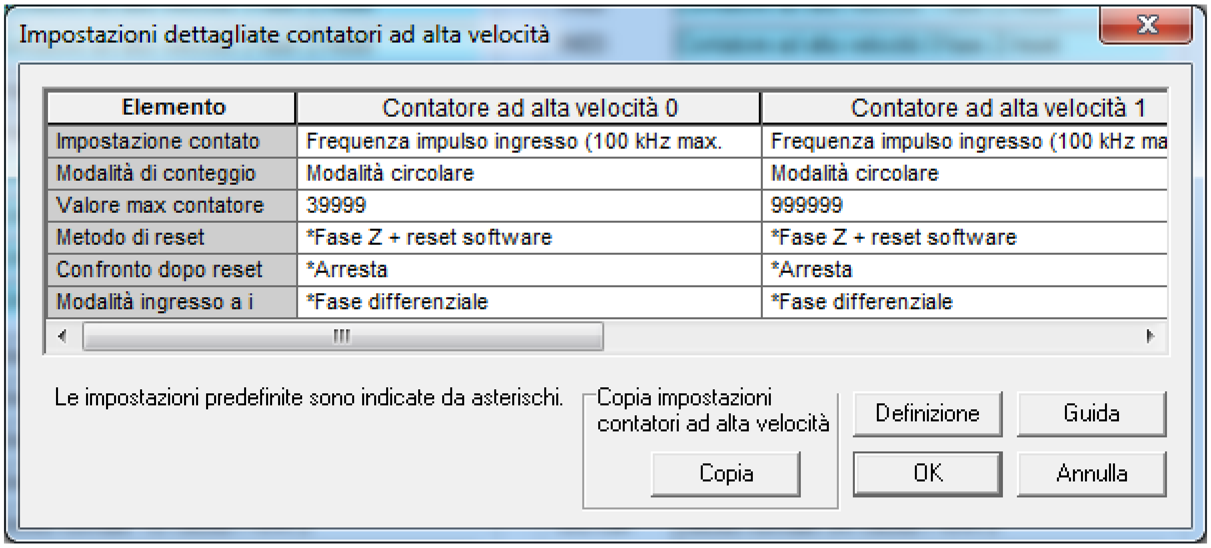
The circular mode has been used to manage electronic axes which proceed indefinitely in the same direction, without incurring count overflow and then positioning errors.

When referring to a Master, the maximum value of count may represent a significant data (such as the maximum linear stroke of the axis, etc.)..

In the case of a Slave axis, it does not have any particular meaning than to prevent an overflow of the HSC. So, high values are certainly better (in terms of program execution). Anyway the maximum value of count of the Slave axes could also coincide with the axis Master.

Anyway it is recommended to work in circular mode ALWAYS.

In the figure below an example of setting up two high-speed counters:


Impostazioni dettagliate contatori ad alta velocità

Elemento	Contatore ad alta velocità 0	Contatore ad alta velocità 1
Impostazione contato	Frequenza impulso ingresso (100 kHz max.	Frequenza impulso ingresso (100 kHz ma
Modalità di conteggio	Modalità circolare	Modalità circolare
Valore max contatore	39999	999999
Metodo di reset	*Fase Z + reset software	*Fase Z + reset software
Confronto dopo reset	*Arresta	*Arresta
Modalità ingresso a i	*Fase differenziale	*Fase differenziale

Le impostazioni predefinite sono indicate da asterischi.

Figure 7: Example of setting up two high-speed counters

In Figure 7, the count value of HSC 0 will be between 0 and 39 999 (and thus it will correspond to a distance per revolution - or "Rep_Dist" - of 40,000 pulses)

For the HSC 1, the "Rep_Dist" will correspond instead to 1,000,000 (being the count value between 0 and 999,999).

It's important to remember the relationship between the maximum count value and the "Rep_Dist" of each counter (see the next section).

2.4 MAIN RULES

The main rules for the proper operation of function blocks are the following:

1: Always set the involved High Speed Counter to circular mode as explained in the previous section

1: The 'Flag of Always On' PLC ("P_On") must be used as FB enable signal (input "EN"). Each FB is provided with an input, named "Start", which determines the real operation.

2: The inputs named "Start" <u>must be set to ON at least one scan after the interrupt task enabling scan.</u>

To avoid mistakes, it's then recommended to enable the scheduled interrupt task at the first scan.

3: Some function blocks have an input labeled "Rep_Dist". This input is linked mathematically to the HSC maximum circular value which the function block refers to.
As explained in the previous section, the value of "Rep_Dist" is obtained by adding 1 to the HSC maximum circular value:

Rep_Dist = Max HSC Counter Value + 1

4: Before using the function blocks in a closed loop always check the correspondence between the direction of the motor rotation and direction of the encoder count!

<u>Please observe these rules to avoid unexpected behavior of the axes!</u>

3 BASIC FB REFERENCE GUIDE

Below the details of the basic function blocks (shown in Figure 6).

3.1 DELTA_HSC (**)

This function block calculates the difference between the actual value of the specified HSC and the value captured at the previous FB execution.

The calculation is performed if the "Start" input is On, and the result will be returned in the FB output named "Delta_Pos". If "Start" is Off, the "Delta_Pos" output value is zero.

Using the function block within a scheduled interrupt task, it then returns the number of pulses generated by the specified high-speed counter within the task interval time.

Note: the "Rep_Dist" input must match the value set for the maximum circular speed counter specified, plus 1 (as described in Section 2.4).

Usage example:

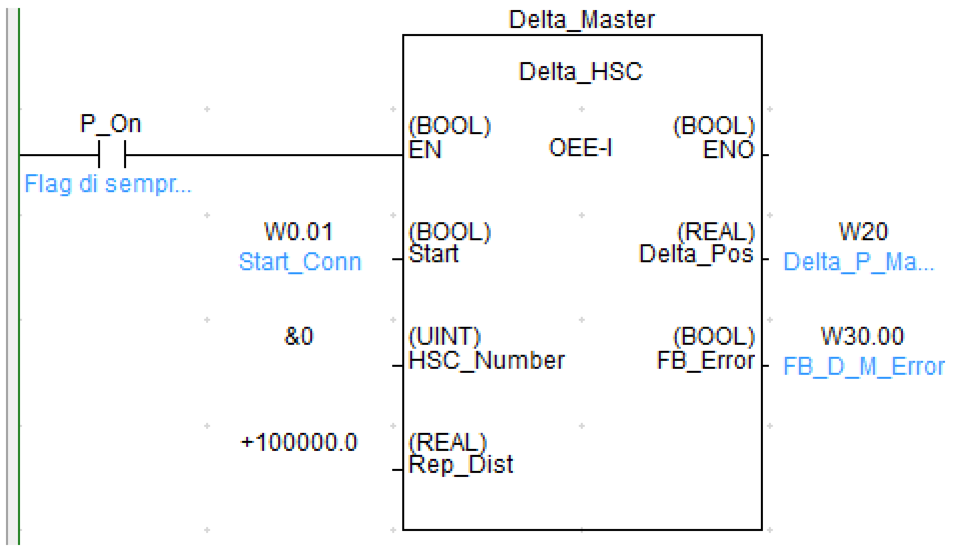


Figure 8: "Delta_HSC" function block example

INPUT VARIABLES:

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block (use the bit "P_On")
Start	BOOL	Starting the calculation of the delta position
HSC_Number	UINT	Number of high-speed counter (0 - 3) to carry out the calculation
Rep_Dist	REAL	Maximum count value set for the specified high-speed counter, plus one

OUTPUT VARIABLES:

NAME	TYPE	DESCRIPTION
ENO	BOOL	Function block enabling state (*)
Delta_Pos	REAL	Calculated delta position
FB_Error	BOOL	Error of the input parameters (HSC_Number out of threshold or Rep_Dist <+10,000)

(*): Only for diagnostic purposes
(**): For CJ1M, use the FB named Delta_HSC_CJ1M

3.2 DELTA_X

Unlike the function block "Delta_HSC" this function block calculates the difference between the actual value of a specified memory address (and not of a high-speed counter) and the value of the same address at the previous execution of the block.

These FB are useful if you need to handle, for example, a position profile of a virtual axis (mathematically generated) or a position information not coming from an HSC (i.e. from NC unit..).

The calculation is performed if the "Start" input is On, and the result will be returned in the FB output named "Delta_Pos". If "Start" is Off, the "Delta_Pos" output value is zero.

Using the function block within a scheduled interrupt task, it then returns the difference of the specified memory address value, between two FB executions.

Note: the "Rep_Dist" input must correspond to the maximum count reached by the channel (as described in Section 2.4).

Usage example:

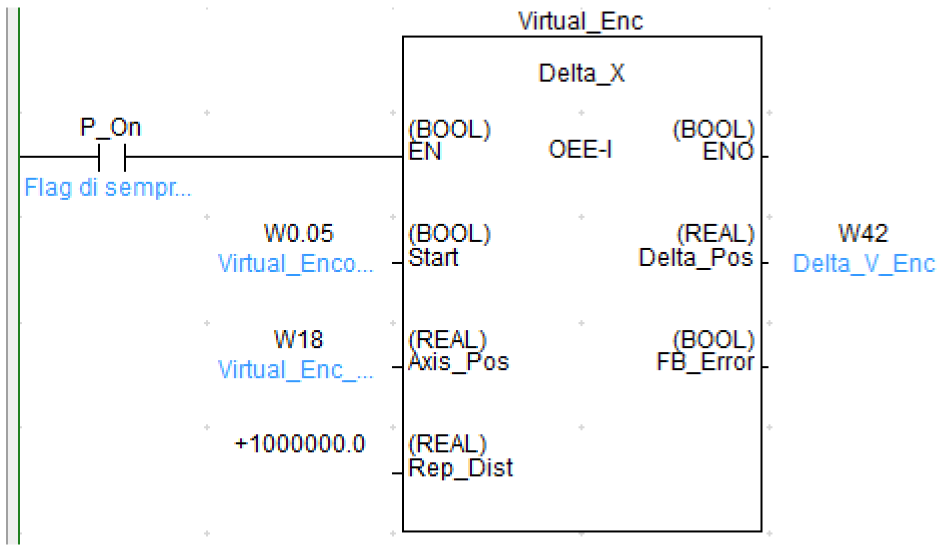


Figure 9: "Delta_X" function block example

INPUT VARIABLES:

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block (use the bit "P_On")
Start	BOOL	Starting the calculation of the delta position
Axis_Pos	REAL	Channel containing the value (of type REAL) position on which to calculate
Rep_Dist	REAL	Maximum value reached by the channel "Axis_Pos"

OUTPUT VARIABLES:

NAME	TYPE	DESCRIPTION
ENO	BOOL	Function block enabling state (*)
Delta_Pos	REAL	Delta position calculated
FB_Error	BOOL	Error of the input parameters (Axis_Pos <0, Rep_Dist <= 0)

(*): Only for diagnostic purposes

3.3 OL_CONNECT

Based on the delta position of a Master axis (calculated with one of the previous FB) this function block calculates the command value of the link (Slave) axis, to realize a function of 'electronic axis' speed link in open loop (hence the prefix 'OL'). The open loop mode is ideal for creating a speed electronic axis link (Such as OL Inverters). The accuracy of the profile is relatively small.

It's important to underline that this function block only perform a calculation and that, physically, does not act on any output of the PLC.
The resulting number may therefore be combined not only to an output pulse train but for example, after appropriate scaling, to an analog output, or to a serial command (e.g. Modbus).

The calculation will be done only if the input of "Start" is On, and the result will be returned in the FB output named " Speed_Hz ".
If "Start" is Off, the value of "Speed_Hz" will be zero.

Usage example:

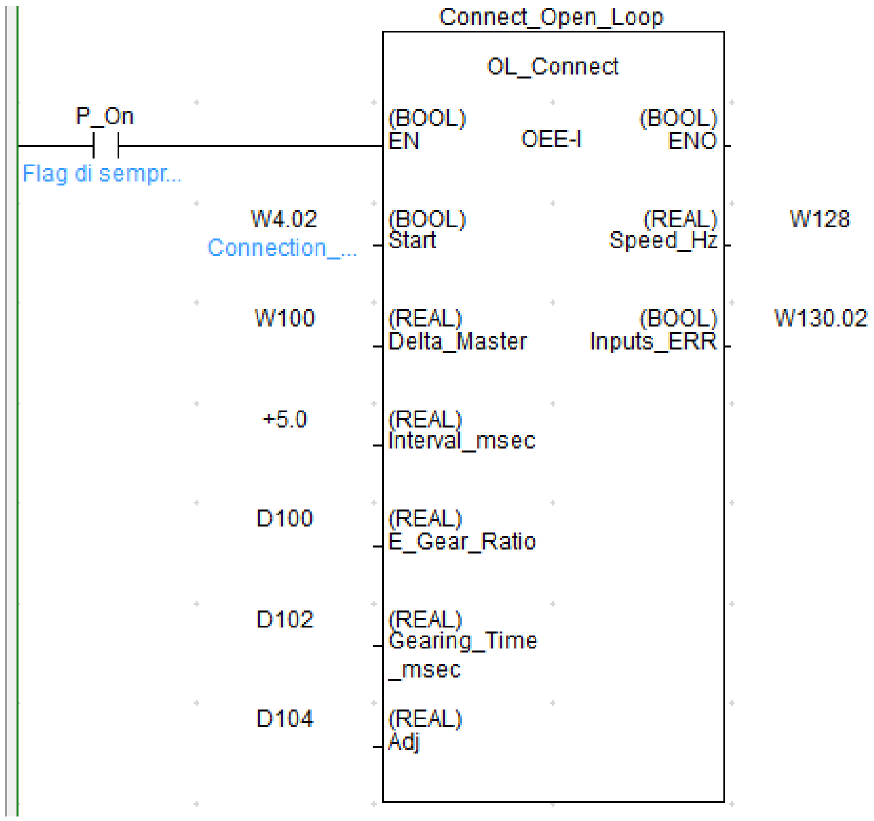


Figure 10: Example of use of the function block "OL_Connect"

The block "OL_CONNECT" manages the relationship of electronic gear between Master axis and Slave axis (E_Gear_Ratio), the time in milliseconds required to give effect to a gear ratio change (Gearing_Time_msec) and an input of Adjustment. (Adj), to vary the actual speed in percentage (1 = 100%).

