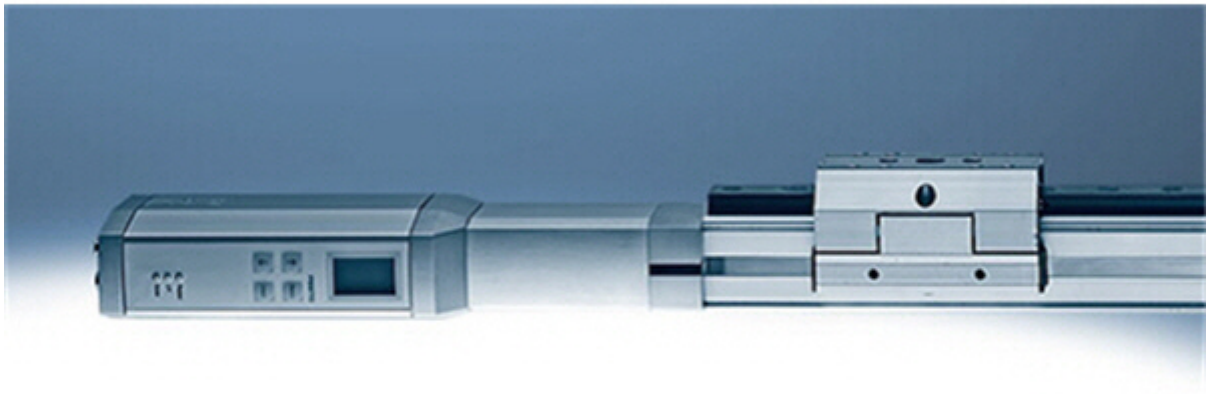
APN 042

Electrical Motor Controllers by Festo on the JX2 System Bus



Application Note

Jetter

Application Note: 042

Item # 60875634

Revision 1.01

October, 2010 / Printed in Germany

Jetter AG reserve the right to make alterations to their products in the interest of technical progress. These alterations need not be documented in every single case.

This Application Note and the information contained herein have been compiled with due diligence. However, Jetter AG assume no liability for printing or other errors or damages arising from such errors.

The brand names and product names used in this document are trademarks or registered trademarks of the respective title owner.

Table of Contents

1 Engineering

Purpose of this Chapter

This chapter is for supporting you in the following activities when engineering a plant equipped with Festo motor controllers:

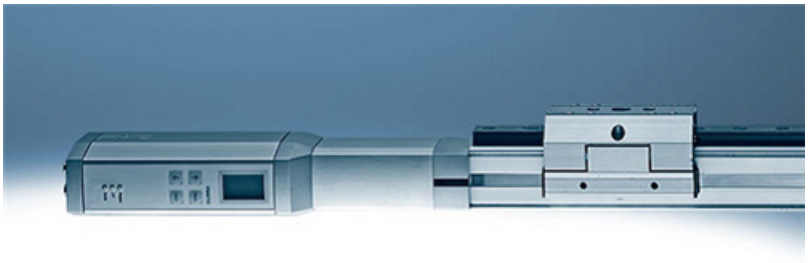
- Selecting ordering date for electric motor controllers by Festo
- Connecting electric motor controllers to the JX2 system bus

Contents

Product Description - Festo MTR-DCI

MTR-DCI

The motor controller MTR-DCI is a product by Festo AG & Co. The motor controller MTR-DCI is an innovative motor with integrated power electronics and has been designed for positioning tasks.



Designation	Description
MTR-DCI-32-xxx-CO	Frame size 32 with CANopen® interface
MTR-DCI-42-xxx-CO	Frame size 42 with CANopen® interface
MTR-DCI-52-xxx-CO	Frame size 52 with CANopen® interface
MTR-DCI-62-xxx-CO	Frame size 62 with CANopen® interface

Technical Data

Number of MTR-DCI units connected to JX2 system bus	10 max. additionally limited by the number of axes of the controller

Minimum Requirements

The motor controller MTR-DCI can be connected to the JX2 system bus of the following controllers and modules by Jetter AG:

Controller / Module	Starting from software release
JC-340	V 1.10.0.00
JC-350	V 1.10.0.00
JC-360	V 1.10.0.00

Product Description - Festo SFC-DC

SFC-DC

The motor controller SFC-DC is a product by Festo AG & Co. The motor controller SFC-DC is used as positioning and position feedback controller for electric mini slides SLTE.

Designation	Description
SFC-DC-xxx-H0-CO	without control panel with CANopen® interface
SFC-DC-xxx-H2-CO	with control panel with CANopen® interface

Technical Data

Number of SFC-DC units connected to JX2 system bus	10 max. additionally limited by the number of axes of the controller

Minimum Requirements

The motor controller SFC-DC can be connected to the JX2 system bus of the following controllers and modules by Jetter AG:

Controller / Module	Starting from software release
JC-340	V 1.10.0.00
JC-350	V 1.10.0.00
JC-360	V 1.10.0.00

Product Description - Festo SFC-LAC

SFC-LAC

The motor controller SFC-LAC is a product by Festo AG & Co. The motor controller SFC-LAC is used as positioning and position feedback controller for linear handling axes HME.

