Program examples for Mitsubishi PLCs FX3U:

FX3UPulsePositioningFRA700 library program examples

Program name	Description
Prg ExampleControlAx1	This program example uses function block PulsposFX3U_FRA700controlAx1 to control the positioning of a FR-A 740 Frequency Inverter with a FR-A7AP plug-in option for vector control, on 1 Axis
Prg ExampleSequenceAx1	This is an example of how a sequence program can be used for positioning.
Prg ExampelControlThreeAxes	This program example uses function blocks PulsposFX3U_FRA700controlAx13, PulsposFX3U_FRA700controlAx23 and PulsposFX3U_FRA700controlAx33 to control the positioning of 3 FR-A 740 Frequency Inverters with FR-A7AP plug-in option for vector control, on 3 Axis
Prg ExampleSequenceThreeAxes	This is an example of how a sequence program can be used for positioning.

ExampleControlAx1

Program description

The example uses a FR-A 740 Frequency Inverter with a FR-A7AP plug-in option for vector control, pulse controlled directly via a FX3U PLC-system's transistor outputs, to control an AC servo motor for an axis when the incremental encoder system is used. The program example for the PLC-system contains a function block for control and a sequence program used for positioning.

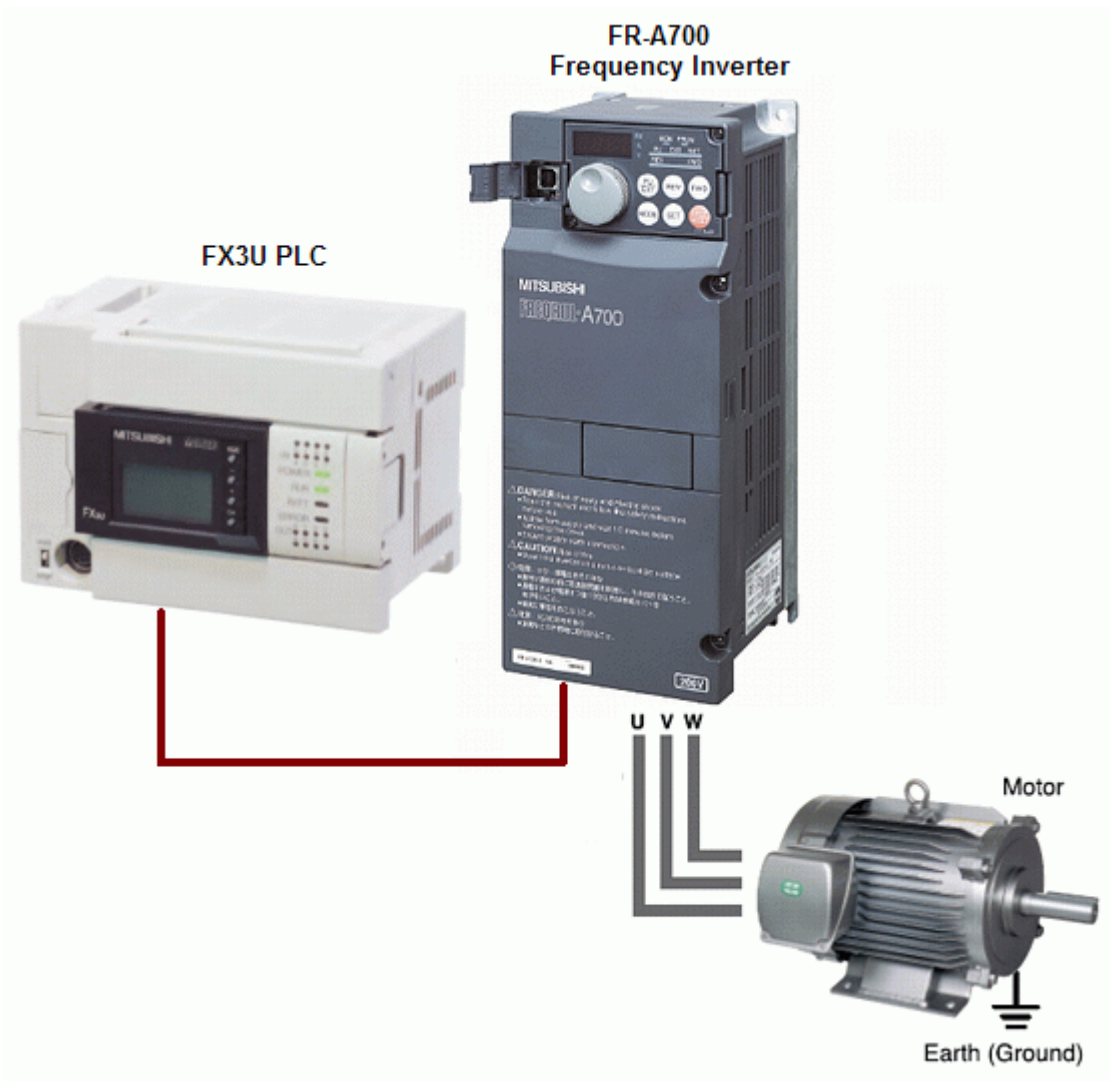
The program example has the following functions:

- Read the servo motor current position into the PLC-system with set of **InverterOn**;
- Enable/disable of **InverterOn** signal;
- Manual control (Jog) with the set speed;
- Absolute and relative positioning to desired position;
- Acceleration and deceleration times with linear ramp;
- Stop of current movement with possibility to restart;
- Possibility to activate hardware or software limit positions;
- Reset of alarm (AlarmReset);
- Digital indication that a positioning is performed.

For details on the configuration of the FR-A740 frequency inverter **click here**.

The project can be used with the following CPUs:FX3U.

System configuration



1. Required hardware

PLC	FX3U
PC	PC with available serial connection
Frequency Inverter	FR-A 740

Servo motor	A servo motor compatible with the frequency inverter (see frequency inverter manual)
PC-PLC connection	SC-09

2. Required software

PLC Software	GX IEC Developer
Library	FX3UPulsePositioningFRA700_Vxxx.sul
Project	FX3UPulsePositioning
Function Blocks	PulsposFX3U_FRA700controlAx1
User Program	ExampleControlAx1 ExampleSequenceAx1

3. Additional documentation (Online manuals)

FX3U Programming manual	Programming Manual	Art. No. 132738
FR-A700 Instruction Manual (Applied)	Instruction Manual	Art. No. IB(NA)-0600257ENG-A

4. Wiring and set-up of frequency inverter

Frequency inverter will be connected using the wiring described in the provided document (three phase 400 VAC, one axis - **1608E.pdf**), The wiring shows a standard connection of a FR-A 740 frequency inverter with a servo motor and connected to a Mitsubishi PLC-system FX3U.