INPUT VARIABLES:

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block (use the bit "P_On")
Start	BOOL	Starting the calculation of the frequency
Delta_Master	REAL	Change in position made in the time axis Master scheduled task
Interval_msec	REAL	Time interval (in milliseconds) of interrupt tasks scheduled within which the block is used
E_Gear	REAL	Speed ratio Slave / Master
Gearing_Time_msec	REAL	Time (in milliseconds) needed to make a real change 'on the fly' electronic report of the Slave / Master
Adj	REAL	This input is useful to obtain percentage changes of speed without changing the electronic report (value 1 corresponds to 100%)

OUTPUT VARIABLES:

NAME	TYPE	DESCRIPTION
ENO	BOOL	Function block enabling state (*)
Speed_Hz	REAL	Real number corresponding to the output frequency (in Hertz) in the Slave axis should move to 'chase' the Master axis, respecting the electronic report and set FF_Gain
Inputs_Error	BOOL	Error values in input: (input 'Interval_msec'e' Gearing_Time_msec '<0)

(*): Only for diagnostic purposes

3.1 CL_CONNECT

Based on the delta position of a Master axis (calculated with one of the 'delta' FBs) this function block calculates the command value to control a Slave axis, to realize a closed loop (hence the prefix 'CL') 'electronic axis' link controlled in position.

Since the function block implements a closed loop control, in addition to the delta position of the axis of the Master, also the delta position of a Slave should is required.

For this reason, if you need to manage the position information coming from two physical encoders, you need two FB type 'Delta_HSC' (one for the Master axis encoder and one for Slave axis encoder) before the 'CL_CONNECT' FB instance, as showed in the picture below:

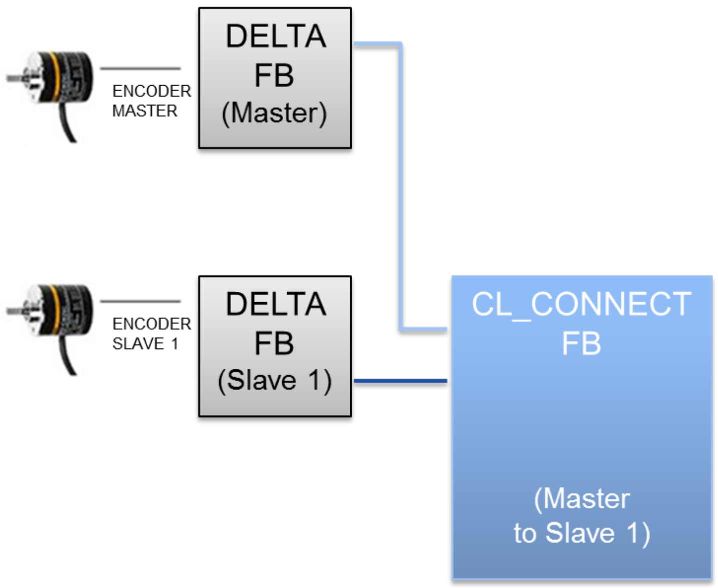


Figure 11: For a closed loop control requires two FB Delta

It's important to underline that this function block only perform a calculation and that, physically, does not act on any output of the PLC.
The resulting number may therefore be combined not only to an output pulse train but for example, after appropriate scaling, to an analog output, or to a serial command (e.g. Modbus).

The calculation will be done only if the input of "Start" is On, and the result will be returned in the FB output named " Speed_Hz ". If the "Start" input is Off, the value of "Speed_Hz" will be zero.

Usage example:

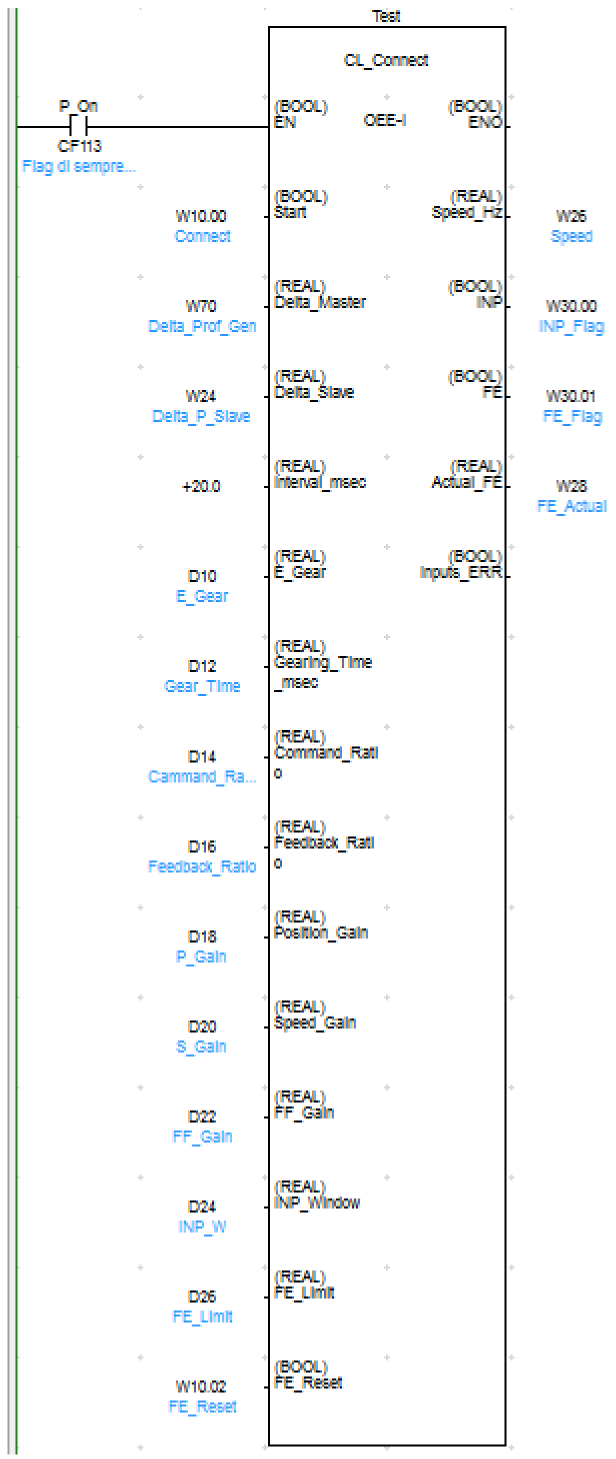


Figure 12: Example of using a "CL_Connect"

As the block "OL_CONNECT" does, this FB manages the electronic gear ratio between the Master and the Slave axis (E_Gear) and the time, in milliseconds, required to give effect to a change in the gear ratio (Gearing_Time_msec).

In addition to the above FB, the block 'CL_Connect' supports all of the functions needed for the closed loop control, such as Position and Speed gains, Feed Forward. Also, relationships between control values and feedback values, 'In Position' window and Following Error are managed.

INPUT VARIABLES:

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block (use the bit "P_On")
Start	BOOL	Starting the control calculation
Delta_Master	REAL	Master delta position
Delta_Slave	REAL	Slave delta position
Interval_msec	REAL	Time interval (in milliseconds) of interrupt tasks scheduled within which the block is used
E_Gear	REAL	Electronic report Slave / Master **(DEFAULT: 1)**
Gearing_Time_msec	REAL	Time (in milliseconds) needed to make a real change 'on the fly' electronic report of the Slave / Master
Command_Ratio	REAL	Command Slave/Master ratio (see below)
Feedback_Ratio	REAL	Feedback Slave / Master ratio (see below)
Position_Gain	REAL	Gain position (during the initial test set this input to 10)
Speed_Gain	REAL	Speed gain (during the initial test set this input to 10)
FF_Gain	REAL	Feed Forward Gain **(DEFAULT: 1)**
INP_Window	REAL	Window 'In-Position' in Slave pulses. Determines the threshold considered negligible tracking error
FE_Limit	REAL	Tracking error limit (in Slave pulses), beyond which it will switch the output "FE"
FE_Reset	BOOL	Following Error reset bit. The bit is active regardless of the status of the "Start"

OUTPUT VARIABLES:

NAME	TYPE	DESCRIPTION
ENO	BOOL	Function block enabling state (*)
Speed_Hz	REAL	Command value
INP	BOOL	The position of a Slave is inside the 'In_Position' window
FE	BOOL	The following error has exceeded the limit specified in the "FE_Limit" input
Actual_FE	REAL	Value of the actual following error (Slave pulses)
Inputs_Error	BOOL	Wrong input values detected (inputs' Interval_msec' and/or 'Gearing_Time_msec' <0)

(*): Only for diagnostic purposes

3.1.1 COMMAND_RATIO AND FEEDBACK_RATIO

To better clarify the meaning of these two inputs, it's proposed a table with different application cases, and the relative values:

PULSE PER REV (Master Encoder)	PULSE CONTROL FOR MOTOR REV (SERVO)	GEAR	PULSE PER REV (Encoder Feedback)	**COMMAND RATIO**	**FEEDBACK RATIO**
8,000	2,500	None	10,000	2.500/8.000	10.000/8.000
8,000	2,500	3:1	10,000	7.500/8.000	10.000/8.000
8,000	4,000	3:1	4,000	12.000/8.000	4.000/8.000
4,000	1,000	2:1	8,000	2.000/4.000	8.000/4.000

NOTE: Setting up 'Command_Ratio' and 'Feedback_Ratio' to control a servo drive is usually easy.
The calculation of the corrected Command_Ratio, may not be equally immediate if you need to control an inverter (due to asynchronous motor performances).

An 'empirical' method for the calculation of the right Command Ratio to control an inverter (with both analog or pulse train signal) is to use the function block 'CL_Connect' in open loop mode (Position_Gain and Speed_Gain equal to zero) and then to change the Command_Ratio until the following error, at a constant speed, remains constant. At that point, stop the connection, reset the gains to their default value (see below) reset the following error and proceed with the test.

Suggestion: To proceed at constant speed you can simply force a fixed delta value to the 'Delta_Master' input.

NOTE: In all of the cases set **to 1** the initial values of E_Gear and FF_Gain

3.1.2 ADDITIONAL NOTES

- Before controlling any movement, check the correspondence between the direction of rotation of Slave motors and the count direction of the respective encoders (for the test is sufficient to put the inputs and Position_Gain Speed_Gain to 0, and FF_Gain to 1) so as to 'open' l 'position loop ..)

- The correct value of the gains depends on the interval of execution of the scheduled task and the type of the controlled system.

- At the beginning, set the gains of position and speed at low values (typically 5 - 10) and then adjust them carefully, since wrong values can cause motor overshooting.

- If the Master is stopped, and the following error is within the 'In Position' window, the block does not perform any control.

CLOSED LOOP WITHOUT ENCODER:

If the feedback encoder is not available, it's possible to close the position loop within the PLC, using the Delta position returned by the function blocks 'P_Train_D' (see Section 3.2.2).

NOTE: In this case, set the 'Feedback_Ratio' same value 'Command_Ratio' and the Speed_Gain to 1!

3.2 P_TRAIN_ (**)

3.2.1 'P_TRAIN' BLOCKS

The function blocks named with the prefix "P_TRAIN_" generate pulse train output (mode CW / CCW or Pulse + Direction mode) at the frequency indicated by the number (type: REAL) specified in the input named "Speed".

In addition, it manages the maximum output frequency for both the positive and negative directions, and a 'dead band' (symmetrical with respect to 0 Hz) within which the pulses are not generated.

If the "Start" input is Off, the pulse train is NOT generated.

These function blocks complete the logic scheme indicated in Figure 6.

Usage example:

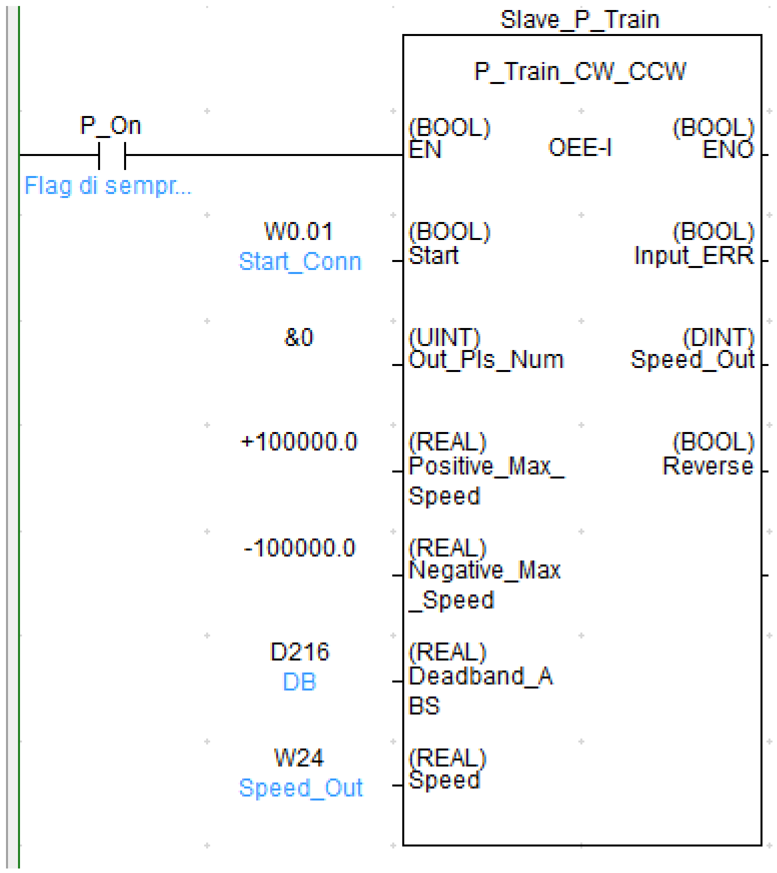


Figure 13: Usage example (output mode: CW / CCW):

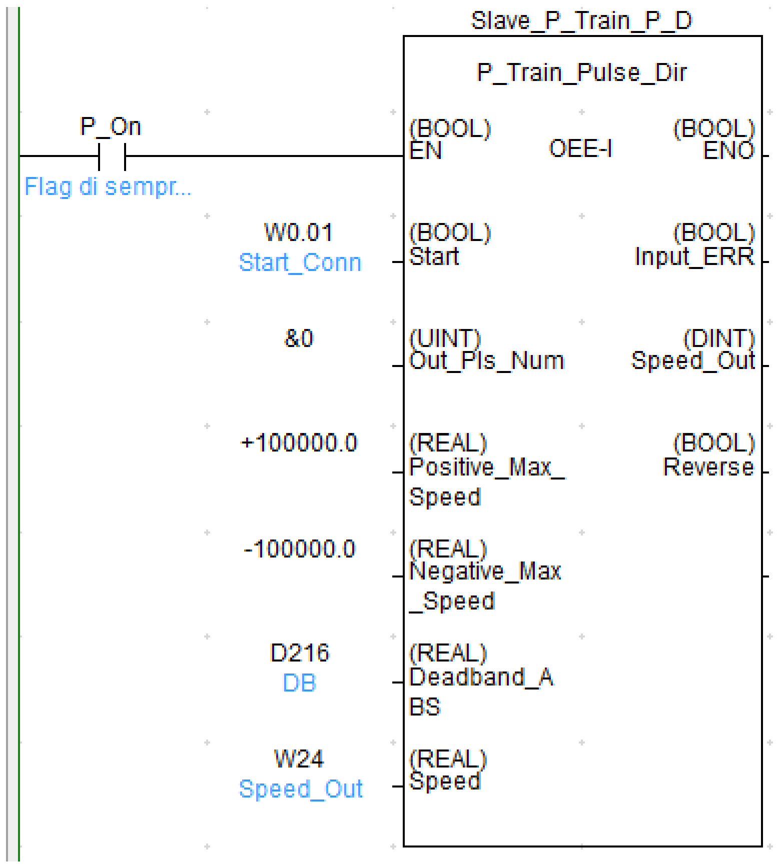


Figure 14: Usage example (output mode: Pulse + Direction)

INPUT VARIABLES (common to both FB):

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block (use the bit "P_On")
Start	BOOL	Pulse Output enabled
Out_Pls_Num	UINT	Number of output pulse train to be used
Positive_Max_Speed	REAL	Maximum output frequency (positive)
Negative_Max_Speed	REAL	Maximum output frequency (negative)
Deadband_ABS	REAL	Value (absolute) of the frequency below which no pulses are generated
Speed	REAL	Frequency at which the pulses must be generated

VARIABLES OUTPUT (common to both FB):

NAME	TYPE	DESCRIPTION
ENO	BOOL	Function block enabling state (*)
Input_ERR	BOOL	Input values out of range
Speed_Out	DINT	Output frequency of the pulse train (*)
Reverse	BOOL	Direction bit (ON if the negative direction) (*)

(*): Only for diagnostic purposes
(**): For CP1L and CJ1M_CPU2x CPUs, use the blocks whose name ends with 'CP1L'.