Designation	Description
SFC-LAC-xxx-H0-CO	without control panel with CANopen® interface
SFC-LAC-xxx-H2-CO	with control panel with CANopen® interface

Technical Data

Number of SFC-LAC units connected to JX2 system bus	10 max. additionally limited by the number of axes of the controller

Minimum Requirements

The motor controller SFC-LAC can be connected to the JX2 system bus of the following controllers and modules by Jetter AG:

Controller / Module	Starting from software release
JC-340	V 1.10.0.00
JC-350	V 1.10.0.00
JC-360	V 1.10.0.00

Product Description - Festo SFC-LACI

SFC-LACI

The motor controller SFC-LACI is a product by Festo AG & Co. The motor controller SFC-LACI is used as positioning and position feedback controller for the following electric drives: DNCE-...-LAS and DFME-...-LAS.

Designation	Description
SFC-LACI-xxx-H0-CO	without control panel with CANopen® interface
SFC-LACI-xxx-H2-CO	with control panel with CANopen® interface

Technical Data

Number of SFC-LACI units connected to JX2 system bus	10 max. additionally limited by the number of axes of the controller

Minimum Requirements

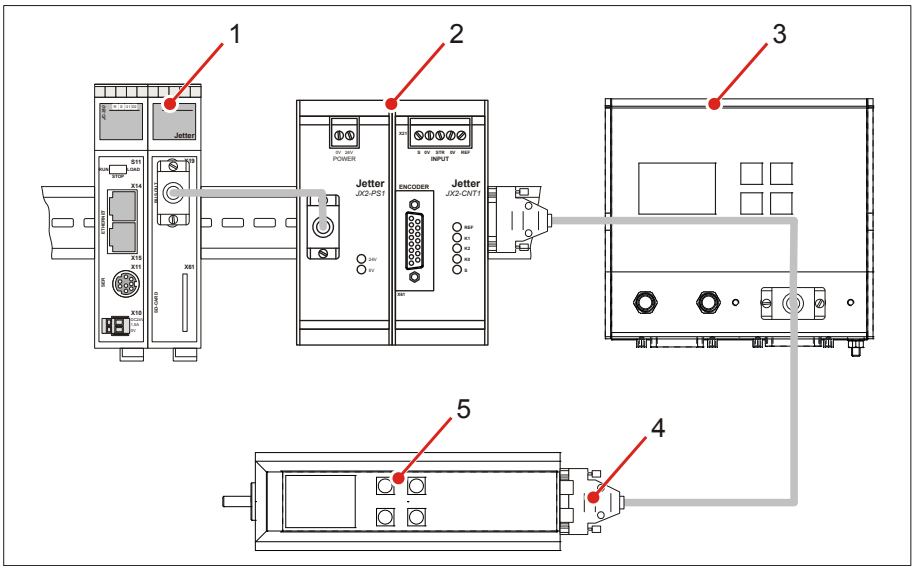
The motor controller SFC-LACI can be connected to the JX2 system bus of the following controllers and modules by Jetter AG:

Controller / Module	Starting from software release
JC-340	V 1.10.0.00
JC-350	V 1.10.0.00
JC-360	V 1.10.0.00

Installing Modules on the JX2 System Bus

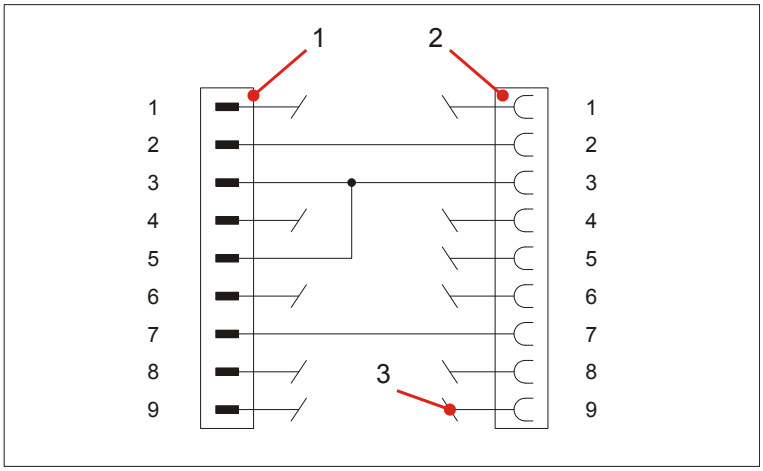
Connection to the JX2 System Bus

Electric motor controllers can directly be connected to the JX2 system bus downstream of JX2-I/O and JX2 slave modules.



Number	Element	Function
1	JC-350	Controller
2	JX2-I/O	JX2-I/O modules on the JX2 system bus
3	SFC-DC	Motor controller by Festo
4	Line end	Line end with integrated bus termination
5	MTR-DCI	Motor unit by Festo

Connection Diagram



Number	Element	Function
1	Male SUB-D connector, 9 pins	for connection to BUS OUT of the last JX2-I/O or JX2-Slave module
2	Female SUB-D connector, 9-pins	for connection to the first CANopen module
3	Not connected	Do not connect these pins

Male SUB-D connector on the JX2 module

Pin	Signal Name	Function
2	CL	Data signal
3	GND	Reference potential
5	TERM	Short-circuited with pin 3
7	CH	Data signal

Female SUB-D connector on CANopen® module

Pin	Signal Name	Function
2	CANL	Data signal
3	GND	Reference potential
7	CANH	Data signal

Connection via Tap Line

If a tap line is used for connecting modules, reflections on the JX2 system bus are caused. When engineering a plant refrain from using tap lines.

