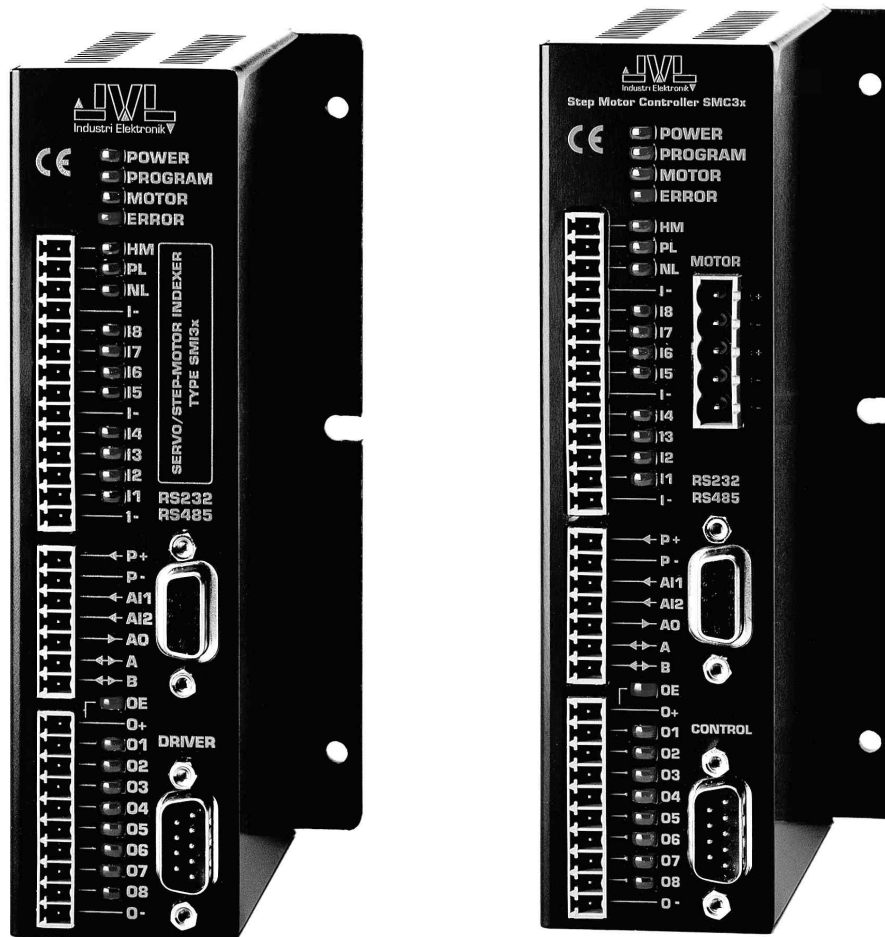


SMI30 / SMI31 Servo/Step Motor Indexer

SMC35A / SMC35B Step Motor Controller

User Manual



TT0119

JVL Industri Elektronik A/S - April 2001

Contents

1	Introduction	5
1.1	Features SMI30/31	6
1.2	Features SMC35	7
1.3	SMI30/31 Indexer Front Panel	8
1.4	SMC35 Controller Front Panel	9
1.5	Quick Start SMI30/31	10
1.6	Quick Start SMC35	11
2	Hardware	13
2.1	Connections	14
2.2	User Inputs	15
2.3	End-of-travel Limit Inputs	16
2.4	Home Input	17
2.5	User Outputs	18
2.6	Connection of Motor (SMC35 only)	19
2.7	Power Supply (SMI30 only)	22
2.8	Power Supply (SMC35 only)	23
2.9	Driver Connection (SMI30 only)	24
2.10	Control Connection (SMC35 only)	25
2.11	Analogue Inputs	26
2.12	Analogue Output	27
2.13	RS232/RS485 Interface	28
3	Software	33
3.1	Use of RS232/RS485 Commands	34
3.2	Standby Mode	35
3.3	Program Execution	36
3.4	Command Description	47
3.5	Control Flags	110
3.6	Error Messages	123
3.7	Alphabetical Overview of Commands	129
4	Appendix	133
4.1	Technical Data SMI30/31 and SMC35	134
4.2	Physical Dimensions	136
4.3	Status and error indication	138
4.4	Common Errors	139
4.5	Connection to Yaskawa servo drives	140
4.6	Connection to JVL step motor driver	144
4.7	Connection to other selected drivers	146
4.8	Accessories	148
4.9	Program examples	149
4.10	Command timing	159
4.11	Connection tables	161
4.12	Calculation of motor movement	163
4.13	Motor Connections (SMC35 only)	164
4.14	Declaration of conformity	166
5	Index	169



Copyright 1996-2002, JVL Industri Elektronik A/S. All rights reserved.
This user manual must not be reproduced in any form without prior written permission of JVL Industri Elektronik A/S.
JVL Industri Elektronik A/S reserves the right to make changes to information contained in this manual without prior notice.
Similarly JVL Industri Elektronik A/S assumes no liability for printing errors or other omissions or discrepancies in this user manual.

MotoWare is a registered trademark of JVL Industri Elektronik A/S

JVL Industri Elektronik A/S
Blokken 42
DK-3460 Birkerød
Denmark
Tlf. +45 45 82 44 40
Fax. +45 45 82 55 50
e-mail: jvl@jvl.dk
Internet: <http://www.jvl.dk>



This User Manual deals with the SMI30-31 Servo/Step Motor Indexer and the SMC35A and SMC35B Step Motor Controller.

The 2 products are fundamentally the same.

The SMI3x Servo/Step Motor Indexer series is a processor unit which requires an external driver to control the motor.

The SMC35 Step Motor Controller series includes a step motor driver which makes it a complete controller unit for step motors.

Allmost everything in this user manual is identical for the SMI3x and the SMC35. Very few additional commands are included in the SMC35 in order to control, for example, the motor current. In hardware sections, some specific pages describe the motor connections.

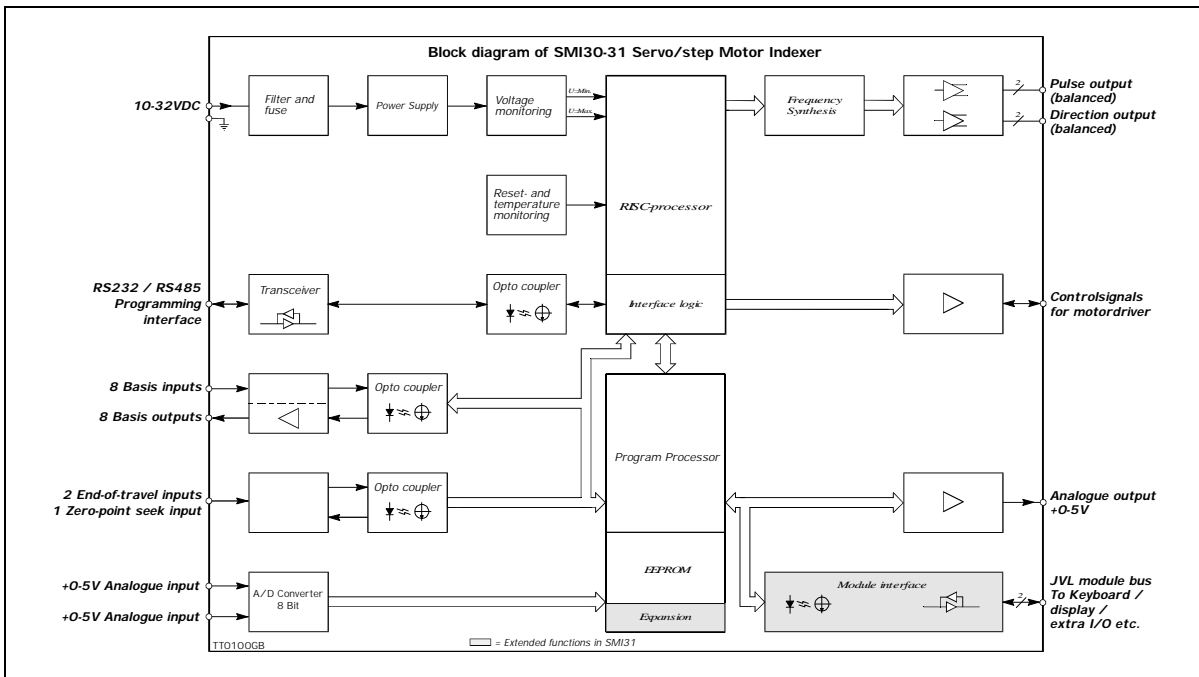
Pages that differ between the SMI3x and the SMC35 are clearly marked.

Note: This manual can only be used for firmware versions higher than V1.59a.

Type	Indexer	Controller	Driver	JVL-bus	Conversion factor	1½ axis
SMI30	Y	-	-	-	-	-
SMI31	Y	-	-	Y	Y	-
SMC35A	Y	Y	3Amp	-	-	-
SMC35B	Y	Y	6Amp	Y	Y	Y
SMC35Q (special version)	Y	Y	1.5Amp	Y	Y	Y

1.1

Features SMI30/31



SMI30 and SMI31 are compact programmable servo or step motor indexers.

The Indexers are characterised by their ability to be controlled either via the RS232/RS485 interface, or via the general inputs in conjunction with a downloaded program.

The Indexer generates a pulse train which is output to the connected servo or step motor driver. This pulse train controls the speed and position of the connected motor. The speed, acceleration, deceleration and distance travelled can be controlled by single commands received via the RS232/RS485 interface or by the program that has been downloaded.

All general purpose inputs and outputs are optically isolated and protected against voltage overloads.

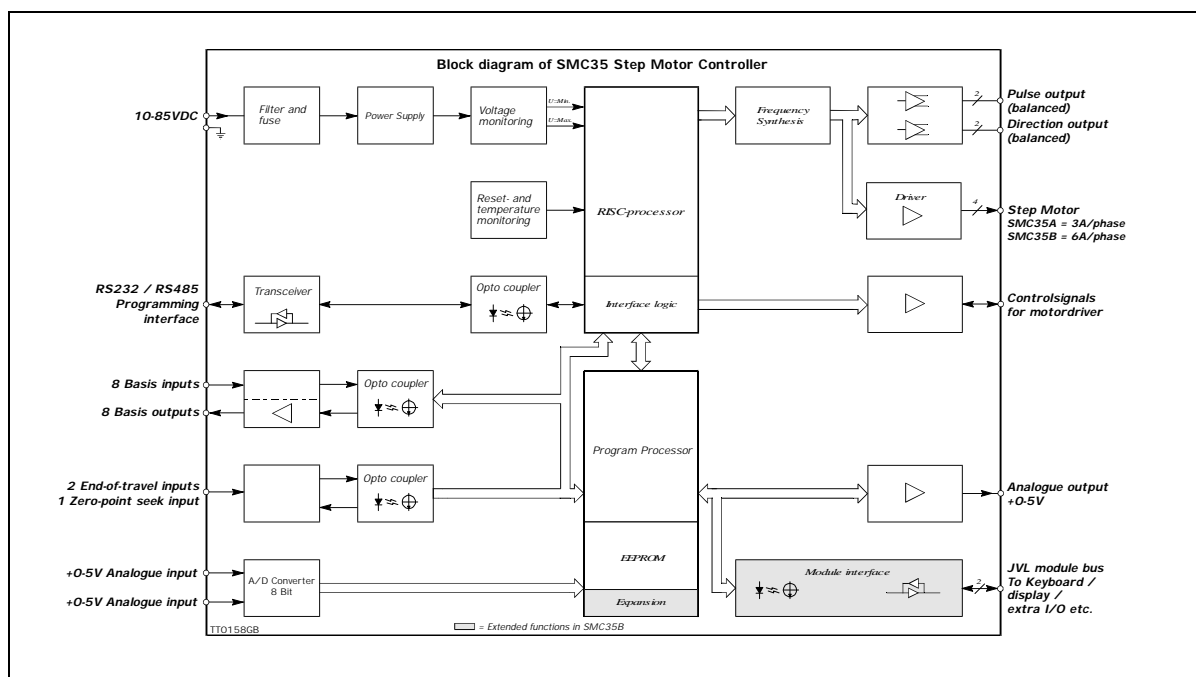
The Indexer is equipped with 8 general-purpose outputs. These can be configured, for example, to give a ready signal when the motor has reached its desired position, or to give an error signal if an obstruction occurs that prevents motor operation. The Indexer can be mounted on a surface.

Main Features:

- Simple programming
- Large speed range: 1 - 2,000,000 pulses/sec.
- Exact speed resolution +/- 0.5 pulse/sec.
- Small physical dimensions
- Connection of up to 255 controllers on the same RS232/RS485 interface bus
- Thermal protection
- Absolute/Relative positioning
- EMC compliant construction - CE approved
- 1 Analogue output +0-5V (1-16Bit)
- 2 Analogue inputs +0-5V (10Bit)
- End-of-travel limit inputs
- 8 general purpose inputs — optically isolated
- 8 general purpose outputs — optically isolated
- Program stored in EEPROM
- Handshake signals to the servo/step driver
- All general purpose I/Os monitored by LEDs
- Plugable screw connectors
- Balanced pulse and direction outputs

1.2

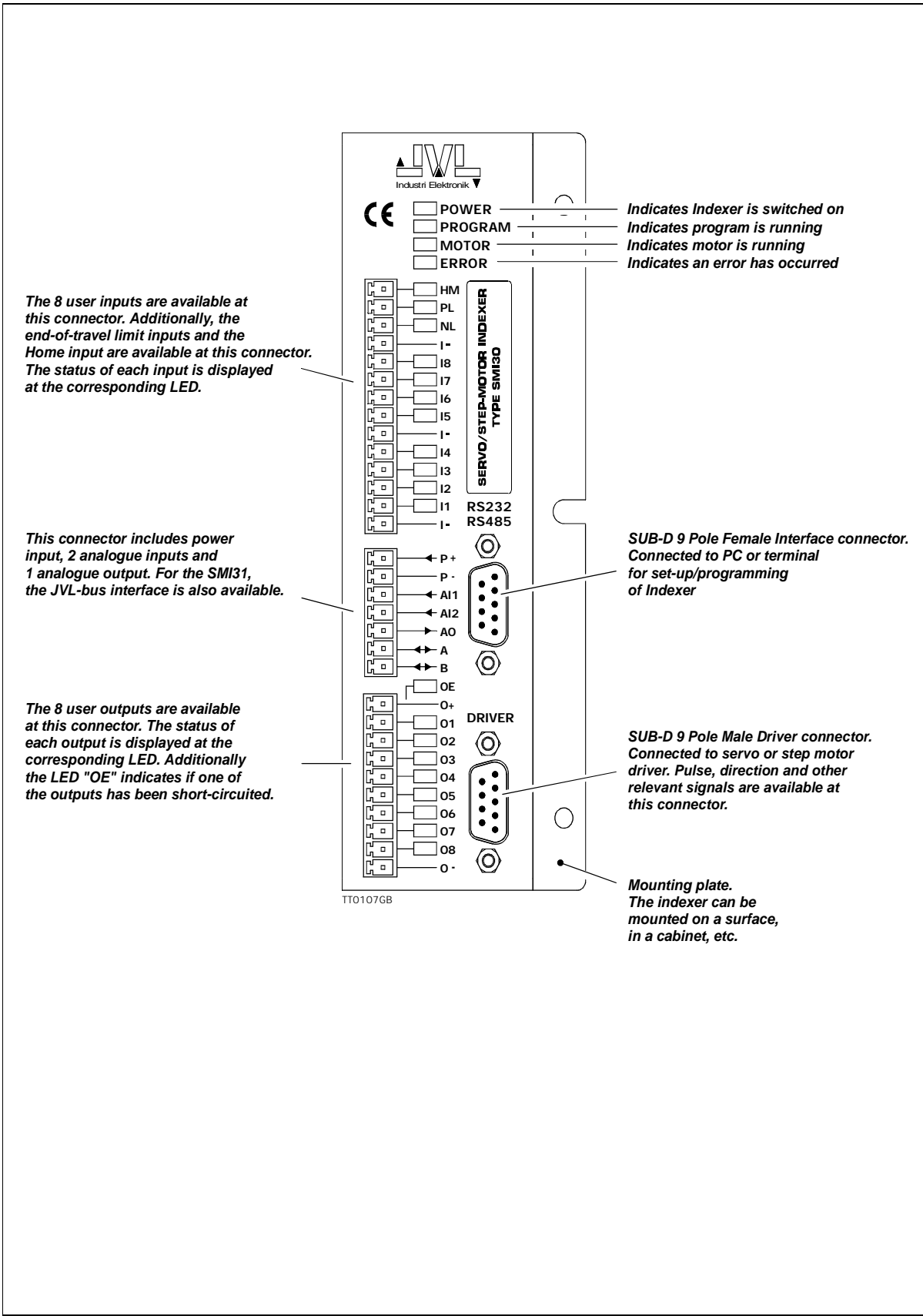
Features SMC35



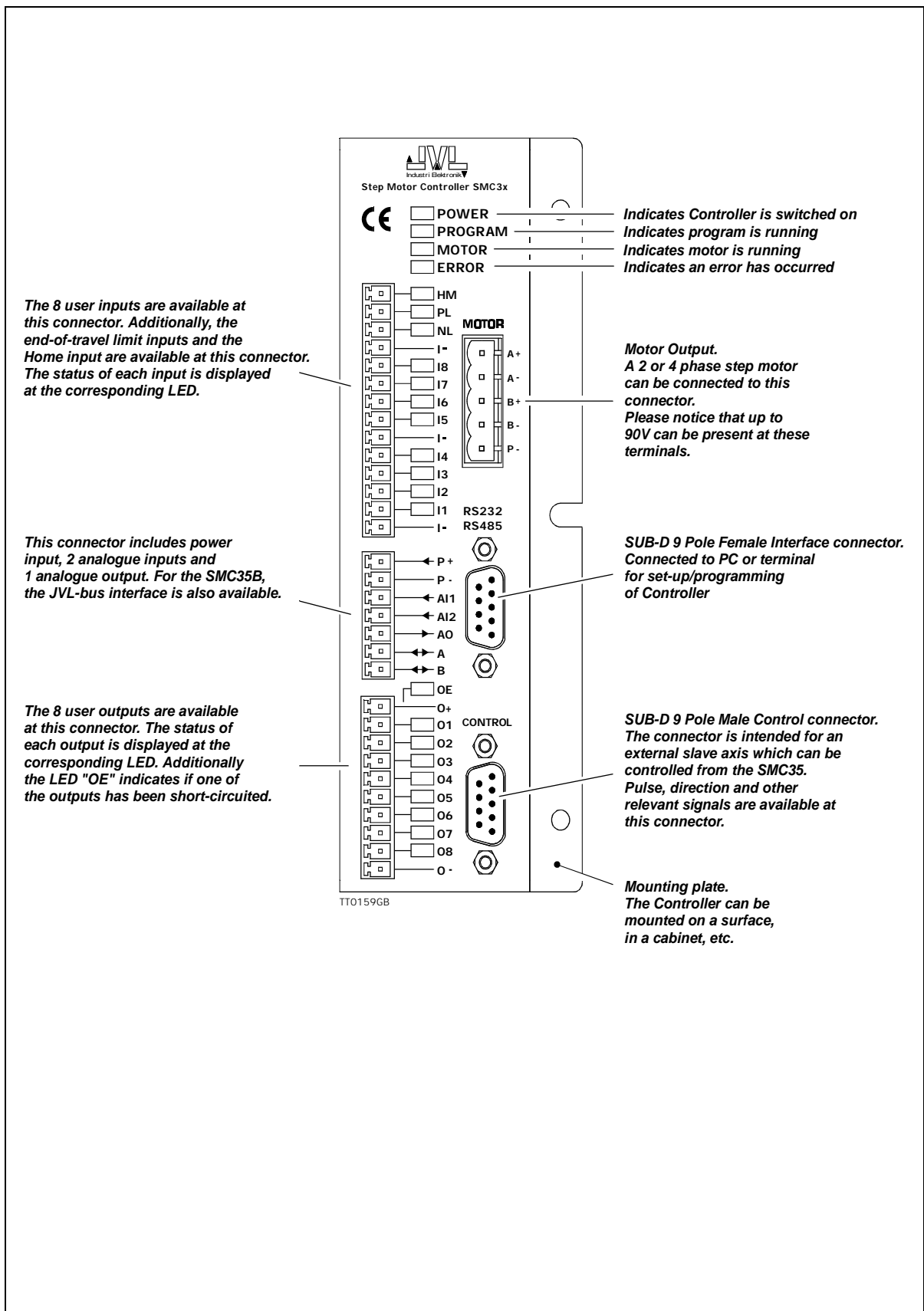
Type overview	Motor Current	JVL Module bus
SMC35A	3 A/RMS per phase	No
SMC35B	6 A/RMS per phase	Yes

- Ultra-compact Step Motor Controller up to 6A/80VDC
- Indexer and driver in one unit
- 1½-axis controller for control of 2 motors from the same program
- Special modes for solving tasks involving dispensing/labelling
- Extremely fast start/stop reaction times
- Programmed via the well-known Windows-based MotoWare program
- Can operate with all 2 or 4 phase step motors
- Selection of ministep resolution via software
- Advanced "all digital" with built-in μ -PLC
- Encoder inputs for monitoring of position and "stall" of motor
- Stores up to 15 errors
- CE approved. Low EMI
- 2 analogue inputs and 1 analogue output 0-5VDC
- User outputs can deliver up to 0.7Amp per channel, i.e. external relays can be avoided
- Positioning range from -2.1billion to +2.1billion
- Multipoint control so that 1 master SMC35 can send data to up to 31 slaves, e.g. SMI30, SMC35, DMC10 and AMC10/12
- Multitasking system with possibility for changing vel., acc., outputs etc. with motor running
- 2 models: SMC35A 3A/20-80VDC or SMC35B 6A/20-80VDC with JVL bus
- Program stored in EEPROM
- Large velocity range: 0 to 2,000,000 pulses/sec.
- Connection of up to 32 indexers on the same RS232/485 interface bus
- Absolute/Relative positioning
- 11 inputs, 8 outputs, end-of-travel inputs, high speed counter/encoder inputs
- All generally used I/O monitored by light emitting diodes
- Small physical dimensions
- Plugable screw connector
- Can be mounted on a surface
- Electronic gear can be coupled in/out