3.2.2 'P_TRAIN_D' BLOCKS

Function blocks of type 'P_Train_D' return, in the output named 'Delta_P_Out', the variation of the pulses generated from the previous execution.

If necessary, this value can be 'passed' to the function block CL_Connect ('Delta_Slave' input) to create a 'closed loop' control between Master pulses and pulse actually sent (something like PLS2 instruction).

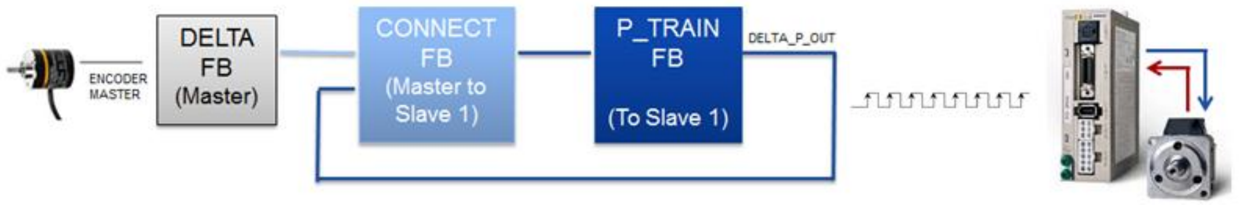


Figure 15: PLC closed loop scheme

IMPORTANT NOTE: If this method is used set 'Feedback_Ratio' to the same value as 'Command_Ratio' in the 'CL_Connect' FB inputs and the Speed_Gain to 1!

Compared to the classic 'open loop' solution, this approach leads to a considerably higher precision control, when servomotors are controlled.

Usage example:

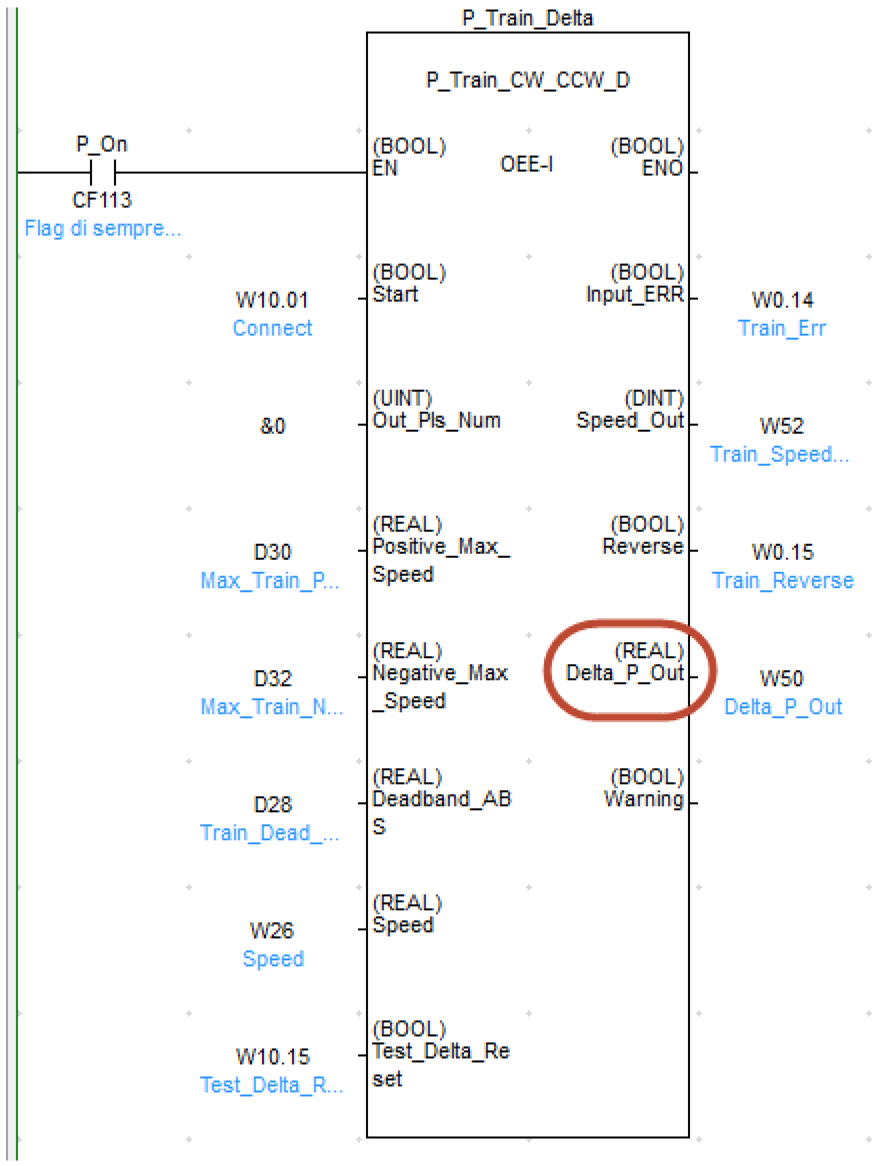
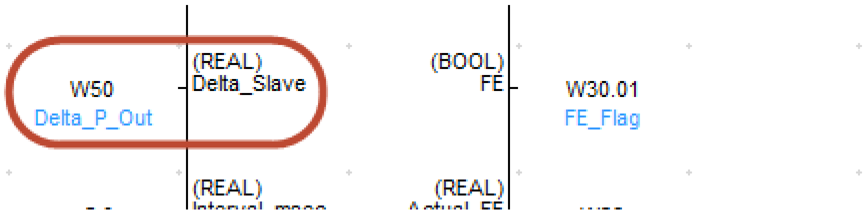


Figure 16: 'P_Train_D' usage example

As showed in Figura 15, to internally close the loop, the Delta_P_Out output can be used as CL_Connect Delta_Slave input:



INPUT VARIABLES (common to both FB):

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block (use the bit "P_On")
Start	BOOL	Pulse Output enabled
Out_Pls_Num	UINT	Number of output pulse train to be used
Positive_Max_Speed	REAL	Maximum output frequency (positive)
Negative_Max_Speed	REAL	Maximum output frequency (negative)
Deadband_ABS	REAL	Value (absolute) of the frequency below which no pulses are generated
Speed	REAL	Frequency at which the pulses must be generated
Test_Delta_Reset	BOOL	This input forces the pulse output internal counter reset (useful to test what happens)

VARIABLES OUTPUT (common to both FB):

NAME	TYPE	DESCRIPTION
ENO	BOOL	Function block enabling state (*)
Input_ERR	BOOL	Input values out of range
Speed_Out	DINT	Output frequency of the pulse train (*)
Reverse	BOOL	Direction bit (ON if the negative direction) (*)
Delta_P_Out	REAL	Pulses generated from the previous execution of the function block (Delta pulses)

(*): Only for diagnostic purposes
(**): For CP1L and CJ1M_CPU2x CPUs, use the blocks whose name ends with 'CP1L_D'.

ADDITIONAL NOTES:

The delta pulse output from the block 'P_Train_D' type, is calculated looking at the PLC Pulse Output internal counters.

These counters have a software limit set between -2 E+9 and +2 E+9 and they are automatically reset (by the function block itself) when the limits are reached.

Since the automatic reset procedure causes temporary interruption of the pulses (at least two scheduled task intervals are required) the function block provides a test input (named ‘Test_Delta_Reset’) to force the reset procedure.

Use this input whenever you need to check if the output pulse interruption causes problems to the product being manufactured (using this input there’s no need to wait till the counter limits are reached).

Moreover a 'Warning' function block output, is activated when the counter is less than one million pulses far from the software limits (both directions).

Theoretically the maximum output pulse counter value is reached in about 5 hours and a half, proceeding constantly at maximum speed (100 KHz).

Anyway, note that at each speed inversion, or when the pulse output is idle, the counter is <u>always</u> cleared automatically, without producing any unexpected behavior.

3.3 SPEED TO ANALOG

The function block named ‘Speed_to_Analog’ converts the number (type REAL) (from +100,000 to -100,000) specified in the input ‘Speed’ in a number (INT) between the specified limits and scaled according to the resolution indicated (for the considered range of positive values).

This function block is ideal to convert the command value calculated by the function block type ‘Connect’ and make it available to an analog output to control the Slave devices:

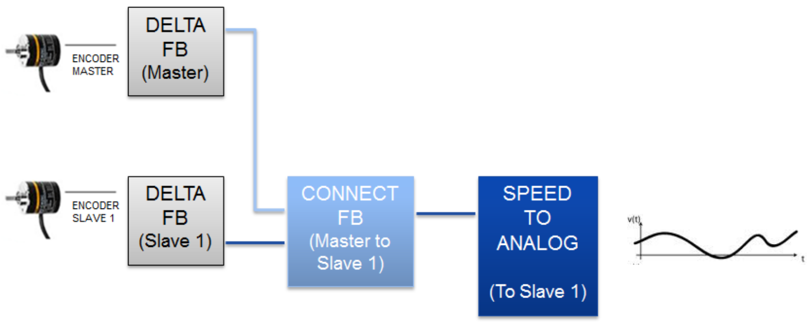
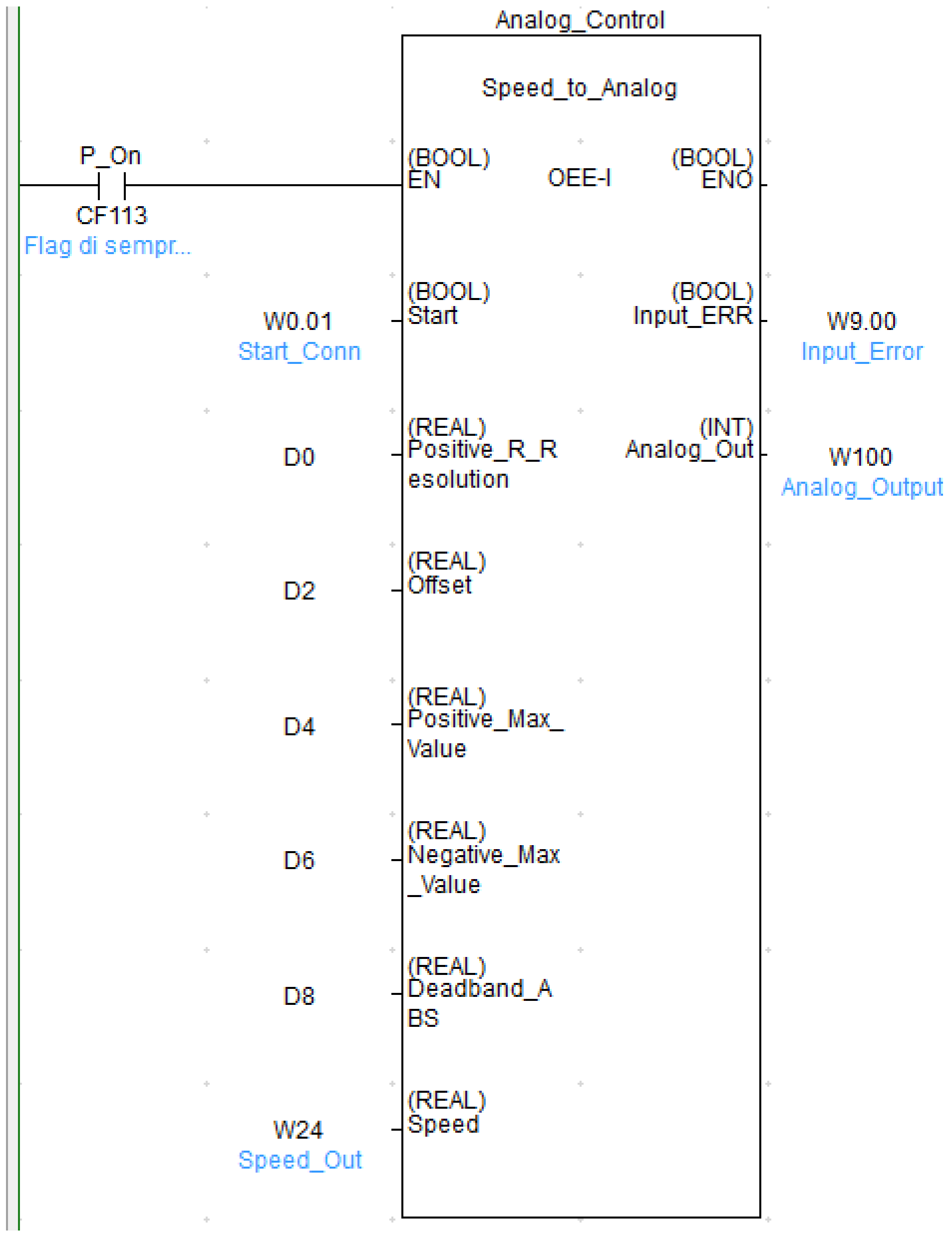


Figure 17: ‘Speed to Analog’ logic usage scheme

Usage example:



INPUT VARIABLES:

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block (use the bit "P_On")
Start	BOOL	Enable output pulse
Positive_R_Resolution	REAL	Output resolution (considered for the range of positive values - maximum value: +20,000)
Offset	REAL	Offset value added to the conversion (between -10,000 and +10,000)

Positive_Max_Value	REAL	Maximum positive output value (up to +32,000)
Negative_Max_Value	REAL	Maximum negative output value (up to -32 000)
Deadband_ABS	REAL	Value (absolute) to below which the output is converted to zero
Speed	REAL	Value to convert and rescale

OUTPUT VARIABLES:

NAME	TYPE	DESCRIPTION
ENO	BOOL	Function block enabling state (*)
Input_ERR	BOOL	Input values out of range
Analog_Out	INT	Output value

(*): Only for diagnostic purposes

Suggestion: use a temporary channel to check the output value before physically copy it to the analog output module!