Bus Terminating Resistor

Electric motor controllers by Festo are not equipped with a bus terminating resistor. Therefore, connect a 120 ohms terminating resistor between signals CANL (pin 2) and CANH (pin 7) of the last motor controller on the JX2 system bus.

Cables for JX2 System Bus

For connecting modules to the JX2 system bus you can order from Jetter AG the following cables :

Item #	Item
10309001	Cable assy # 530 0.2 m
10309002	Cable assy # 530 0.5 m
10309003	Cable assy # 530 1.0 m
10309004	Cable assy # 530 1.5 m
10309006	Cable assy # 530 2.0 m
10309016	Cable assy # 530 2.5 m
10309015	Cable assy # 530 3.0 m
10309007	Cable assy # 530 4.0 m
10309008	Cable assy # 530 5.0 m

Configuring the Controller JC-3xx

Introduction

The controller JC-3xx is able to automatically detect and commission the connected motor controllers.

Configuration using a CANopen® EDS file is not needed.

Baud Rate

The baud rate setting depends on the number of modules connected to the JX2 system bus.

JX2-I/O modules JX2-Slave modules JetMove	Electric motor controller	1000 kBaud	500 kBaud	250 kBaud	125 kBaud
x		x	x	x	x
	x	x	x	x	x
x	x	x			x

Configuring Electric Motor Controllers

Software Tool FCT by Festo	Electric motor controllers are configured using the FCT software tool by Festo.
Setting the Baud Rate	Set the baud rate of the motor controllers to the same values set in JetControl.
Setting the Node ID	Set the node ID at the Festo direct drives to a value between 70 and 79.

Identification via Module Array

Introduction

The controller JC-3xx enters all modules found on the JX2 system bus into the module array. Each module has a unique module code.

Module Codes of Electric Motor Controllers

Electric motor controller	Module code
SFC-LAC	108
SFC-LACI	109
SFC-DC	110
MTR-DCI	111

2 Initial Commissioning

Purpose of this Chapter

This chapter gives a simple description of the initial commissioning of the electric motor controller by Festo and covers the following functions:

- Carrying out a reference run

Prerequisites

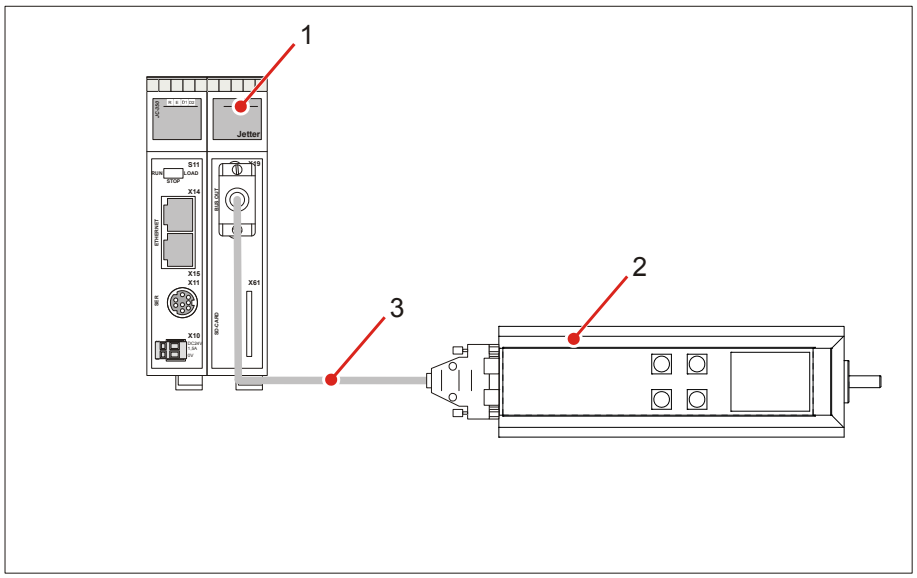
To be able to commission the electric motor controller the following prerequisites have to be fulfilled:

- The node ID of the motor controller is 70.
- The controller JC-3xx and the motor controller have been set to the same baud rate.
- The electric motor controller is connected to the JX2 system bus of a JC-3xx controller.
- The controller JC-3xx is connected to a PC.
- The programming tool JetSym is installed on the PC.
- The minimum requirements regarding modules, controllers and software are fulfilled.

Contents

Preparatory Work for Initial Commissioning

Connection to the JX2 System Bus



Number	Element	Function
1	JC-3xx	Controller
2	MTR-DCI	Motor unit by Festo
3	Cable	JX2 System Bus Cable

Condition of the LEDs

If no errors are present, the LEDs on the controller JC-3xx and on the motor controller have the following states.

Module	LED	State
JC-3xx	R	Green, lit
MTR-DCI	Power	Green, lit
MTR-DCI	I/F	Green, lit

Starting JetSym

Step	Action
1	Launch the software tool JetSym on your PC.
2	Create a new project.
3	Establish in JetSym a connection to the controller JC-3xx.
4	In JetSym, open a setup window.