1.3 SMI30/31 Indexer Front Panel

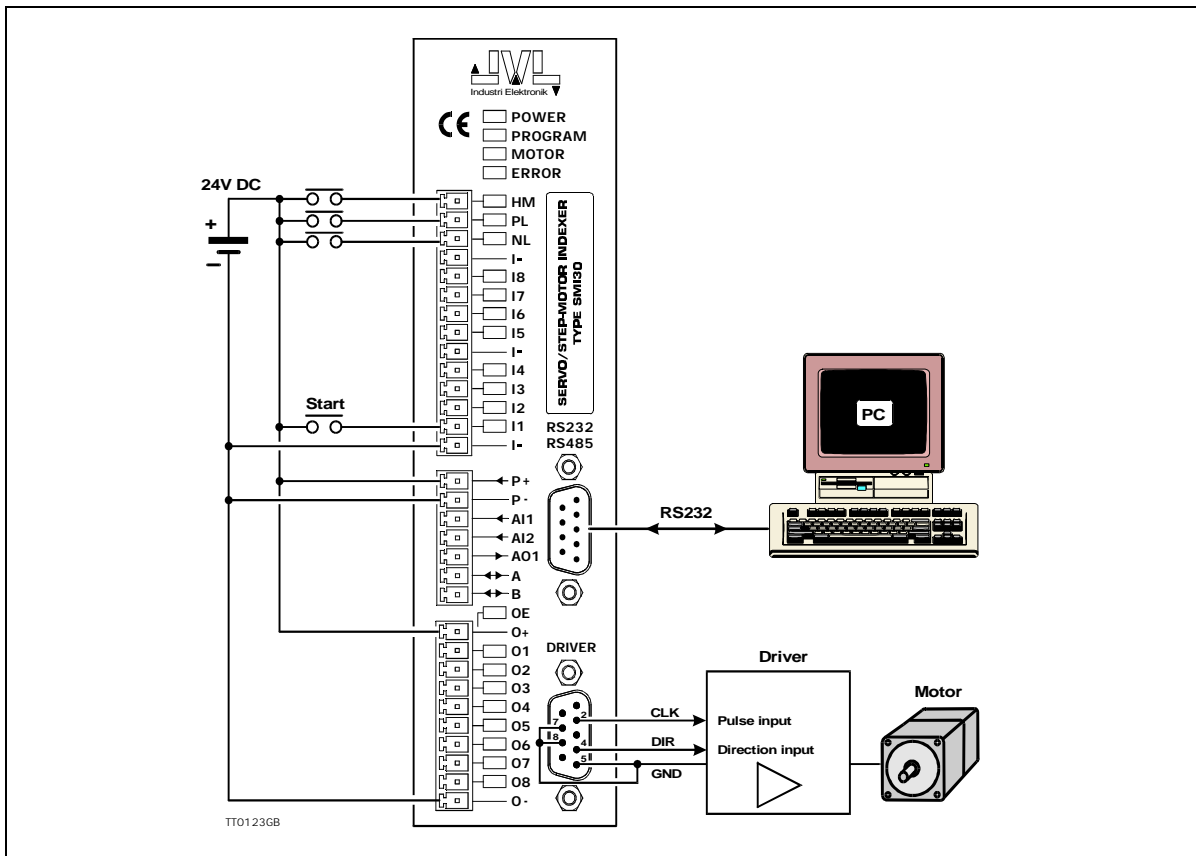


1.4 SMC35 Controller Front Panel



1.5

Quick Start SMI30/31

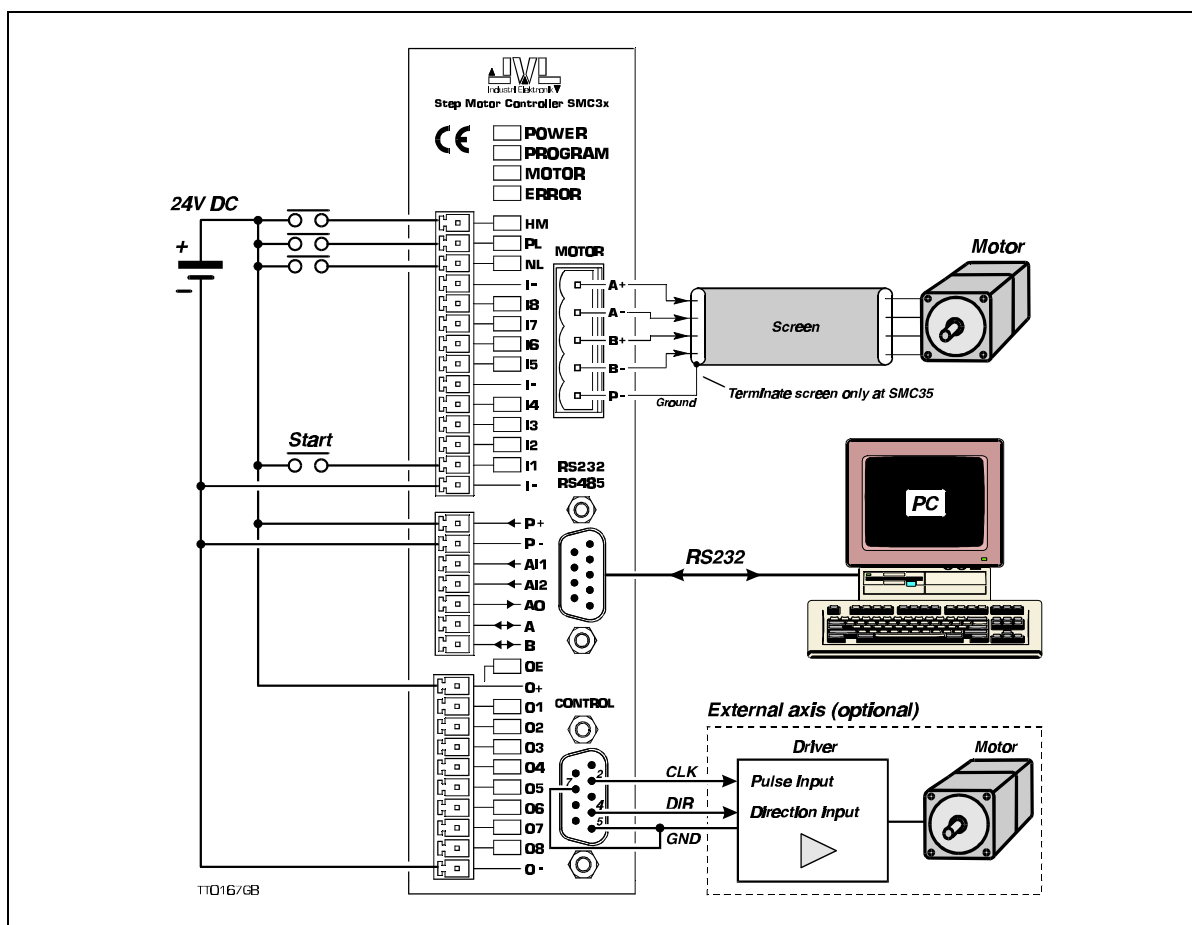


1.5.1 How to get started with the SMI30 and MotoWare

1. Switch on the SMI30. The Power lamp should be lit.
2. Start MotoWare and set it to SMI30 mode by selecting SMI30 in the **Set-up** menu, **Controller specs**.
3. Select **OnLine Editor** in the **Applications** Menu or from the Toolbox. Key-in SON=1. (servo on). This will enable the servo driver.
4. Key in "OUT1=1". The green LED **O1** on the front panel of the Indexer will be lit if +24 volts are connected to O+ and 0 volts connected to O-.
5. In the OnLine Editor, key-in "?". This will display status information, giving all the current register values.
6. In the OnLine editor, key in "SR=1000". This will cause the motor to move 1000 pulses forward. If this does not happen, the Indexer will probably display errors E46 or E39. To solve the problem, check the cable connection to the motor controller and the servo-parameters.
7. When the motor runs, return to the main menu. Select **Open** from the **File** menu. From the MotoWare directory select the test program: **Test_SMI.mcp** and press **OK**. Send the program by activating the **Send** button. This test program activates the outputs in succession and moves the motor 8000 pulses forwards and backwards. When this has taken place 5 times, it will stop. (See *Test-program, quick start*, page 153.)

1.6

Quick Start SMC35

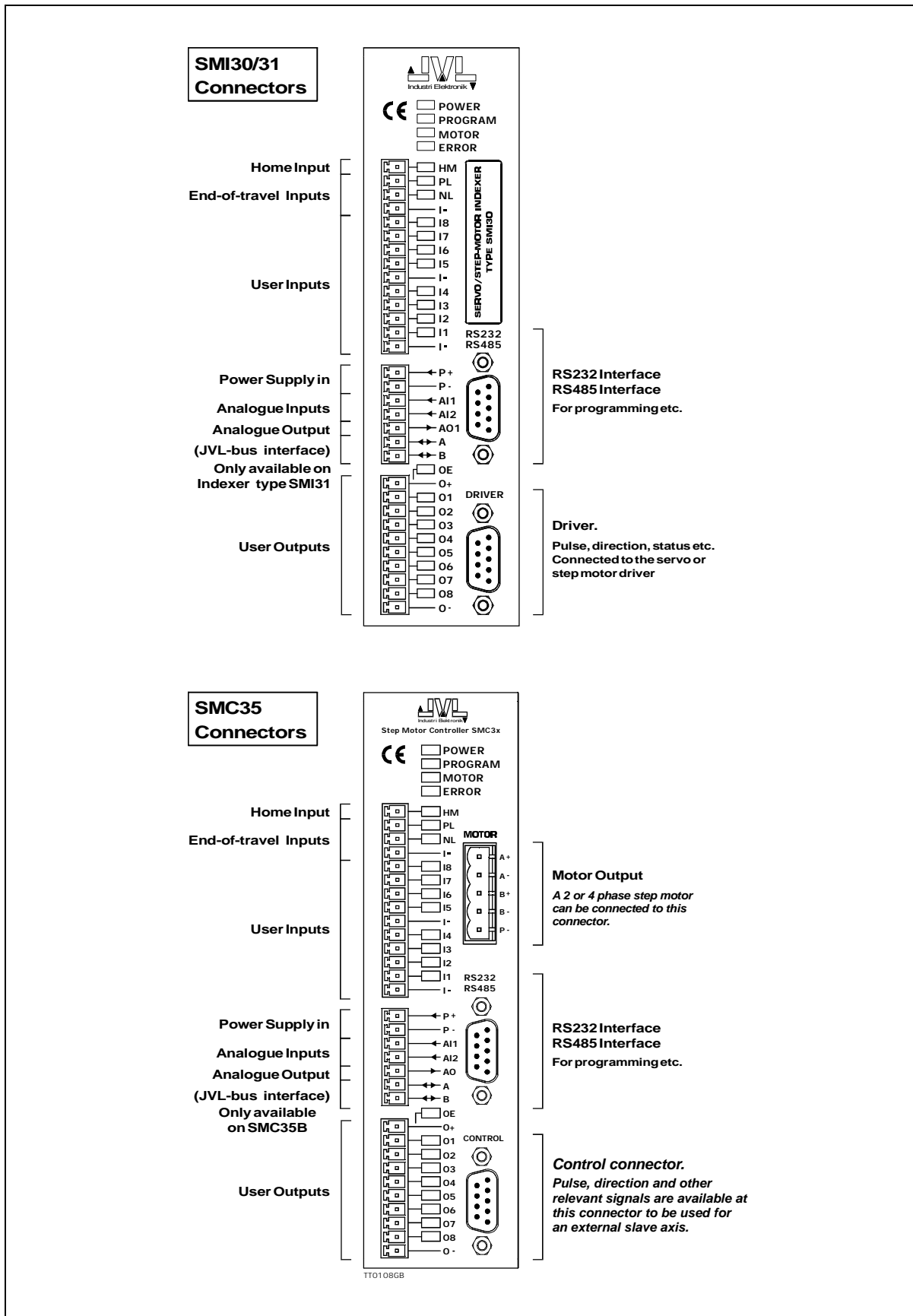


1.6.1 How to get started with the SMC35 and MotoWare

1. Switch on the SMC35. The Power lamp should be lit.
2. Start MotoWare and set it to SMC35 mode by selecting SMC35 in the **Set-Up** menu, **Controller specs**.
3. Select **OnLine Editor** in the **Applications** Menu or from the Toolbox. Key-in SON=1. (servo on). This will enable the motor output.
4. Key in "OUT1=1". The green LED **O1** on the front panel of the Indexer will be lit if +24 volts are connected to O+ and 0 volts connected to O-.
5. In the OnLine Editor, key-in "?". This will display status information, giving all the current register values.
6. In the OnLine editor, key in "SR=1000". This will cause the motor to move 1000 pulses forward. If this does not happen, check the cable connection to the motor or adjust the current to the motor with the CS and CT commands.
7. When the motor runs, return to the main menu. Select **Open** from the **File** menu. From the MotoWare directory select the test program: **Test_SMI.mcp** and press **OK**. Send the program by activating the **Send** button. This test program activates the outputs in succession and moves the motor 8000 pulses forwards and backwards. When this has taken place 5 times, it will stop. (See *Test-program, quick start*, page 153.)

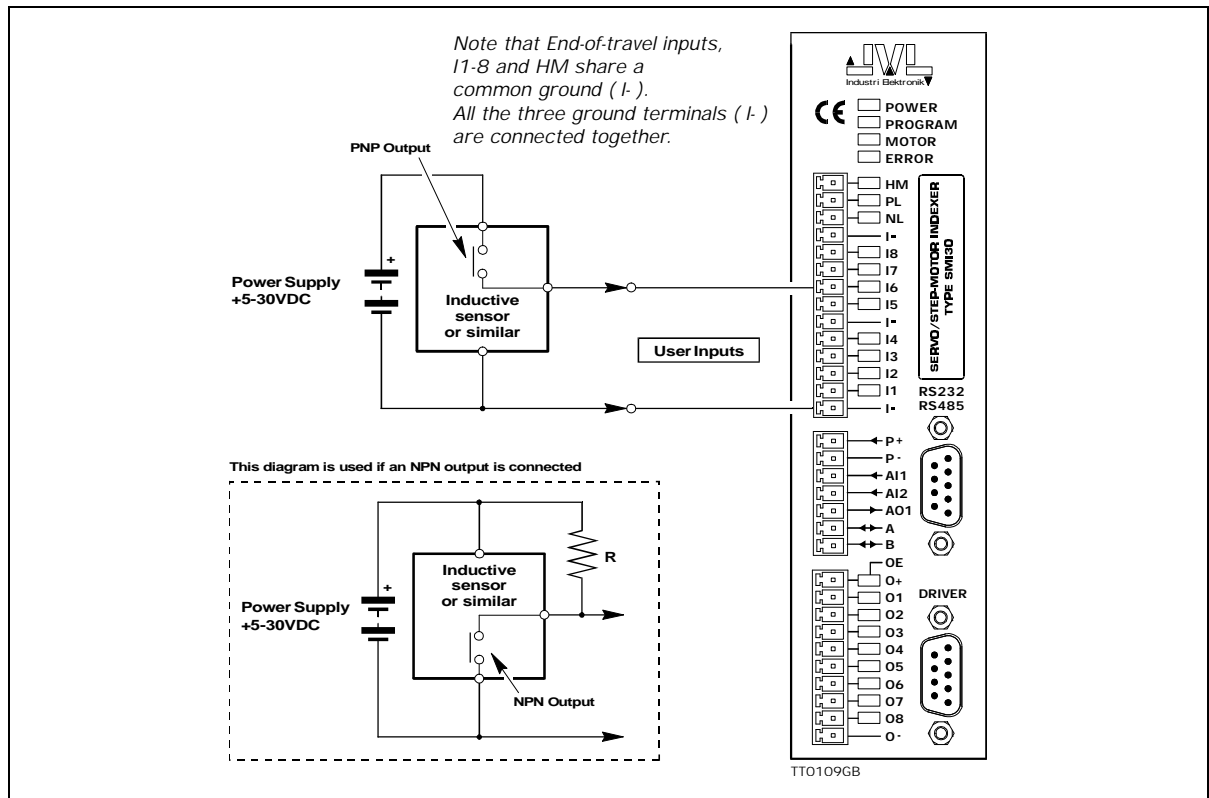
2.1

Connections



2.2

User Inputs



2.2.1 General

The Indexer is equipped with a total of 8 digital inputs. Each input can be used for a variety of purposes depending on the actual application. Each of the inputs can be detected from the actual program that has been downloaded to the Indexer.

The Inputs are optically isolated from other Indexer circuitry. All of the Inputs have a common ground terminal, denoted *I-*. Note that this terminal is also used for the End-of-Travel Limit Inputs (*Section 2.3*, page 16) and Home Input (*Section 2.4*, page 17). Each Input can operate with voltages in the range 5 to 30VDC. Note that the Inputs should normally be connected to a PNP output since a positive current must be applied for an input to be activated.

2.2.2 Connection of NPN Output

If an Input is connected to an NPN output, a Pull-Up resistor must be connected between the Input and the + supply. See the illustration above.

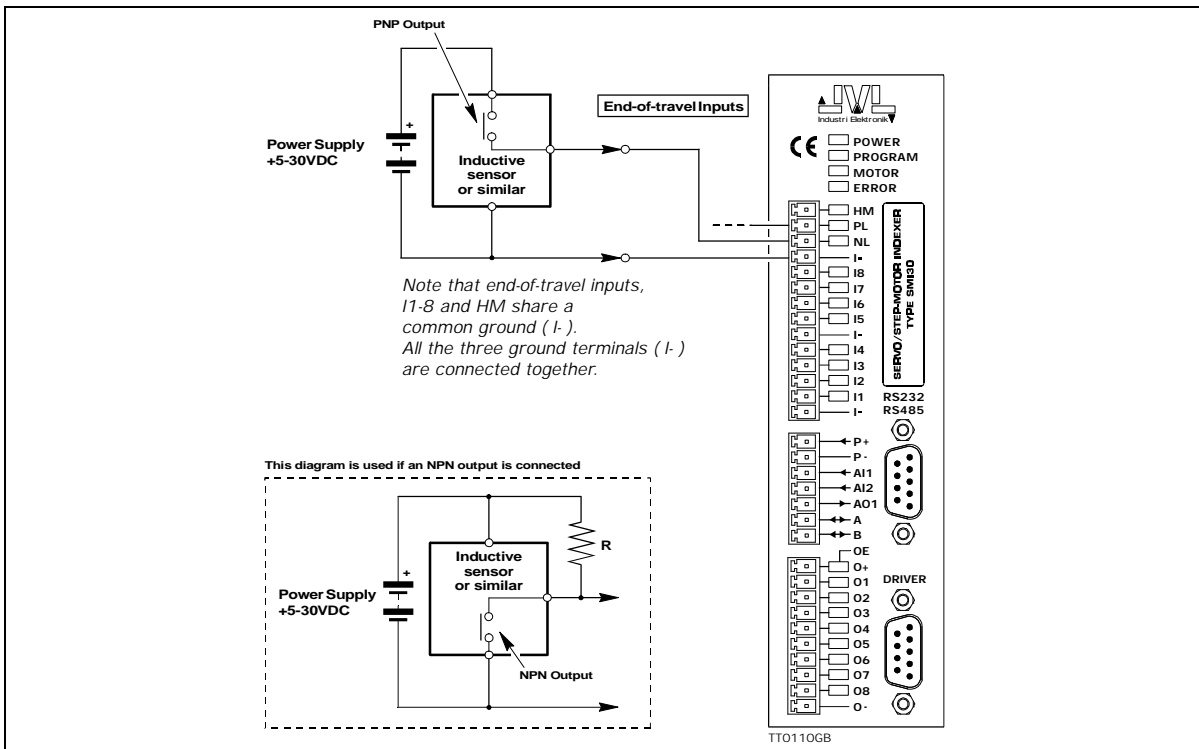
The value of the resistance used depends on the supply voltage. The following resistances are recommended:

Supply Voltage	Recommended Resistance
5-12VDC	1kOhm / 0.25W
12-18VDC	2.2kOhm / 0.25W
18-24VDC	3.3kOhm / 0.25W
24-30VDC	4.7kOhm / 0.25W

2.2.3 Indication of Input Status

To indicate the status of each Input, the Indexer's front panel is equipped with LEDs denoted I1, I2,..... I8. These LEDs are lit when the respective Input is activated.

2.3 End-of-travel Limit Inputs



2.3.1 General

The Indexer is equipped with end-of-travel limit inputs denoted *NL* (negative limit) and *PL* (positive limit). The Inputs are, together with *I1-I8* and *HM* (Home input) optically isolated from other Indexer circuitry. All of these inputs have a common ground denoted *I-*. The End-of-travel Limit Inputs operate with voltages in the range 5 to 30VDC. Note that the Inputs must normally receive a signal from a PNP output since a positive current must be applied for the Inputs to be activated.

Activation of the *PL* Input will halt motor operation if the motor is moving in a positive direction. The motor can however operate in a negative direction even if the *PL* Input is activated. Activation of the *NL* Input will halt motor operation if the motor is moving in a negative direction. The motor can however operate in a positive direction even if the *NL* Input is activated.

An error message will be set in the Indexer's error register if either the *NL* or *PL* Inputs has been activated. See *Error Messages*, page 123.

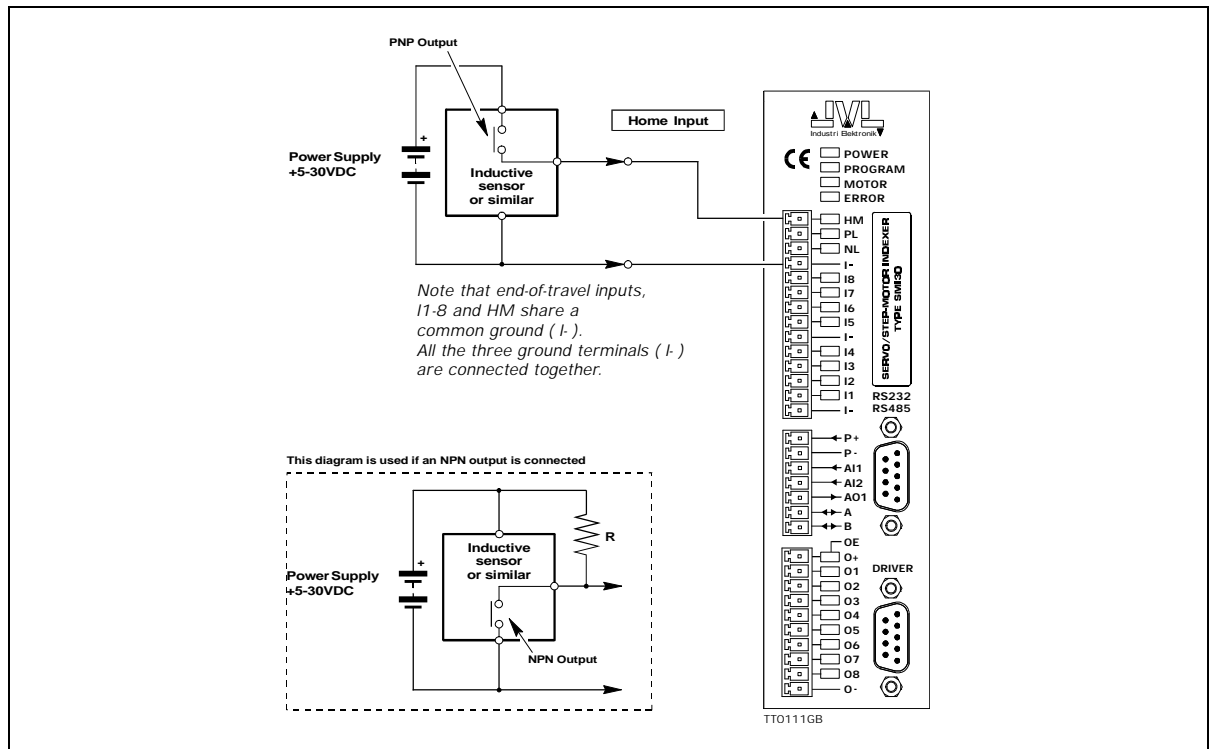
The *NL* and *PL* inputs can also be used as general inputs (i.e. the same as *IN1-8*). See also the *Positive Limit Switch (PLS)* command, page 88 and the *Negative Limit Switch (NLS)* command, page 83.

2.3.2 Connection of NPN Output

To connect an end-of-travel input to an NPN output, a Pull-Up resistor must be connected between the Input and the + supply. See above illustration. The size of the resistance depends on the supply voltage used. The following resistances are recommended:

Supply Voltage	Recommended Resistance
5-12VDC	1kOhm / 0.25W
12-18VDC	2.2kOhm / 0.25W
18-24VDC	3.3kOhm / 0.25W
24-30VDC	4.7kOhm / 0.25W

2.4 Home Input



2.4.1 General

The Input *HM* (Home) is used during the zero-point seek function. A zero-point seek occurs after one of the following conditions:

1. The Indexer receives the seek zero command *SZ* (reset). See the *Seek Zero Point (SZ)* command, page 106
2. The Indexer is switched on and is set to automatically execute a program containing a *SZ* command (only if *MRI=1*).