4 ADVANCED FB

This section presents the guidelines related to advanced' Easy Positioning ' FBs. These function blocks realize 'advanced' functions to complete the electronic link axis functionalities.

4.1 PROFILE_GEN

This Function Block includes a degree 5 polynomial to generate of a position profile.

The block can be used individually, to generate a positioning profile (Figure 18) or combined to an existing movement to superimpose ('ADDAX') a movement to a movement already running (Figure 19).

Logical pattern of use in the case of generation of a single profile, to control the positioning of a Slave axis in a closed loop:

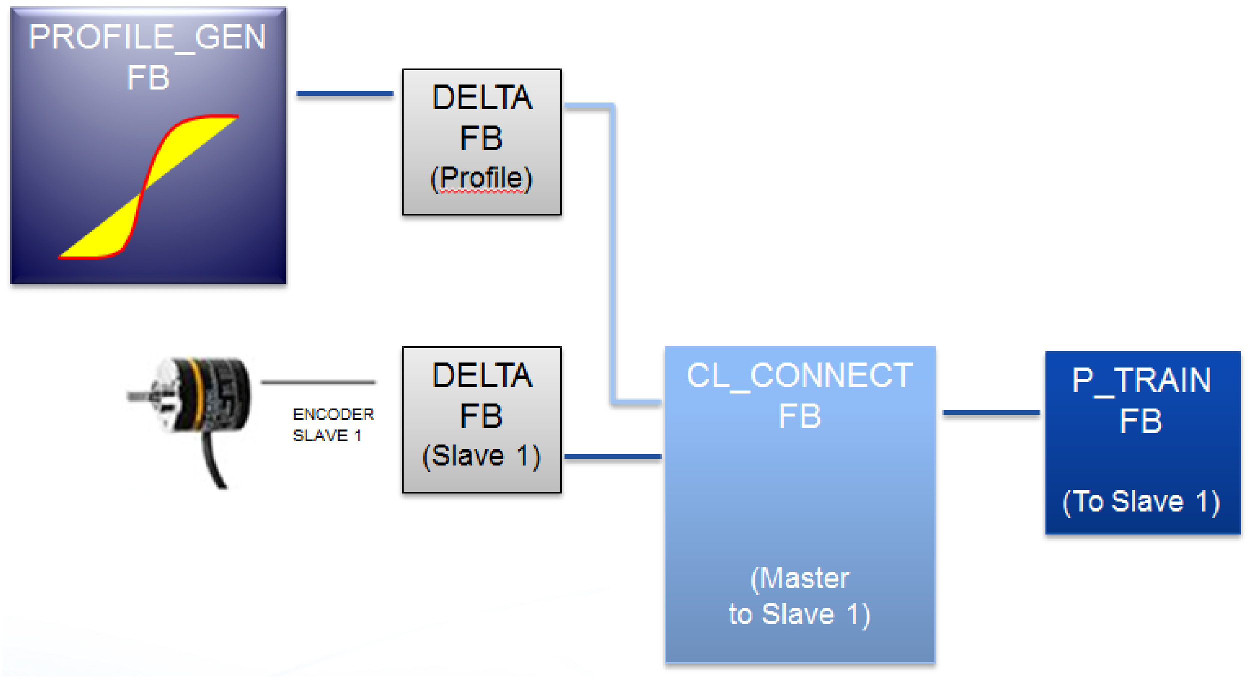


Figure 18: Block 'Profile_Gen' used individually

Logical pattern of use in the case of profile generation that is going to add (or subtract) to a profile already exists (function Addax) and which controls a Slave axis in a closed loop:

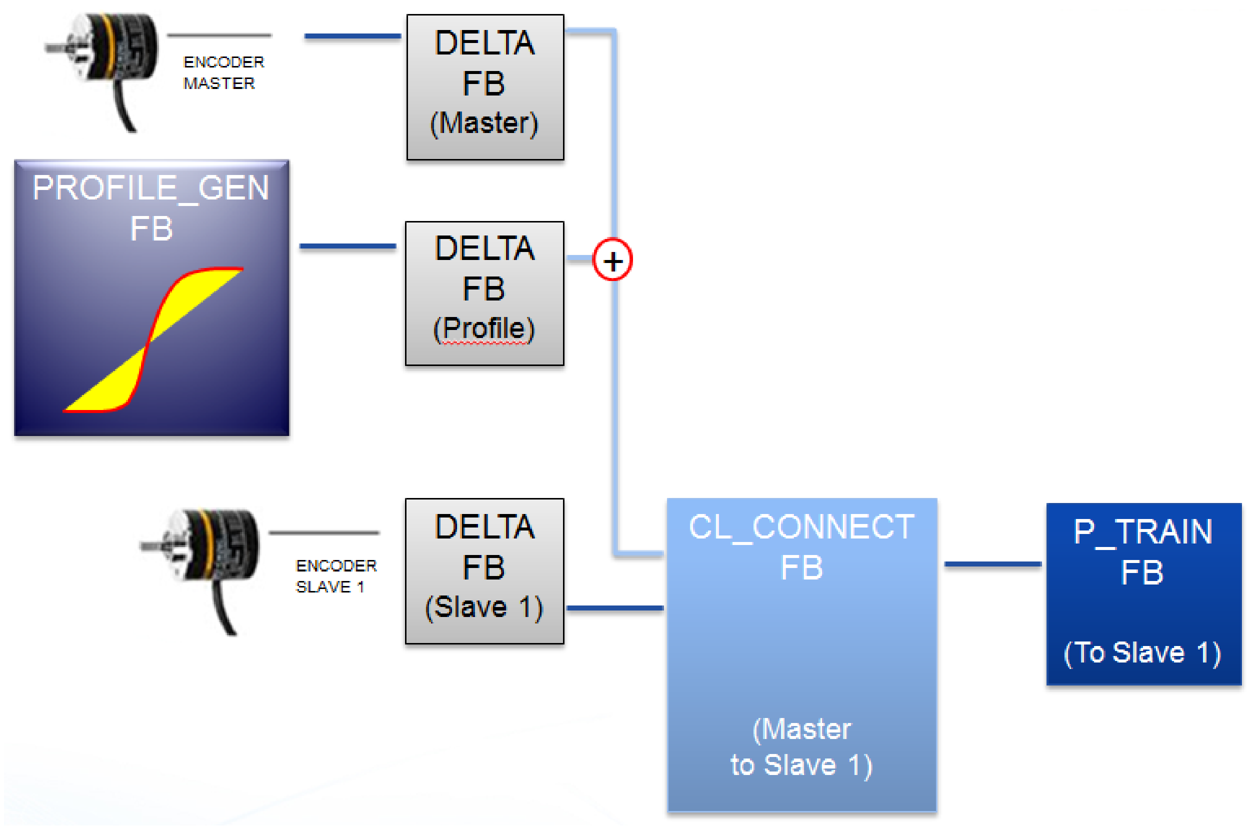


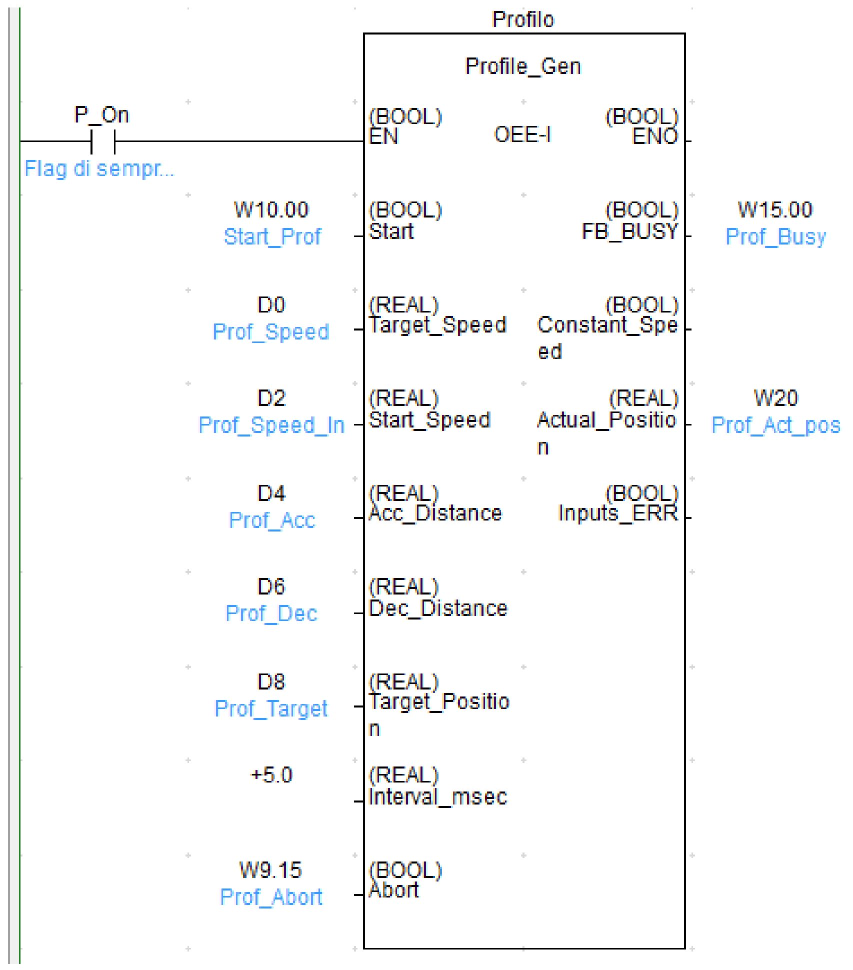
Figure 19: Block 'Profile_Gen' used to create a 'Addax' function

As shown schematically in Figure 19, the position delta induced by the function block is added to the position delta derived from the encoder Master, before being sent to the input 'Delta_Master' of the function block 'Connect'.

The result is then interpreted as an overall movement given by the two contributions, and the Slave axis will move accordingly.

GENERAL NOTE: The function block accepts only positive speeds and distances as input. To proceed in the opposite direction (negative profile) subtract (rather than add) the contribution of the delta FB or multiply it by -1.

Usage example:



INPUT VARIABLES:

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block (use the bit "P_On")
Start	BOOL	Starting the profile generation
Target_Speed	REAL	Target frequency (pulses / sec - only positive values)
Start_Speed	REAL	Start (only positive values)
Acc_Distance	REAL	Space Acceleration (no spaces invalid or inconsistent with the range of interrupt)
Dec_Distance	REAL	Space deceleration (no spaces or inconsistent with the null interval for)
Target_Position	REAL	Positioning of the total space (including the area of acceleration and deceleration, positive value)
Interval_msec	REAL	Time interval (in milliseconds) of the scheduled interrupt task where the block is used
Abort	BOOL	Stops the generation of the profile

OUTPUT VARIABLES:

NAME	TYPE	DESCRIPTION
ENO	BOOL	Function block enabling state (*)
FB_BUSY	BOOL	The profile is being generated(*)
Constant_Speed	BOOL	On at constant speed (*)
Actual_Position	REAL	Actual profile position
Inputs_ERR	BOOL	Inconsistency input data (*)

(*): Only for diagnostic purposes

The rising edge of the Start input starts the profile generation, moving the Actual_Position output value from zero to the value specified in the 'Target_position' input.

The input 'Abort' freezes the actual position at the last calculated value and aborts the profile generation.

If the start signal is set while the Abort input is still active, the profile will not be generated.

The 'Busy' output is active when the profile is being generated; it is inactive in all of the other cases.

NOTE: If 'Abort' input is low, the ‘Actual_Position’ is always reset to zero on the rising edge of the 'Start' input. Take this into account if the position data is processed by a delta function block (the reset could be interpreted as a sudden change in position ..)

To properly manage the 'delta' FB, it is recommended to use the Profile_Gen ‘FB_BUSY’ output as Delta ‘Start’ input, as showed in the figure below:

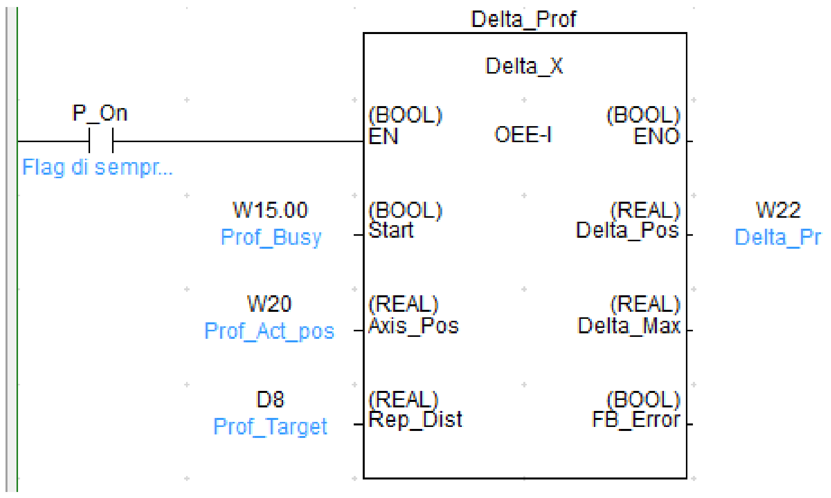


Figure 20: Example of correct use of the function block 'Delta' downstream of a 'Profile_Gen'

Negative profiles: the function block handles only positive speeds and distances. To proceed in the opposite direction (and thus create negative profile) is sufficient to multiply the delta value (coming from the Delta FB) by -1, as shown below:

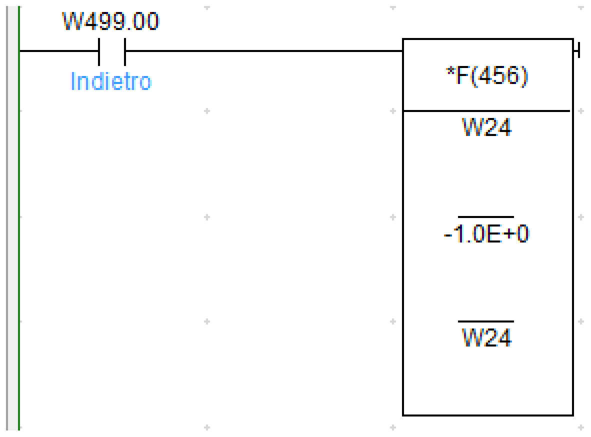


Figure 21: Positive/negative direction control

4.2 CYLOIDAL_CAM

The function block 'Cycloidal_Cam' creates a 'cycloidal' cam.
The cycloid curve has the characteristic of having a zero acceleration in the initial and final moments, and then show no discontinuities (leading, high speed and in certain types of application, side effects).

The function block operates as input the position of a Master real (or virtual) between zero and Rep_Dist, and returns the equivalent cycloidal cam.