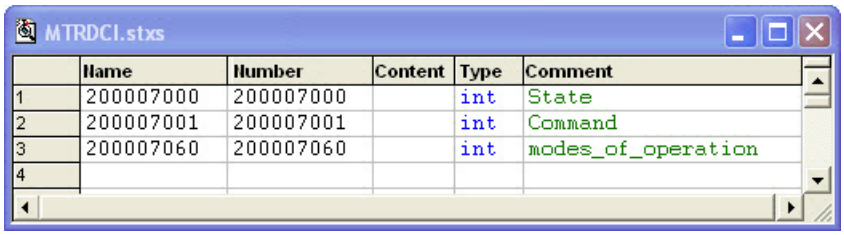
Initial Commissioning Along with a JC-3xx

Introduction

Initial commissioning is for activating the referencing process of the electric motor controller.

Overview of Registers in JetSym Setup

The node ID of the motor controller MTR-DCI is 70. From this node ID the following register numbers result for initial commissioning:



Registers	Name	Description
200007000	State	State of MTR-DCI
200007001	Command	Commands to MTR-DCI
200007060	modes_of_operation	Operating mode

Initiating the Referencing Process

Enter in the setup window of JetSym the values given below into the registers of the motor controller MTR-DCI in the following order:

Step	Action
1	Enter command *shutdown* into the MTR-DCI register: R 200007001 = 6
2	Enter command *switch on* into the MTR-DCI register: R 200007001 = 7
3	Enter command *enable operation* into the MTR-DCI register: R 200007001 = 15
4	Change *modes_of_operation* to "referencing": R 200007060 = 6
5	Trigger the referencing process using the following command: R 200007001 = 31
⇨	**Result:** The electric motor controller MTR-DCI starts the referencing process.

3 Programming

Introduction

This chapter is for supporting you when programming Festo motor controllers of the MTR-DCI and SFC-... series in the following fields of activity:

- Determining the register numbers depending on the system configuration.
- Programming electric motor controllers by way of example.

Prerequisites

To be able to program electric motor controllers the following prerequisites have to be fulfilled:

- The electric motor controller is connected to a JetControl controller.
- The controller is connected to a PC.
- The programming tool JetSym is installed on the PC.
- The minimum requirements regarding modules, controllers and software are fulfilled.

Contents

3.1 Numbering Registers and I/Os for a JC-3xx Controller

Introduction

Controllers and modules produced by Jetter AG offer a host of functions which can be accessed by the user via registers. A unique number is assigned to each register and each digital input or output.

Usage: Register Number

Register numbers are used in the following cases:

- A module register is to be read or written in the Setup section of JetSym.
- A module register is to be declared as a variable in the application program of JetSym.
- A module register is to be declared as a tag in JetViewSoft.

Usage: I/O Number

I/O numbers are used in the following cases:

- A digital input is to be read in the Setup section of JetSym.
- A digital output is to be read or written in the Setup section of JetSym.
- A digital input or output is to be declared as a variable in the application program of JetSym.
- A digital input or output is to be declared as a tag in JetViewSoft.

Contents

Registers and I/O Numbers of CANopen® Modules on the JX2 System Bus

I/O Module Numbers of CANopen® Modules

I/O module numbers of CANopen® modules connected to the JX2 system bus of a JC-3xx are determined as follows:

- In most cases, the I/O module numbers correspond to the node ID of the CANopen® module.
- Exceptions: SMC EX120 and frequency inverters by Lenze.

Register Numbers for CANopen® Modules

Register numbers for CANopen® modules connected to the JX2 system bus of a JC-3xx consist of the following elements:

2	0	0	0	0	x	x	z	z

Element	Meaning	Value range
xx	I/O Module Number	70 ... 79
z	Module register number	00 99

I/O Numbers for CANopen® Modules

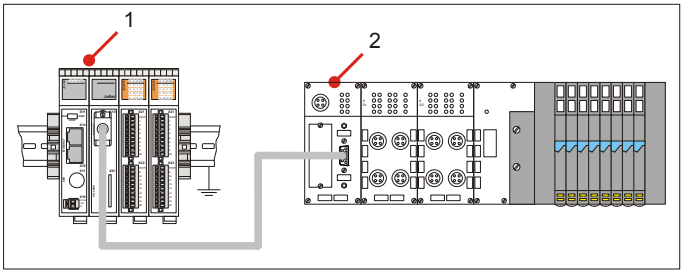
I/O numbers for CANopen® modules connected to the JX2 system bus of a JC-3xx consist of the following elements:

2	0	0	0	0	m	m	z	z

Element	Meaning	Value range
mm	Module specific I/O module number	70 ... 79
zz	Module specific I/O number	1 ... 64

Example

A CANopen® module is connected to a JC-3xx controller.



Number	Module	I/O Module Number	Register(s)	I/O
1	JC-3xx	1	see JC-3xx documentation	
2	Festo CPX	2	2000070zz	2000070zz

3.2 Programming Electric Motor Controllers made by Festo

Introduction

This chapter is for supporting you when programming Festo motor controllers of the MTR-DCI and SFC-... series and provides the following information:

- Description of the interface between CANopen® objects of motor controllers and register numbers.
- Access to CANopen® objects of motor controllers from within the application program.

Contents

Register/Object Assignment

Operating principle

The controller JC-3xx connects the CANopen® interface of motor controllers with its registers. The electric motor controllers by Festo retain their full scope of functions. The documentation provided by Festo remains valid.