The Home Input is primarily used if the Indexer is used for positioning purposes.

The Input is optically isolated from other Indexer circuitry, with the exceptions of *I1 - I8*, and *NL* and *PL* (End-of-travel Limit Inputs). All these inputs have a common ground denoted *IN-*. The Home Input can operate with voltages in the range 5 to 30VDC. Note that the Input is designed to receive a signal from a PNP output since a positive current must be applied for the Input to be activated.

The *HM* input can also be used as a normal input which means it can be included, for example, in an *IF* statement.

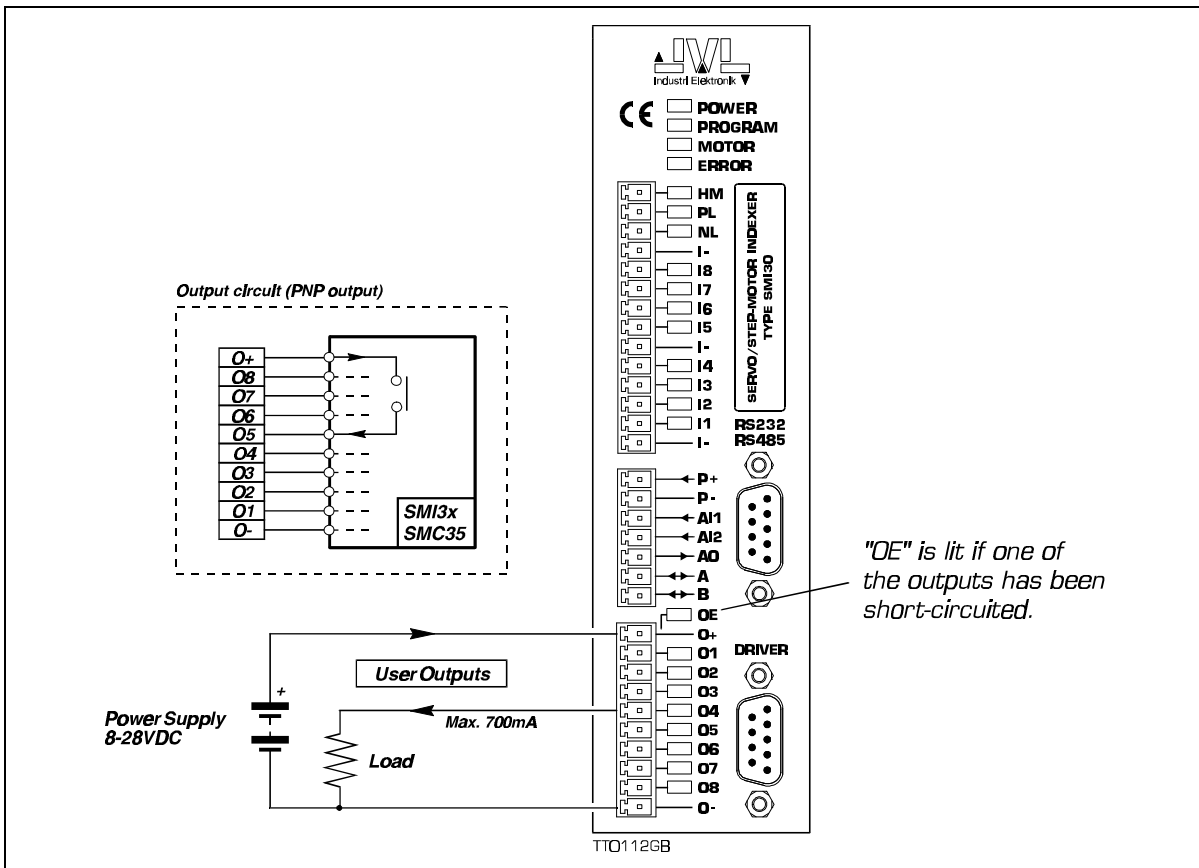
2.4.2 Connection of NPN Output

To connect the Input to an NPN output, a Pull-Up resistor must be connected between the Input and the + supply. See above illustration. The size of the resistance depends on the supply voltage used. The following resistances are recommended:

Supply Voltage	Recommended Resistance
5-12VDC	1kOhm / 0.25W
12-18VDC	2.2kOhm / 0.25W
18-24VDC	3.3kOhm / 0.25W
24-30VDC	4.7kOhm / 0.25W

2.5

User Outputs



2.5.1 General

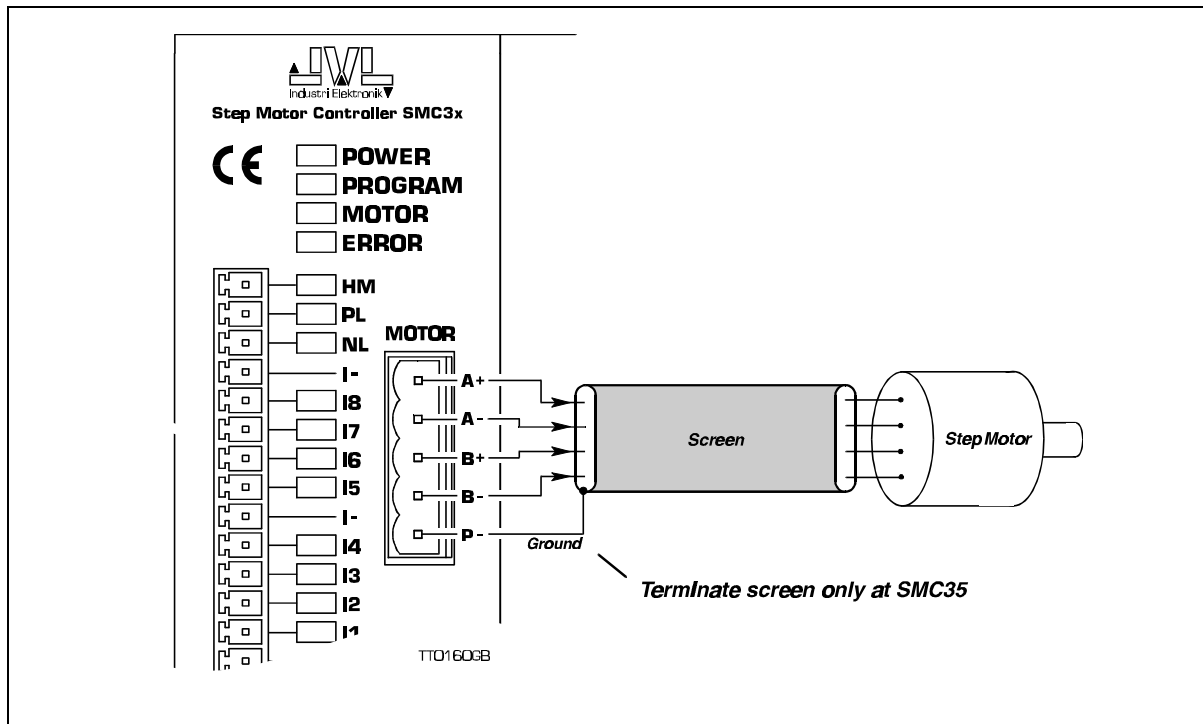
The Indexer is equipped with a total of 8 digital outputs. Each output can be used for a variety of purposes depending on the Indexer's basic mode of operation. The Outputs are optically isolated from other Indexer circuitry. The output circuitry must be powered from an external power supply. This power supply is connected to the terminals O+ and O-. The output circuitry operates with voltages in the range 8-28VDC. Each output can supply a continuous current of 700mA. The Outputs are all source drivers, i.e. if a given Output is activated, contact is made between the +supply (O+) and the respective output terminal. See above illustration. To indicate the level of each output, the Indexer front panel is equipped with LEDs, denoted IO1, IO2,..... IO8. These LEDs are lit when the respective Output is activated.

2.5.2 Overload of User Outputs

All of the Outputs are short-circuit protected, which means that the program and the motor is stopped and the output is automatically disconnected in the event of a short circuit. The Output will first function normally again when the short-circuit has been removed. The *Out Error (OE)* LED on the Indexer's front panel is lit when one or more of the Outputs has been short-circuited. The LED also indicates if the output circuitry has been overheated due to an overload. It is possible via software control to detect an overload using the command *EST* or *ES2*. The error message *E43: 01-08 Output error*, page 127 will appear.

Note: Do not connect a voltage greater than 30VDC to the O+ terminal as the output circuitry may be seriously damaged and the unit will require factory repair.

2.6 Connection of Motor (SMC35 only)



2.6.1 Cabling

For SMC35A that supply a phase current in the range 0 to 3 A, it is recommended that 0.5mm² cable (minimum) is used to connect the motor to the controller.

For SMC35B that supply a phase current in the range 0 to 6 A, it is recommended that 0.75mm² cable (minimum) is used to connect the motor to the controller.

Cable lengths used to connect the motor to the Driver should not exceed 10 metres because of impedance loss. It is possible to use longer cables but motor performance will decrease.

Cables should be securely connected since a poor connection can cause heating and destruction of the connector. Similarly, tinned conductors should be avoided.

Important!

To minimise spurious noise emission from the motor cables and fulfil CE requirements, screened cable must be used.

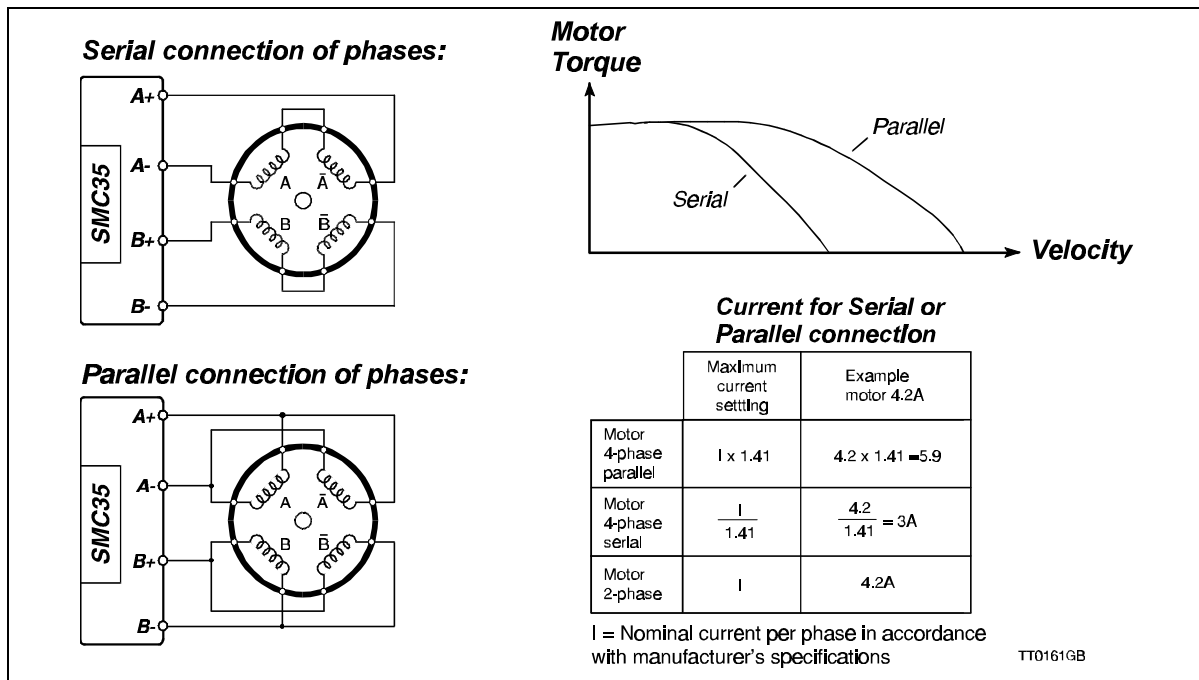
If screened cable is not used, other electronic equipment in the vicinity may be adversely affected.

The removable connector must never be removed while a voltage is connected as this will significantly reduce the lifetime of the connector. Note also that the connector's lifetime is reduced by repeated connecting/disconnecting since the contact resistance of the pins is increased.

Note that P- is connected to the chassis and functions as the main ground on the SMC35.

See also *Motor Connections (SMC35 only)*, page 164 which describes how various models of motor should be connected to the SMC35.

2.6 Connection of Motor (SMC35 only)



2.6.2 Connection of Step Motor

Various types of step motor are available:

1. 2-phase Bipolar (4 connectors)
2. 4-phase Bipolar/Unipolar (8 connectors)
3. 4-phase Unipolar (6 connectors).

Note that Type 3 motors indicated above (Unipolar motors) produce 40% less torque. This motor type can be used with success but is not recommended if a 4 or 8 wire motor is available instead. This section will not describe the unipolar type further.

2-phase or 4-phase motors can be connected to the Controllers as follows:

2-phase Motors (4 wires).

This type of motor can be directly connected to the Controller's motor terminals. The Controller current adjustment must not exceed the manufacturer's specified rated current for the motor.

4-phase Motors (8 wires).

This type of motor can be connected to the Driver in one of the 2 following ways:

1. Serial connection of phases.
2. Parallel connection of phases.

Selection of serial or parallel connection of the motor phases is typically determined by the speed requirements of the actual system.

If slow speeds are required (typically less than 1 kHz), the motor phases can be connected in serial. For operation at higher speeds (greater than 1 kHz), the motor phases can be connected in parallel.

2.6 Connection of Motor (SMC35 only)

2.6.3 Serial Connection

Using serial connection of the phases, a motor provides the same performance (up to 1kHz) as parallel connection, but using only approximately half the current. This can influence the selection of Controller model and enables a Controller rated for a lower motor current to be used. See illustration on previous page.

If the phases of a 4-phase step motor are connected in series, the motor's rated phase current should be divided by 1.41. For example, if the rated current is 4.2A, the maximum setting of the Controller phase current must not exceed 3 A when the motor phases are connected in series.

2.6.4 Parallel Connection

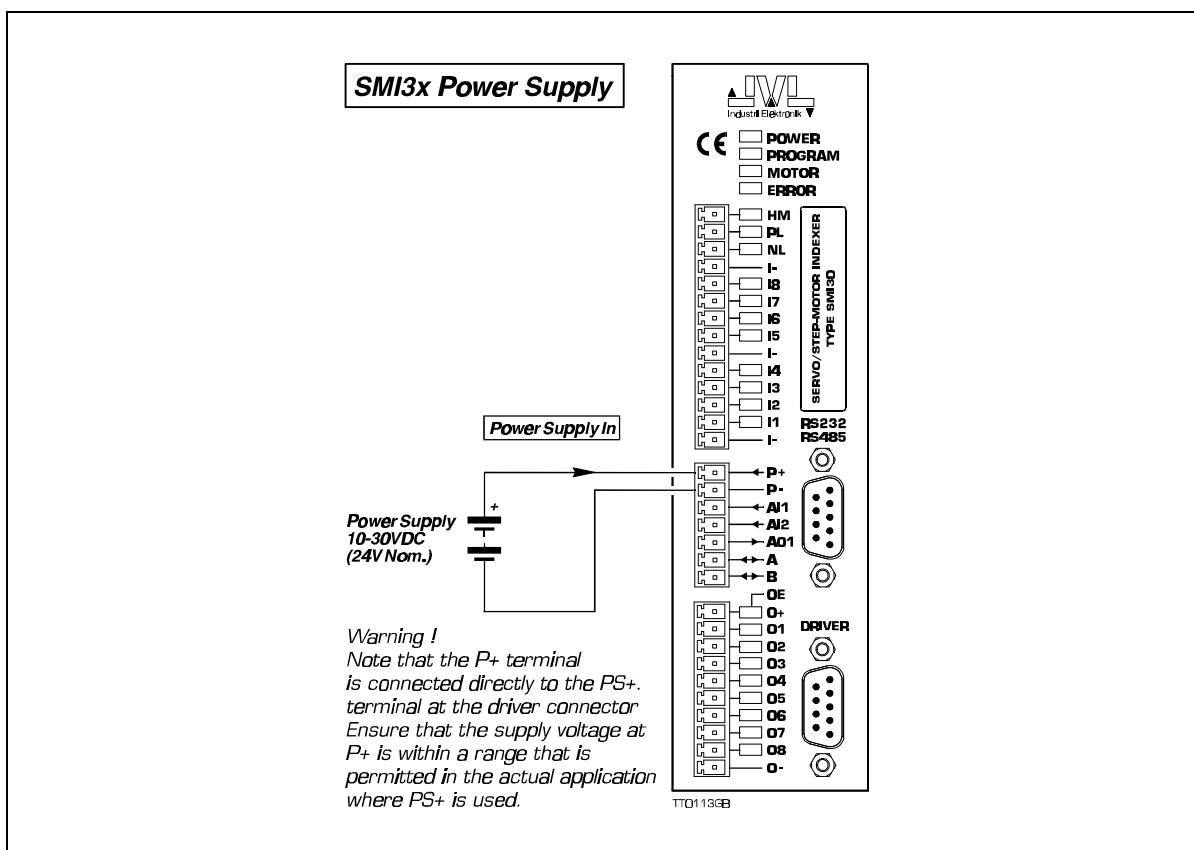
With parallel connection of motor phases, a motor will provide better performance at frequencies greater than 1kHz compared to serially connected phases, but requires approximately twice the current. This can influence the choice of Controller since it is necessary to select a Controller that can supply twice the current used for serial phase connection. See illustration on previous page.

When the phases of a 4-phase motor are connected in parallel, the specified rated current of the motor must be multiplied by a factor of 1.41. For example, if the rated current is 4.2 A, the maximum setting of the Controller phase current must not exceed 5.9 A when the phases are connected in parallel.

It should be noted that the lower the self-induction of the motor, the better since this influences the torque at high speeds. The torque is proportional to the current supplied to the motor.

The applied voltage is regulated by the Controller so that the phase current is adjusted to the selected value. In practice this means that if a motor with a large self-inductance (e.g. 100mH) is used, the Controller cannot supply the required phase current at high speeds (high rotational frequencies) since the output voltage is limited.

2.7 Power Supply (SMI30 only)



2.7.1 General Aspects of Power Supply (SMI30 only)

Powering of the Indexer is relatively simple. Indexer types SMI30/31 require a supply voltage in the range 10-30VDC.

2.7.2 Power Supply of Indexer (SMI30 only)

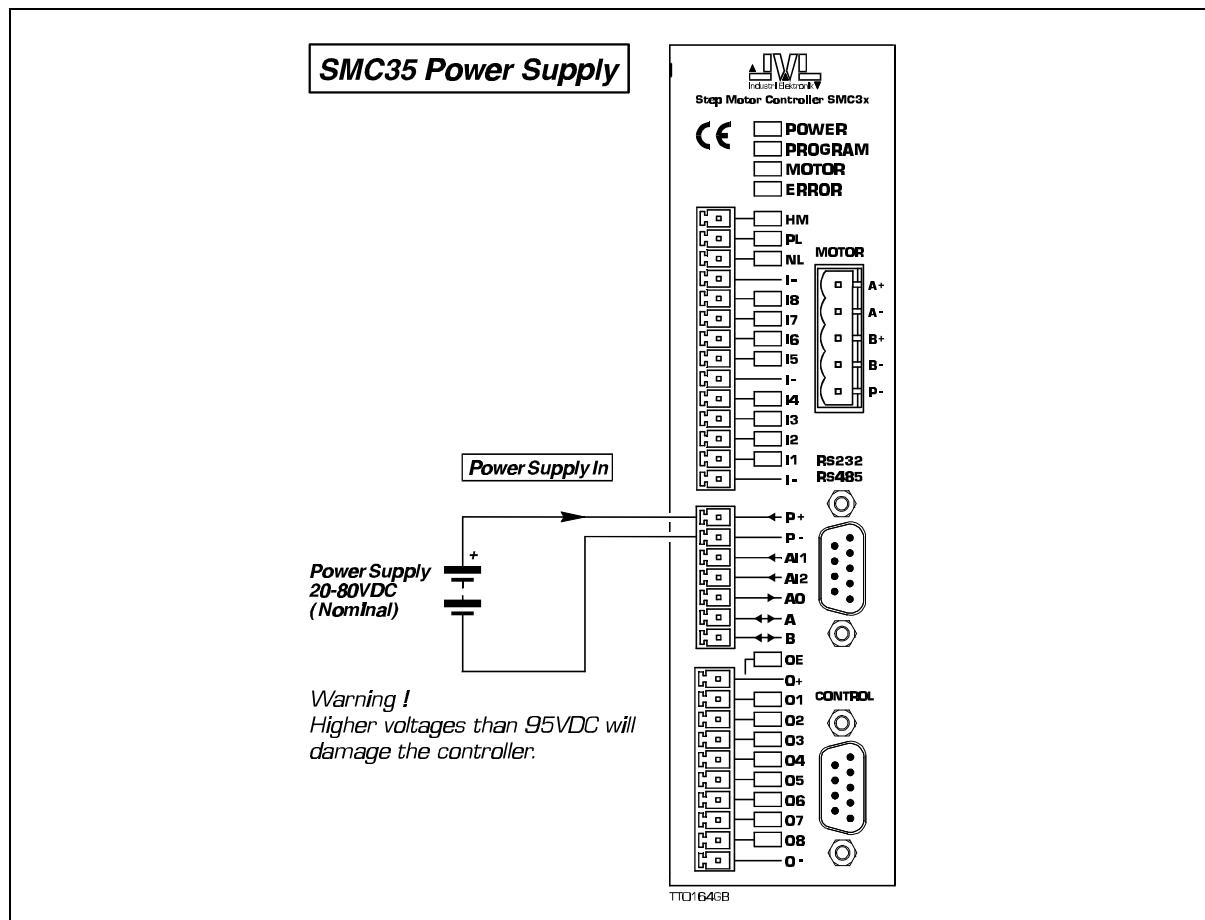
To ensure that powering of the Indexer is as simple as possible, only a single supply voltage is connected to the Indexer. Internal supply circuitry ensures the correct supply voltages for the driver, control circuits, etc.

For optimum performance, it is recommended that a capacitance of minimum 1000 μ F is connected to the supply. Similarly, it is recommended that 0.75mm cable is used to connect the power supply to the Indexer. If the Indexer supply voltage falls below 8V, the internal reset circuitry will reset the driver. Provision should therefore be made to ensure that the supply voltage is always maintained at a minimum of 10-15V, even in the event of a mains voltage drop.

2.7.3 Power Supply Faults (SMI30 only)

The Indexer is protected against incorrect polarity connection and voltage overload. If a voltage overload of the Indexer supply occurs, or the supply is connected with incorrect polarity, the Indexer's internal fuse will be blown. The fuse can only be replaced by an authorised service centre.

2.8 Power Supply (SMC35 only)



2.8.1 General Aspects of Power Supply (SMC35 only)

Powering of the Controller is relatively simple. The Controller types SMC35A and SMC35B require a supply voltage in the range 20-80VDC nominal. It is strongly recommended to use a voltage as high as possible since it will give the best torque performance of the motor at high speeds.

2.8.2 Power Supply (SMC35 only)

To ensure that powering of the Controller is as simple as possible, only a single supply voltage is connected to the Controller. Internal supply circuitry ensures the correct supply voltages for the driver, control circuits, etc.