Logical pattern of use in the case of a real Master:

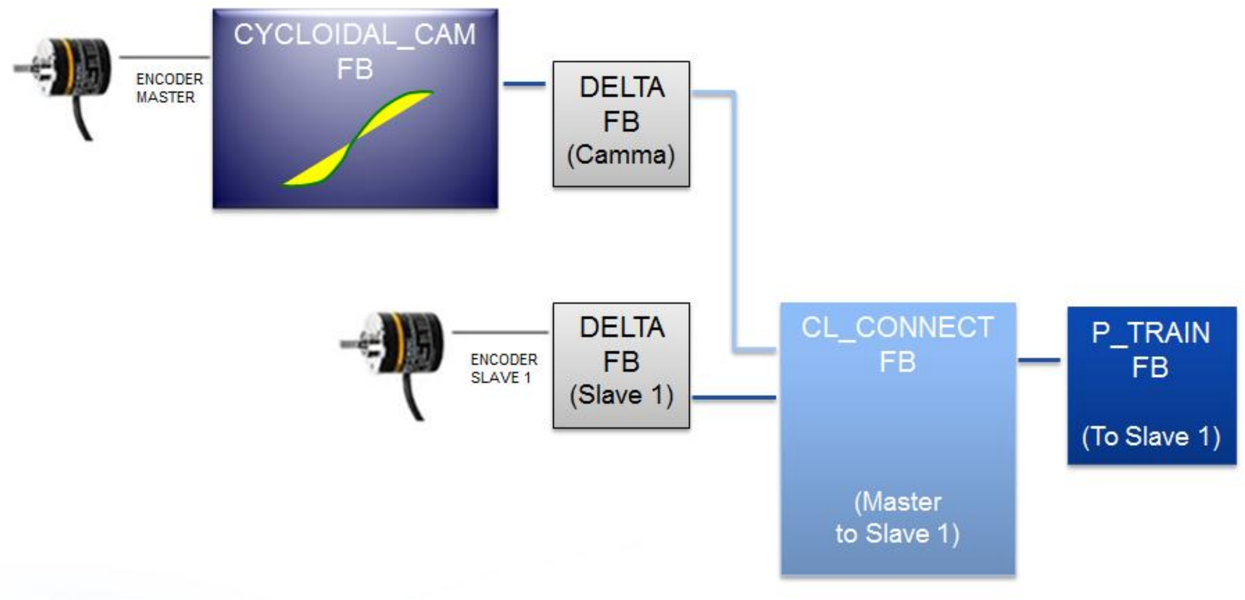
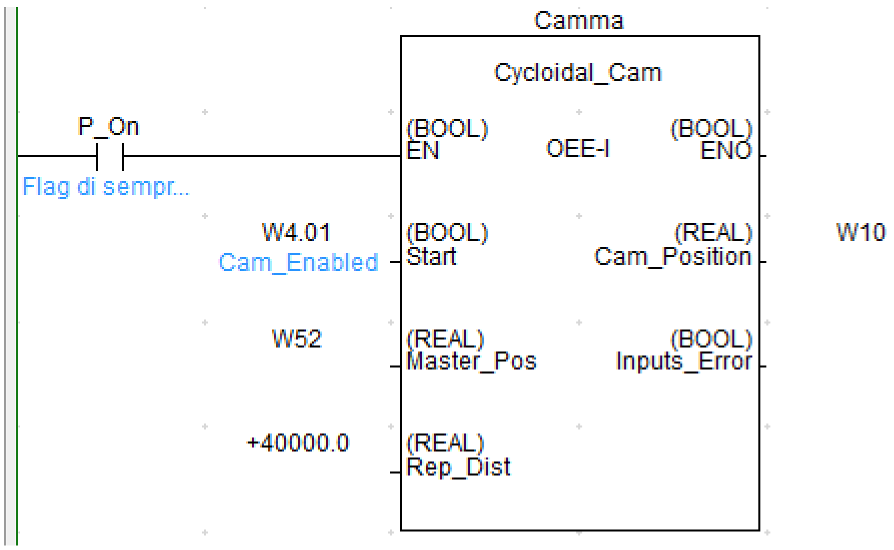


Figure 22: The function block generates the cycloidal cam

NOTE: the "Master_Pos" input type is 'REAL'. If you wish to use the value coming from an HSC don't forget to convert it from DINT to REAL (instruction FLTL) previously.

Usage example:



INPUT VARIABLES:

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block (use the bit "P_On")
Start	BOOL	Enabling the conversion profile
Master_Pos	REAL	Master position (between 0 and Rep_Dist
Rep_Dist	REAL	Maximum value reached by the Master position

OUTPUT VARIABLES:

NAME	TYPE	DESCRIPTION
ENO	BOOL	Function block enabling state (*)
Cam_Position	REAL	Generated actual position of the cam
Error	BOOL	Inconsistency input data (Rep_Dist less than or equal to zero; Master_Pos greater than Rep_Dist).

(*): Only for diagnostic purposes

4.3 MOVELINK

This function block uses a degree 5 polynomial to generate a profile of space, connected to the movement of a Master axis (hence the name 'MOVELINK').

The profile of space is calculated by dividing the distance traveled by the Master in three parts (three spaces): acceleration space, the space at a constant speed and deceleration space.

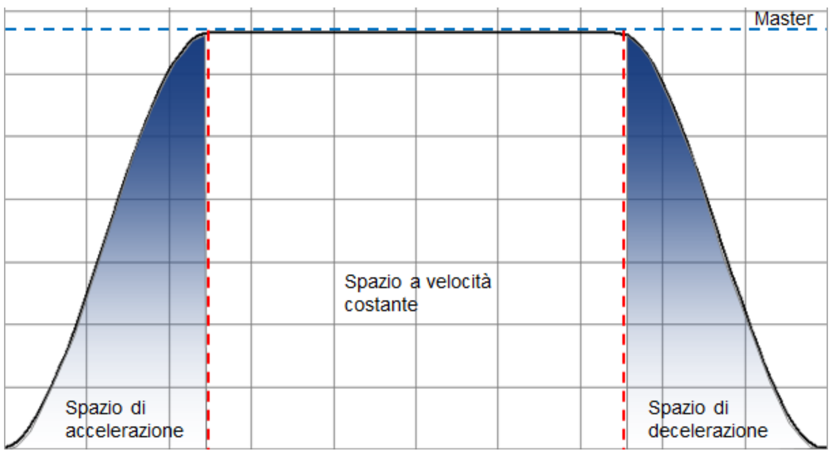


Figure 23: Speed profile generated with the "Acc_Mode" input = 0

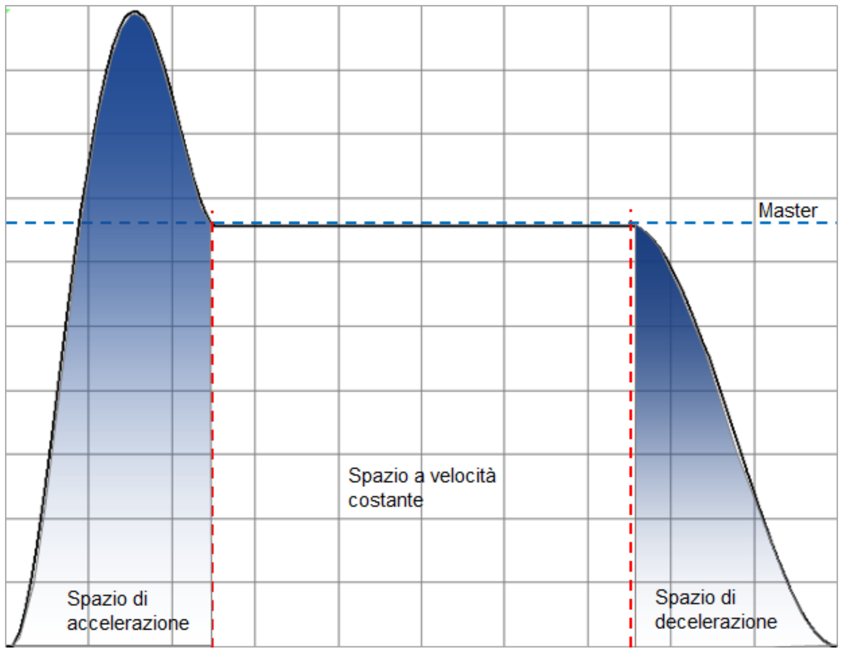


Figure 24: Speed profile generated with the "Acc_Mode" input =1

The distance traveled during the acceleration phase depends on the acceleration mode indicated by the input "Acc_Mode":

- If "Acc_Mode" is equal to zero, the distance traveled during the acceleration phase corresponds to half the space Master (Figure 23)
- if "Acc_Mode" is equal to one, the distance traveled during the acceleration phase corresponds to the space Master (Figure 24)

The areas underlying the speed profiles indicated in the previous figures represent the space paths.

Obviously, in the stretch at a constant speed, the distance traveled is equivalent to the space of the Master, while, during the deceleration phase, the space covered by the profile generated is always equal to half the distance traveled by the Master.

The total space resulting from the function block, thus corresponds to the sum of the spaces of acceleration, deceleration and constant speed, calculated as indicated above.

Logical pattern of use of function blocks to create a MOVELINK between a Master and a Slave axis encoder:

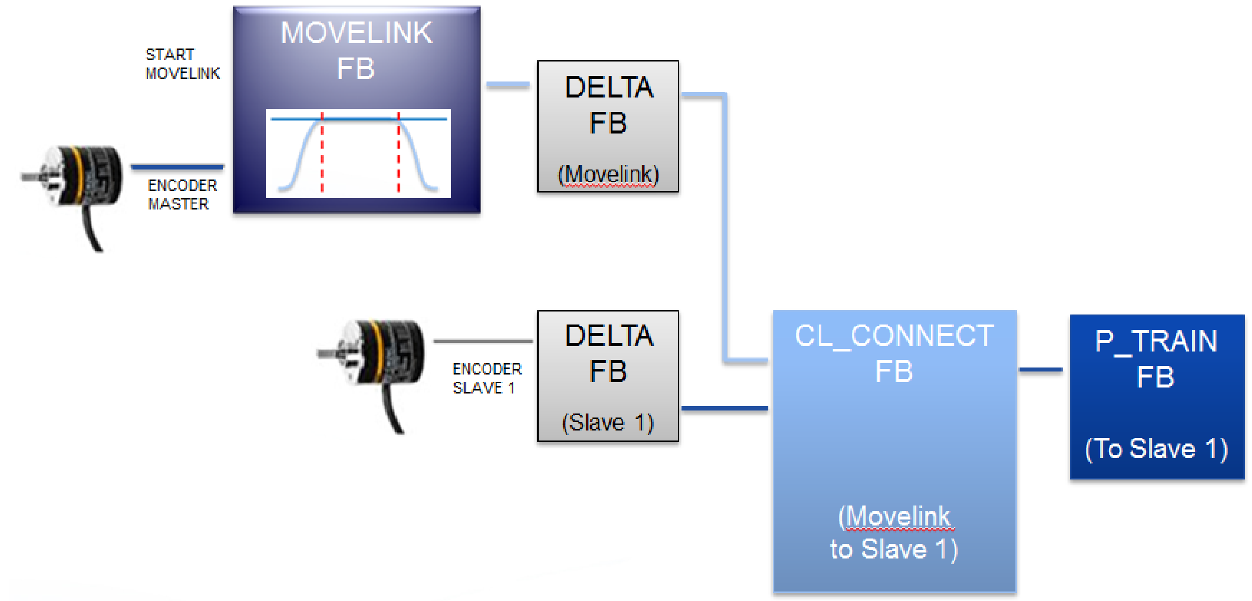
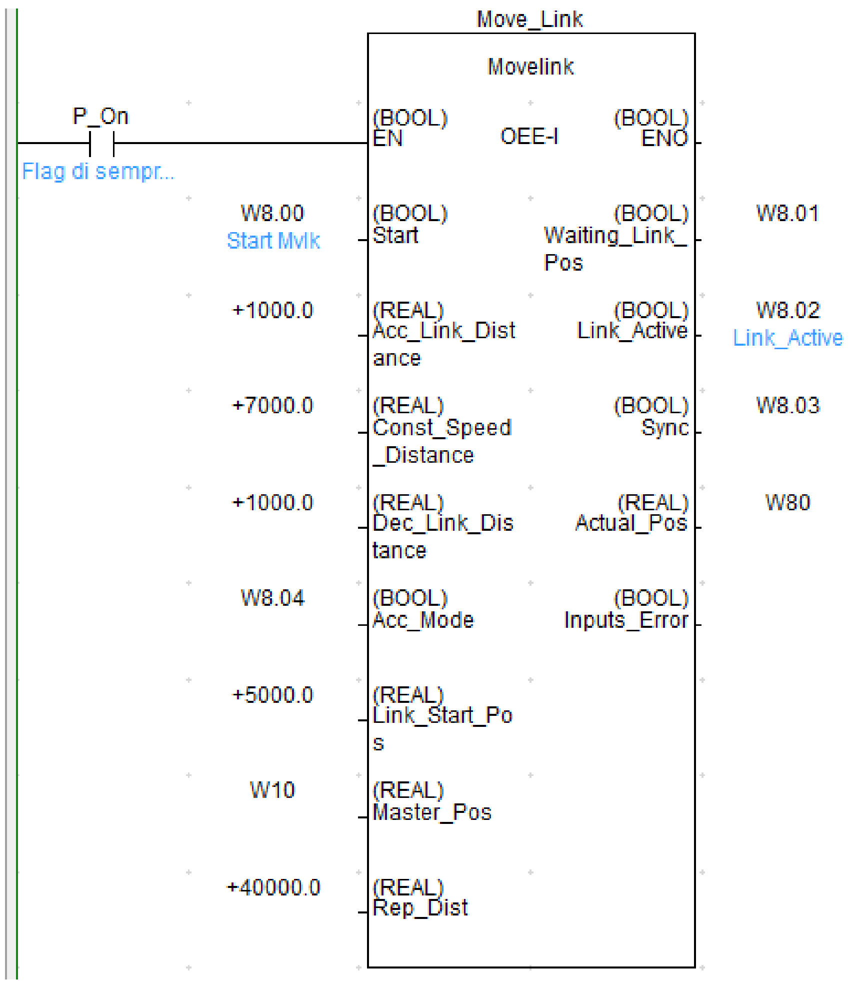


Figure 25: Creating a 'MOVELINK' axis between a Master encoder and a Slave axis

Usage example:



INPUT VARIABLES:

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block (use the bit "P_On")
Start	BOOL	Enabling the generation of the profile
Acc_Link_Distance	REAL	Master space used for the acceleration phase
Const_Speed_Distance	REAL	Space Master used for the section at constant speed
Dec_Link_Distance	REAL	Master space used for the deceleration phase
Acc_Mode	BOOL	Acceleration mode: to 0 space generated during the acceleration phase corresponds to half the distance traveled by the Master; to 1 the space generated during the acceleration phase corresponds to the distance traveled by the Master
Link_Start_Pos	REAL	Quote Master for the start of MOVELINK
Master_Pos	REAL	Actual share of the Master
Rep_Dist	REAL	Maximum value reached by the channel "Master_Pos"

OUTPUT VARIABLES:

NAME	TYPE	DESCRIPTION
ENO	BOOL	Function block enabling state (*)
Waiting_Link_Pos	BOOL	Pending share coupling
Link_Active	BOOL	MOVELINK active (the portion of coupling has been shut off and the MOVELINK is in progress) (*)
Sync	BOOL	On to the stretch at a constant speed (Master and Slave have the same speed) (*)
Actual_Pos	REAL	Actual position generated
Inputs_Error	BOOL	Inconsistency input data (*)

(*): Only for diagnostic purposes

Additional notes:

- When the FB is waiting for the link position, the Master must only move in a positive direction; only when the link is active (output 'Link_Active' to On) the Master is followed in both directions until the exit of MOVELINK;
- To start the coupling on the signal of "Start" (without waiting then a share) use the same channel for both "Master_Pos" and "Link_Start_Pos" inputs
- To generate negative movements, act on the function blocks' Delta' using the same strategy suggested for the function block' Profile_Gen (Figure 24)
- The function block is ideal as a 'flying start'. For this purpose use the output of Sync to know when the speeds are synchronized

NOTE:
The output 'Actual_Pos' is reset on the rising edge of the 'Start'. Take this into account if the position data is processed by a delta function block (the reset could be interpreted as a sudden change in position ..)