Register/Object Assignment

The table below shows the register/object assignment:

Module Register	Object	Description
7x00	0x6041	State
7x01	0x6040	Command
7x02	0x607A	TargetPos
7x03	0x6081	ProfileVelocity
7x04	0x607E	Polarity
7x05	0x6083	ProfileAcceleration
7x06	0x6084	ProfileDeceleration
7x07	0x6067	PositionWindow
7x35	0x6064	PositionActualValue
7x10	0x6062	PositionDemandValue
7x11	0x6068	PositionWindowTime
7x12	0x606C	ActualSpeed
7x16	0x6510	ToolMass
7x17	0x20E0:6	WorkItemMass only SCL-LAC / SFC-LACI
7x18	0x20E0:5	JerkAcceleration only SCL-LAC / SFC-LACI
7x19	0x20E0B	JerkDeceleration only SCL-LAC / SFC-LACI
7x20	0x6065	FollowingErrorWindow
7x30	0x6072	MaxTorque
7x31	0x6073	MaxCurrent
7x32	0x6075	MotorRatedCurrent
7x33	0x6076	MotorRatedTorque
7x34	0x6077	TorqueActualValue
7x37	0x607F	MaxProfileVelocity
7x39	0x6086	MotionProfileType
7x44	0x6044	VlControlEffort
7x50	0x6071	TargetTorque
7x56	0x605A	QuickStopOptionCode

Module Register	Object	Description
7x00	0x6041	State
7x57	0x6085	QuickStopDecceleration
7x59	0x609A	HomingAcceleration
7x60	0x6061	ModesOfOperation
7x61	0x6098	HomingMethod
7x62	0x607C	HomeOffset
7x63	0x6099:1	HomingspeedSwitch
7x64	0x6099:	HomingSpeedZero
7x65 ... 7x80	configurable	free CANopen registers
7x90	0x1001	ErrorRegister
7x91	0x1002	manufacturer status
7x92	Index of MR 7x93	
7x93	0x1003	error field
7x97	0x1018:4	serial number
7x98	0x100D	life time factor
7x99	0x100A	software version

Implementation of CANopen® Services

SDO Upload/Download

At almost every read/write access to a module register of an electric motor controller JetControl 3xx launches an SDO upload or downlad.

PDO Access

The controller JC-3xx sends the object listed below to the motor controller using PDO(tx). In the event of a read access to these module registers/CANopen® objects the controller JC-3xx reads the contents from its own process image. No read access via CANopen® happens.

- MR 7x01 / Object 0x6040: Command
- MR 7x02 / Object 0x607A:TargetPos
- MR 7x60 / Object 0x6061: modes_of_operation
- MR 7x50 / Object 0x6071: TargetTorque
- MR 7x03 / Object 0x6081: ProfileVelocity
- MR 7x17 / Object 0x20E0:6: WorkItemMass

Update des PDO1(rx)

The controller JC-3xx does not send a PDO with every write access to the module registers of PDO1(rx).

Step	Action
1	The user enters a value into MR 7x02 *TargetPos* or MR 7x50 *TargetTorque*
2	The user enters a value into MR 7x60 *modes_of_operation*
3	The user enters a value into MR 7x01 *Command*
4	The controller sends the PDO1(rx) to the electric motor controller. The set number in PDO1(rx) is always 0.

Free CANopen® Registers

Introduction

Read or write access to any object of an electric motor controller can be made via free CANopen® registers.

Overview of Registers

Registers	Description
MR 7x65 ... 7x80	Free CANopen® Registers
MR 7x94	Index (65 ... 79)
MR7x65	Object information

Configuring a Free CANopen® Register

Step	Action
1	Enter the number of the free CANopen® register into MR 7x94. 7x94 := 65;
2	Enter the object information into MR 7x95. The format is: object, sub-index, length. 7x95 := 0x10180104; // object 0x1018, sub-index 0x01, length 4
3	A read access to MR 7x65 will return the contents of object x. 7x65 = 0x0000001D; // Festo vendor ID

Example: Axis Referencing

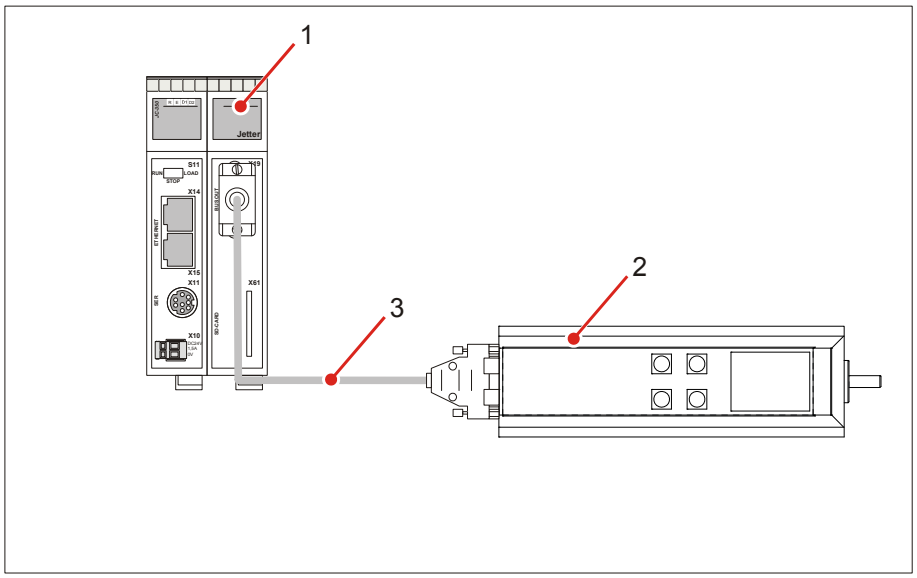
Task

The referencing process is to be carried out automatically in a JetSym STX program.