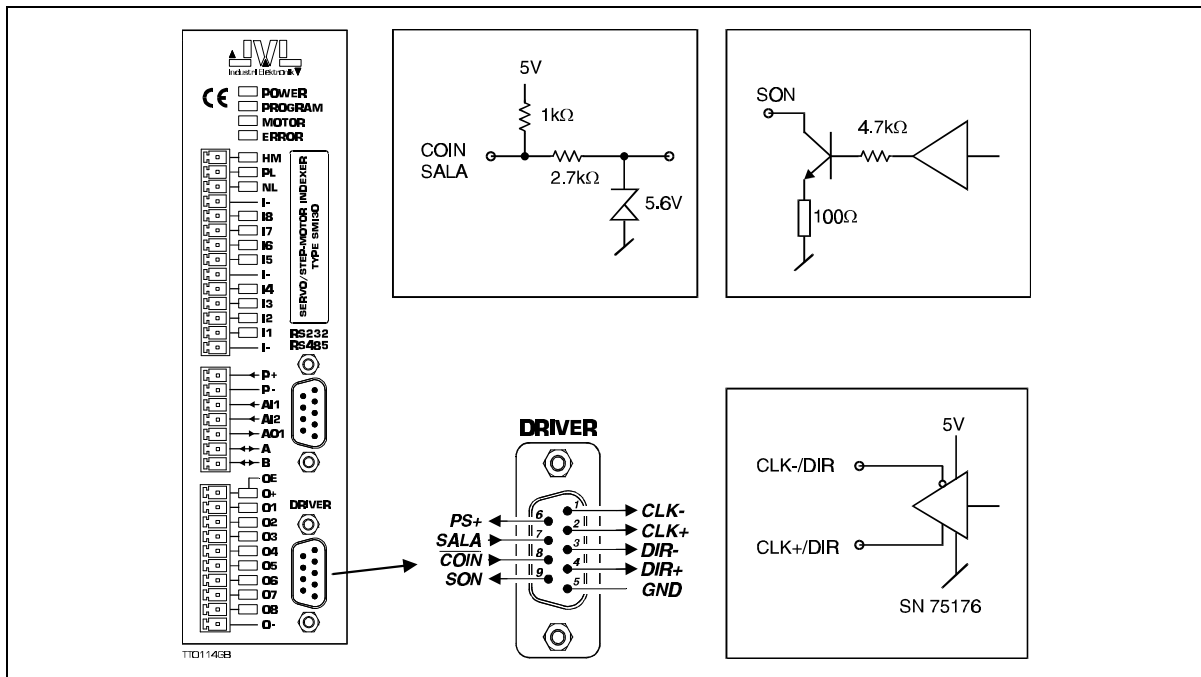
For optimum performance, it is recommended that a capacitance of minimum 1000mF is connected to the supply. Similarly, it is recommended that 0.75mm cable is used to connect the power supply to the Controller. If the Controller supply voltage falls below 15V, the internal reset circuitry will reset the driver. Provision should therefore be made to ensure that the supply voltage is always maintained at a minimum of 20V, even in the event of a mains voltage drop.

2.8.3 Power Supply Faults (SMC35 only)

The Controller is protected against incorrect polarity connection and voltage overload. If a voltage overload of the Controller supply occurs, or the supply is connected with incorrect polarity, the Controller's internal fuse will be blown.

The fuse can only be replaced by an authorised service centre.

2.9 Driver Connection (SMI30 only)



2.9.1 General

All the necessary input and outputs for the connected servo or step motor driver are available at this connector. The following signals can be used:

CLK - / CLK +

This is the main pulse signal output for the connected driver. The 5V output is balanced which means that CLK - is the inverted signal of CLK +. If the driver connected to the Indexer has an unbalanced input, CLK- must be left unconnected.

DIR - / DIR +

This is the direction output for the connected driver. The 5V output is balanced which means that DIR - is the inverted signal of DIR +. If the driver connected to the Indexer has an unbalanced input, DIR- must be left unconnected.

GND

The ground terminal must be connected to the driver to ensure proper operation. The GND is also reference for the PS+, SALA, COIN and SON terminals.

PS+

This terminal is a voltage output for the input or output circuitry in the connected driver. Note that the voltage at this terminal is the same as at the Supply terminal P+. Inside the indexer it is protected by the main fuse so that any short circuit results in limited damage.

SALA

Servo Alarm input. This terminal ensures that all activity in the Indexer is stopped if the terminal is left open. The terminal must under normal operation be kept logic low which means at the same level as GND. The input is active high. See *CB15 Servo alarm signal (SALA) flag*, page 113, to change active level.

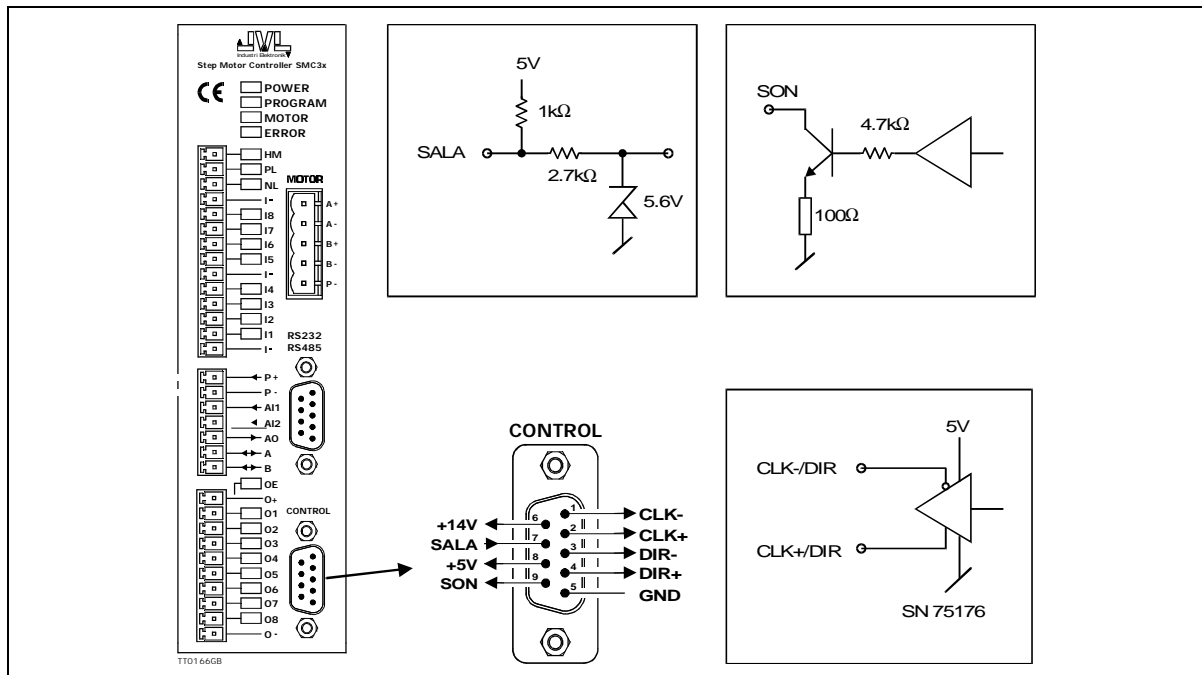
COIN

Servo-in-position input. This terminal ensures that the Indexer waits for the servo motor until it has reached its final position. If the terminal is left open, the Indexer will keep waiting. The terminal must under normal operation be kept logic low which means at the same level as GND. See *CB16 Motor in position (COIN) flag*, page 113, flag to change active level.

SON

This output is low (active NPN) if the command *SON=1* is executed.

2.10 Control Connection (SMC35 only)



2.10.1 General

The Control connector is intended to be used if an external step or servo driver has to be controlled. The control signals are shared with the internal driver and therefore it is not possible to operate both motors at the same time unless it with the same speed and target position. The following signals can be used:

CLK - / CLK +

This is the main pulse signal output for the connected driver. The 5V output is balanced which means that CLK - is the inverted signal of CLK +. If the driver connected to the controller has an unbalanced input, CLK- must be left unconnected.

DIR - / DIR +

This is the direction output for the connected driver. The 5V output is balanced which means that DIR - is the inverted signal of DIR +. If the driver connected to the Controller has an unbalanced input, DIR- must be left unconnected.

GND

The ground terminal must be connected to the driver to ensure proper operation. The GND is also reference for the +5V, SALA, +14V and SON terminals.

+14V

This terminal is a voltage output for the input or output circuitry in the connected driver. Note that this output voltage can only withstand shortcircuit for 5-10 seconds. The maximum allowable current drawn from this terminal is 50mA.

SALA

Servo Alarm input. This terminal ensures that all activity in the Controller is stopped if the terminal is left open. The terminal must under normal operation be kept logic low which means at the same level as GND. The input is active high. See *CB15 Servo alarm signal (SALA) flag*, page 113, to change active level.

+5V

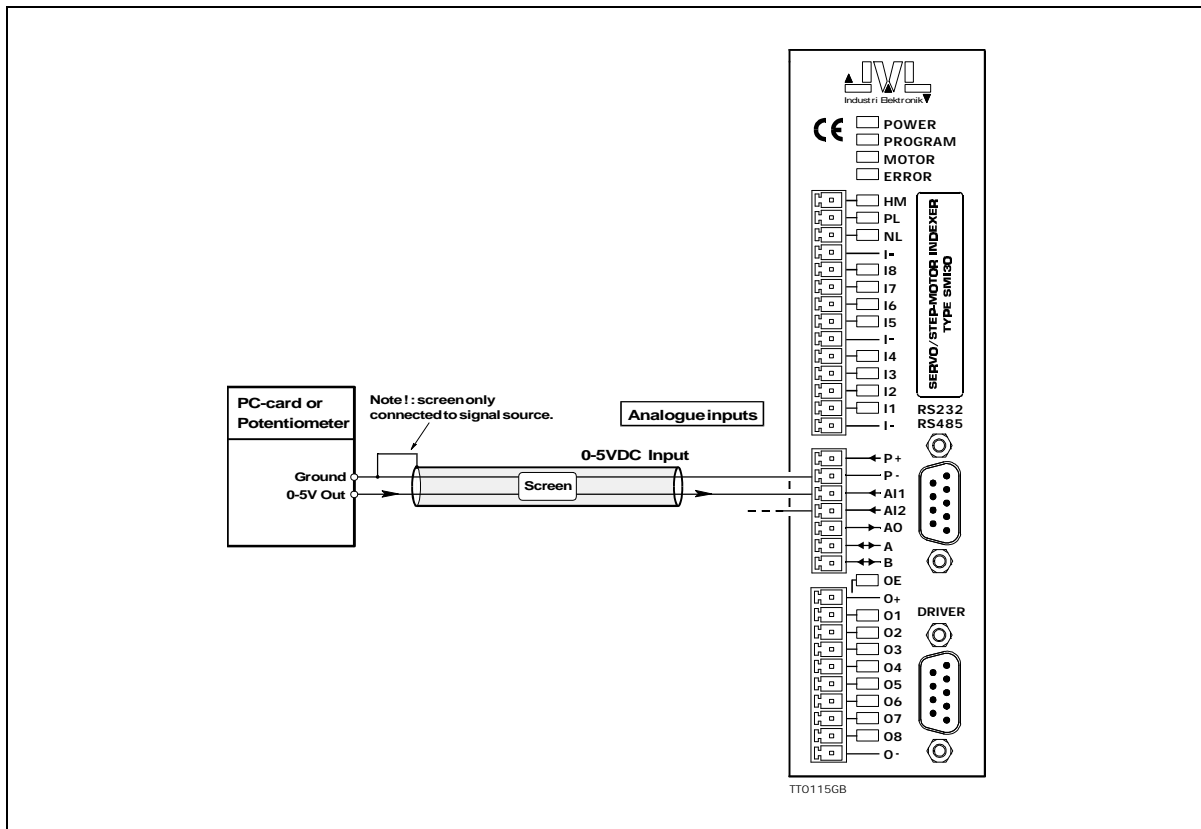
This terminal is a voltage output for the input or output circuitry in the connected driver. Note that this output voltage can only withstand shortcircuit for 5-10 seconds. The maximum allowable current drawn from this terminal is 50mA.

SON

This output is low (active NPN) if the command $SON=1$ is executed.

2.11

Analogue Inputs



2.11.1 General

The 0-5V Analogue Inputs are used for example when the Indexer is operated as a stand-alone unit. In this kind of application it can be an advantage to use a potentiometer, joystick or other device for adjusting speed, position, acceleration, etc.

In these modes of operation, the motor is controlled to produce a velocity or position, etc., which is determined by, and proportional to, the voltage applied to the Analogue Input. The Analogue Inputs share a common internal supply with the P+ and P- terminal and also the signals at the *Driver* connector, but are optically isolated from all other input and outputs. The Analogue Inputs are protected against voltage overload up to 50V peak and have a built-in filter which removes input signal noise.

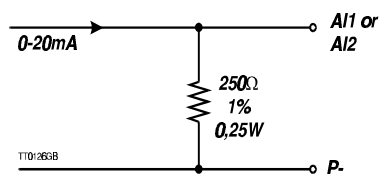
Always use screened cable to connect the source used to control an Analogue Input since the motor, etc., can easily interfere with the analogue signal and cause instability.

The Indexer is equipped with an analog-to-digital converter (ADC) which converts the detected analogue signal level. The ADC has a resolution of 8 bit.

Via software, the A/D converter can be adjusted to 10, 12, or 14 bit resolution using a special integration technique.

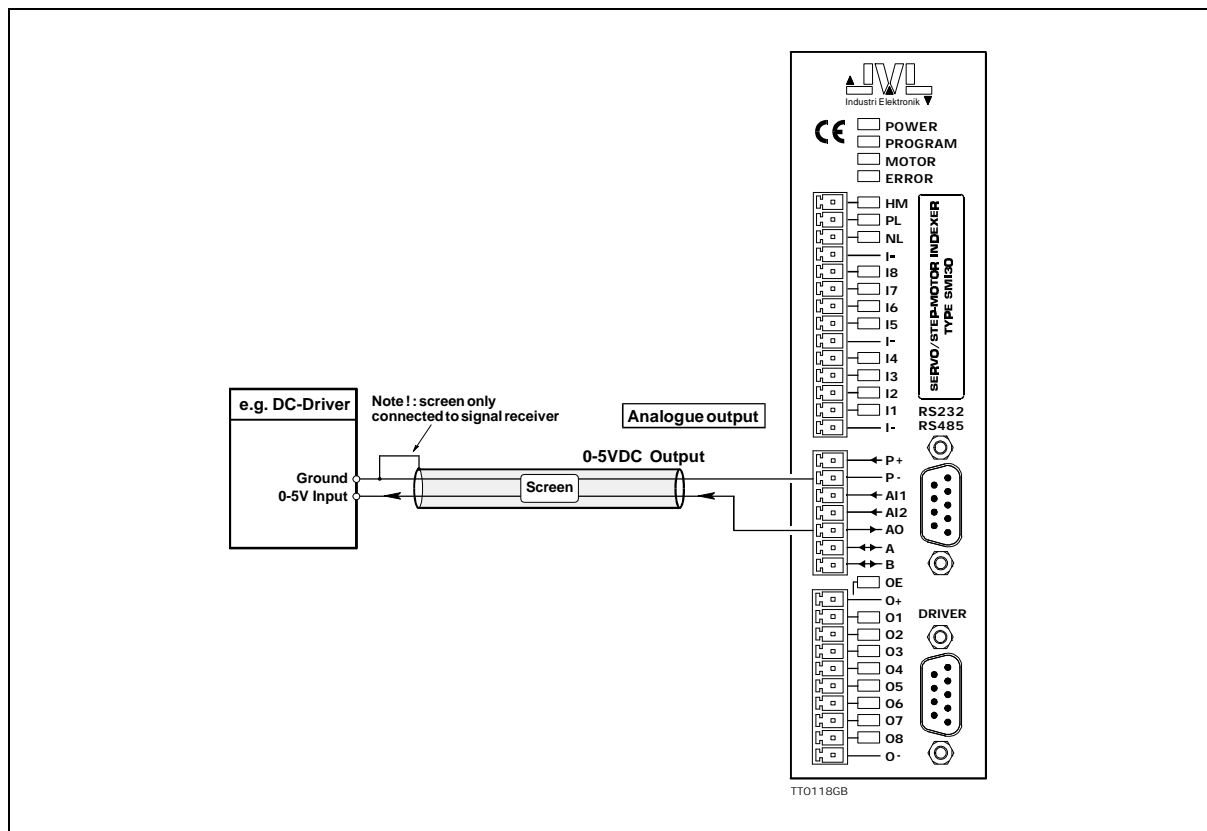
See *Analogue input (AI1 / AI2)*, page 52 for further details.

In order to use the Analogue Inputs as 0-20 mA inputs, a 250 Ω , 1% resistor must be connected between AI and P-.



2.12

Analogue Output



2.12.1 General

The 0-5V Analogue Output is used, for example, when the Indexer must control secondary functions, such as a small DC-motor + driver or a temperature controller. The voltage at the Analogue Output is set using a single command either inserted in a program or sent directly to the Indexer.

The Analogue Output shares a common internal supply with the P+ and P- terminal and also the signals at the *Driver* connector, but is optically isolated from all other input and outputs.

The Analogue Output is not protected against long-duration short circuits, although the Output can withstand short (<100ms) short-circuit or capacitive loads up to 10 nF.

Always use screened cable when connecting the Analogue Output to external units since the motor, etc., can easily interfere with the analogue signal and cause instability.

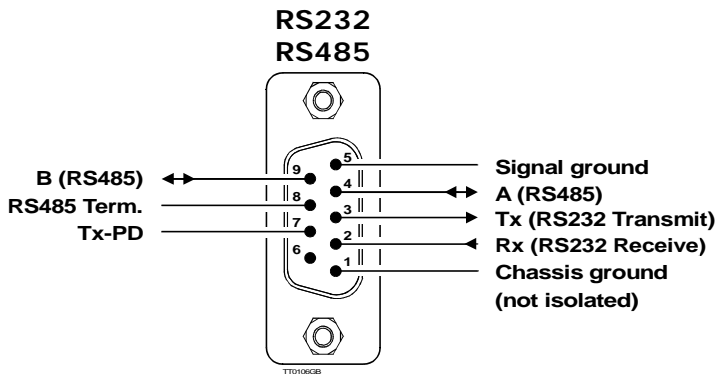
The Analogue Output can be set to a resolution between 1 and 16 bit. The output voltage however is always from 0.00 to 5.00 VDC.

See also *Analogue output (AOUT)*, page 54.

2.13 RS232/RS485 Interface

2.13.1 Interface Connection

The Indexer Interface uses the widespread RS232/RS485 standard, offering the advantage that all Personal Computers and standard terminals can be connected via the interface. The 3 interface signals Rx, Tx and ground are used. The interface cable length should not exceed 10 meters in the case of RS232 or 100 meters in the case of RS485. Indexer Interface:



2.13.2 Communication Protocol

The Indexer uses the following format: (1 startbit), 7 databits, Odd parity, 1 Stop bit. Note that a startbit is always used in the RS232(V24)/RS485 protocol. It is also possible to use 8 databit, no parity. See the *Baud Rate on RS232/RS485 (Baud)* command, page 56.

2.13.3 Communication Rate

The Indexer operates at a fixed communication rate (Baud rate) of 9600 or 19200 Baud. The Baud Rate must be set accordingly on the terminal or PC used to communicate with the Indexer. See the *Baud Rate on RS232/RS485 (Baud)* command, page 56.

2.13.4 Command Syntax

Communication with the Indexer must follow a specific command syntax:

[Address] Command [=Argument] [Checksum] <CR>

Text in square brackets [] may be included or omitted depending on the set-up.

Address: This address must be used when more than one Indexer is connected to the same interface. See also the *Address (ADDR) - Only SMI31 and SMC35B* command, page 51.

Command: The command itself. See *Command Description*, page 47 for an overview of commands.

Argument: The subsequent numeric argument for the command. An argument always begins with the equal-to sign “=”. Certain commands do not use arguments. (e.g. commands that display set-ups).

Checksum: In situations where long communication lines are used, a checksum can be used to ensure that the commands are received correctly. If an error occurs, the error message E9 is received and the command must be re-transmitted. See also the *RS232/RS485 Interface Checksum (CHS)* command, page 56.

<CR>: ASCII value 13. This character terminates the command line.

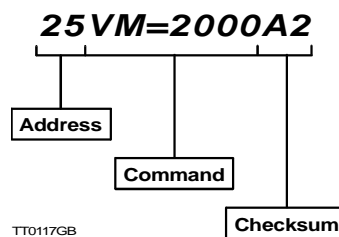
2.13 RS232/RS485 Interface

2.13.5 Synchronisation

During communication with the Indexer, each command string must be terminated by a <CR> (ASCII 13) or a semi-colon ";" to separate the commands. These delimiters tell the Indexer that the command string is complete and interpretation can begin. When a checksum is used, command interpretation will not begin until the entire command line has been received, i.e. the command line is terminated by a <CR>. A maximum of 120 characters may be sent in a single command line.

2.13.6 Checksum

In industrial applications, electrical noise from motors etc. often occurs. This noise is quite arbitrary and random and cannot be eliminated 100% even by effective electrical filtering. To ensure correct transmission of Indexer commands therefore, a checksum can be used. A typical command line may be as follows:



In this example, addressing is used (address 25). A command is transmitted followed by a checksum. The checksum consists of two characters. The checksum is a 'simple' checksum and is calculated in the following way: First the ASCII value of each of the characters in the command line is determined. These values are summed and the two least significant characters (the least significant byte) of the result's hexadecimal value are used.

The two least significant digits are converted to ASCII values and transmitted along with the command line. The actual calculation in this example is as follow:

$$50+53+86+77+61+50+48+48+48 = 418 \text{ (decimal)} = 1A2 \text{ (hexadecimal)}$$

The checksum is thus A2 (only the 2 least significant characters) which is sent as ASCII 65 (decimal) and 50 (decimal). The hexadecimal characters a-f can also be sent as capitals, i.e. can also be sent as ASCII 65 - 70 (decimal).

In the event that the command string is corrupted during transmission, the checksum will not correspond and the Indexer will report an error message "E9", indicating that a checksum error has occurred. The command string must then be re-transmitted. See the *RS232/RS485 Interface Checksum (CHS)* command, page 56, for activation of the checksum function.

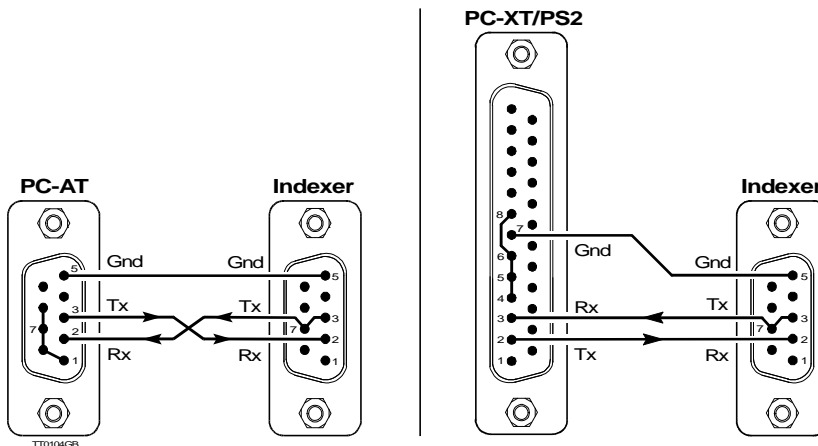
2.13.7 Parity Bit Error

In the event that one byte is corrupted during transmission, there will be a parity bit error and the indexer will report an error message E14 after the entire command has been received. The command string must be re-transmitted.