5. Programming

This program uses the function block **"PulsposFX3U_FRA700controlAx1**" from library ***"FX3UPulsePositioningFRA700_Vxxx.sul"*** .

5.1 Internal data registers and auxiliary relays in the FX3U

The table below describes the system special auxiliary relays and data registers used in the PLC-program with usage of internal positioning instrunctions.

Input	Type	Description
D8340+D8141	Current position	Current position for PLC-output Y0 [pulse].
D8345	ZeroReturnCreepSpd	Zero Return creep speed.
D8346	ZeroReturnHighSpd	Zero Return high speed.
D8343-D8344	Maximum speed	Maximum frequency of output pulse, 10-100.000 [Hz].
D8348	Acceleration time	Acceleration time [msec] for DDRVI and DDRVA instructions.
D8349	Deceleration time	Deceleration time [msec] for DDRVI and DDRVA instructions.
M8340	-	Pulse output monitor.

5.2 Global Variables

The images below describe the global variables used in the program example. These variables are mapped to the internal system registers and external I/O terminals of the PLC system used for the positioning control functionality.

Global Variable List

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	
10	VAR_GLOBAL	CurrPosAx1	D8340	%MD0.8340	DINT	...
11	VAR_GLOBAL	MaxSpeedAx1	D8343	%MD0.8343	DINT	...
12	VAR_GLOBAL	HighHomingSpeedAx1	D8346	%MD0.8346	DINT	...

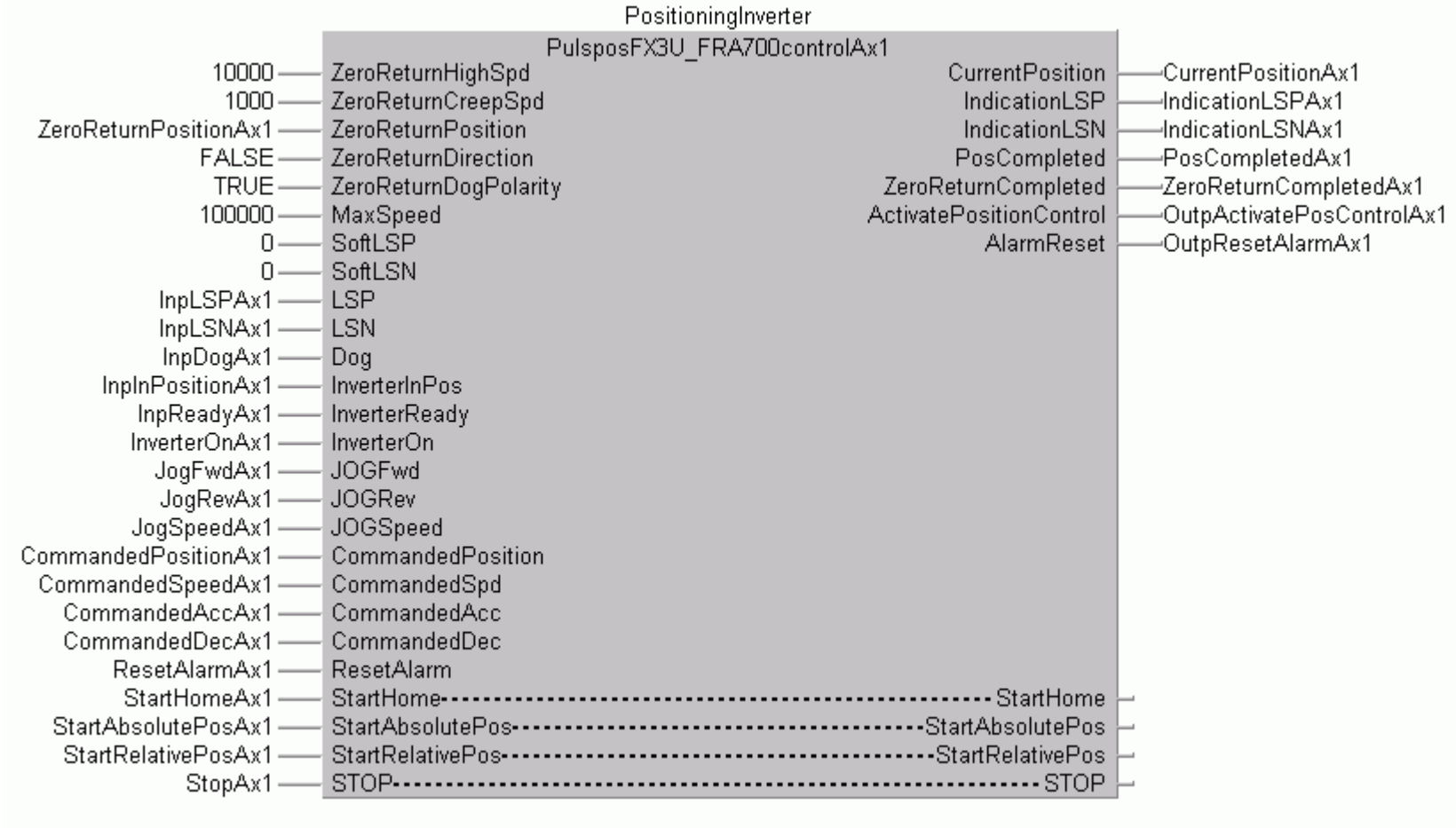
Global Variable List

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	
24	VAR_GLOBAL	InpInPositionAx1	X0	%IX0	BOOL	...
25	VAR_GLOBAL	InpReadyAx1	X1	%IX1	BOOL	...
26	VAR_GLOBAL	InpAlarmAx1	X2	%IX2	BOOL	...
27	VAR_GLOBAL	InpLSPAx1	X3	%IX3	BOOL	...
28	VAR_GLOBAL	InpLSNAx1	X4	%IX4	BOOL	...
29	VAR_GLOBAL	InpDogAx1	X5	%IX5	BOOL	...
30	VAR_GLOBAL	OutpPulsetrainAx1	Y0	%QX0	BOOL	...
31	VAR_GLOBAL	OutpSignPulsetrainAx1	Y1	%QX1	BOOL	...
32	VAR_GLOBAL	OutClearsignalToAx1	Y2	%QX2	BOOL	...
33	VAR_GLOBAL	OutpActivatePosControlAx1	Y3	%QX3	BOOL	...
34	VAR_GLOBAL	OutpResetAlarmAx1	Y4	%QX4	BOOL	...