Solution

First, put the electric motor controller into the state "switched on". Then, start the referencing process.

Configuration



Number	Element	Description
1	JC-3xx	Controller
2	MTR-DCI	Motor unit by Festo Node ID = 70
3	Cable	JX2 system bus cable

JetSym STX Program

```
#Define __OFFSET__(n) n*SizeOf(Int)
Type
    TYPE_MTR_DCI:
    Struct
        State:                  Int At __OFFSET__(0);
        Command:                Int At __OFFSET__(1);
        ModesOfOperation:       Int At __OFFSET__(60);
    End_Struct;
End_Type;

Var
    MTR_DCI:  TYPE_MTR_DCI At %VL 200007000;
End_Var;

Task MTRDCI Autorun
```

```
// Resetting to "Switch on disabled"
MTR_DCI.Command := 0x00;
When
    BitSet(MTR_DCI.State, 6)
Continue;

// Command "shutDown"
// Setting to state "Ready to Switch On"
MTR_DCI.CommAND := 6;
When
    BitSet(MTR_DCI.State, 0) AND
    BitSet(MTR_DCI.State, 5)
Continue;

// Command "Switch on"
// Setting to state "Switched On"
MTR_DCI.Command := 7;
When
    BitSet(MTR_DCI.State, 0) AND
    BitSet(MTR_DCI.State, 1) AND
    BitSet(MTR_DCI.State, 4) AND
    BitSet(MTR_DCI.State, 5)
Continue;

// Command "enable operation"
// Setting to state "Operation Enabled"
MTR_DCI.Command := 0x0F;
When
    BitSet(MTR_DCI.State, 0) AND
    BitSet(MTR_DCI.State, 1) AND
    BitSet(MTR_DCI.State, 2) AND
    BitSet(MTR_DCI.State, 4) AND
    BitSet(MTR_DCI.State, 5)
Continue;

// Operating mode "Referencing"
MTR_DCI.ModesOfOperation := 6;
// Initiating axis referencing
MTR_DCI.Command := 0x1F;
When
    BitSet(MTR_DCI.State, 15)
Continue;
```

Example: Axis Positioning

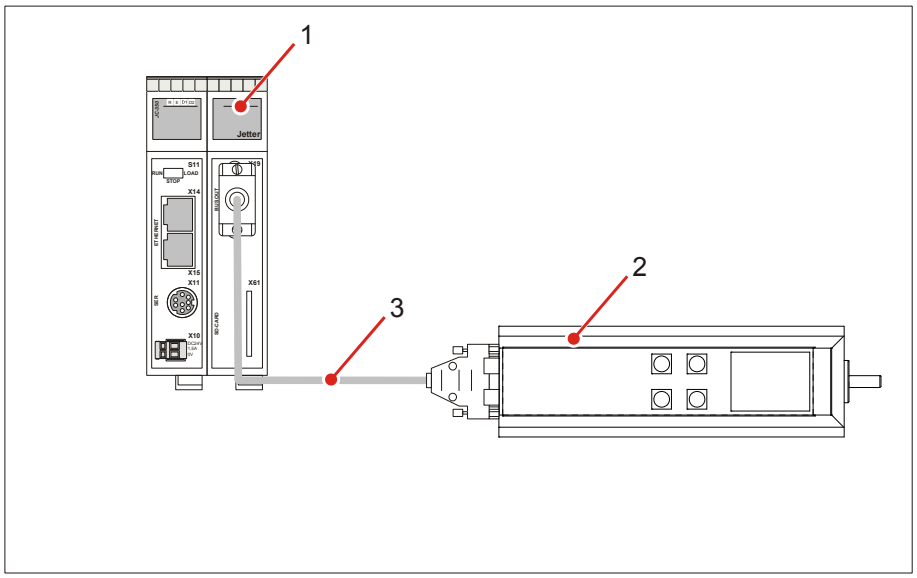
Task

In a JetSym STX program, an MTR-DCI axis is to be moved between two positions.

Solution

First, reference the electric motor controller. Then, start the positioning process.

Configuration



Number	Element	Description
1	JC-3xx	Controller
2	MTR-DCI	Motor unit by Festo Node ID = 70
3	Cable	JX2 system bus cable

JetSym STX Program

```
#Define __OFFSET__(n) n*SizeOf(Int)
Type
    TYPE_MTR_DCI:
    Struct
        State:                  Int At __OFFSET__(0);
        Command:                Int At __OFFSET__(1);
        ModesOfOperation:       Int At __OFFSET__(60);
        TargetPos:              Int At __OFFSET__(2);
    End_Struct;
End_Type;

Var
    MTR_DCI:  TYPE_MTR_DCI At %VL 200007000;
End_Var;
```

```
Task MTRDCI Autorun

    // Axis referencing must be performed first
    // ....
Loop
    MTR_DCI.Command := 0x0F;
    Delay(T#5ms);
    MTR_DCI.TargetPos := 30000;
    MTR_DCI.Command := 0x1F;
    When
        BitSet(MTR_DCI.State, 10) AND
        BitClear(MTR_DCI.State, 8)
    Continue;

    MTR_DCI.Command := 0x0F;
    Delay(T#5ms);
    MTR_DCI.TargetPos := 1000;
    MTR_DCI.Command := 0x1F;

    When
        BitSet(MTR_DCI.State, 10) AND
        BitClear(MTR_DCI.State, 8)
    Continue;
End_Loop;
```