To properly manage the 'delta' of the generated position, use the Profile_Gen ‘Link_Active’ output as Start input for the 'Delta_X’ FB:

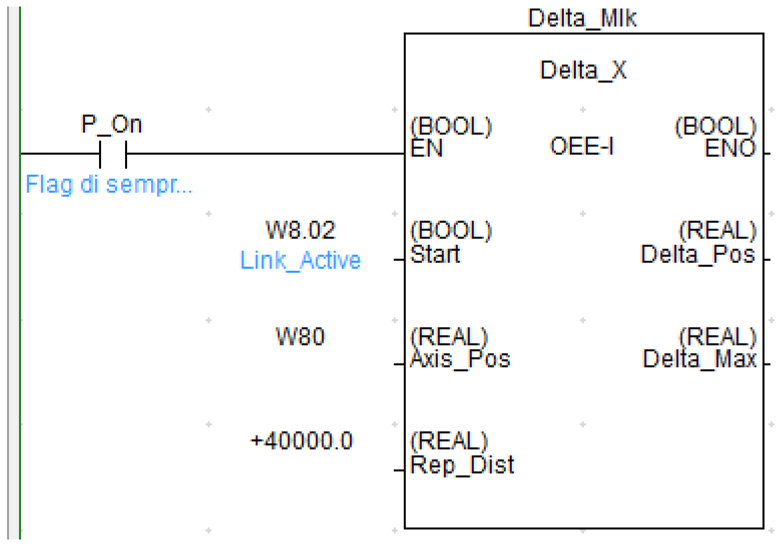


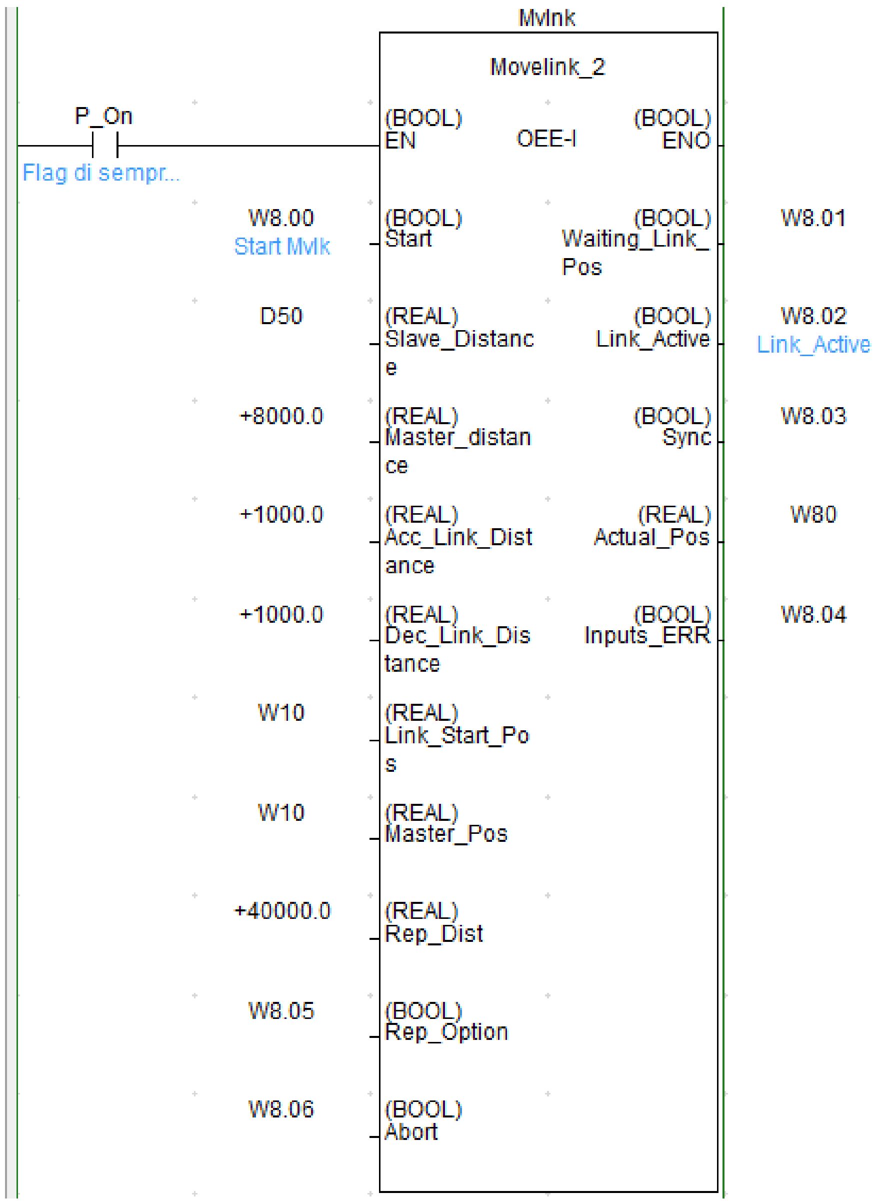
Figure 26: Example of correct use of the function block 'Delta_X'

To generate negative movements, act on the function blocks' Delta' using the same strategy suggested for the function block' Profile_Gen (Figure 18)

4.4 MOVELINK_2

The function block "Movelink_2" allows to perform a synchronized movement relative to a Master axis, where, it's also possible to define the space of a Slave axis (virtual) and link in a repetitive way (input 'Rep_Option').

Usage example:



INPUT VARIABLES:

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block (use the bit "P_On")
Start	BOOL	Enabling the generation of the profile
Slave_Distance	REAL	Space TOTAL Slave axis (virtual)
Master_Distance	REAL	TOTAL Area Master axis
Acc_Link_Distance	REAL	Master space used for the acceleration phase
Dec_Link_Distance	REAL	Master space used for the deceleration phase
Link_Start_Pos	REAL	Quote Master for the start of MOVELINK
Master_Pos	REAL	Actual share of the Master
Rep_Dist	REAL	Maximum value reached by the channel "Master_Pos"
Rep_Option	BOOL	If this bit is set to ON on fornte rising edge of the 'Start, the MOVELINK is performed on a repetitive
Abort	BOOL	Aborts MOVELINK (exit 'Actual_Pos' stops on the actual value)

OUTPUT VARIABLES:

NAME	TYPE	DESCRIPTION
ENO	BOOL	Function block enabling state (*)
Waiting_Link_Pos	BOOL	
Link_Active	BOOL	MOVELINK active (the portion of coupling has been shut off and the MOVELINK is in progress) (*)
Sync	BOOL	On to the stretch at a constant speed (Master and Slave have the same speed) (*)
Actual_Pos	REAL	Actual position generated (virtual Slave)
Inputs_Error	BOOL	Inconsistency input data (*)

(*): Only for diagnostic purposes

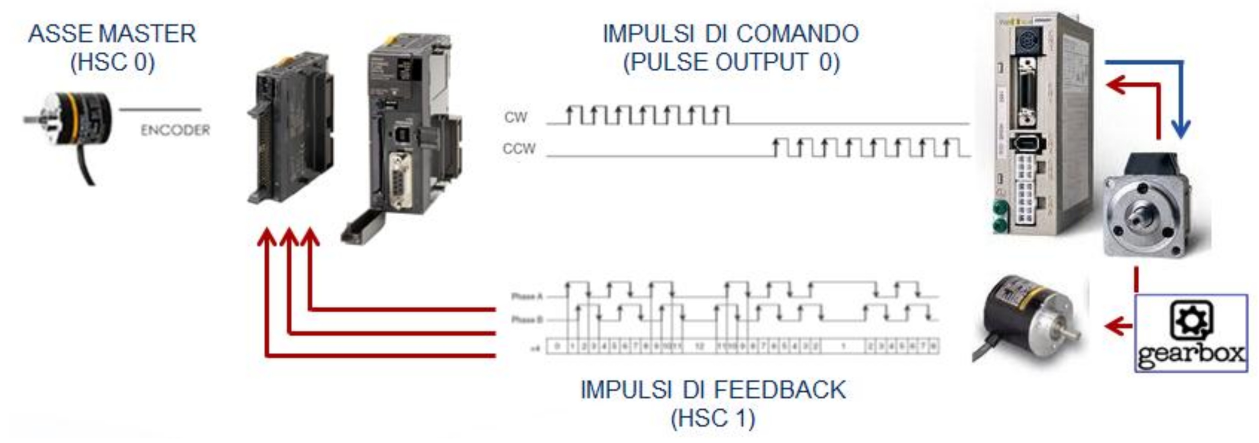
Additional notes:

- Unlike the function block 'MOVELINK' it is NOT possible to define the acceleration mode. The space of acceleration and deceleration of the Slave virtual corresponds always to half the space Master

- When the FB is waiting for the link position, the Master must only move in a positive direction; only when the link is active (output 'Link_Active' to On) the Master is followed in both directions until the exit of MOVELINK;
- To start the coupling on the signal of "Start" (without waiting then a share) use the same channel for both "Master_Pos" and "Link_Start_Pos" inputs
- To generate negative movements, act on the function blocks' Delta' using the same strategy suggested for the function block' Profile_Gen (Figure 21)
- The function block is ideal as a 'flying start'. For this purpose use the output of Sync to know when the speeds are synchronized

5 PROGRAM EXAMPLE

Below the steps needed to implement electronic shaft between a Master axis (an encoder wired on HSC 0) and a servo drive (Slave axis) controlled via pulse train (Pulse Output 0). The feedback encoder is wired to HSC 1.



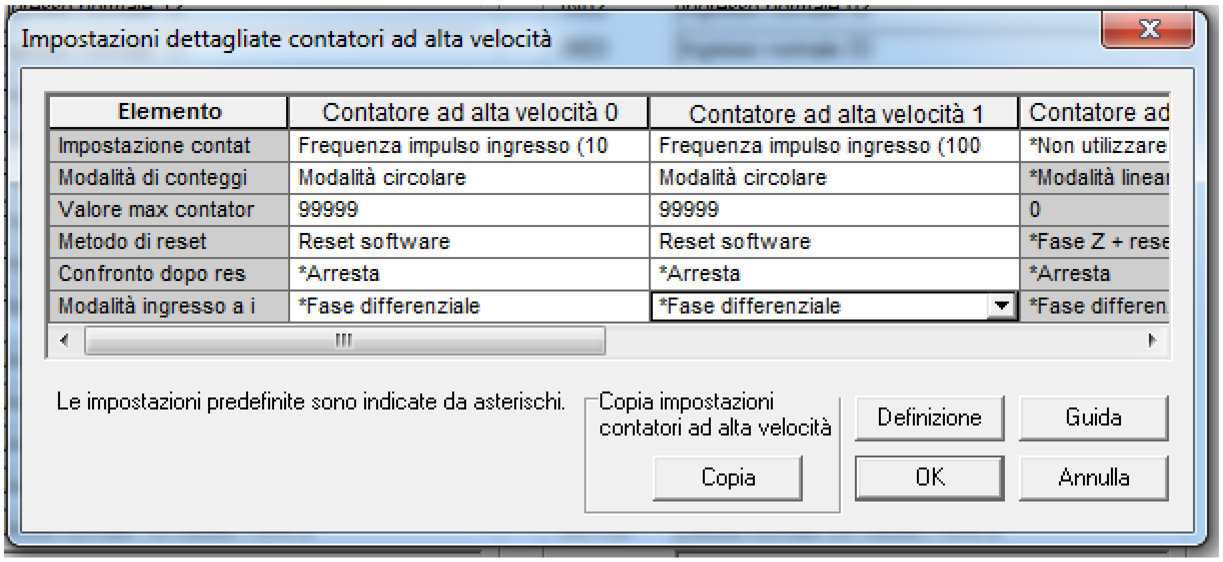
It is assumed that:

- the Master encoder generates **8,000** pulses per revolution on PLC high-speed counter 0 (HSC0)
- **2,500** pulses are needed to the servo driver to perform one motor shaft revolution
- after the servo motor a **3:1** gearbox is mounted
- the feedback encoder (mounted after the gearbox) generates **4,000** pulses per revolution, on PLC high-speed counter 1 (HSC1) PLC.

Finally, it is assumed that both encoders have a 'Differential phase' output type.

5.1 PLC SETTINGS

In agreement with what is indicated in Section 2.3, we will set, as an example, 99,999 as maximum value for both circular counters (this value corresponds to a Rep_Dist of 100,000 pulses ..)


Impostazioni dettagliate contatori ad alta velocità

Elemento	Contatore ad alta velocità 0	Contatore ad alta velocità 1	Contatore ad
Impostazione contat	Frequenza impulso ingresso (10	Frequenza impulso ingresso (100	*Non utilizzare
Modalità di conteggi	Modalità circolare	Modalità circolare	*Modalità linea
Valore max contator	99999	99999	0
Metodo di reset	Reset software	Reset software	*Fase Z + rese
Confronto dopo res	*Arresta	*Arresta	*Arresta
Modalità ingresso a i	*Fase differenziale	*Fase differenziale	*Fase differen

Le impostazioni predefinite sono indicate da asterischi.

Transfer the settings and power cycle the PLC.