2.13 RS232/RS485 Interface

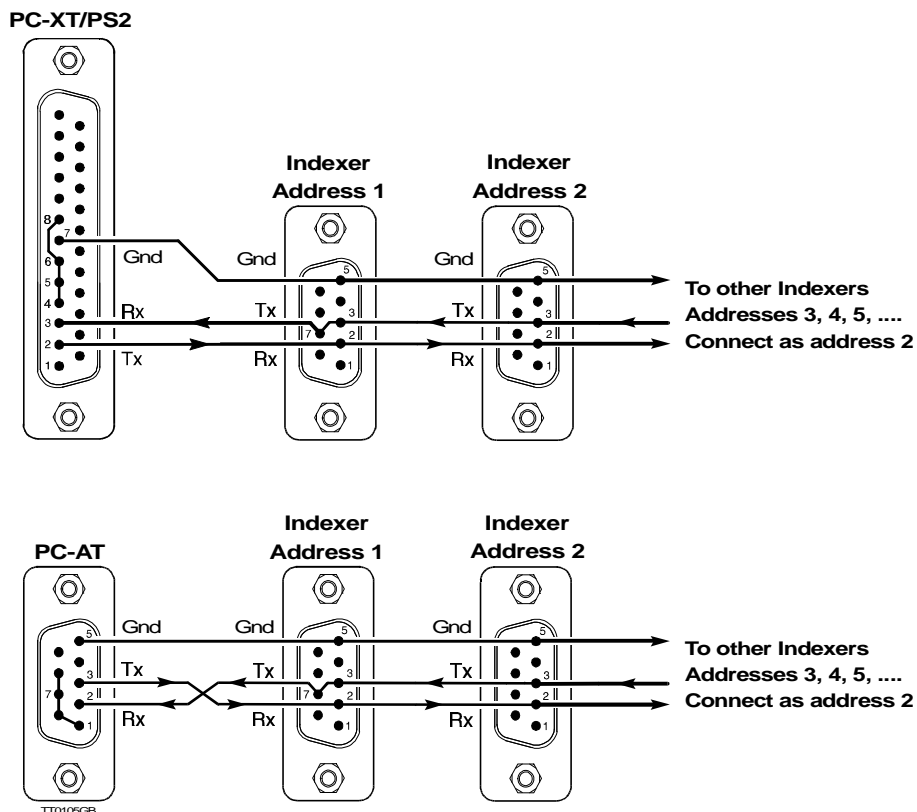
2.13.8 Connection to a PC

For communication from a PC, the following connection diagrams can be used. These show the connections between the Indexer and an IBM AT or IBM-XT/PS2:



2.13.9 Connection of Several Indexers to a PC

For connection of more than 1 Indexer to a PC (i.e. using addressing), the connection diagrams given below can be used. Note that Tx (pin 3) must be connected to TX-PD (pin 7) on one of the Indexers included in the system. It is possible to combine SMI30 and 31 and other JVL products on the same interface bus. The diagrams show the connections between Indexers and an IBM AT or IBM-XT/PS2:



2.10 RS232/RS485 Interface

JVL can deliver cables in different lengths and with up to 7 leads. See Accessories in Appendix. To make your own cable, use a male 9-pin D-Sub connector at the SMI30 and a female 9-pin D-Sub connector at the PC:

3.1 Use of RS232/RS485 Commands

3.1.1 Use of RS232/RS485 Commands

The Indexer can be controlled via its RS232/RS485 interface. Indexer commands are sent as ASCII characters terminated by <CR> ASCII 13 (decimal) or delimited by a semicolon ";". See also *RS232/RS485 Interface*, page 28.

Some of the Indexer commands have associated command parameters, others do not. For those commands which use parameters, transmitting the command alone without specifying the parameter will provoke the Indexer to respond with the command and the currently set value of the parameter. If no addressing is used, the Indexer always responds when a command has been received. If the purpose of the command is to display a value or set-up, the required information will be sent as a reply, or a 'Y' will be transmitted to indicate that the command has been received. In the event that incorrect information has been sent to the Indexer, for example a command that does not exist or a value that is out of range, the Indexer will respond with an error message. Error messages consist of an 'E' followed by a number followed by an explanatory text. See *Error Messages*, page 123.

Example:	Sent to Indexer	VM<CR>
	Received from Indexer	VM=500<CR>
	Sent to Indexer	VM=600<CR>
	Received from Indexer	Y<CR>
	Sent to Indexer	VM=-5<CR>
	Received from Indexer	E2: Out of range<CR>

Any errors in communication will be stored in the error status register 1. This register can be read using the *Read-out of Error Status (ES)* command, page 69. See also the *Error Status Text (EST)* command, page 72

Commands may be sent as both upper-case and lower-case characters. With the exception of error messages, replies from the Indexer are always upper-case.

The following sections described all of the RS232/RS485 commands. As mentioned above, all commands must be terminated by a carriage-return character <CR> or semi-colon ";" before they will be interpreted by the Indexer. These characters are not included in the description of the individual commands.

3.2

Standby Mode

3.2.1 Standby Mode

In this mode of operation the Indexer will position the motor via commands transmitted over the RS232/RS485 interface. Various operating parameters can be continuously adjusted via the interface while the motor is running. This mode is primarily used in systems in which the Indexer is permanently connected to a PC via the RS232/RS485 interface. The Program must be aborted by use of the *Halt of Motor and Program (H,K)* command, page 73 or the *Smooth Halt of Motor (SH)* command, page 102.

The position is specified in terms of pulses. The motor's instantaneous position can be read regardless of whether it is running or stationary. When a new position is set up, the motor moves to the new position using the pre-programmed velocity profile. See the following commands: *Acceleration (AC)*, page 49, *Start Rate (VS)*, page 108, and *Maximum Velocity (VM)*, page 107.

Motor operation can use a programmed velocity profile by programming a maximum velocity and acceleration. In this mode when the motor is operated to move to a new position, it will operate using the programmed velocity profile and the profile will always follow the acceleration/deceleration values. This means that the motor may not always attain maximum velocity if the distance is short. Motor status can be read using the *Report Motor Status (RS)* command, page 94.

At any time the motor can be stopped using either the *Halt of Motor and Program (H,K)* command, page 73 or the *Smooth Halt of Motor (SH)* command, page 102.

Note: In order to achieve the correct velocity and acceleration, the number of encoder pulses per revolution must be set up using the *Pulses per revolution (PR)* command, page 89.

Set a maximum velocity using the *Maximum Velocity (VM)* command, page 107

If necessary, set start velocity using the *Start Rate (VS)* command, page 108

Set an acceleration using the *Acceleration (AC)* command, page 49.

The motor can now be set to move to various positions using the *Set new absolute Position (SP)* command, page 103 or the *Relative Positioning (SR)* command, page 104.

Commands of particular interest for operation in this mode are:

Pulses per revolution (PR), page 89

Set new absolute Position (SP), page 103

Relative Positioning (SR), page 104

Start Rate (VS), page 108

Maximum Velocity (VM), page 107

Acceleration (AC), page 49

Read Status of Inputs (IN), page 75

Read/Set Status of Outputs O1 - O8 (OUT), page 86

Halt of Motor and Program (H,K), page 73

Smooth Halt of Motor (SH), page 102.

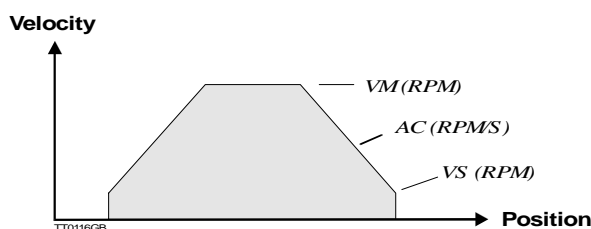


Figure -1 - Velocity profile

3.3

Program Execution

3.3.1 General Description

The Indexer and Controller provide the feature that they can be programmed using a simple and flexible programming language which is built up around the interface command set. Thus all commands can be used for developing or executing programs. During program execution, all parameters in the Controller can be read or changed. All values that can be set and read using the same single command are called registers and can be used in arithmetic expressions.

Program execution is line based. A program can consist of up to 2000 program lines, beginning with line number 0. A program line is executed every 0.8 - 7 milliseconds. The Indexer can thus take care of all the functions required in a step or servo system. For example, it is possible to communicate via the RS232 interface when a program is executed.

The programming language itself is very simple and resembles BASIC. The program is not compiled, but is interpreted during execution. This gives the advantage that in principle only a terminal is required to program the Controller.

3.3.2 Use of Commands in a Program

The inclusion of a command, such as one of the "show value" commands, will result in the returned value being sent over the RS232 interface. For example, if the current acceleration is 100, the command *AC* alone will result in the following string on the interface: *AC=100*. The command *AC=200* however will change the acceleration to 200. When a command is included in an arithmetic expression, the value of the register is substituted into the expression. For example, the program line *VM=AC+100* will set the maximum velocity to the value of the acceleration plus 100. When register values are included in expressions in this way, no account is taken of the implied units (velocity and acceleration in this case). When, for example, velocity is changed using the *VM* command, the effect on motor operation occurs instantaneously. Changes in motor parameters must therefore be made with great care.

Examples of the use of commands in a program:

```
AC=330           ; Set acceleration to 330 RPM/s
VM=500          ; Set max. velocity to 500 RPM
SR=100000       ; Advance the motor 100000 pulses
AP              ; Show actual position via the RS232 interface
```

3.3.3 User Registers

All registers can be used for temporary storage of values. Since some registers have direct effect on motor movement, as mentioned above, the Controller is equipped with 221 user-definable 32 bit registers denoted R0-R220. The memory space can also be used as 440 user-definable 16 bit registers called RI, or 880 user-definable 8 bit registers called RB, see RI and RB description. All registers are signed integer. These can be used freely to store intermediate value or can be used and included in arithmetic expressions in the same way as any other parameter such as the velocity (*VM*) or acceleration (*AC*). The user registers can store values in the range -2.147.483.647 to +2.147.483.647 and can be saved in the Controller's non-volatile memory using the command *MS2*. When the contents of the user registers are saved in non-volatile memory, they must be recalled using the *MR2* command before they can be used.

Examples of the use of user registers:

```
R1=R2           ; Set register 1 (R1) equal to register 2 (R2)
R1=R1/R2        ; Divide R1 with R2 and save the result in R1
R3=R1*R2        ; Multiply R2 by R1 and save result in R3
R1=VM*10        ; Multiply VM by 10 and save the result in R1
```

3.3

Program Execution

The user registers can also be used for indirect addressing by the use of square brackets [and]. R[3] and R3 will give the same result. [and] give the possibility of using another register or an equation as the index for the register. The following illustrate examples of indirect addressing:

VM=R[R5]
R15=R[R5+1]

3.3.4 Programming the SMI3x or SMC35 using MotoWare

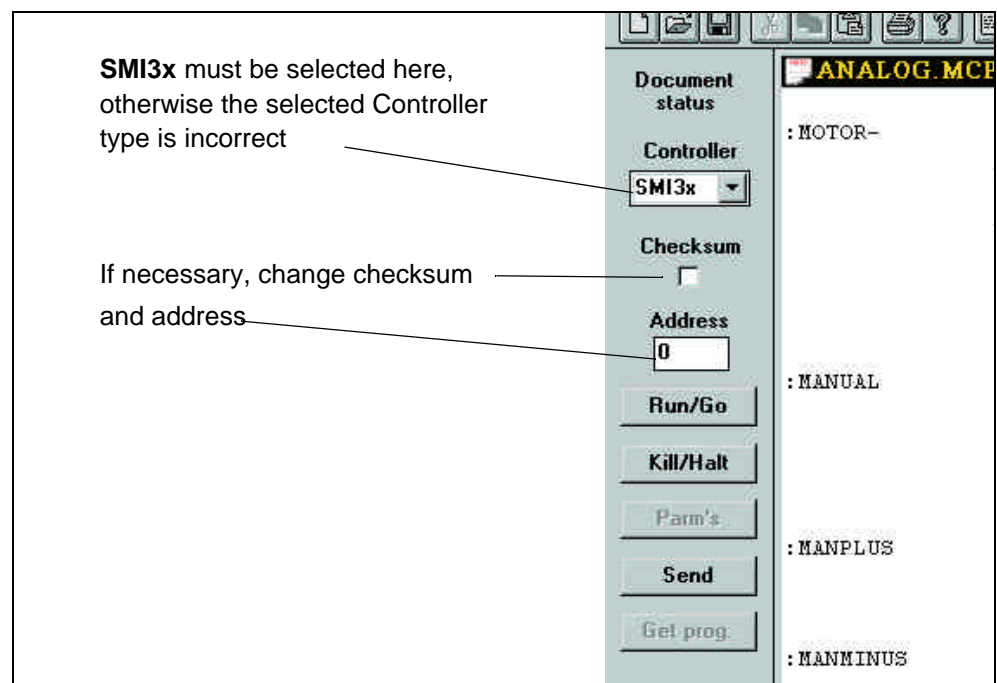
Using *MotoWare*, programs can be easily developed and saved in the Controller.

Proceed as follows to create a new program:

- 1) First, open a new program document: either by selecting **FILE** and then **New**, or by clicking the new document icon.

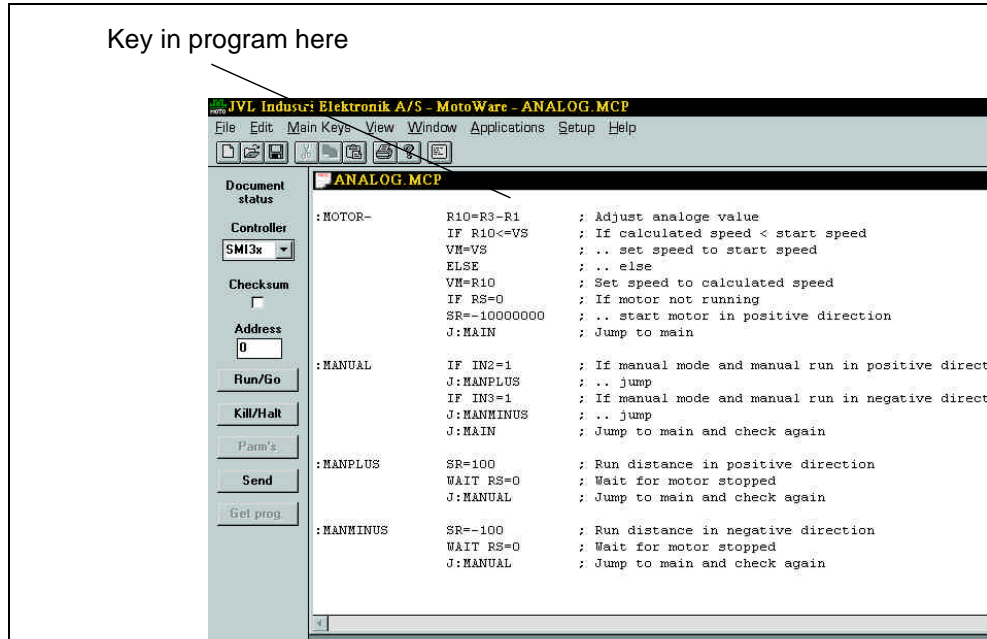


- 2) Select the correct Controller type and, if required, whether addressing and checksum are to be used.

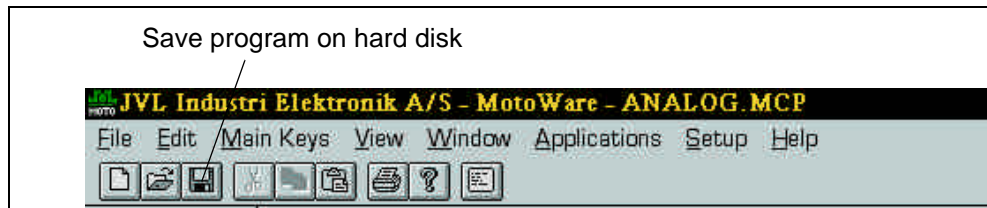


3.3 Program Execution

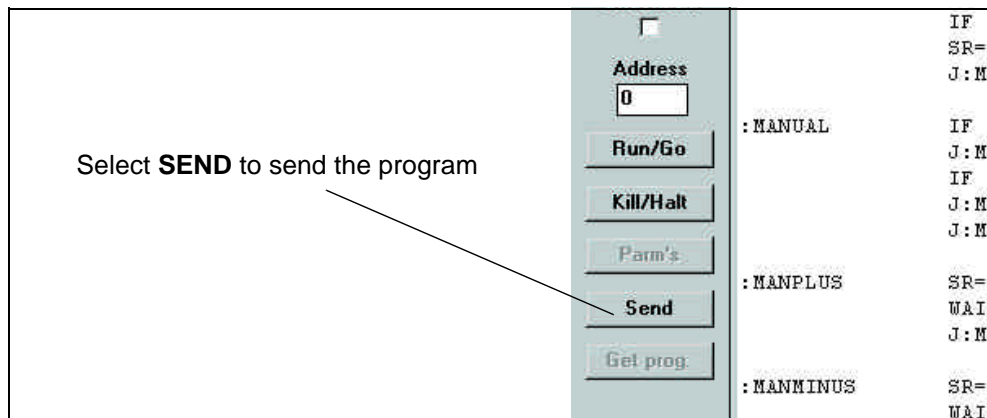
- 3) Key in the program in the program document editor window



- 4) Once the program is complete, it can be saved on the hard disk.



- 5) Once the program has been saved to hard disk, it can be sent to the Controller. Select **SEND**. If an error occurs, an error message will be displayed. See *Error messages during programming and program execution*, page 43.

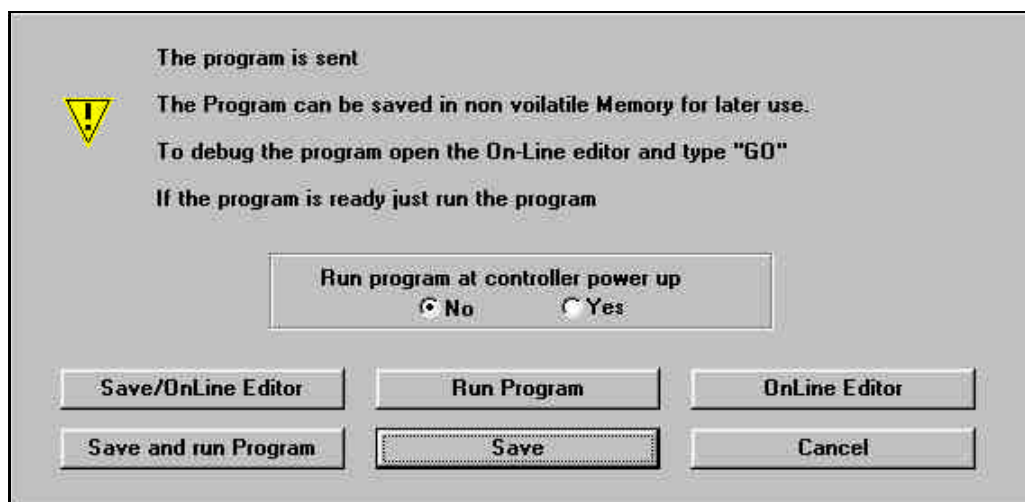


3.3

Program Execution

6) Once the program has been sent to the Controller, the dialogue box shown below is displayed. This provides several options. For example, you can choose to start the program automatically when the Controller is powered up. In this case *No* is selected followed by *Save*. The six command buttons have the following function:

- Save/Online Editor:** Saves the program in non-volatile memory and opens the On-Line Editor. When this option is selected, the *MS1* command is sent to the Controller. Then the OnLine Editor is started. The program can then be executed using the *GO* command. It is important to use the OnLine Editor during tests. In the event of program errors, the Controller sends error messages which are automatically displayed in the OnLine Editor.
- Save and run Program:** Saves the program in non-volatile memory and starts program execution. When this option is selected, the *MS1* command is sent to the Controller, followed by the *GO* command. The program is saved and then executed.
- Run Program:** Starts the program. When this option is selected, the *GO* command is sent to the Controller and program execution begins.
- Save:** Saves the program in non-volatile memory.
- OnLine Editor:** Starts the OnLine Editor directly. The OnLine Editor is opened and the program can be executed using the *GO* command. It is important to use the OnLine Editor during tests. In the event of program errors, the Controller sends error messages which are automatically displayed in the OnLine Editor.
- Cancel:** Closes the dialogue box without any further action.



3.3

Program Execution

3.3.5 Arithmetic expressions

All registers can be assigned a value by following the register name with an "equal to" sign "=", followed by an absolute value, a register name, or an arithmetic expression. Absolute values, register values and the following four operators can be used in arithmetic expressions.

Arithmetic operators used in expressions:

+	addition
-	subtraction
*	multiplication
/	division

All calculations are performed as 32-bit integers (-2.147.483.647 to +2.147.483.647). Integers are signed and have approximately 10 significant digits.

The Indexer is equipped with 221 User Storage Registers which can be used for storing intermediate results etc. These are designated R0-R220. These registers can also be used as 8 bit or 16 bit registers by using the RB0 - RB880 or the RI0 - RI440 command. In addition the Indexer is equipped with predefined registers which can only be used for specific purposes. Register VM for example is used to determine the motor top speed. The user- and predefined registers enable parameters such as lengths, speeds, acceleration, delay times, etc. to be continuously changed and controlled during program execution. In addition the User Register contents can be stored permanently in the Indexer's EEPROM memory.

Example 1:

```
R2=3000 ; Sets the value of register R2 to 3000.  
VM=R2+100 ; Sets the Top Rate to 3100 pulses/seconds for the next motor  
; operation.
```

Example 2:

```
R34=400 ; Sets the value of register R34 to 400.  
D=R34+AI1 ; Waits (400+A/D conversion of Analogue input 1 Level) x 10ms.
```

Example 3:

```
R1=AP+R2 ; Sets the value of R1 to value of the Position Counter in  
; pulses + R2.
```

3.3

Program Execution

Example 4:

```
R1=350+750 ; Sets the value of R1 to 1050.
```

Example 5:

```
R30=100 ; Sets the value of R30 to 100.
```

```
R31=200 ; Sets the value of R31 to 200.
```

```
R34=R30+R31 ; Sets the value of R34 to value of R30+R31.
```

Note:

If the # sign is used, the number will be interpreted as a binary number consisting of zeros and ones.

For example: OUT = #00001111 or R1 = #10101010 or, if the value should be read out as a binary number, the following should be written:

OUT# or R1#

3.3.6 Operator precedence and order of evaluation

The following table gives the rules of operator precedence and order of evaluation for operators that can be used in arithmetic and/or logical expressions. Operators on the same line of the table have the same rank, i.e. multiplication * and division / are ranked equally and an expression is evaluated from left to right. For example, $2*35/3$ results in a value of 23, and $35/3*2$ gives a value of 22.

Operators that can be used in arithmetic and logical expressions:

Operator	Order of evaluation
* /	left to right
+ -	left to right
< > = <= >= <>	left to right
AND	left to right
OR	left to right
= (value assignment)	right to left

3.3.7 Binary notation using the # operator

Using the # operator, it is possible to express a number in binary notation using zeroes or ones. For example it can be practical to set the outputs in this way. In addition, it is possible to express an arbitrary number or register in binary notation by setting a # after the register name. Note: R1 = #xxxx1111 is not allowed.