Program example has two programs in the POU Pool:

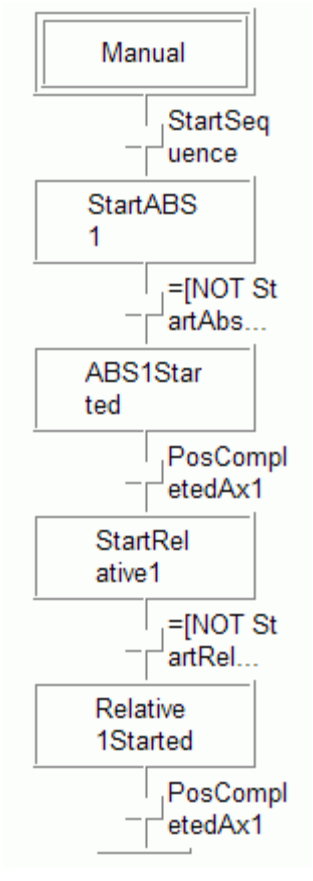
5.3 ExampleControlAx1

This program uses the function block **PulsposFX3U_FRA700controlAx1** to control positioning of the FR-A 740 frequency inverter.



5.4 ExampleSequence

The sequence program uses the input/output variables from the function block **PulsposFX3U_FRA700controlAx1** using the correct settings. The program example *ExampleSequence*, written in SFC, shows how this is done. Obviously you can use another SFC sequence according to the desired operation.
An absolute positioning from position A to position B is performed using the sequence program.



In the start step of the sequence the acceleration/deceleration time, speed and position are defined. Positioning starts with a SET instruction of the StartAbsolutePos variable. When positioning starts and the motor starts rotating, the variable StartAbsolutePos is automatically reset to 0 in the function block. This is how the program jumps to the next step, "Performing positioning". When positioning is done, it generates a pulse with the duration of one program scan, in the output variable PosCompleted. This is used to drive the program to the next step, "Positioning done". A relative positioning starts with StartRelativePos and works the remaining steps according to the same principle.

6. Description of the functions supported by the program example

6.1 Positioning limits setting

Hardware position limit setting can be connected to both the frequency inverter and PLC system. The mechanical setting of a positioning limit controls the motor for an equilibrium of both PLC-system and frequency inverter.

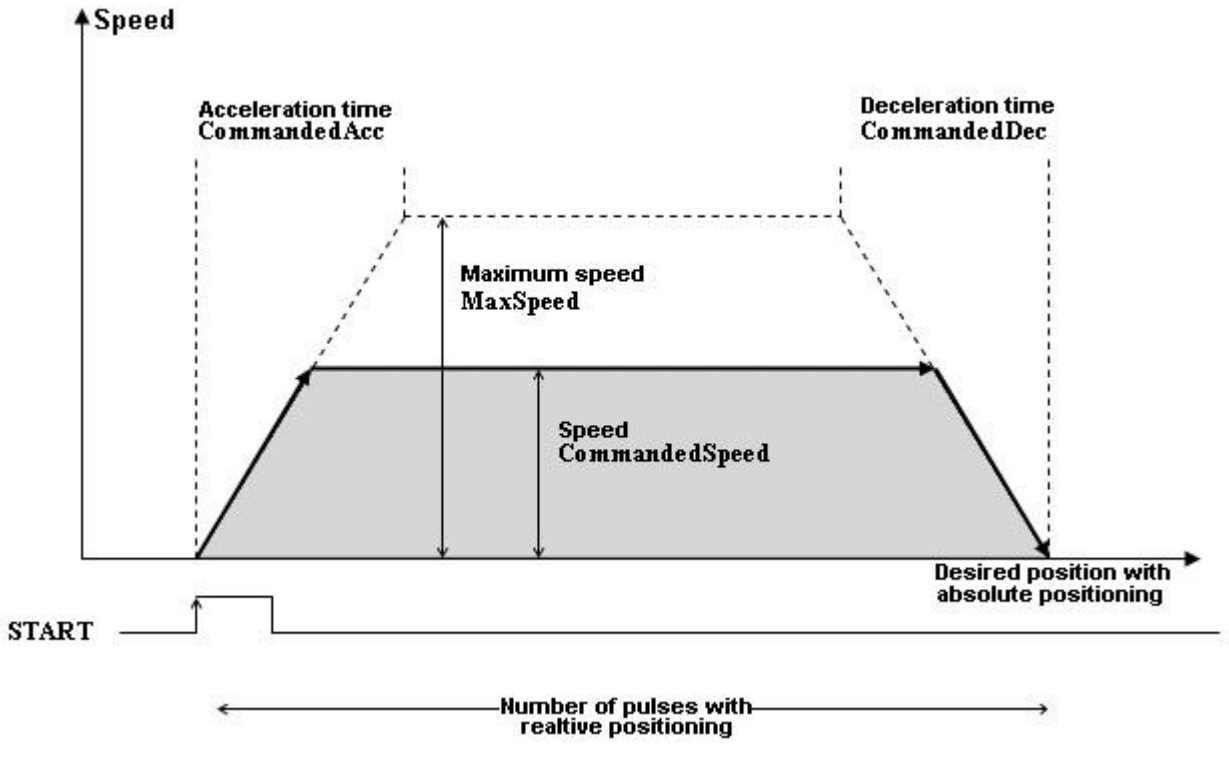
6.2 JOG operation, manual control

Using the JOG-operation a motor can be manually driven with the desired speed without a target position. Speed can not change under operation.

6.3 Absolute positioning

Absolute positioning implies movement control of the servo motor to a specific position with a desired speed, acceleration time and deceleration time.

Position and speed are written into ***CommandedPosition*** respectively ***CommandedSpeed*** input variables. Regardless of the motor current position, it is driven into the position stated in ***CommandedPosition*** input variable.
Speed for positioning is set in ***CommandedSpeed***, acceleration and deceleration times are set in ***CommandedAcc*** and ***CommandedDec*** input variables.
The positioning start signal is given with a SET instruction of the variable ***StartAbsolutePos***.
After the positioning is started the variable is reset automatically. When positioning is done, the variable ***PosCompleted*** is set. To stop a current positioning use the variable ***STOP***. Restart of a stopped positioning is done using the variable ***Restart***.