Type Definitions for MTR-DCI in JetSym

Introduction

The following JetSym type definitions are for specifying the module registers of the electric motor controller MTR-DCI. To be able to access these module registers from the application program, you only have to define __OFFSET__ and create a variable of this type.

```
#ifdef __OFFSET__
#undef __OFFSET__
#endif
#Define __OFFSET__(n) n*SizeOf(Int)

Var
    MTR_DCI:  TYPE_MTR_DCI At %VL 200007000;
End_Var;
```

Type Definition

```
Type
    TYPE_MTR_DCI:
    Struct
        State       :           Int At __OFFSET__(0);
        Command:                Int At __OFFSET__(1);
        TargetPos:              Int At __OFFSET__(2);
        ProfileVelocity:        Int At __OFFSET__(3);
        Polarity:               Int At __OFFSET__(4);
        ProfileAcceleration:    Int At __OFFSET__(5);
        ProfileDeceleration:    Int At __OFFSET__(6);
        PositionWinDow:         Int At __OFFSET__(7);
        PositionActualValue:    Int At __OFFSET__(35);
        PositionDemandValue:    Int At __OFFSET__(10);
        PositionWinDowTime:     Int At __OFFSET__(11);
        ActualSpeed:            Int At __OFFSET__(12);
        ToolMass:               Int At __OFFSET__(16);
        FollowingErrorWinDow:   Int At __OFFSET__(20);
        MaxTorque:              Int At __OFFSET__(30);
        MaxCurrent:             Int At __OFFSET__(31);
        MotorRatedCurrent:      Int At __OFFSET__(32);
        MotorRatedTORque:       Int At __OFFSET__(33);
        TorqueActualValue:      Int At __OFFSET__(34);
        MaxProfileVelocity:     Int At __OFFSET__(37);
        MotionProfileType:      Int At __OFFSET__(39);
        VlControlEffort:        Int At __OFFSET__(44);
        TargetTorque:           Int At __OFFSET__(50);
        QuickStopOptionCode:    Int At __OFFSET__(56);
        QuickStopDecceleration: Int At __OFFSET__(57);
        HomingAcceleration:     Int At __OFFSET__(59);
        ModesOfOperation:       Int At __OFFSET__(60);
        HomingMethod:           Int At __OFFSET__(61);
        HomeOffset:             Int At __OFFSET__(62);
        HomingspeedSwitch:      Int At __OFFSET__(63);
```

```
        HomingSpeedZero:        Int At __OFFSET__(64);
        FreeObjetcs:
                  Array[15] Of Int At __OFFSET__(65);
        ErrorRegister:          Int At __OFFSET__(90);
        ManufacturerStatus:     Int At __OFFSET__(91);
        ErrorFieldIndex:        Int At __OFFSET__(92);
        ErrorField:             Int At __OFFSET__(93);
        FreeObjectIndex:        Int At __OFFSET__(94);
        FreeObjectConfig:       Int At __OFFSET__(95);
        SerialNumber:           Int At __OFFSET__(97);
        SWVersion:              Int At __OFFSET__(99);
    End_Struct;
End_Type;
```

Type Definitions for SFC-LAC / SFC-LACI in JetSym

Introduction

The following JetSym type definitions are for specifying the module registers of the electric motor controller SFC-LAC / SFC-LACI. To be able to access these module registers from the application program, you only have to define __OFFSET__ and create a variable of this type.

```
#ifdef __OFFSET__
#undef __OFFSET__
#endif
#Define __OFFSET__(n) n*SizeOf(Int)

Var
    SFC_LAC:  TYPE_SFC_LAC At %VL 200007000;
End_Var;
```

Type Definition

```
Type
    TYPE_SFC_LAC:
    Struct
        State       :           Int At __OFFSET__(0);
        Command:                Int At __OFFSET__(1);
        TargetPos:              Int At __OFFSET__(2);
        ProfileVelocity:        Int At __OFFSET__(3);
        Polarity:               Int At __OFFSET__(4);
        ProfileAcceleration:    Int At __OFFSET__(5);
        ProfileDeceleration:    Int At __OFFSET__(6);
        PositionWinDow:         Int At __OFFSET__(7);
        PositionActualValue:    Int At __OFFSET__(35);
        PositionDemandValue:    Int At __OFFSET__(10);
        PositionWinDowTime:     Int At __OFFSET__(11);
        ActualSpeed:            Int At __OFFSET__(12);
        ToolMass:               Int At __OFFSET__(16);
        WorkItemMass:           Int At __OFFSET__(17);
        JerkAcceleration:       Int At __OFFSET__(18);
        JerkDeceleration:       Int At __OFFSET__(19);
        FollowingErrorWinDow:   Int At __OFFSET__(20);
        MaxTorque:              Int At __OFFSET__(30);
        MaxCurrent:             Int At __OFFSET__(31);
        MotorRatedCurrent:      Int At __OFFSET__(32);
        MotorRatedTORque:       Int At __OFFSET__(33);
        TorqueActualValue:      Int At __OFFSET__(34);
        MaxProfileVelocity:     Int At __OFFSET__(37);
        MotionProfileType:      Int At __OFFSET__(39);
        VlControlEffort:        Int At __OFFSET__(44);
        TargetTorque:           Int At __OFFSET__(50);
        QuickStopOptionCode:    Int At __OFFSET__(56);
        QuickStopDecceleration: Int At __OFFSET__(57);
        HomingAcceleration:     Int At __OFFSET__(59);
        ModesOfOperation:       Int At __OFFSET__(60);
```

```
        HomingMethod:           Int At __OFFSET__(61);
        HomeOffset:             Int At __OFFSET__(62);
        HomingspeedSwitch:      Int At __OFFSET__(63);
        HomingSpeedZero:        Int At __OFFSET__(64);
        FreeObjetcs:
                  Array[15] Of Int At __OFFSET__(65);
        ErrorRegister:          Int At __OFFSET__(90);
        ManufacturerStatus:     Int At __OFFSET__(91);
        ErrorFieldIndex:        Int At __OFFSET__(92);
        ErrorField:             Int At __OFFSET__(93);
        FreeObjectIndex:        Int At __OFFSET__(94);
        FreeObjectConfig:       Int At __OFFSET__(95);
        SerialNumber:           Int At __OFFSET__(97);
        SWVersion:              Int At __OFFSET__(99);
    End_Struct;
End_Type;
```