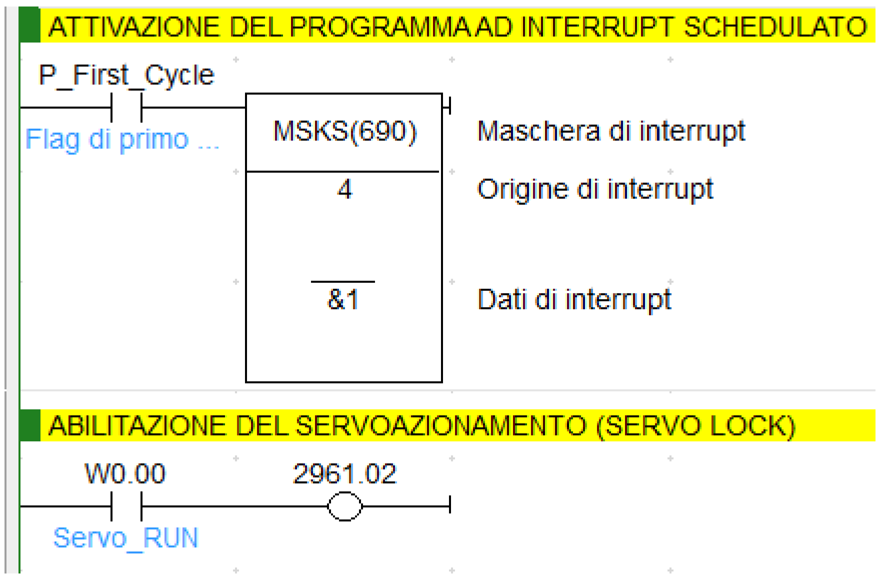
5.2 TASK MANAGEMENT AND SERVO RUN

The code below enable, at the first scan, the interrupt tasks scheduled with a time interval of **10 ms.**

The second rung handles Servo Lock / Unlock (wired, in the example, to the PLC integrated output addressed to CIO 2961.02).

NOTE:
The interpretation of the time value, indicated in the second operand MSKS, depends on the time constant expressed in the PLC settings window (Figure 5).

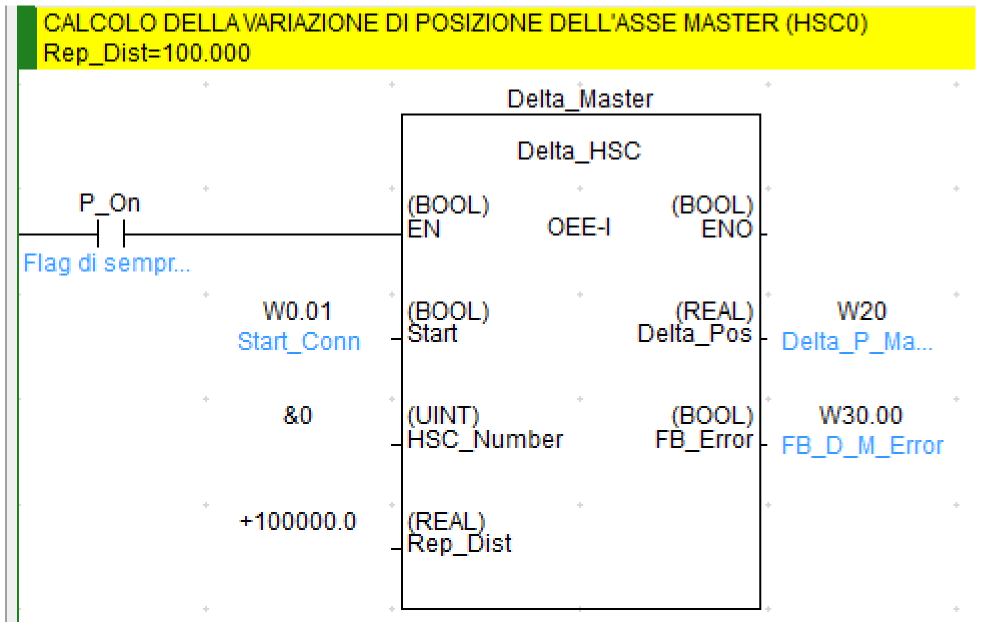
The default value is 10, before proceeding, always check that the value residing in the PLC matches and, if you are in doubt, power cycle the PLC and read settings from PLC again.



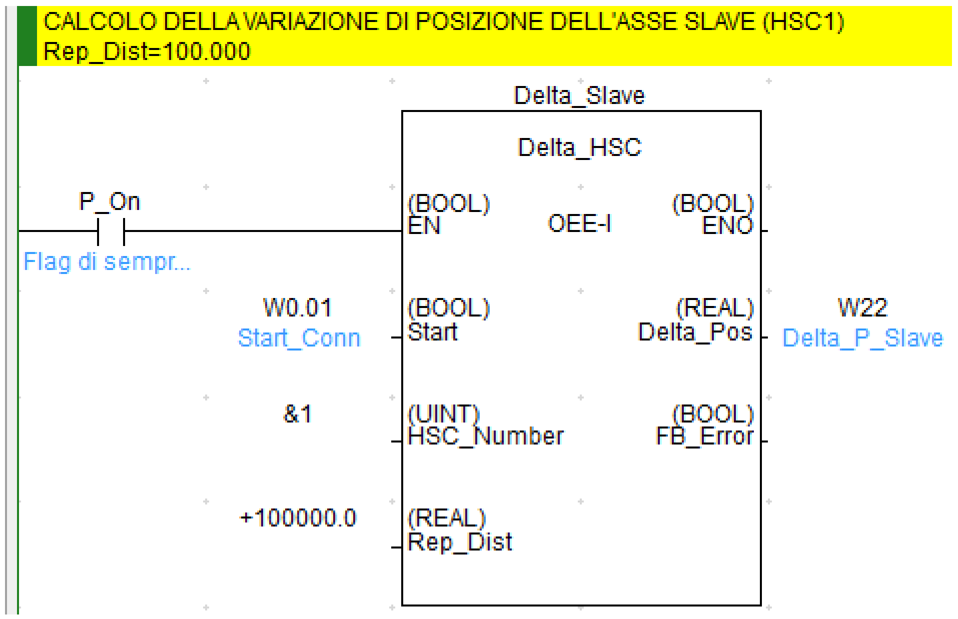
5.3 'DELTA_HSC' FB

Into the scheduled interrupt task section, two function block type 'Delta_HSC' are needed, since both axes return their position from an high-speed counter (HSC).

Delta FB for the Master encoder (wired to PLC high-speed counter 0):



Delta FB for the feedback encoder (wired to PLC high-speed counter 1) to know the Slave axis position:



Note:

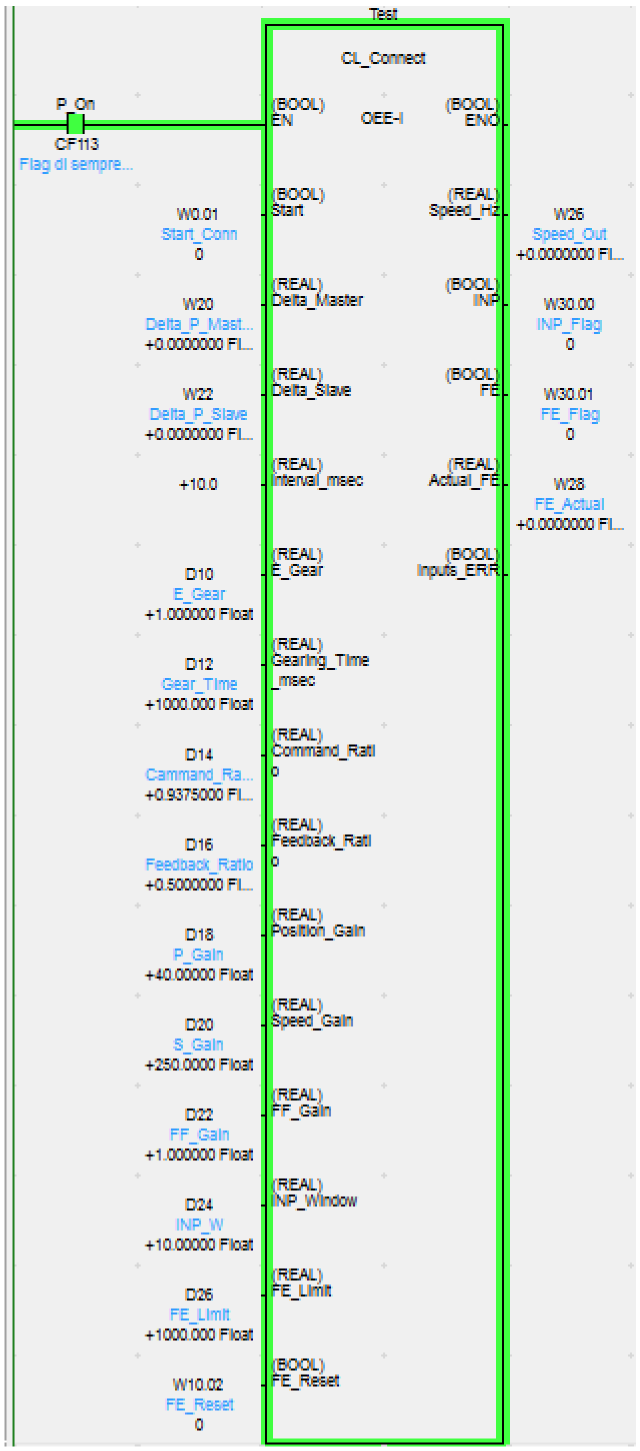
- Both function blocks are enabled using the same bit W0.01 (Start_Conn).
- Both blocks have the same function 'Rep_Dist', because the same maximum count value circular (99999) has been set for both counters in the PLC Settings

5.4 'CL_CONNECT' FB

The two previous function blocks return respectively, in the channels W20 and W22, the distance traveled by the Master axis and Slave axis in the scheduled task time interval (Interval_msec).

These two values (type: REAL) are sent to the 'Delta_Master' and 'Delta_Slave' inputs of the function block 'CL_CONNECT'.

Function block 'CL_Connect':



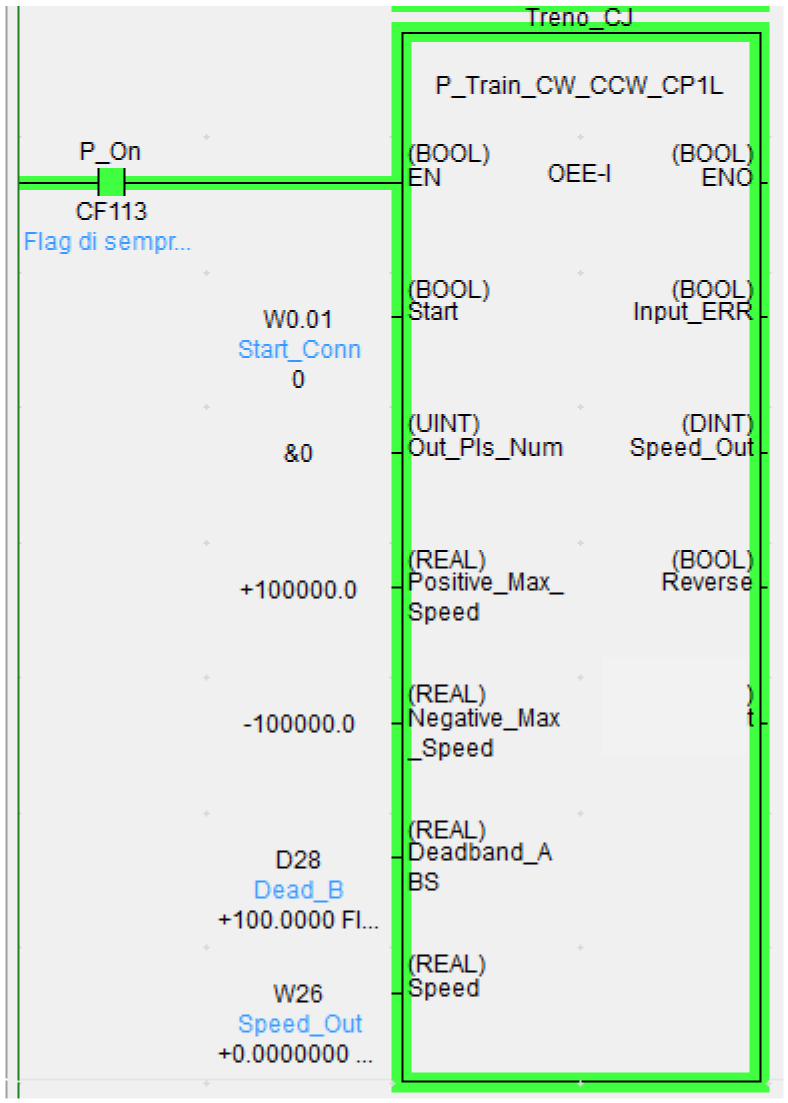
In agreement with what is indicated in the table in Section 3.4.1 the values of Command_Ratio and Feedback_Ratio have been set to 7.500/8.000 (i.e. +0.9375) and 4.000/8.000 (i.e. +0.5).

The values for the other inputs are purely indicative.

5.5 'P_TRAIN' FB

The actual speed value calculated by the previous function block (channel W24) is used by the function block that generates the pulse train.

Function block for the pulse train generation (mode CW/CCW):



5.6 ELECTRONIC SHAFT AXIS TEST

Before enabling the electronic connection (bit W0.01) RUN the servo motor (bit W0.0 to ON).

Set Position_Gain and Speed_Gain inputs to zero (to control the Slave axis in open loop) in the function block 'CL_Connect', and set the input FF-Gain to 1.
After a final check on the correctness of the FBs input parameters, enable the electronic connection (bit W0.01 to ON).

Move the Master encoder and verify that the Slave axis rotates in the correct directions and that these directions are consistent with the value returned by the encoder feedback.

If everything is correct, reset the following error (bit W10.02 to ON) and assign values other than zero to Position_Gain and Speed_Gain (in this way the position loop is closed); proceed bearing in mind what is described in Section 3.4.2 .

Adjust the FF-Gain input only if necessary.

6 FB APPENDIX

In support of the Easy Positioning function blocks some other blocks, with related and useful function, have been developed.
These function blocks can be executed in a cyclic task (it's not mandatory to run them in an interrupt scheduled task)
.

6.1 'JOG' FUNCTION BLOCKS(**)

These function blocks can be used to control axis in a Jog movement.

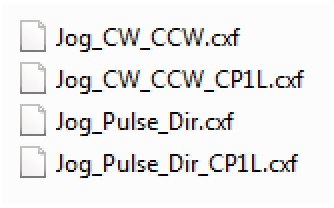
The calculated speed value can be output directly as a pulse train or even in different ways (i.e. using the function block 'Virtual_Jog' that simply returns a number).

The function blocks handle acceleration and deceleration time (in msecs) and speed limits.

Note: it is recommended to disable the function block if unused.