6.4 Relative positioning

Relative positioning implies control of the servo motor with a desired speed, acceleration time and deceleration time to a specific position. The new current position = the former current position + the positioning distance.
The motor drive distance is given in the input variable ***CommandedPosition***. If the motor should go in a negative direction, indicate that by setting a negative positioning distance.
Speed for positioning is set in ***CommandedSpeed***, acceleration and deceleration times are set in ***CommandedAcc*** and ***CommandedDec*** input variables.
Start signal is given with a SET instruction of the variable ***StartRelativePos***.
The positioning continues with resetting the variables in the function block while operating rewriting the actual position. When positioning is done, the variable ***PosCompleted*** is set. To stop a current positioning use the variable ***STOP***. Restart of a stopped positioning is done using the variable ***Restart***.

Project parts

- **Global Vars** - This is the global variable list. Here you find the name of the global variables used by the program.
- **POU_Pool** - The POU pool contains the programs: **ExampleControlAx1** and **ExampleSequenceAx1**, and the function blocks used by them.
- **Task_Pool** - In the task pool only insert the programs **ExampleControlAx1** and **ExampleSequenceAx1**. See details on how to add a program to the task pool.

ExampleControlThreeAxes

Program description

The example uses 3 FR-A 740 Frequency Inverters with a FR-A7AP plug-in option for vector control, pulse controlled directly via a FX3U PLC-system's transistor outputs, to control 3 AC servo motors for 3 axes, when the incremental encoder system is used. The program example for the PLC-system contains a function block for control and a sequence program used for positioning.

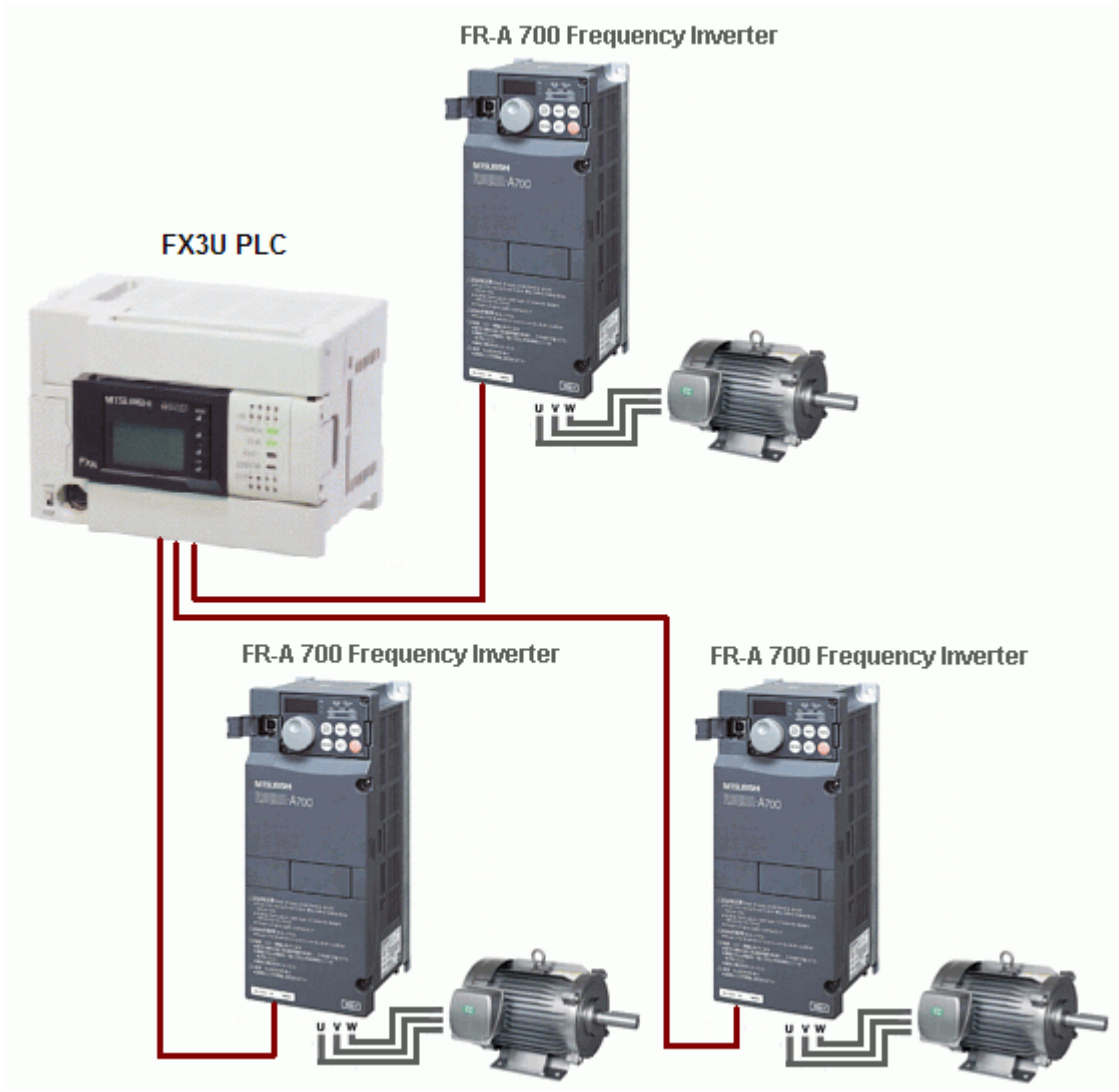
The program example has the following functions:

- Read the servo motor current position into the PLC-system with set of **InverterOn**;
- Enable/disable of **InverterOn** signal;
- Manual control (Jog) with the set speed;
- Absolute and relative positioning to desired position;
- Acceleration and deceleration times with linear ramp;
- Stop of current movement with possibility to restart;
- Possibility to activate hardware or software limit positions;
- Reset of alarm (AlarmReset);
- Digital indication that a positioning is performed.

For details on the configuration of the FR-A740 frequency inverter **click here**.

The project can be used with the following CPUs:FX3U.

System configuration



1. Required hardware

PLC	FX3U
PC	PC with available serial connection
Frequency Inverter	FR-A 740
Servo motor	A servo motor compatible with the frequency inverter (see frequency inverter manual)
PC-PLC connection	SC-09

2. Required software

PLC Software	GX IEC Developer
Library	FX3UPulsePositioningFRA700_Vxxx.sul
Project	FX3UPulsePositioning
Function Blocks	PulsposFX3U_FRA700controlAx13 PulsposFX3U_FRA700controlAx23 PulsposFX3U_FRA700controlAx33
User Program	ExampelControlThreeAxes ExampleSequenceThreeAxes

3. Additional documentation (Online manuals)

FX3U Programming manual	Programming Manual	Art. No. 132738
FR-A700 Instruction Manual (Applied)	Instruction Manual	Art. No. IB(NA)-0600257ENG-A

4. Wiring and set-up of frequency inverter

The frequency inverters will be connected using the wiring described in the provided document (three phase 400 VAC, three axes - **1610E.pdf**), The wiring shows a standard connection of 3 FR-A 740 frequency inverters with the corresponding servo motors and connected to a Mitsubishi PLC-system FX3U.