Examples

```
Sent to Indexer      Out=#00001010 ;Outputs 2 and 4 set active
Received from Indexer R1=#10101010 ;Register R1 is assigned
                                     the value 170
```

```
Sent to Indexer      R1#
Received from Indexer R1=#10101010
```

```
If IN = 0001010 ;If Input 2 and 4 are high and the rest
OUT1=1 ;are low, output 1 is activated
```

```
If IN = #xxxxxx10 ;If input 1 is low and Input2 is high, and
OUT2=1 ;the rest are either high or low, Out 2 is
;activated.
```

3.3 Program Execution

3.3.8 Logic operations

It is possible to perform bit-operations using the ANDL (and), ORL (or) and INV (invert) commands. Thus it is possible to set or remove single bits in a number. INV (invert) changes all "1"s to "0"s and "0"s to "1"s.

Example

```
R1=#01001111      ;Assign R1 the value 79
R2=#00110110      ;Assign R2 the value 54
R3=R1 ORL R2       ;Logic "OR" between R1 and R2
R3                 ;R3=#01111111 or 127
R3=R1 ANDL R2      ;Logic "AND" between R1 and R2
R3                 ;R3=#00000110 or 6
R3=0 INV R1        ;R3=#111111111111111111111111111111110110000 or -80
```

3.3.9 IF statement

Logical expressions can be evaluated using an IF statement. Together with ELSE, the IF statement can be used to express "decisions" within the programming sequence. Formally the syntax for the IF statement is as follows:

```
IF expression
    action1
ELSE
    action2
```

in which the ELSE clause is optional. The conditional test is performed by evaluating *expression*. If it is true, *action1* is carried out. If *expression* is false, and if an ELSE clause is included, then *action2* is carried out. The IF statement is line based: *action1* must be specified on the lines following the IF statement, and if an ELSE clause is used, ELSE and *action2* must be specified on the subsequent lines.

3.3

Program Execution

3.3.10 Error messages during programming and program execution

Three types of error message can occur during programming and program execution: grammatical errors, syntactic errors, and errors during execution (runtime errors). A check for grammatical errors is carried out immediately during transfer of a program to the Indexer. A check is made to ensure that the individual commands and operators exist, that absolute values are not too large, etc. A check is also made to ensure that commands are used in the correct context. For example, the following program line:

```
AC=H
```

will result in the error message: *E6: Parameter error or out of range*. The H command is not of the register type. When a program is transferred via the *MotoWare* program editor and an error occurs, transfer is interrupted and the line containing the error is highlighted.

When a program is interpreted during execution, any syntax errors are found while the program is in use. During testing therefore, it is important to use *MotoWare* with the On-Line editor window open. During execution, the Controller will automatically transmit any error messages. The following is an example:

```
:START VM=500
      JS : CALC
      J : START

:CALC  VM=VM+5
      J : START           ;This line has incorrect syntax. Use RET in this line
```

The above program segment will result in the error message: *E34. Too many gosub, max. 32*. The Indexer has detected 32 JS commands without RET command.

The third type of error is those that occur during normal operation of a program that functions. These are not program errors as such but errors for example in the use of registers. Assigning a value which is too great or too small to a register during online control will normally result in the error message: *E2: Out of range*. During program execution however, this type of error will not generate error messages on the RS232 interface. Instead, information about previous errors is stored in a register which can be read using the EST command. These types of error can thus be handled during program execution and therefore do not require the program to be stopped. The following example illustrates how such errors can be avoided:

```
ES0=0           // Clear any error messages
AC=100000       // Set acceleration to 100000
IF ES0>0        // If error, ES0 is greater than 0
  AC=50000      // Set acceleration to 50000
```

resulting in the acceleration being set to 50000.

3.3

Program Execution

3.3.11 Jumping to program lines and the use of labels

The Jump command *J* provides a facility for program control by jumping to a specified program line number. The Jump command can only be understood correctly by the Indexer when it is used together with an absolute value, for example *J50* (jump to line number 50). Using absolute line number values can give problems when programs are modified. When *MotoWare* is used however, labels can be used. *MotoWare* interprets and translates the individual labels and sends the correct command to the Controller. Label names may in principle consist of all displayable characters, but it is recommended that only numerals and letters (a-z) are used since problems may occur if programs are moved between computers with different set-ups. Labels are case sensitive.

The following program segment:

```
:START  IF IN1=1           ; If IN1 is equal to 1, next line is executed
        J:OK              ; Jump to label OK
        ELSE              ; If IN1 is 0, execute line after ELSE
        J:ERROR           ; Jump to label ERROR
:OK      OUT5=1            ; Set OUT5
        J:START           ; Jump to label START. Begin again
:ERROR   OUT5=0           ; Clear OUT5
        J:START           ; Jump to label START. Begin again
```

is translated to:

```
IF IN1=1
J4
ELSE
J6
OUT5=1
J0
OUT5=0
J0
```

3.3.12 Call of sub-routine

If the same sequence of commands is used often, it is a good idea to specify a sub-routine. A sub-routine is started with a label and terminated by the *RET* command. A sub-routine is called by the *JS* (Jump Subroutine) command. When the *JS* command is executed, program execution will continue from the line number specified by the *JS* command in the form of a number or a label. When the *RET* (Return) command is encountered in the sub-routine, the program returns to the main program at the line immediately after the *JS* command and continues from there. The following is an example of the use of a sub-routine:

```
        R5=500
        R6=1000
        R1=5
        JS:TEST ; set acceleration to 500
        R1=6
        JS:TEST ;set acceleration to 1000
        J:END

:TEST   AC=R[R1]
        RET

:END
```

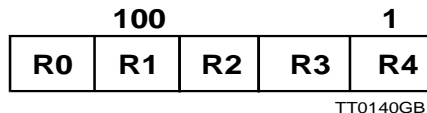
3.3 Program Execution

3.3.13 Array and pointer functions

R[Rx] Pointer for 32 bit signed numbers
 RI[Rx] Pointer for 16 bit signed number
 RB[Rx] Pointer for 8 bit signed numbers

With these registers it is possible to point at a register using the contents of another register. This is often required when the Indexer is used for prescribed repetitive tasks operating with many subjects and associated variables.

Example:



R4 contains the value 1. R1 contains the value 100
 R0 = R[R4] will transfer the value 100 which is in register R1 to register R0

Location of registers in the memory

Note that RB, RI and R are placed "on top" of each other



Example:

The example below shows how a register can be used to point to the content of another register. R10 to R15 contain velocity, lengths, etc., for subject 1. R20 to R25 contains the variables for subject 2. To access these, R1 would be set to 20.

```

:INIT  R10=1000      ;Velocity1
        R11=500       ;Acceleration
        R12=20000    ;Position1
        R13=2000     ;Velocity2
        R14=100000   ;Position2
        R15=#00001111 ;Out 1-4 = 1

:START R1=10         ;Pointer to one subject
        VM=R[R1]     ;Transfer R10 to VM
        R1=R1+1      ;Pointer = 11
        AC=R[R1]     ;Transfer R11 to AC
        R1=R1+1      ;Pointer = 12
        SP=R[R1]     ;Run to absolute position in R12
        Wait RS=0    ;Wait until motor has finished running
        R1=R1+1      ;Pointer = 13
        VM=R[R1]     ;Transfer R13 to VM
        R1=R1+1      ;Pointer = 14
        SR=R[R1]     ;Run to relative position: R14
        WAIT RS=0    ;Wait until motor stands still
        R1=R1+1      ;Pointer = 15
        OUT=R[R1]    ;Set outputs
J: START
  
```

3.3 Program Execution

3.3.14 Monitoring of inputs and errors

The program below illustrates how external events can be handled with simple commands and interrupt routines while a main program is being executed

```

                CB7=1          ;If any error bit active in ES register, output O5 will be
                                activated
:MAIN  SR=10000              ;Run relative 10000 pulses
        WAIT RS=0           ;Wait until motor has stopped
        J: MAIN             ;Jump to Main and loop

        INT 146             ;If error 46 (Alarm signal from motor drive) occurs while in
        OUT4=1              ;Main routine, activate output O4
        RETI                ;Return to main routine

        INT1                ;If IN1 is activated
        SH                  ;Stop the motor and
        OUT6=1              ;Activate output O6
        OUT6=0              ;Deactivate output O6 again
        RETI                ;Return to main routine
```

3.3.15 Pause in program execution (Delay)

The D command pauses program execution. The resulting break in program execution is specified in units of 10 msec by writing D=pause. For example:

```

        R1=20               ; Set R1 to 20
        D=R1                ; Wait for 200 msec.
```

3.4

Command Description

This section gives a brief description of the SMI30-31 Indexer command set, including the following main points: Command Name, Mode, Range, Usage and Examples. All the commands are listed in alphabetical order.

Command name:	The name of the command
Modes:	In which mode the command is available — Standby mode, Programming mode or Running mode
Range:	The valid numerical range of the command. For example: 0 - 100000 RPM/sec.
Selection:	The valid prefix to the command
Default:	The default value of the command or register content after Power up and SD command
Description:	A brief explanation of the command
Usage:	Describes the syntax of the command
Example	An example how the command could be used

3.4 Command Description

3.4.1 Show set-up (?)

<u>Command</u>	?
<u>Modes</u>	Stand by
<u>Description</u>	The most important details of the Indexer status and set-up can be displayed using this single command.
<u>Usage</u>	? Display values.
<u>Example</u>	<pre>Sent to Indexer ? Received from Indexer: **JVL Industri Elektronik A/S, SMI30, 27-01-1998,1.5, Addr=0 Max. Velocity (RPM): VM =100 Acceleration (RPM/S): AC =100 Start Velocity (RPM): VS =10 Actual Pos. (Unit,Pulses): AP =0, APP =0 Puls/rev. Motor: PR =8192 Acc. S-curve (RPM/S/S): ACS =0 Conversion fac.(pulse/unit): CON =1.0000 Counter 1 (Pulses): CN1 =0 Counter 2 (Pulses): CN2 =0 Counter 1 mode,divider: CTM1 =1, CND1 =1 Counter 2 mode,divider: CTM2 =1, CND2 =8 Input (HM,PL,NL,I8-I1): IN =#0,0,00000000 Output (O8-O1): OUT =#00000000 Analogue input 1 (Volt): AI1 =0.00 Analogue input 2 (Volt): AI2 =0.00 Analogue output (Volt): AOOUT =0.00 Input voltage P+ (Volt): VOL =25.05</pre>

3.4.2 Indexer Type (!)

<u>Command</u>	!
<u>Modes</u>	Stand by
<u>Description</u>	This command (an exclamation mark) can be used to obtain information about the Indexer type and its address. The Indexer will reply to this command regardless of whether addressing or checksum is used. Thus only 1 Indexer may be connected to the interface if this command is used without an address. The command can be used alone, or together with an address.
<u>Usage</u>	! Show Indexer type and address.
<u>Example</u>	<pre>Sent to Indexer ! Received from Indexer SMI30:ADDR=0</pre>

Note that the above is only an example. If the Indexer is a type SMI31, the response would be SMI31. Similarly the address (0 in the above example) will also depend on the actual address of the Indexer in question.

3.4 Command Description

3.4.3 Delete EEPROM (##)

Command ##

Modes Standby

Description This command can be used to reset the content of the Indexer's permanent memory. (EEPROM). Program, error messages, address, and checksum data will be erased and set to default factory settings. (MR1=0, Baud=2, Addr.=0, CHS=0). The Indexer will reply to this command regardless of whether addressing or checksum are used. The command can also be used in extreme cases where the Indexer is functioning improperly because of noise, has been programmed incorrectly, or includes a fault.

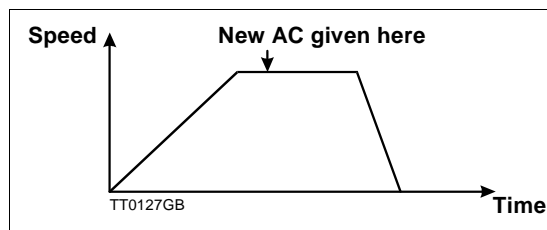
3.4.4 Acceleration (AC)

Command AC

Modes Stand by, Programming, Running

Range 1 - 2000000 RPM/sec.

Description This command is used to specify the acceleration/deceleration profile. AC can be changed while the motor is running. This gives for instance the possibility to accelerate very softly and to decelerate very fastly in the same run. If AC is changed during an acceleration/deceleration, the new value will not be active until the next acceleration/deceleration. Remember that if AC is changed while the motor is running, this must be done so that the motor has time to decelerate with the new slope. If AC is set to a value higher than 16777215*5455/PR, AC is set to 16777215*5455/PR.



Usage AC = x Set acceleration in RPM/sec.
AC Show the current acceleration value.

3.4.5 Acceleration pulses (ACP)

Command ACP

Modes Stand by, Programming, Running

Range 1-1000000 Pulses.

Description This command is used to specify the acceleration profile expressed in pulses. Using this command it is possible to make the acceleration and deceleration with a fixed length. If the motor is running when ACP is changed, the acceleration will be changed only when the motor has stopped. If the acceleration is to be changed "on the fly", use the AC command.

Usage ACP = x Set acceleration in fixed number of pulses.
ACP Show the actual acceleration in number of pulses.

3.4 Command Description

3.4.6 Acceleration Time (ACT)

Command ACT

Modes Stand by, Programming, Running

Range 1-10000 ms

Description This command is used to specify the acceleration profile expressed in time. Using this command it is possible to make the acceleration and deceleration with a fixed time during. If the motor is running when the acceleration time is changed, the acceleration will be changed only when the motor has stopped. If the acceleration is to be changed "on the fly", use the AC command.

Usage **ACT** = x Set acceleration in fixed time.

ACT Show the current acceleration time in msec.

3.4 Command Description

3.4.7 Address (ADDR) - Only SMI31 and SMC35B

Command ADDR

Modes Stand by, Programming, Running

Range 0 - 255

Description The Indexer can be configured to react to all communication via the interface bus (Point-to-Point communication). In this case, the Indexer address must be set to 0, which is the factory default. When the address is set to 0, the address must not be transmitted together with any command during communication with the Indexer. It is also possible to connect several Indexers to the same interface bus. In this case each Indexer must be assigned its own unique address in the range 1-255. This is done by writing an address command to each Indexer and connecting all Indexers in parallel to the same RS232/RS485 port. When the address is changed, it will be effective immediately and be stored in the EEPROM so that the address is remembered, also at power down. Commands can then be sent to each individual Indexer by preceding each command with the respective address between 1 and 255. The number of Indexers that can be simultaneously controlled is however dependent on the system hardware. Note: If the address of an Indexer has been forgotten, the *Indexer Type (!)* command, page 48 can be used.

Usage ADDR=x Set address to x.

ADDR Show address.

Example

5SR=1000	Send command SR=1000 to Indexer with address 5.
Y	"Y" is received in return as an indication that the command is received.
5R1	Ask Indexer with address 5 for the value of R1.
R1=1000	Indexer returns R1=1000.

3.4 Command Description

3.4.8 Analogue input (AI1 / AI2)

Command AI1 or AI2

Modes Stand by, Programming, Running

Range 0 - 16383

Description The AI1 and AI2 commands refer to the two analogue inputs. These commands make it possible to read these inputs. This can be done as a part of a program or when the Indexer is in standby mode. For example, an analogue input can be used to adjust the speed as a function of an analogue voltage applied to the input. Another application is the adjustment of motor position or length according to the analogue voltage applied to these inputs. The analogue inputs are fed to an A/D converter with 8 bit resolution. Internally the Indexer uses an oversampling technique which makes the actual resolution much higher. By use of oversampling, it is possible to achieve resolutions of 8, 10, 12 or 14 bit. To set the resolution, control bit *CB2* and *CB3* can be used (See the *CB2 / CB3 Analogue conversion flags* command, page 110.) The default resolution is 8 bit. Regardless of the resolution, the value returned will always be in the range 0-16383. This makes it possible to adjust the resolution without making any further conversion of the measured values. Notice that higher resolution also makes the execution time longer.

Usage **AI1** Shows actual level at analogue input 1 (AI1)

AI2 Shows actual level at analogue input 2 (AI2)

Example 1 High resolution is required, therefore 14 bit-resolution is set. The motor must move to an absolute position determined by the analogue input voltage. This movement must start (be updated) every time input 1 is activated. The example code is as follows:

```
:INIT  CB2=1          ;SELECT 14 BIT RESOLUTION
        CB3=1          ; -
        VM=1500        ;SET THE SPEED TO 1500 RPM
:START  WAIT IN1=1     ;WAIT FOR A START SIGNAL AT INPUT 1
        SP=AI1*6       ;READ THE ANALOGUE INPUT AND MULTIPLY
                          ;THE VALUE WITH 6. LET THE MOTOR MOVE TO
                          ;THE RESULTING POSITION.
        WAIT RS=0      ;WAIT UNTIL MOTOR HAS STOPPED
        J:START        ;JUMP TO START AND REPEAT
```

Example 2

```
        SR=10000
:Start  VM=AI1/10+200
        IF RS>0
        J:Start
```

3.4 Command Description

3.4.9 Activate flag in external module (AO) - Only SMI31

Command AO

Modes Standby, Programming, Run

Range Address 0-31, Flag 0 - 65535

Description The Activate command is used to activate a flag in an external module whose address is specified by "a".
The Flag number is specified by "o". For example, the flag may refer to an output on a JVL IOM11 module (Input/Output module). When the flag is activated, an output will be activated. A flag in a different module may refer to a completely different function. For example if flag 3 in a JVL Keyboard/Display KDM10 module is activated, the cursor on the module's LCD display will blink. Flags with the same number in different modules can have different functions. Remember that all modules should have their own unique address.

See the instruction manual for the individual module for a description of the function of the module's flags.

Format: AO{1<=a<=31}.{1<=o<=255}

Example 1: A Keyboard-Display Module has address 4. The module display is to be erased so that new text can be displayed. The following command will erase the display and position the cursor at the top left-hand corner of the display.

AO4 . 1 ; Erase LCD display

Example 2: An IOM11 module and the Indexer are connected together in a system. The IOM11 module address is 10. Output 4 is to be activated. The following command is used:

AO10 . 4

3.4 Command Description

3.4.10 Analogue output (AOUT)

Command AOUT

Modes Standby, Programming, Run

Range 0 - 65535

Description The AOUT command is used to set the analogue output of the Indexer. The value specified by the AOUT command is converted into an analogue voltage between 0.00 and 5.00 VDC. Note that the analogue output uses a PWM technique and is not recommended for use in very precise applications.

Normally the output can be used for temperature control, or for speed control of secondary motors, etc.

The resolution can be set in the range from 1-16 bit which corresponds to a dynamic range from 1 to 65535. The resolution is set by the CND2 register (see page 58). The CND2 register must be set in the range 1 to 16 which corresponds to the number of bits.

To enable the analogue output, CTM2 must be set to 7 (see page 65).

It is recommended that the resolution is set in the range 5 to 9.

Note that if the output value is specified outside the allowable range, the output voltage will not be valid and an error message will occur.

Bit resolution (CND2)	Range (0-5V)	Ripple frequency Hz	Ripple voltage 2,5V
1	0-1	460800	-
2	0-3	230400	-
3	0-7	115200	-
4	0-15	57600	-
5	0-31	28800	-
6	0-63	14400	-
7	0-127	7200	-
8	0-255	3600	-
9	0-511	1800	-
10	0-1023	900	-
11	0-2047	450	-
12	0-4095	225	-
13	0-8191	112.5	-
14	0-16383	56.25	-
15	0-32767	28.12	-
16	0-65535	14.06	-

Example In this example the analogue output is set to 2.5 volt.

```
:START CND2=8 ; SET ANALOGUE OUTPUT TO 8 BIT RESOLUTION
      CTM2=7 ; SET TIMER 2 AS PWM GENERATOR
      AOUT=127 ; SET ANALOGUE OUTPUT TO 2.5 VOLT
```

3.4 Command Description

3.4.11 Actual Position (AP)

Command AP

Modes Stand by, Programming, Running

Range -2147483648 - +2147483647 units or pulses

Description The motor position can be read at any given time. The position is given in terms of units relative to the zero point. The motor's position can also be "reset" by specifying an argument to the AP command. If the conversion factor is set to 1 (CON=1.0000) the units will be expressed in pulses. AP is the desired position and not necessarily the motor position. There can be a difference if a servo motor is used and it has a positioning error during running and standstill.

The position can be changed only when the motor is stationary.

Usage **AP = x** Sets motor's current position to x.

AP Show motor's position in units.

3.4.12 Actual position expressed in number of pulses (APP)

Command APP

Modes Stand by, Programming, Running

Range -2147483648 - +2147483647 pulses

Description The motor position can be read at any given time. The position is given in terms of pulses relative to the zero point. APP is the desired position and not necessarily the motor position. There can be a difference if a servo motor is used and it has a positioning error during running and standstill.

The position can be changed only when the motor is stationary.

3.4 Command Description

3.4.13 Baud Rate on RS232/RS485 (Baud)

Command BAUD

Modes Stand by

Range 1-8

Description This command is used to select the baud rate of the serial interface. The table below shows the selectable values.

Baud	Baud Rate bits/sec	Parity	Databits
1	1200	odd	7
2	9600 (default)	odd	7
3	19200	odd	7
4	reserved		
5	1200	None	8
6	9600	None	8
7	19200	None	8
8	reserved		

When the baud rate is changed, the new baud rate is valid immediately. The value is not stored in the Indexer EEPROM, and will not be remembered after power has been switched off or Reset has been activated.

The default value is Baud = 2 (9600 bits/sec, odd parity, 7 databits).

3.4.14 Control Bit (CB1 - CB31)

See *Control Flags*, page 110.

3.4.15 RS232/RS485 Interface Checksum (CHS)

Command CHS

Modes Standby, Programming, Running

Selection 0 = no, 1 = yes

Description As described in *Checksum*, page 29 a checksum can be used for communication via the interface.

Usage
CHS=0 Do not use checksum.
CHS=1 Use checksum.
CHS Show checksum status.

3.4 Command Description

3.4.16 Counter 1 / Timer 1 (CN1)

Command CN1

Modes Standby, Programming, Running

Range 0 - 65535

Description The contents of the counter / timer 1 are available and can be set using this command. CN1 can be verified, reset, or preset at any time under standby or program control. See also the CTM1 (page 63) and CND1 (page 58) commands.

Usage **CN1** Shows the contents of Counter Timer 1

CN1 = 100 Set the Counter/Timer 1 equal to 100

3.4.17 Counter 2 / Timer 2/ Encoder (CN2)

Command CN2

Modes Standby, Programming, Run

Range 0 - 65535 (-2 mia. to +2 mia. if CTM2=10)

Description The contents of the counter / timer 2 are available and can be set using this command. CN2 can be verified, reset, or preset at any time under standby or program control. See also the CTM2 (page 65) and CND2 (page 58) commands. If CTM2=10, then CN2 is used to show encoder pulses on IN7 and IN8. See CTML page 56.

Usage **CN2** Shows the contents of Counter Timer 2

CN2 = 100 Sets the Counter/Timer 2 equal 100

3.4 Command Description

3.4.18 Counter 1 / Timer 1 divider (CND1)

Command CND1

Modes Standby, Programming, Running

Range 1 - 2.147.483.648

Description This command can only be used when counter 1 (CN1) is set to mode 5 or 6 (timer with divided frequency). See the *Counter mode (CTM1)* command, page 63.
An overrun in the counter will normally cause an interrupt and the routine INT7 will be executed, if it is implemented in the actual program.
When the counter 1 is set to mode 5 or 6, the interrupt will first be accessed after the interrupt has occurred the number of times specified by CND1.
If the counter 1 (CN1) is driven by the internal counter frequency of 921600 Hz, the interrupt will occur 14 times per second. This also means that if an interrupt is requested each second, the command CND1=14 must be executed.