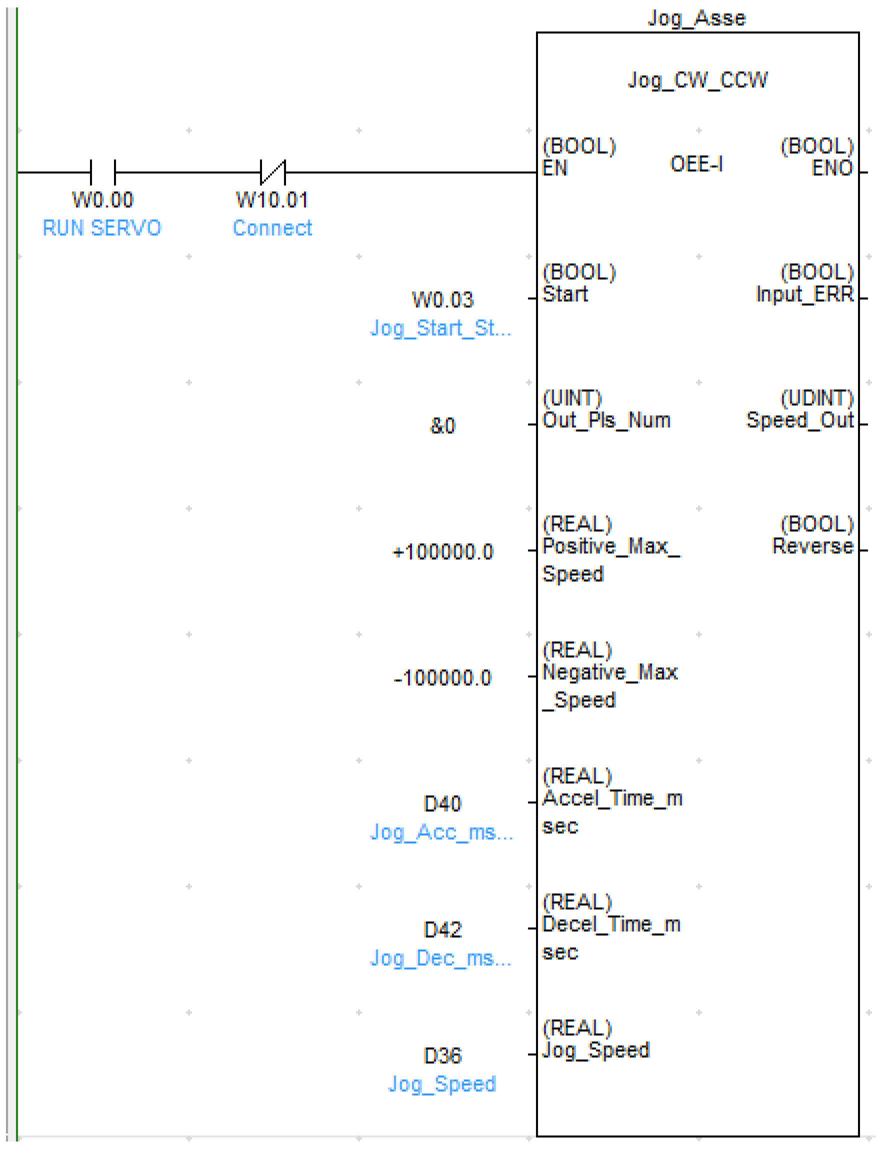
6.1.1 PULSE TRAIN JOG

In order to support CP1/CJ1 CJ2 CPUs, different FB version have been developed:


Jog_CW_CCW.cxf
Jog_CW_CCW_CP1L.cxf
Jog_Pulse_Dir.cxf
Jog_Pulse_Dir_CP1L.cxf

(**): For CP1L and CJ1M_CPU2x CPUs, blocks whose name ends with 'CP1L' must be used.

Usage example:



INPUT VARIABLES (common to all FB types):

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block
Start	BOOL	Pulse Output enabled
Out_Pls_Num	UINT	Number of output pulse train to be used
Positive_Max_Speed	REAL	Maximum output frequency (positive)
Negative_Max_Speed	REAL	Maximum output frequency (negative)
Accel_Time_msec	REAL	Time (in milliseconds) to reach the target Jog Speed starting from zero speed

Decel_Time_msec	REAL	Time (in milliseconds) to reach the zero speed starting from the target Jog Speed
Jog_Speed	REAL	Target jog speed

OUTPUT VARIABLES (common to all of the FB types):

NAME	TYPE	DESCRIPTION
ENO	BOOL	Function block enabling state (*)
Input_ERR	BOOL	Input values out of range
Speed_Out	DINT	Pulse train output frequency (*)
Reverse	BOOL	Direction bit (ON if the negative direction) (*)

(*): Only for diagnostic purposes
(**): For CP1L and CJ1M_CPU2x CPUs use the blocks whose name ends with 'CP1L'.

If the Start INPUT goes to OFF, the output speed is set to zero.

If the direction is reversed (by specifying a value of speed of opposite sign) while movement is in progress, the motor will decelerate to zero respecting the specified deceleration, and will then accelerate the new speed respecting the specified acceleration.

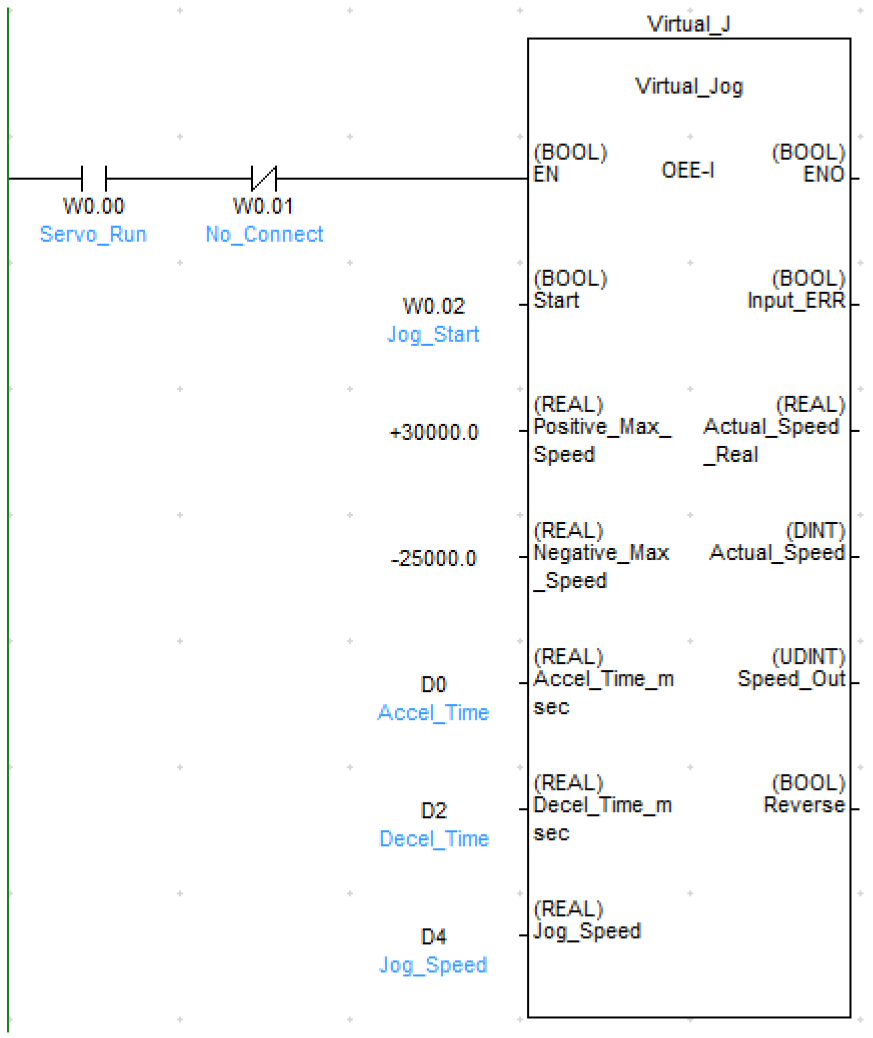
Accelerations and decelerations are calculated to reach the target jog speed, starting from zero speed and viceversa.

6.1.2 VIRTUAL JOG

As the previous blocks, this block calculates a value of speed respecting the acceleration and deceleration times. Unlike the previous blocks it does not physically output any speed from the PLC.

The calculated output is then a pure number (to be considered in Hz) that can be sent to a device through communication, or rescaled appropriately (e.g. via the 'Speed_to_Analog' function block) to control an analog output.

Usage example:



Input Variables:

NAME	TYPE	DESCRIPTION
EN	BOOL	Enabling Function Block
Start	BOOL	Output speed calculation enabled
Positive_Max_Speed	REAL	Maximum output frequency (positive)
Negative_Max_Speed	REAL	Maximum output frequency (negative)
Accel_Time_msec	REAL	Time (in milliseconds) to reach the target Jog Speed starting from zero speed
Decel_Time_msec	REAL	Time (in milliseconds) to reach the zero speed starting from the target Jog Speed
Jog_Speed	REAL	Target jog speed

VARIABILI DI USCITA:

NOME	TIPO	DESCRIZIONE
ENO	BOOL	Function block enabling state (*)
Input_ERR	BOOL	Input values out of range
Actual_Speed_Real	REAL	Calculated output speed [REAL]
Actual_Speed	DINT	Calculated output speed [DINT]
Speed_Out	UDINT	Calculated output speed [UDINT]
Reverse	BOOL	Direction bit (ON if the negative direction)

(*): Only for diagnostic purposes

If the Start INPUT goes to OFF, the output speed is set to zero.

If the direction is reversed (by specifying a value of speed of opposite sign) while movement is in progress, the motor will decelerate to zero respecting the specified deceleration, and will then accelerate the new speed respecting the specified acceleration.

Accelerations and decelerations are calculated to reach the target jog speed, starting from zero speed and viceversa.

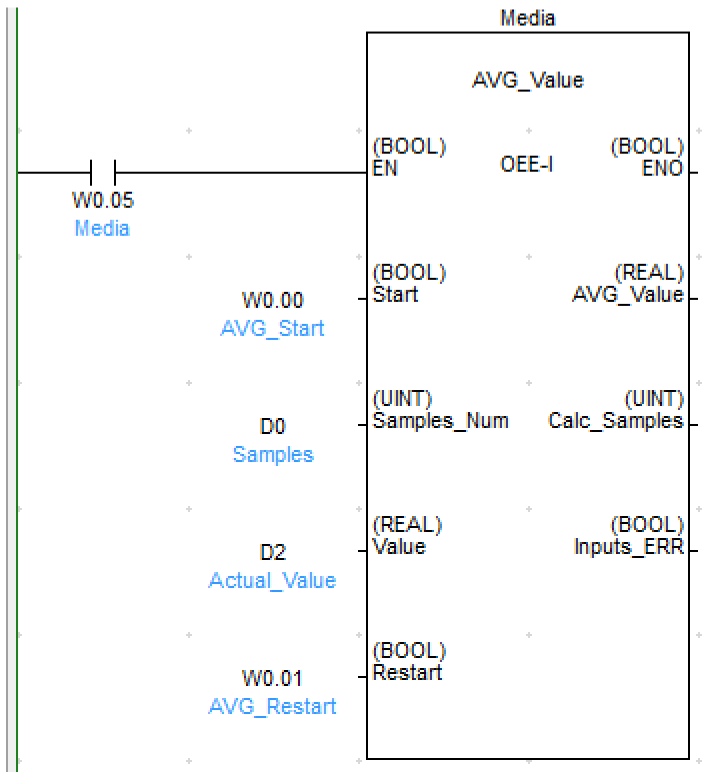
The block returns the calculated value of speed in the most common data types, in order to avoid subsequent conversions.

6.2 AVERAGE 'AVG_VALUE'

The ' AVG_Value ' function block returns the average of the value specified in the 'Value' input, based on the number of samples specified in the input named 'Samples_Num'.

The maximum number of samples settable is equal to 30,000.

Usage example:



INPUT VARIABLES:

NAME	**TYPE**	**DESCRIPTION**
EN	BOOL	Enabling Function Block
Start	BOOL	Average calculation starts
Samples_Num	UINT	Number of samples to consider (max: **30.000**)
Value	REAL	Sampled value
Restart	REAL	Sampling restart (same effect as set OFF and set ON the Start input)

OUTPUT VARIABLES:

NAME	**TYPE**	**DESCRIPTION**
ENO	BOOL	Stato di abilitazione del Function Block (*)
AVG_Value	REAL	Calculated average value
Calc_Samples	REAL	Number of samples considered
Inputs_ERR	BOOL	Input values out of range

(*): Only for diagnostic purposes

If the Start input is set to ON, the samples are captured at each function block execution.

If the Start input is set to OFF, the average calculated value is set to zero.

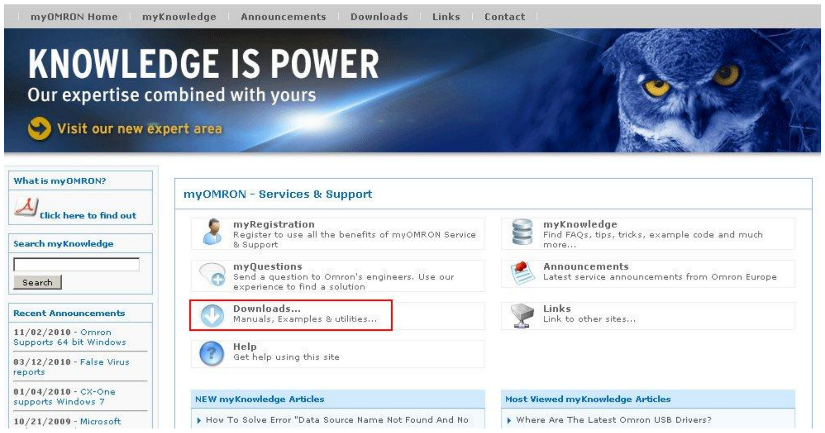
The number of samples computed is reset at the rising edge of the input 'Start', or at the rising edge of the input ' Restart '.

The ‘Restart’ input restart the calculation.

7 APPENDIX

A comprehensive collection of documents like this one are available for free, after registration, in the Downloads section → 'Local Material' → 'Italy' section, on the site:

www.myomron.com



What is myOMRON?

Click here to find out

Search myKnowledge

Search

Recent Announcements

11/02/2010 - Omron Supports 64 bit Windows

03/12/2010 - False Virus reports

01/04/2010 - CX-One supports Windows 7

10/21/2009 - Microsoft

myOMRON - Services & Support

myRegistration
Register to use all the benefits of myOMRON Service & Support

myKnowledge
Find FAQs, tips, tricks, example code and much more...

myQuestions
Send a question to Omron's engineers. Use our experience to find a solution

Announcements
Latest service announcements from Omron Europe

Downloads...
Manuals, Examples & utilities...

Links
Link to other sites...

Help
Get help using this site

NEW myKnowledge Articles

- How To Solve Error "Data Source Name Not Found And No

Most Viewed myKnowledge Articles

- Where Are The Latest Omron USB Drivers?