5. Programming

This program uses the function blocks **"PulsposFX3U_FRA700controlAx13**" ,**"PulsposFX3U_FRA700controlAx23**" and **"PulsposFX3U_FRA700controlAx33**" from library ***"PulsposFX3U_FRA700controlAx3_Vxxx.sul"*** .

5.1 Internal data registers and auxiliary relays in the FX3U

The table below describes the system special auxiliary relays and data registers used in the PLC-program with usage of internal positioning instrunctions.

Input	Type	Description
D8340+D8141	Current position	Current position for PLC-output Y0 [pulse], axis 1.
D8345	ZeroReturnCreepSpd	Zero Return creep speed for axis 1.
D8346	ZeroReturnHighSpd	Zero Return high speed for axis 1.
D8343-D8344	Maximum speed	Maximum frequency of output pulse for axis 1, 10-100.000 [Hz].
D8348	Acceleration time	Acceleration time [msec] for DDRVI and DDRVA instructions for axis 1.
D8349	Deceleration time	Deceleration time [msec] for DDRVI and DDRVA instructions for axis 1.
M8340	-	Pulse output monitor for axis 1.
D8350+D8151	Current position	Current position for PLC-output Y1 [pulse], axis 2 .
D8355	ZeroReturnCreepSpd	Zero Return creep speed for axis 2.
D8356	ZeroReturnHighSpd	Zero Return high speed for axis 2.
D8353-D8354	Maximum speed	Maximum frequency of output pulse for axis 2, 10-100.000 [Hz].
D8358	Acceleration time	Acceleration time [msec] for DDRVI and DDRVA instructions for axis 2.
D8359	Deceleration time	Deceleration time [msec] for DDRVI and DDRVA instructions for axis 2.
M8350	-	Pulse output monitor for axis 2.
D8360+D8161	Current position	Current position for PLC-output Y2 [pulse], axis 3 .
D8365	ZeroReturnCreepSpd	Zero Return creep speed for axis 3.
D8366	ZeroReturnHighSpd	Zero Return high speed for axis 3.
D8363-D8364	Maximum speed	Maximum frequency of output pulse for axis 3, 10-100.000 [Hz].
D8368	Acceleration time	Acceleration time [msec] for DDRVI and DDRVA instructions for axis 3.
D8369	Deceleration time	Deceleration time [msec] for DDRVI and DDRVA instructions for axis 3.
M8360	-	Pulse output monitor for axis 3.

5.2 Global Variables

The images below describe the global variables used in the program example. These variables are mapped to the internal system registers and external I/O terminals of the PLC system used for the positioning control functionality.

Global Variable List

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	
25	VAR_GLOBAL	CurrPosAx1	D8340	%MD0.8340	DINT	...
26	VAR_GLOBAL	MaxSpeedAx1	D8343	%MD0.8343	DINT	...
27	VAR_GLOBAL	HighHomingSpeedAx1	D8346	%MD0.8346	DINT	...
28	VAR_GLOBAL	CurrPosAx2	D8350	%MD0.8350	DINT	...
29	VAR_GLOBAL	MaxSpeedAx2	D8353	%MD0.8353	DINT	...
30	VAR_GLOBAL	HighHomingSpeedAx2	D8356	%MD0.8356	DINT	...
31	VAR_GLOBAL	CurrPosAx3	D8360	%MD0.8360	DINT	...
32	VAR_GLOBAL	MaxSpeedAx3	D8363	%MD0.8363	DINT	...
33	VAR_GLOBAL	HighHomingSpeedAx3	D8366	%MD0.8366	DINT	...