Example In this example the INT7 routine will be called every 10 second

```
:START   CTM1=5           ; SET TIMER TO DIVIDED FREQUENCY
          CND1=140        ; (921600/65536)*10 SECOND
          .
          .
          INT7            ; INT7 ROUTINE
          OUT8=1
          D=100
          OUT8=0
          RETI           ; RETURN TO MAIN PROGRAM
```

3.4.19 Counter 2 / timer 2 divider (CND2)

Command CND2

Modes Standby, Programming, Running

Range 1 - 2.147.483.648

Description This command can only be used when counter 2 (CN2) is set to mode 5 or 6 (timer with divided frequency) or mode 7 (analogue output). See the *Counter mode (CTM2)* command, page 65.
An overrun in the counter will normally cause an interrupt and the routine INT8 will be executed if it is implemented in the actual program.
When the counter 2 is set to mode 5 or 6, the interrupt will first be accessed after the interrupt has occurred the number of times specified in CND2.
If the counter 2 (CN2) is driven by the internal counter frequency of 460800 Hz, the interrupt will occur 7 times per second. This also means that if an interrupt is requested each second, the command CND2=7 must be executed.
If CTM2 is set to mode 7 (analogue output) the CND2 command is used to specify the number of bits used to produce the analogue output voltage.

3.4 Command Description

3.4.20 Clear flag in external module (CO) - Only SMI31 and SMC35B

Command CO

Modes Standby, Programming, Run

Range Address 0-31, Flag 0 - 65535

Description The Clear command is used to clear a flag in an external module. The number of flags which can be cleared in different external modules varies, but each module has at least 1 flag. For the JVL KDM10 module (Keyboard-Display Module) for example, the Clear command can be used to clear the LCD display; in the IOM10 module (I/O module) the Clear command can be used to deactivate one of the Module's outputs, etc. See page 79.

Format: CO {1<=a<=31}.{1<=o<=255}

Example 1: The Indexer and a KDM10 Module are connected in a system via the RS485 interface. The address of the Indexer is 1 and the KDM10 module address is 3. The Cursor on the KDM10's LCD display is to be switched off. If the cursor is active while text is being printed using the PRINT command, the display may flicker. This is avoided by switching off the cursor as follows:

```
CO3.3 ; Deactivate cursor
```

Example 2: The Indexer and an IOM10 module are connected in a system via the RS485 interface. The IOM11 module's address is 5. The IOM11's output 7 is to be de-activated. The command is as follows:

```
CO5.7 ;Deactivate output 7 on IOM11 module with address 5.
```

3.4.21 Compare Memory (COMP)

Command COMP

Modes Stand by

Range 0 - 1000

Description The program which is in the permanent EEPROM memory is compared, byte by byte, with the program in the temporary memory. If the two programs are identical, a "Y" is returned. If there is an error, the response indicates in which line, e.g. COMP = 12 (error in line 12).
If there is no program in temporary memory, the response "E21: No program in RAM" is given.

3.4 Command Description

3.4.22 Conversion factor (CON) - Only SMI31 and SMC35B

Command CON

Modes Stand by, Programming, Running

Range 0.0001 - 9999.9999

Description The *CON* command makes it possible to specify a conversion ratio between the number of pulses a motor moves and a unit of measurement such as length, volume, position, etc. The command *CON* specifies the number of pulses per well-known unit - length, volume (mm, ml, cm, dl, etc.). The conversion factor can be specified as a real number in the range 0.0001 to 9999.9999, with up to 4 decimal points.

The *CON* command is inserted at the beginning of a program and stays in effect until any subsequent conversion factor is specified at run-time. When a motor operation is performed, the number of units specified by the motor command is multiplied by the conversion factor and the motor moves the resulting number of pulses.

If, for example, a motor must move 2.3456 steps to dose a volume of 1 millilitre, the conversion factor is set to a value of 2.3456 using the command *CON=2.3456*.

To dose a volume of 450 ml in a subsequent motor command, the value 450 is specified. The conversion results in $450 * 2.3456$ (1055.52) pulses. The motor will then move 1055 pulses and the remainder (0.52 pulses) will be stored. The remainder is used in the next motor operation to correct for the 0.52 dosage pulses.

Example A system requires motor operation of 14.654 pulses to dose a volume of 1 ml. The following program is send to the indexer.

```
CON=14.654      ;THE CONVERSION FACTOR IS SET TO 14.654
                ;PULSES PER MILLILITRE
R1=290          ;A VALUE OF 290 IS ASSIGNED TO REGISTER R1
SR=R1          ;THE MOTOR IS MOVED TO PROVIDE A DOSE OF
                ;290 ml. THE NUMBER OF PULSES RUN IS
                ;290 * 14.654 = 4249. THE PULSE REMAINDER
                ;IS 0.66
D100           ;DELAY OF 1 SECOND
SR=+18         ;DOSE 18 ml. THE NUMBER OF PULSES IS
                ;18 * 14.654 + PULSE_REMAINDER = 264
                ;THE NEW PULSE REMAINDER IS 0.432
```

Note that after a "Home" operation using the *SZ* command, or after a new *CON* command is executed, the pulse remainder is reset to 0 since the motor is set to its absolute reference point.

An extra pulse will be executed if the remainder in the positive direction is ≥ 0.5 or if the remainder in the negative direction is > 0.5 .

Note: The length or the number below the decimal point, multiplied by the conversion factor, must give a number of maximum $\pm 2.147.483.647$.

If *CON* is used in arithmetic expressions, only the whole numbers are included in the calculation.

Usage **CON = x** Sets the conversion factor to x.

CON Show the actual conversion factor.

3.4 Command Description

3.4.23 Motor current at standby (CS) - Only SMC35

Command: CS

Modes: Standby, Programming, Running

Range: SMC35B 0-6000 mA, SMC35A 0-3000mA

Description : The current supplied to the step motor can be adjusted to specified values for standby using the CS command. Set the current at top speed, acceleration and deceleration VM using the CT command. See the CT command for further description, examples etc.

Usage: **CS=1000** Sets the motor current when the motor is stationary to 1000mA
CS Show the motor current at standstill

3.4.24 Motor current when running (CT) - Only SMC35

Command: CT

Modes: Standby, Programming, Running

Range: SMC35B 0-6000 mA, SMC35A 0-3000mA

Description : The current supplied to the step motor can be adjusted to specified values for standby using the CS command. Set the current at top speed, acceleration and deceleration VM using the CT command. Normally only a small current is required when the motor is stationary since the static inertia of a typical step motor is much less than the inertia while the motor is rotating, depending on the speed range of the motor.

The torque of a step motor is directly proportional to the applied current, up to the specified phase current (see the specifications for a given motor). If a 4 phase motor with 8 leads is used, the current should be set to the value specified for the motor multiplied by a factor of 1.4. See motor connections section for further information.

In the nominal current is exceeded, the motor will overheat and only very little increase in torque will result.

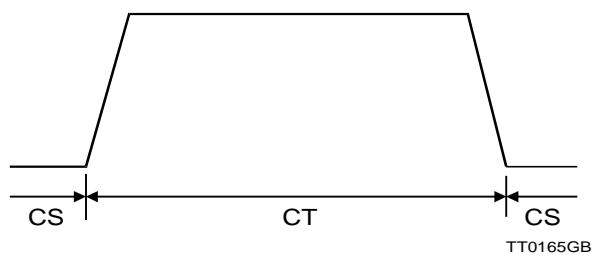
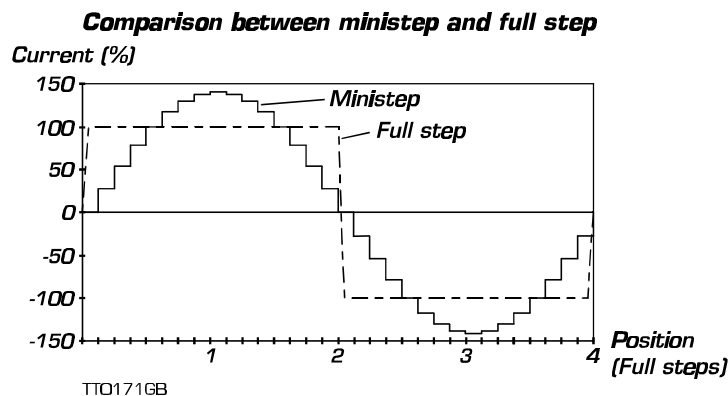
The commands can be used at any point in a program and can be specified and changed continuously throughout a program.

The current supplied to each of the step motor's phases can be adjusted for standby and operating currents using software commands CS and CT. The Driver automatically switches between the two currents by detecting the presence of step-pulses. Values for the two currents are typically adjusted so that the Operating Current is significantly higher than the Standby Current, since the motor must be supplied with more power to drive its load during acceleration and constant operation than when it is stationary. The only overriding consideration that must be made in the adjustment of motor phase currents is that the thermal output of the motor must not exceed the maximum operating temperature of the step motor — see the manufacturer's product data for the motor in question. Independent of which model of SMC35 is used, the current will always be divisible by 64 (6 bit resolution)

Usage: **CT=4000** Sets the current at acceleration, deceleration and at VM velocity to 4000mA
CT Show the motor current when running

3.4 Command Description

When mini-step resolution is changed using the PR command, the motor current is sinusoidal and the peak current will be greater than the specified CT current by a factor of 1.41. This is done to maintain the correct RMS current so that the motor is utilised 100%. When the current in one phase is at the peak value (1.41 times the specified CT current), current in the other phase will be zero.



Example:

```
CS=400      ;Sets motor Standby Current to 400mA (0.4A).
CT=5000    ;Sets motor Current during acceleration/de-
            ;celeration and running to 6000mA (6A).
AC=1000    ;Sets acceleration/deceleration to 1000 rpm/s
VM=1000    ;Set max. velocity to 1000 rpm
SR=10000   ;Run the motor 10000 steps forward
WAIT RS=0  ;Wait until motion completes.
```

3.4 Command Description

3.4.25 Counter mode (CTM1)

Command CTM1

Modes Standby, Programming, Run

Range 1 - 10

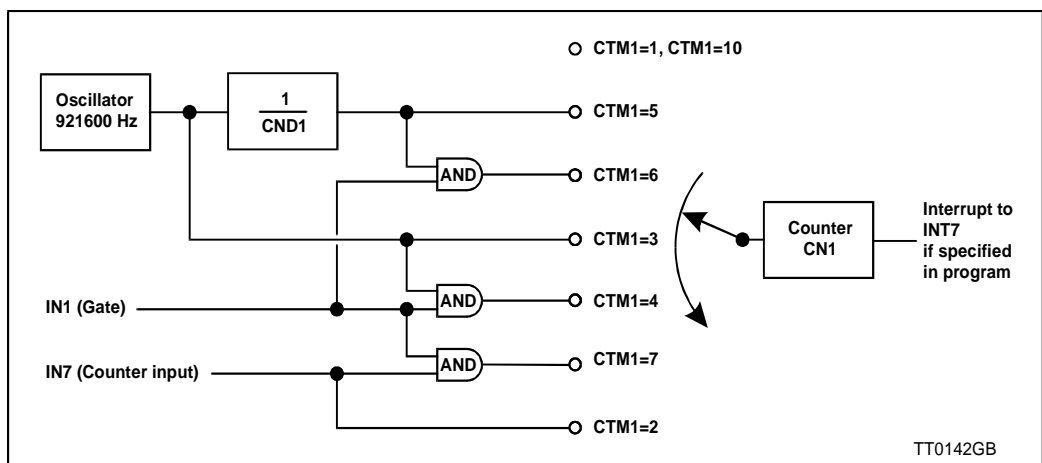
Description The Indexer provides facilities for solving advanced applications using 2 counters/timers denoted CN1 and CN2. These counter/timer registers have a resolution of 16 bit (0-65535).

The registers can be configured for different purposes by setting the CTM1 and CTM2 registers.

The counter / timers can be set as an ordinary incremental-counter, timer, gated timer, timer with divider (slow speed), etc. Note that both counters are set in encoder mode if CTM2 (see page 65) is set to 10. This makes counter 1 disabled for normal use.

Usage

Different counter / timer modes	
CTM1 = 1	Counter 1 (CN1) is not active
CTM1 = 2	Incremental counter (0-65535). Input 7 is counter input
CTM1 = 3	Timer. The timer frequency is 921600 Hz. If overflow, timer starts from 0.
CTM1 = 4	Gated timer. Input 1 must be high to keep the counter running
CTM1 = 5	Timer with divided frequency. Division ratio is set up by the CND1 command
CTM1 = 6	Similar to CTM1 = 5 but as gated timer.
CTM1 = 7	Gated counter. Gate is input 1 and count input is input 7.
CTM1=10	Encoder Mode. See CTM2 command



3.4 Command Description

Example

In an actual application there is a requirement to count an external event. For every 10 counts, program execution must be interrupted and an unconditional jump be made to an interrupt routine that drives the motor 10000 pulses. The program is as follows:

```
:START  CN1=65535-10  ;SET COUNTER REGISTER EQUAL 65525.  
        CTM1=2        ;ACTIVATE COUNTER MODE  
        .  
        (more program)  
  
INT7    ;BEGINNING OF INTERRUPT 7 ROUTINE  
CN1=65535-10 ;SET COUNTER REGISTER TO 65525  
SR=10000    ;DRIVE MOTOR +10000 PULSES  
WAIT RS=0   ;WAIT HERE UNTIL MOTOR HAS STOPPED  
RETI       ;RETURN TO MAIN ROUTINE
```

Note !

When the counter (CN1) overflows from 65535 to 0, the interrupt routine INT7 is executed if this routine is included in the program memory.

If the command NSTOP=7 is included in a program at the same time as the counter overflows, the motor will be stopped.

3.4 Command Description

3.4.26 Counter mode (CTM2)

Command CTM2

Modes Standby, Programming, Run

Range 1 - 10

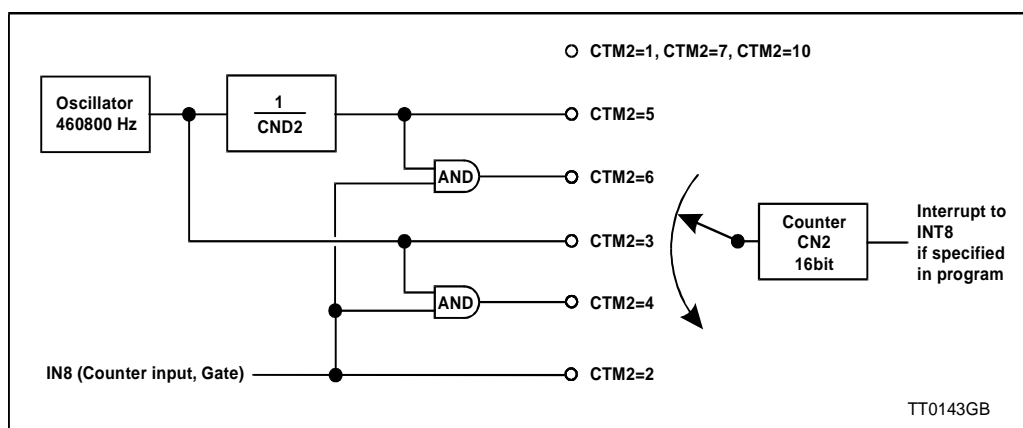
Description The Indexer provides facilities for solving advanced applications using 2 counters/timers denoted CN1 and CN2. These counter/timer registers have a resolution of 16 bit (0-65535).

The registers can be configured for different purposes by setting the CTM1 and CTM2 registers.

The counter / timers can be set as ordinary incremental-counters, timer, gated timer, timer with divider (slow speed), etc. Note that both counters are set in encoder mode if CTM2 is set to 10. This makes counter 1 disabled for normal use.

Usage

Different counter / timer modes	
CTM2 = 1	Counter 2 (CN2) is not active
CTM2 = 2	Incremental counter (0-65535). Input 8 is counter input
CTM2 = 3	Timer. The timer frequency is 460800 Hz. If overflow, timer starts from 0.
CTM2 = 4	Gated timer. Input 8 must be high to keep the counter running
CTM2 = 5	Timer with divided frequency. Division ratio is setup by the CND2 command
CTM2 = 6	Similar to CTM2 = 5 but as gated timer. Input 8 is gate.
CTM2 = 7	Analogue output is enabled - see also the <i>AOUT</i> command
CTM2 = 10	Encoder mode. Inputs 7 and 8 are encoder inputs. CN2 contains the number of pulses in 32 bit.



3.4 Command Description

Example In an actual application there is a requirement to count an external event. For every 10 counts, program execution must be interrupted and an unconditional jump be made to an interrupt routine that drives the motor 10000 pulses. The program is as follows:

```
:START  CN2=65535-10  ;SET COUNTER REGISTER EQUAL 65525.
        CTM2=2        ;ACTIVATE COUNTER MODE
        .
        .
        (more program)

        INT8          ;BEGINNING OF INTERRUPT 8 ROUTINE
        CN2=65535-10  ;SET COUNTER REGISTER EQUAL 65525
        SR=10000      ;DRIVE MOTOR +10000 PULSES
        WAIT RS=0     ;WAIT HERE UNTIL MOTOR SPEED IS ZERO
        RETI          ;RETURN TO MAIN ROUTINE
```

Note !

When the counter (CN2) overflows from 65535 to 0, the interrupt routine INT8 is executed if this routine is included in the program memory.
If the command NSTOP=8 is included in a program at the same time as the counter overflows the motor will be stopped.

Example If it is required to monitor if a step motor runs the required number of steps, an encoder can be connected to I7 and I8. CN2 shows the number of pulses received multiplied by a factor of 4. CN2 can show numbers in 32 bit. An encoder giving 100 pulses/rev. is connected. This gives 400 pulses in CN2 for each revolution.

```
:INIT   CTM2=10        ;Encoder Mode.
        CN2=0          ;Zero-set encoder counter
        AP=0

:START  SR=400         ;Run one motor revolution
        Wait RS=0     ;Wait until motor is stopped
        If CN2 ≤ SR-5
        J:Error
        If CN2 ≥ SR+5
        J:Error
        J:Start

:ERROR  OUT1=1        ;Set output showing that motor is posi-
                    ;tioned wrongly
        SR=SR-CN2     ;Run the missing steps
        Wait RS=0
        AP=CN2       ;Correct position counter so it corresponds
                    ;to the number of pulses run.
```

3.4.27 Current Velocity (CV)

Command CV

Modes Stand by, Programming, Running

Description The motor velocity in RPM can be read at any time using this command.

Usage CV Show current velocity in RPM.

3.4 Command Description

3.4.28 Current Frequency (CVI)

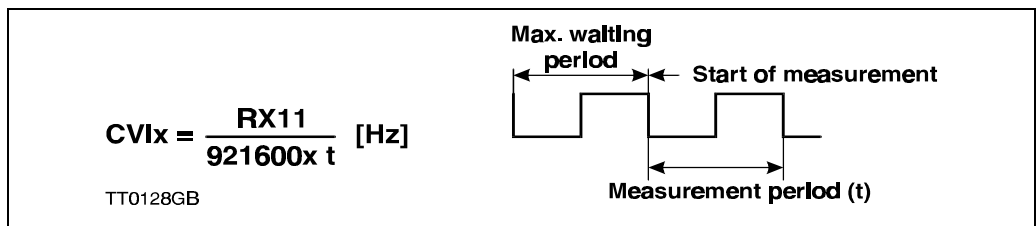
Command CVI

Modes Stand by, Programming, Running

Selection 1, 2, 3, 4, 7, 8

Description With this command it is possible to determine the frequency of a signal at digital inputs IN1, IN2, IN3, IN4, IN7 and IN8. When the command is executed, a high to low transition on the input is awaited, whereafter a timer is started which measures the following period time.

CVI is calculated according to the following formula:



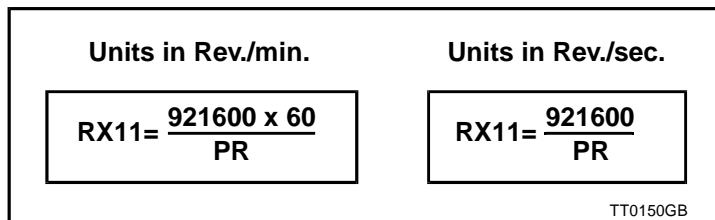
Using CB28 it is possible to select whether measurement should take place on fast frequencies, CB28=1, or on slow frequencies, CB28=0. Note that timer 1 (CN1) is used for measurement of time if CB28=0. (See *CB28 CVIx Frequency range*, page 116.)

Note: CVI 7 and CVI 8 does not work in encoder mode. (PIF=2 or CTM2=10)

CB28	Frequency Range	Formula	Default
0	1Hz to 7kHz	$CVIx = \frac{RX11}{921600 \times t}$	$CVIx = \frac{921600}{921600 \times t} = \frac{1}{t}$ CVI expressed in Hz
1	14Hz to 60kHz		

RX11 is set to 921600 in order to get CVI to be expressed in pulses/sec. RX11 can be changed if it is desired to have CVI expressed in another unit. TT0149GB

Unit	PR	RX11
Pulses/sec	-	921600
Rev./min.	2000	27648
Rev/sec.	500	1843



3.4 Command Description

CVI is best suited for measurement of frequencies below 50kHz. Measurement above this frequency will cause measurement errors. The measurement period will have a tolerance of ± 8 ms. Note that as the frequency is measured for one period only, the value shown will differ much if the frequency differs much from one period to the next. This is, for instance, the case if an encoder signal from a servo motor is measured.

If the frequency is too low, an error bit 41 will be set. CVI is then set to the VS value +1.

Example

```
RX11 = 921600      ;CVI CONTAINS VELOCITY IN PULSES/SEC.
:START CVI7        ;SHOW VELOCITY ON IN7
J:START           ;REPEAT
```

3.4.29 Delay (D).

Command D

Range 1 - 32000 (0.01 - 320 sec)

Modes Stand by, Programming, Running

Description The Delay command pauses program execution. The command character must be followed by a parameter value between 1 and 32,000 which specifies the Delay duration in 1/100 second.

Example D=27 - results in a delay of 0.27 seconds.

3.4.30 Global Execute (E)

Command E

Modes Standby, Programming, Running

Description The motor runs to the absolute position given by the SPT register. The *E* command is always received by the Indexer independent of the address. This enables many motors to be started precisely at the same time. This makes it possible to make linear interpolation between many axes with a high degree of precision.
See also *Set new global absolute position (SPT)*, [page 103](#).
The Indexer will respond to the *E* command regardless of whether addressing or checksum is used.

3.4.31 Else (ELSE).

Command ELSE

Modes Programming, Running

Description See *Control of program flow (IF)*, [page 74](#), and *IF statement*, [page 42](#).

3.4 Command Description

3.4.32 Error bit (ERR)

Command ERR

Modes Stand by, Programming, Running

Selection 0 - 47

Description The *ERR* (0-47) command can be used to test if an error bit has been set in the program. For example, a test can be made to determine whether there is an alarm from the servo driver (Flag ERR46). If this is the case, the program will stop.

The *ERR* command can also be used to change the Flag value by writing to the *ERR* register. Writing for instance ERR46=1, will cause bit 14 in error register ES2 to be set to 1. Thus a possible error routine can be tested for proper functioning without the error actually being present. When an ERR command is activated, it is possible to use it as an interrupt function by use of the INT 100-147 command. See the INT command.