Type Definitions for SFC-DC in JetSym

Introduction

The following JetSym type definitions are for specifying the module registers of the electric motor controller SFC-DC. To be able to access these module registers from the application program, you only have to define __OFFSET__ and create a variable of this type.

```
#ifdef __OFFSET__
#undef __OFFSET__
#endif
#Define __OFFSET__(n) n*SizeOf(Int)

Var
    SFC_LAC:  TYPE_SFC_DC At %VL 200007000;
End_Var;
```

Type Definition

```
Type
    TYPE_SFC_DC:
    Struct
        State       :           Int At __OFFSET__(0);
        Command:                Int At __OFFSET__(1);
        TargetPos:              Int At __OFFSET__(2);
        ProfileVelocity:        Int At __OFFSET__(3);
        Polarity:               Int At __OFFSET__(4);
        ProfileAcceleration:    Int At __OFFSET__(5);
        ProfileDeceleration:    Int At __OFFSET__(6);
        PositionWinDow:         Int At __OFFSET__(7);
        PositionActualValue:    Int At __OFFSET__(35);
        PositionDemandValue:    Int At __OFFSET__(10);
        PositionWinDowTime:     Int At __OFFSET__(11);
        ActualSpeed:            Int At __OFFSET__(12);
        ToolMass:               Int At __OFFSET__(16);
        FollowingErrorWinDow:   Int At __OFFSET__(20);
        MaxTorque:              Int At __OFFSET__(30);
        MaxCurrent:             Int At __OFFSET__(31);
        MotorRatedCurrent:      Int At __OFFSET__(32);
        MotorRatedTORque:       Int At __OFFSET__(33);
        TorqueActualValue:      Int At __OFFSET__(34);
        MaxProfileVelocity:     Int At __OFFSET__(37);
        MotionProfileType:      Int At __OFFSET__(39);
        VlControlEffort:        Int At __OFFSET__(44);
        TargetTorque:           Int At __OFFSET__(50);
        QuickStopOptionCode:    Int At __OFFSET__(56);
        QuickStopDecceleration: Int At __OFFSET__(57);
        HomingAcceleration:     Int At __OFFSET__(59);
        ModesOfOperation:       Int At __OFFSET__(60);
        HomingMethod:           Int At __OFFSET__(61);
        HomeOffset:             Int At __OFFSET__(62);
        HomingspeedSwitch:      Int At __OFFSET__(63);
```

```
        HomingSpeedZero:        Int At __OFFSET__(64);
        FreeObjetcs:
                  Array[15] Of Int At __OFFSET__(65);
        ErrorRegister:          Int At __OFFSET__(90);
        ManufacturerStatus:     Int At __OFFSET__(91);
        ErrorFieldIndex:        Int At __OFFSET__(92);
        ErrorField:             Int At __OFFSET__(93);
        FreeObjectIndex:        Int At __OFFSET__(94);
        FreeObjectConfig:       Int At __OFFSET__(95);
        SerialNumber:           Int At __OFFSET__(97);
        SWVersion:              Int At __OFFSET__(99);
    End_Struct;
End_Type;
```

Jetter AG

Graeterstrasse 2

D-71642 Ludwigsburg

Germany

Phone:	+49 7141 2550-0
Phone - Sales:	+49 7141 2550-433
Fax - Sales:	+49 7141 2550-484
Hotline:	+49 7141 2550-444
Internet:	http://www.jetter.de
E-Mail:	sales@jetter.de

Jetter Subsidiaries

Jetter (Switzerland) AG

Münchwilerstrasse 19

CH-9554 Tägerschen

Switzerland

Phone:	+41 71 91879-50
Fax:	+41 71 91879-69
E-Mail:	info@jetterag.ch
Internet:	http://www.jetterag.ch

Jetter USA Inc.

13075 US Highway 19 North

Florida - 33764 Clearwater

U.S.A

Phone:	+1 727 532-8510
Fax:	+1 727 532-8507
E-Mail:	bschulze@jetterus.com
Internet:	http://www.jetter.de