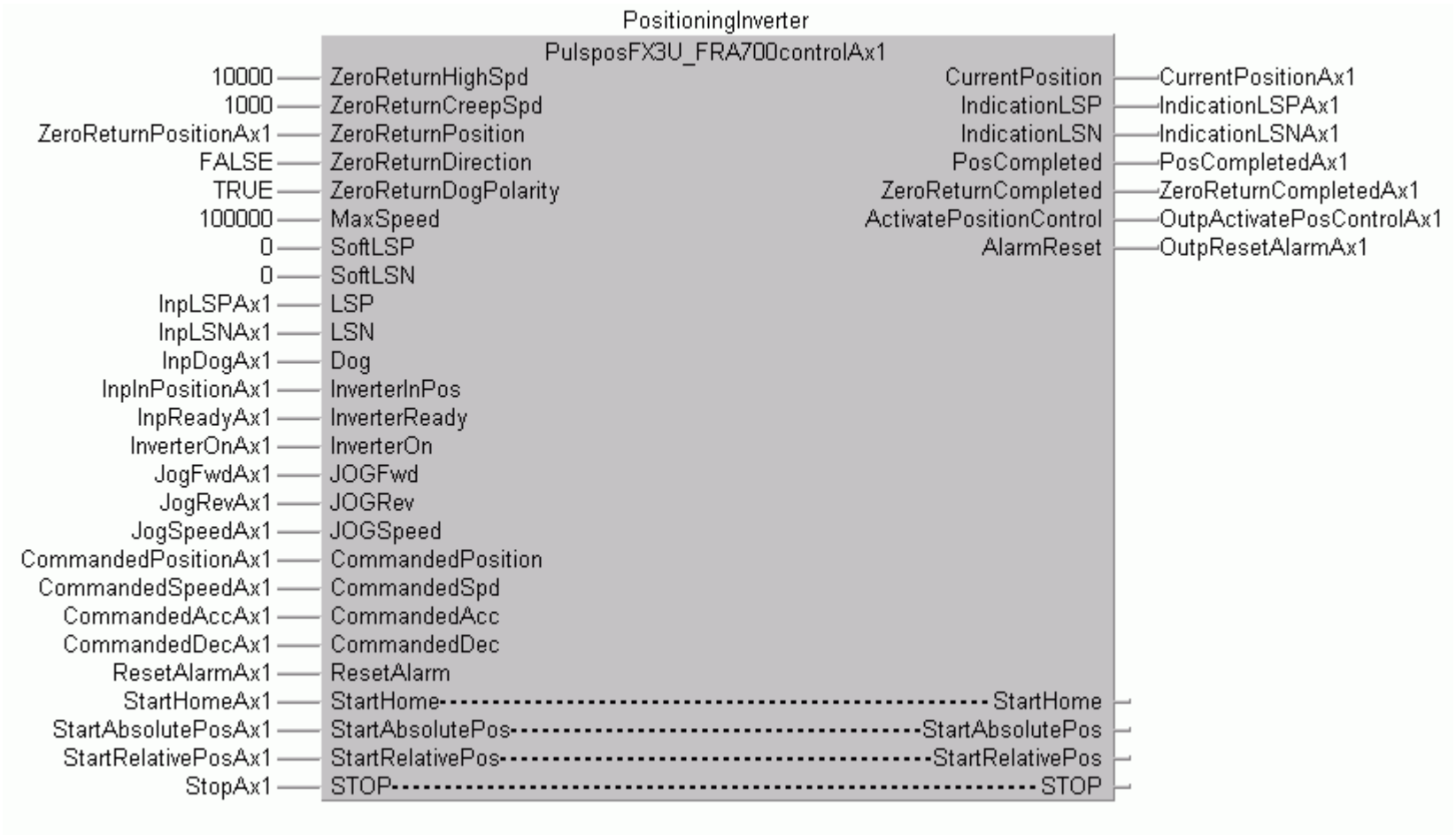
Global Variable List

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type
70	VAR_GLOBAL	InpInPositionAx1	X0	%IX0	BOOL
71	VAR_GLOBAL	InpReadyAx1	X1	%IX1	BOOL
72	VAR_GLOBAL	InpAlarmAx1	X2	%IX2	BOOL
73	VAR_GLOBAL	InpLSPAx1	X3	%IX3	BOOL
74	VAR_GLOBAL	InpLSNAx1	X4	%IX4	BOOL
75	VAR_GLOBAL	InpDogAx1	X5	%IX5	BOOL
76	VAR_GLOBAL	InpDogAx2	X6	%IX6	BOOL
77	VAR_GLOBAL	InpDogAx3	X7	%IX7	BOOL
78	VAR_GLOBAL	InpReadyAx2	X10	%IX8	BOOL
79	VAR_GLOBAL	InpLSPAx2	X11	%IX9	BOOL
80	VAR_GLOBAL	InpLSNAx2	X12	%IX10	BOOL
81	VAR_GLOBAL	InpInPositionAx2	X13	%IX11	BOOL
82	VAR_GLOBAL	InpAlarmAx2	X14	%IX12	BOOL
83	VAR_GLOBAL	InpInPositionAx3	X15	%IX13	BOOL
84	VAR_GLOBAL	InpReadyAx3	X16	%IX14	BOOL
85	VAR_GLOBAL	InpAlarmAx3	X17	%IX15	BOOL
86	VAR_GLOBAL	InpLSPAx3	X20	%IX16	BOOL
87	VAR_GLOBAL	InpLSNAx3	X21	%IX17	BOOL
88	VAR_GLOBAL	OutpPulsetrainAx1	Y0	%QX0	BOOL
89	VAR_GLOBAL	OutpPulsetrainAx2	Y1	%QX1	BOOL
90	VAR_GLOBAL	OutpPulsetrainAx3	Y2	%QX2	BOOL
91	VAR_GLOBAL	OutpSignPulsetrainAx1	Y3	%QX3	BOOL
92	VAR_GLOBAL	OutpSignPulsetrainAx2	Y4	%QX4	BOOL
93	VAR_GLOBAL	OutpSignPulsetrainAx3	Y5	%QX5	BOOL
94	VAR_GLOBAL	OutClearsignalToAx1	Y6	%QX6	BOOL
95	VAR_GLOBAL	OutClearsignalToAx2	Y7	%QX7	BOOL
96	VAR_GLOBAL	OutClearsignalToAx3	Y10	%QX8	BOOL
97	VAR_GLOBAL	OutpActivatePosControlAx1	Y11	%QX9	BOOL
98	VAR_GLOBAL	OutpResetAlarmAx1	Y12	%QX10	BOOL
99	VAR_GLOBAL	OutpActivatePosControlAx2	Y13	%QX11	BOOL
100	VAR_GLOBAL	OutpResetAlarmAx2	Y14	%QX12	BOOL
101	VAR_GLOBAL	OutpActivatePosControlAx3	Y15	%QX13	BOOL
102	VAR_GLOBAL	OutpResetAlarmAx3	Y16	%QX14	BOOL

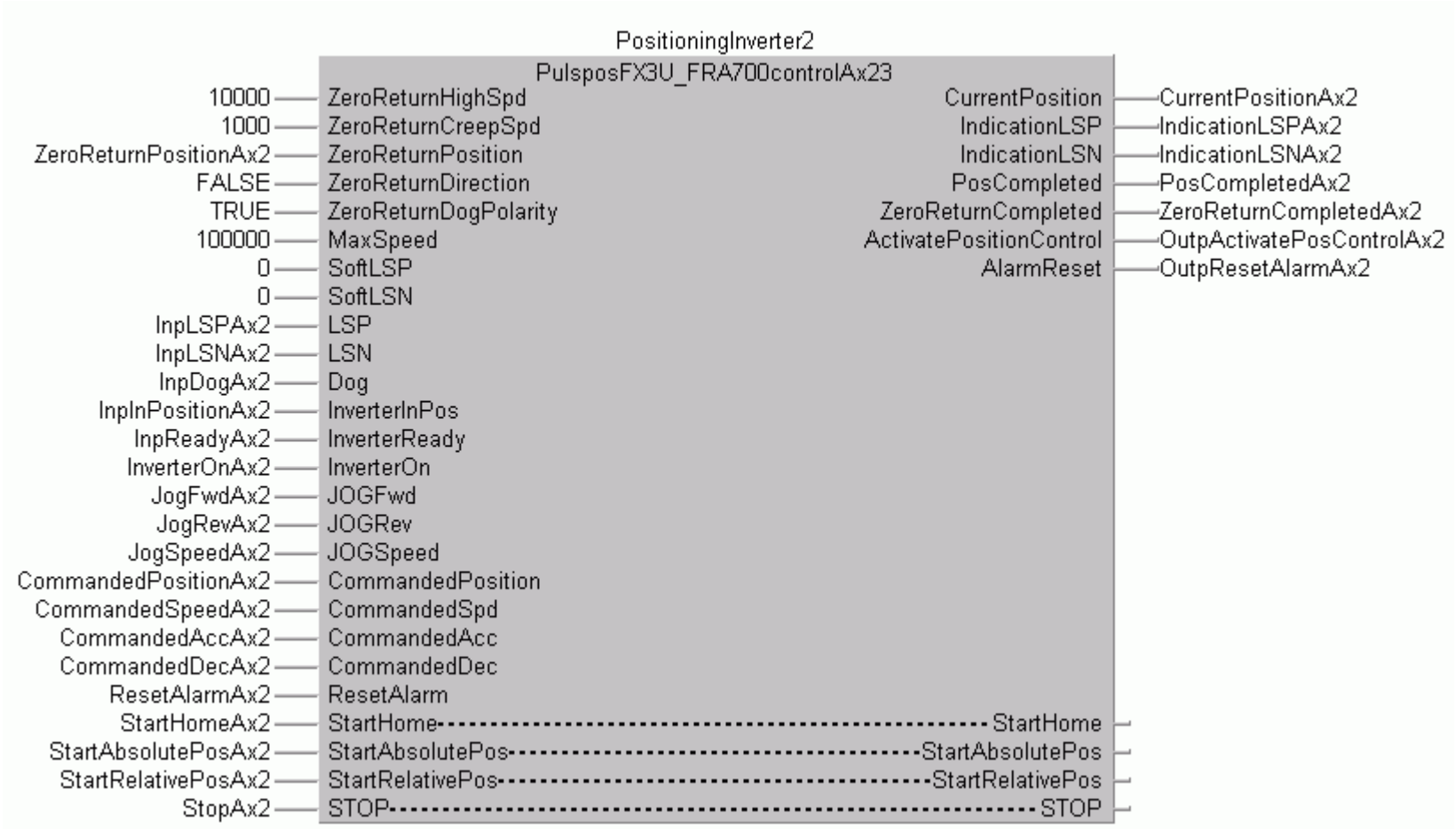
Program example has two programs in the POU Pool:

5.3 ExampleControlThreeAxes

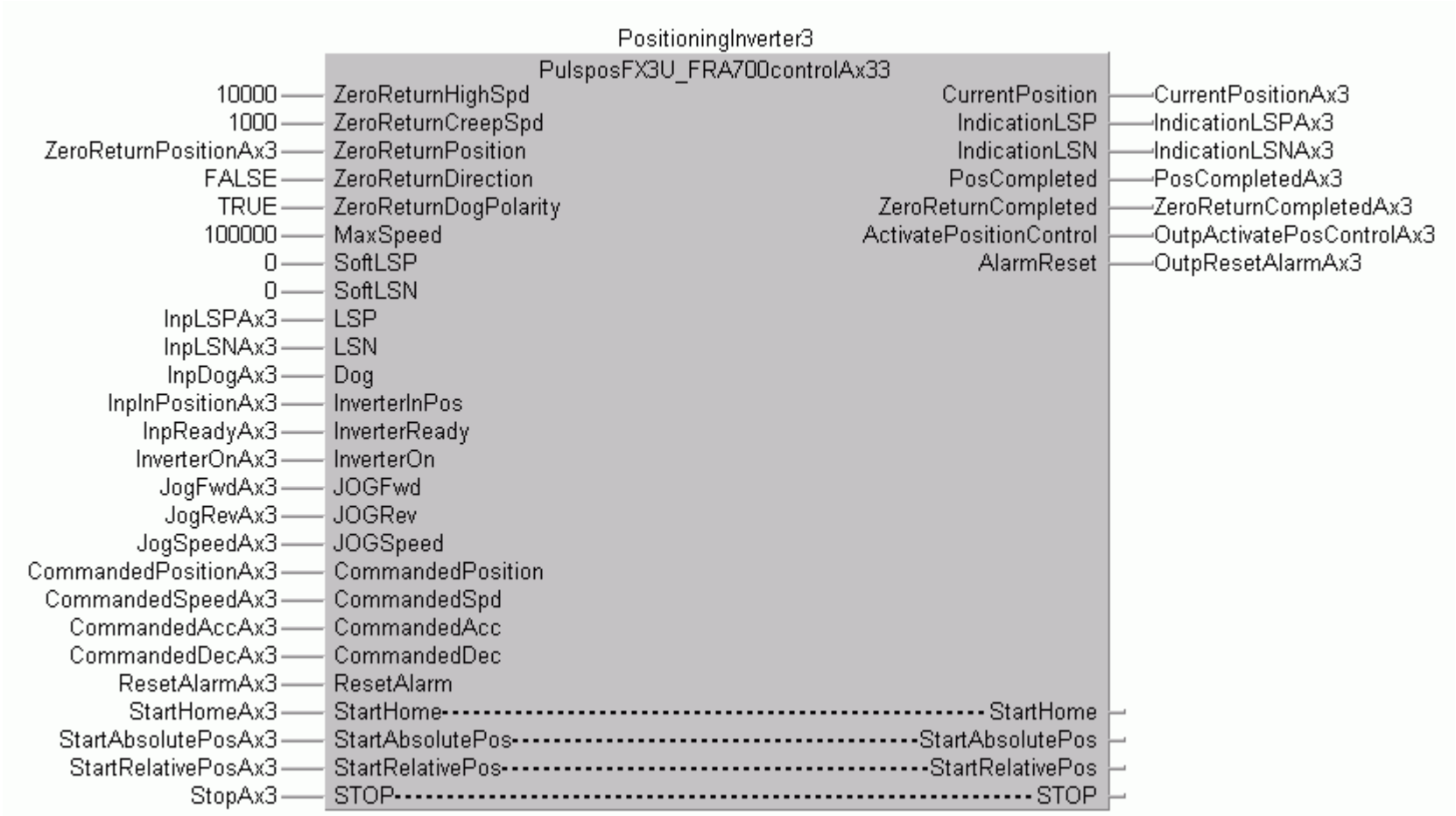
This program uses the function block **PulsposFX3U_FRA700controlAx13** to control positioning on axis 1 using the the first FR-A740 frequency inverter.



The program uses the function block **PulsposFX3U_FRA700controlAx23** to control positioning on axis 2 using the the second FR-A740 frequency inverter.

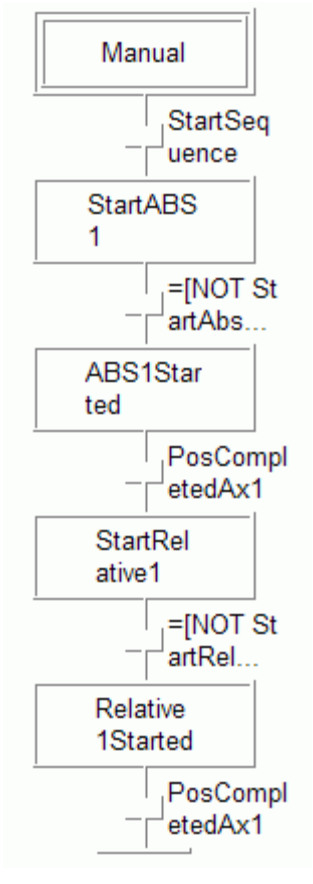


The program uses the function block **PulsposFX3U_FRA700controlAx33** to control positioning on axis 3 using the the third FR-A740 frequency inverter.



5.4 ExampleSequence

The sequence program uses the input/output variables from the function block **PulsposFX3U_FRA700controlAx1** using the correct settings. The program example *ExampleSequence*, written in SFC, shows how this is done. Obviously you can use another SFC sequence according to the desired operation.
An absolute positioning from position A to position B is performed using the sequence program.



In the start step of the sequence the acceleration/deceleration time, speed and position are defined. Positioning starts with a SET instruction of the StartAbsolutePos variable. When positioning starts and the motor starts rotating, the variable StartAbsolutePos is automatically reset to 0 in the function block. This is how the program jumps to the next step, "Performing positioning". When positioning is done, it generates a pulse with the duration of one program scan, in the output variable PosCompleted. This is used to drive the program to the next step, "Positioning done". A relative positioning starts with StartRelativePos and works the remaining steps according to the same principle.

6. Description of the functions supported by the program example

6.1 Positioning limits setting

Hardware position limit setting can be connected to both the frequency inverter and PLC system. The mechanical setting of a positioning limit controls the motor for an equilibrium of both PLC-system and frequency inverter.

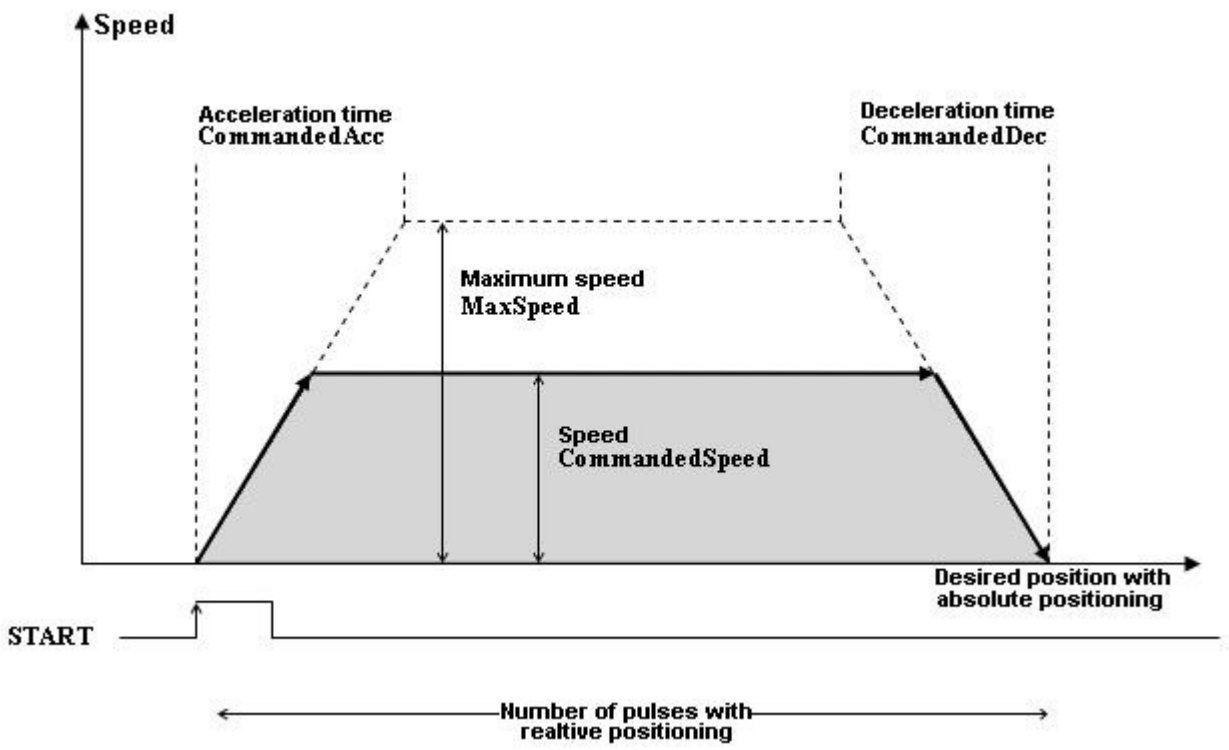
6.2 JOG operation, manual control

Using the JOG-operation a motor can be manually driven with the desired speed without a target position. Speed can not change under operation.

6.3 Absolute positioning

Absolute positioning implies movement control of the servo motor to a specific position with a desired speed, acceleration time and deceleration time.

Position and speed are written into ***CommandedPosition*** respectively ***CommandedSpeed*** input variables. Regardless of the motor current position, it is driven into the position stated in ***CommandedPosition*** input variable.
Speed for positioning is set in ***CommandedSpeed***, acceleration and deceleration times are set in ***CommandedAcc*** and ***CommandedDec*** input variables.
The positioning start signal is given with a SET instruction of the variable ***StartAbsolutePos***.
After the positioning is started the variable is reset automatically. When positioning is done, the variable ***PosCompleted*** is set. To stop a current positioning use the variable ***STOP***. Restart of a stopped positioning is done using the variable ***Restart***.



6.4 Relative positioning

Relative positioning implies control of the servo motor with a desired speed, acceleration time and deceleration time to a specific position. The new current position = the former current position + the positioning distance.
The motor drive distance is given in the input variable ***CommandedPosition***. If the motor should go in a negative direction, indicate that by setting a negative positioning distance.
Speed for positioning is set in ***CommandedSpeed***, acceleration and deceleration times are set in ***CommandedAcc*** and ***CommandedDec*** input variables.
Start signal is given with a SET instruction of the variable ***StartRelativePos***.
The positioning continues with resetting the variables in the function block while operating rewriting the actual position. When positioning is done, the variable ***PosCompleted*** is set. To stop a current positioning use the variable ***STOP***. Restart of a stopped positioning is done using the variable ***Restart***.

Project parts

- **Global Vars** - This is the global variable list. Here you find the name of the global variables used by the program.
- **POU_Pool** - The POU pool contains the programs: **ExampleControlThreeAxes** and **ExampleSequenceThreeAxes**, and the function blocks used by them.
- **Task_Pool** - In the task pool only insert the programs **ExampleControlThreeAxes** and **ExampleSequenceThreeAxes**. See details on how to add a program to the task pool.