3.4.33 Read-out of Error Status (ES)

Command ES

Modes Stand by, Programming, Running

Selection 0, 1, and 2

Description During operation of a system, various error conditions can arise. Some errors can be attributed to communication and set-up (error status register 0) and others attributed to hardware and motor control errors. The error status can be read using the *ES* (Error Status) command. The command invokes the Indexer to transmit a series of zeroes (0) and ones (1). A quick overview of error messages is thus obtained and can also be interpreted by other software programs. The separate *EST* command can be used to give an overview of text responses.

There are three error status registers.

ES0 : General errors caused by communication (RS232/RS485) or commands that have bad syntax.

ES1 : This register contains errors that have occurred while motor was running.

ES2 : This register contains errors that have been discovered under program execution. Bit 11-15 are critical errors that always are directly transmitted at the RS232/RS485 interface.

Register ES0 provides information about RS232/RS485 communication and set-up errors. This register accumulates and stores all errors that have occurred since the register was last read. When the register is read, the information is automatically erased.

3.4

Command Description

Error status bits, Register ESO

Bit no.	Error no.	Explanation	Error Flag
0	E0	No error	ERR0
1	E1	Error	ERR1
2	E2	Out of range	ERR2
32	E3	Number of parameters is wrong	ERR3
4	E4	Instruction does not exist	ERR4
5	E5	It is not an instruction	ERR5
6	E6	Parameter error or out of range	ERR6
7	E7	Register number error or out of range	ERR7
8	E8	Data cannot be saved in EEPROM	ERR8
9	E9	Checksum error	ERR9
10	E10	Parameter will be truncated	ERR10
11	E11	Limit switch activated	ERR11
12	E12	Limit switch active. Motor run command ignored	ERR12
13	E13	Reserved	ERR13
14	E14	Odd parity error in RS232 receive	ERR14
15	E15	Reserved	ERR15

Error status bits, Register ES1

Bit no.	Error no.	Explanation	Error Flag
0	E16	Command not allowed in standby mode	ERR16
1	E17	Memory full, (EEPROM)	ERR17
2	E18	Command not allowed. Program is running	ERR18
3	E19	Command not allowed in programming mode	ERR19
4	E20	Command not allowed. Motor is running	ERR20
5	E21	No program in RAM	ERR21
6	E22	Command ignored. Communication error on JVL BUS	ERR22
7	E23	Command ignored. Timeout error on JVL BUS	ERR23
8	E24	Command ignored. Unknown register/flag on JVL BUS	ERR24
9	E25	Command ignored. A and B connected wrong on JVL BUS	ERR25
10	E26	AC lower than 1. AC value changed to 1	ERR26
11	E27	AC higher than 1250000. AC value changed to 1250000	ERR27
12	E28	Internal AC=0. PIC value = 1	ERR28
13	E29	AOUT parameter out of resolution range	ERR29
14	E30	Warning. Wrong supply voltage	ERR30
15	E31	Reserved	ERR31

3.4 Command Description

Error status bits, Register ES2

Bit no.	E no.	Explanation	Error Flag
0	E32	VM specified lower than VS. VM value changed to VS	ERR32
1	E33	Error in CTM command parameter	ERR33
2	E34	Too many GOSUB, max 32	ERR34
3	E35	Program end	ERR35
4	E36	Too many ELSE after IF (ELSE)	ERR36
5	E37	Too many interrupts (INT)	ERR37
6	E38	Return without jump. (RET,RETI)	ERR38
7	E39	Warning!. Motor drive running	ERR39
8	E40	Address not allowed. Max. 255	ERR40
9	E41	Timeout on IN7/IN8. CVI command	ERR41
10	E42	Motorprocessor fault	ERR42
11	E43	O1-O8 Output error	ERR43
12	E44	RAM/EEPROM memory error	ERR44
13	E45	Fatal case error. Contact JVL!!	ERR45
14	E46	Error !. Alarm signal from motor drive	ERR46
15	E47	Unknown error	ERR47

The error indication is cleared after reading the error status. For critical (vital) errors, motor operation is interrupted and the error information remains in the register. The user must then either switch the system off and on again to reset the error status, or use the *RESET* command.

Note: Control bit CB8 will be 1 if there is one or more errors in the three error registers. (See also *Control Flags*, page 110, and *Error Messages*, page 123.)

Usage **ES0** Show error status register 0.

Example Sent to Indexer ES0
Received from Indexer ES0=0000000001000101

Note: bit 0 is the right-most bit.

3.4 Command Description

3.4.34 Error Status Text (EST)

<u>Command</u>	EST
<u>Modes</u>	Stand by, Running
<u>Description</u>	The <i>EST</i> command has exactly the same function as the <i>ES</i> command described above with the exception that the error status is reported as plain text for all three status registers. The <i>EST</i> command produces a list in English of the error status. If there are no errors, the error response is <i>E0: No errors</i> . A list of the error messages is given in <i>Error Messages</i> , page 123.
<u>Usage</u>	EST Read out error status register 0, 1, 2 as text.
<u>Example</u>	Send to indexer EST Received from indexer E0: No errors E2: Out of range E46: Error!. Alarm signal from motor driver

3.4.35 Error Status Text (ESTG)

<u>Command</u>	ESTG
<u>Modes</u>	Standby, Running
<u>Description</u>	The <i>ESTG</i> command can be used to print out last 15 errors which have occurred. The errors are stored in the permanent memory (EEPROM) and earlier errors are therefore stored even if the Indexer has been switched off. All errors above Error 7 will be stored. The command is typically used for example if a machine has periodical errors. It is then possible to determine which errors have occurred earlier. All the errors messages can be deleted using the <i>CB22=1</i> command (see <i>CB22 Diverse flag</i> , page 114).

3.4.36 Exit Programming Mode (EXIT)

<u>Command</u>	EXIT or PX
<u>Modes</u>	Programming
<u>Description</u>	The <i>EXIT</i> or <i>PX</i> command is used to exit Programming Mode and set the Indexer to Stand by Mode. A program can then be executed or a new program keyed in. If JVL's MotoWare software is used for programming the Indexer, this command should not be included in the program as MotoWare sends it automatically.

3.4 Command Description

3.4.37 Start Program execution (GO)

Command GO

Modes Stand by

Description Starts Program execution. The *GO* command can also be used to complete a programming sequence. The command can be used when the Indexer is in Stand by Mode. After the program has been started, the only way to stop it is by sending the *H* (Halt), *SH* (Soft Halt) or *STOP* command.

3.4.38 Halt of Motor and Program (H,K)

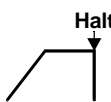
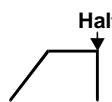
Command H,K

Modes Stand by, Program, Running

Description This command is used to stop the motor at once, regardless of velocity, deceleration etc. The Halt command has the highest priority since it stops program execution regardless of motor movement. The Halt command is effective immediately, i.e. as soon as the command is issued the Indexer is set to Standby Mode. To begin program execution once more, a new Execute command must be used. The program will start from the beginning. It is often necessary to use the *SZ* (Home) command before starting a new execution of a program since the motor position will be arbitrary owing to the instantaneous stop resulting from the Halt command.

To stop the motor without stopping the program execution, the *STOP* command can be used. (See *Stop motor (STOP)*, page 106, and *Smooth Halt of Motor (SH)*, page 102). Note also that the halt command will disable the high speed start input (CB20=0) permanently in the same manner as *RESET* or *SD* (set default). See also *CB20 High speed start trigger at IN1*, page 114.

Usage **H** Halt motor (and disable high speed start)

	Command given in Standby mode	Command given in Program Running
Motor		
Program	Stopped	Stopped

TT0130GB

3.4 Command Description

3.4.39 Control of program flow (IF)

Command IF

Modes Programming, Run

Description The *IF* command is used for comparison of 2 numeric values. These values may be the contents of registers such as *R1*, *IN1*, *VM* etc., or simply integer values such as 10500, 420, etc. All registers described in *Arithmetic expressions*, page 40, can be used in *IF* expressions. The comparison operator may be one of the following >, <, <>, =. It is also possible to use all the input designations, IN1 - IN8, NL, PL, HM, AI1 and AI2 directly in the expressions.

If the condition specified by the *IF* expression is fulfilled, the next line of the program is executed. If the condition is not fulfilled, the next line is omitted and execution continues from there. The *ELSE* command can be combined with the *IF* expression to provide more flexible possibilities.

Example 1:

```
:START IF R10 < 9800 ; If the content of register R10 is less than
      OUT1=1 ; 9800, activate output 1.
      J:PROG2 ; Jump to label PROG2.
```

Example 2:

```
      IF IN2=1 ; If Input2 is active.
      J:START ; Jump to label START.
      SP=100 ; Set position to 100.
:START SP=AP+50 ; Increase position by 50.
      IF AP=4000 ; If position has reached 4000 ...
      SP=100 ; ...Move to absolute position 100.
      ELSE ; Else ...
      SR=100 ; ...Move 100 pulses (relative) clockwise.
      WAIT RS=0
      J:START ; Jump to label START, where the position is increased.
```

The above program moves the motor 100 pulses and increases the motor position 50 pulses for each cycle until position 4000 is reached. Then the motor is moved back to position 100 and the cycle is repeated.

3.4 Command Description

3.4.40 Read Status of Inputs (IN)

Command IN

Modes Stand by, Programming, Running

Description The Indexer has 8 inputs. The status of these inputs can be read using the IN command as a single bit IN1 to IN8 or as an 8-bit value with a value from 0-255 (IN).

Usage IN Read inputs.

Example Sent to Indexer IN#
Received from Indexer IN=#00010100
Note that IN8 is the left-most digit (MSB)
Sent to Indexer IN
Received from Indexer IN=20
Sent to Indexer IN3
Received from Indexer IN3=1

IN can also be used in a program

Example If IN=1
OUT=1

If IN=#00001100
OUT8=1

If IN=#xxx101x ;Where x is immaterial
SR=10000

It is also possible to read other inputs than IN1 to IN8. For example, the analogue input can be read as a digital value directly.

IN9	IN10	IN11	IN12	IN13	IN14	IN15	IN16
NL	PL	HM	AI1	AI2	(SOK) Reserved	OE Output Error LED	Reserved

3.4.41 Read data from external module (INPUT) - Only SMI31 and SMC35B

Command INPUT

Modes Stand by, Programming, Running

Description The *INPUT* command is used to read-in data from external modules connected to the JVL bus RS485 interface. It can be used to read-in data from modules such as Keyboard, Display, thumbwheel, BCD data from PLC equipment, printer, extra inputs, digital-to-analogue modules, etc. All of the above-mentioned external modules are intelligent and will therefore contain registers whose contents can be read into the Indexer's registers using the *INPUT* command. The size and number of registers in external modules may vary, but each module has at least 1 register. Remember that all modules should have their own unique address.

3.4 Command Description

Format Rx = Input n1.n2

n1 Specifies the address of the external module from which input is required. The address parameter must be specified as a value between 0 and 31. The RS485 interface enables up to 32 modules to be connected to the interface. The address of each module must be set via DIP switches on the individual module.

n2 Specifies the register in the external module from which input is to be read. n2 must be specified in the range 0-255.

Example A JVL Input/Output Module IOM11 that has 16 inputs and 8 outputs is used. The Module address is 5. All 16 inputs are to be read and tested to determine if the value is 255. If this is the case, the module Counter is read and the program continues. In the instruction manual for the IOM11 module, the Counter register is specified as register 2 and the register for all 16 inputs is 3.

```
:READINP      R10=INPUT5.2      ;READ ALL 16 INPUTS AND TRANSFER
                                   ;CONTENTS TO R10
                                   IF R10=255      ;IF INPUTS NOT EQUAL TO 255 READ AGAIN
J:READ_COUNT
J:READINP      ;ELSE READ COUNTER VALUE AND CONTINUE
                                   ;PROGRAM
:READ_COUNT    R30=INPUT5.3      ;READ COUNTER AND TRANSFER TO R30
                                   R[R1]=R30      ;TRANSFER COUNTER VALUE TO AN ARRAY
                                   ;REGISTER USING R1 AS ARRAY POINTER
```

3.4.42 Read flag from external module (I) - Only SMI31 and SMC35B

Command I

Modes Stand by, Programming, Running

Description The *I* command is used to read-in data flags from external modules connected to the JVL bus RS485 interface. It can be used to read-in data from modules such a Keyboard, Display, thumbwheel, BCD data from PLC equipment, printer, extra inputs, digital-to-analogue modules, etc. All of the above-mentioned external modules are intelligent and will therefore contain flags whose contents can be read into the Indexer's registers using the *I* command. The size and number of flags in external modules may vary, but each module has at least 1 flag. Remember that all modules should have their own unique address.

Format Rx = I n1.n2

n1 Specifies the address of the external module from which input is required. The address parameter must be specified as a value between 0 and 31. The RS485 interface enables up to 32 modules to be connected to the interface. The address of each module must be set via DIP switches on the individual module.

n2 Specifies the flag in the external module from which input is to be read. n2 must be specified in the range 0-255.

Example A JVL Input/Output Module IOM11 is used and has address 2.

```
:START          IF I2.3=1          ;IF INPUT3 ON IOM11
J:RUN+          ;IS HIGH JUMP TO RUN+
                IF I2.4=1          ;IF INPUT4 ON IOM11
J:RUN-          ;IS HIGH JUMP TO RUN-
:RUN+          SR=100              ;RUN MOTOR 100 PULSES
                WAIT RS=0          ;WAIT UNTIL MOTOR STOPPED
```

3.4

Command Description

```

                J:START          ;JUMP BACK TO START

:RUN-          SR=0-100        ;RUN MOTOR -100 PULSES
                WAIT RS=0      ;WAIT UNTIL MOTOR STOPPED
                J:START          ;JUMP BACK TO START
    
```

3.4.43 Interrupt (INT)

Command INT

Modes Stand by, Programming, Running

Description The Indexer can monitor different inputs while the program is being executed. When a specified input condition is fulfilled, the Indexer finishes the command line that is under execution. Program execution is then continued from the program line where the corresponding *INT* command is inserted.

The following interrupts are available		
Command	Description	Trigger condition
INT1	Input 1 (I1)	Transition from logic 1 to 0. (CB12=0). Default Logic 0. (CB12=1).
INT2	Input 2 (I2)	Transition from logic 1 to 0. (CB13=0). Default Logic 0. (CB13=1).
INT3	Input 3 (I3)	Transition from logic 1 to 0. (CB14=0). Transition from logic 0 to 1. (CB14=1). Default
INT7	Counter 1 (CN1)	Overflow in counter 1 from 65535 to 65536
INT8	Counter 2 (CN2)	Overflow in counter 2 from 65535 to 65536
INT9	Input NL	Transition from 0 to 1. CB25=0. Default. Transition from 1 to 0. CB25=1.
INT10	Input PL	Transition from 0 to 1. CB26=0. Default Transition from 1 to 0. CB26=1.
INT100-147	ERR1-47	Error E1 to E47

Example An application is written so that user output *O8* will be activated for 100 ms if input 1 is activated. The program is as follows:

```

:START SR=100000 ; RUN THE MOTOR 100,000 PULSES
        WAIT RS=0 ; WAIT HERE UNTIL THE MOTOR HAS STOPPED
        D=10      ; DELAY 100ms
        J:START   ; JUMP TO THE PROGRAM START

        INT1      ; THIS INTERRUPT ROUTINE IS EXECUTED IF INPUT 1
                  ; IS ACTIVATED (I1=1)
        OUT8=1    ; SET OUTPUT 8 TO LOGIC 1
        D=10      ; DELAY 100ms
        OUT8=0    ; SET OUTPUT 8 TO LOGIC 0
        RETI      ; RETURN FROM INTERRUPT ROUTINE.
                  ; CONTINUE THE MAIN PROGRAM
    
```

Several inputs can be monitored at the same time.

If an interrupt is being executed while another interrupt occurs, the first interrupt will be finished before the new interrupt routine is executed.

The interrupt routines can be inserted at any point in the program. If an interrupt routine is executed in a program without any interrupt having occurred, program execution is halted and an error message will occur when the *RETI* is executed.

3.4 Command Description

3.4.44 Jump Command (J)

Command J

Range 0 - 2000

Modes Programming, Run

Description The *J* command is used to make an unconditional jump to a specified line number in the program.

The program line number can be specified in the range 0-2000. If MotoWare is used, a label can be inserted instead. See also *Jumping to program lines and the use of labels*, page 44.

Example:

```
0 OUT1=1           ;Activate output 1
1 SR=100          ;Run motor 100 pulses forward
2 OUT1=0          ;Deactivate output 1
3 WAIT RS=0       ;Wait until motor is finished
4 J0              ;Goto line 0
```

3.4.45 Jump-Sub command (JS)

Command JS

Range 0 - 2000

Modes Programming, Run

Description In contrast to the *J* command which jumps to a specified program line number, the *JS* command makes a jump to a program sub-routine.

When a *JS* command is executed, the Indexer first stores the number of the next line after the *JS* command and then goes to the line number specified by the *JS* command. When the *RET* (Return) command is encountered in the sub-routine, the program returns to the main program at the line immediately after the *JS* command and continues from there.

The *JS* command can be used up to 32 times corresponding to 32 nested sub-routines. Note that a sub-routine in the program always must be placed *after* the corresponding *JS* command. See also *Call of sub-routine*, page 44.

Example:

```
:
JS:SetOut1
:
:SetOut1OUT1=1;1ACTIVATE OUTPUT 1
D=10 ;WAIT 100MS
OUT1=0 ;DEACTIVATE OUTPUT1
RET ;RETURN TO MAIN PROGRAM
```

3.4 Command Description

3.4.46 Verify line number (LINE)

Command LINE

Modes Stand by, Programming, Running

Description The *LINE* command returns the line number of the program in the controller (not in the document). If the command is used in standby mode, it will return the line number for the last command executed.

Usage **LINE** Show line number

3.4.47 Macro functions (MAKRO)

Command MAKRO

Modes Standby, Programming, Running

Range 1 - xx

Description In cases where the Indexer is used for very special applications, the macro functions can be used. Macro functions are made upon customer request. Please contact JVL Industri Elektronik regarding this feature.

Usage **MAKRO = x** Start macro function x.

3.4 Command Description

3.4.48 Memory Checksum (MCHS)

Command MCHS

Modes Stand by, Programming, Running

Range 0-2.147.483.648

Selection 0-4

Description

Register	Memory	Description
MCHS0	Received on RS232	All bytes received on the RS232 interface are added in this checksum register. This makes it possible to check if the Indexer has received the data correctly. The checksum register is only reset under power up or after the command Program (start programming) has been received. The MCHS0 register can be set to 0 by sending the command <i>MCHS0=0</i> or simply by reading the MCHS0 register.
MCHS1	Temporary memory RAM	Here the checksum of the program which is in the temporary memory (RAM) is shown. It is only the checksum of the program itself which is calculated.
MCHS2	Permanent memory EEPROM	Calculates the checksum of the entire EEPROM, independent of the size of the program. This number will be an expression for the content of the program, which is in EEPROM. This register can for instance be used in the beginning of a program to check if the content of the EEPROM is ok or contains errors. Special parameters such as address, checksum, Baud rate etc. are part of the checksum calculation.
MCHS3	Program memory EPROM	Calculates the checksum for the program memory (EPROM)
MCHS4	Total RAM	The checksum is calculated for the entire RAM (32Kbyte)

Note:

The checksum is calculated by adding all bytes in a 32 bit number. Calculations of the checksum can take up to 15 sec. During this period it is not possible to communicate with the Indexer.

3.4.49 Show used memory (MEM)

Command MEM

Modes Standby, Programming, Running

Range 0 -100%

Description Use this command to verify how much memory has been used. When the *MEM* command is used, the Indexer will return a number indicating the used memory in bytes and the percentage free memory.

Example Sent to Indexer MEM
Received from Indexer MEM=292 byte used. 95% free.
This indicates that 292 bytes of the memory is used and 95% is still free.

3.4 Command Description

3.4.50 Recall Program (MR1)

Command MR1

Modes Stand by

Range 0 - 3

Description An Indexer program can be permanently stored in non-volatile EEPROM memory, i.e. without the need for current to retain the data. The Memory Recall command *MR1* is used to recall data from the EEPROM memory and set-up the Indexer and system using these values. It is also possible to recall the program automatically at power up. Note that each time the *MR1* command is used to load a program from permanent memory into working memory, any instructions in the Indexer's working memory will be erased.

Usage **MR1=x** Restore status:
0 = Do not recall program at power up.
1 = Recall program and execute at power up.
2 = Recall program now, but not at power up.
3 = Recall program now and at power up. After power up the program is executed.
4 = Recall program now and send it in a special format via RS232 to PC (MotoWare). MotoWare will then receive the program and save it on disk if required. The program can also be transferred to another unit using the special CB41 and PROGRAM command (from version 1.7)

MR1 Show restore status.

3.4.51 Recall Registers (MR2)

Command MR2

Modes Stand by, Programming, Running

Description The user registers R0 - R220 can be permanently stored in non-volatile EEPROM memory, i.e. without the need for current to retain the data. The Memory Recall Register command *MR2* is used to recall all 220 registers from the EEPROM memory.

Usage **MR2** Recall all user registers R0 to R220.

3.4.52 Save Program and user registers (MS)

Command MS

Modes Stand by

Description Saves program and user registers. Performs the same operation as if MS1 and MS2 were used simultaneously

3.4.53 Save Program (MS1)

Command MS1

Modes Stand by

Description An Indexer program can be permanently stored in non-volatile EEPROM memory, i.e.

3.4

Command Description

without the need for current to retain the data. The Memory Save program *MS1* command is used to store the Indexer program in the permanent memory. Only one program can be stored in permanent memory at a time. If the *MRI* register is set to 1, the program stored in permanent memory is automatically recalled and executed when the Indexer is switched on. **Note:** Storage should not be made more than 100 000 times to the EEPROM. See also: *Save user registers (MS2)*, page 83.

Usage

MS1 Save program in EEPROM

3.4 Command Description

3.4.54 Save user registers (MS2)

Command MS2

Modes Stand by, Programming, Running

Description The user registers R0 to R220 can be permanently stored in non-volatile EEPROM memory, i.e. without the need for current to retain the data. The Memory Save register *MS2* command is used to store the registers in the permanent memory. The predefined registers, for example VM, cannot be saved in permanent memory,

Usage **MS2** Save all registers R0 to R220 in EEPROM

Note: The MS2 command must not be used in a program in such a way that a register is repeatedly being stored. Writing more than 100 000 times to the EEPROM must be avoided.

3.4.55 Negative Limit Switch (NLS)

Command NLS

Modes Stand by, Programming, Running

Range 1=enabled or 2=disabled

Default 2 (disabled)

Description The PL and NL Inputs function as end-of-travel limits. If the motor is moving in a negative direction and NL is activated, the motor is stopped at once. The PL Input is the positive end-of-travel input. The NL input can be set to active high (NLS=1 and CB25=1), to active low (NLS=1 and CB25=0), or be disabled (NLS=2).
For connection of the end-of-travel inputs, see *End-of-travel Limit Inputs*, page 16.
Please note following limitation when using the limit switch (NLS=1)
If NLS= 1, it is not possible to use the NL input as a stop input in connection with the *NSTOP* command. Neither is it possible to use the input as an interrupt input. If one of these 2 modes is chosen, the limit function will have the highest priority.
If the limit input is activated when a motor run command is about to be executed, the command will be ignored and a bit will be set in the error status register.
Control bit 4 (CB4) will go high when the limit input is activated. The bit will first go low when a new motor run command is executed without the limit activated.
If the user wants to make a controlled limit with proper acceleration, it should be done with CB5=1 or with an interrupt program and the *INT* command. See *Interrupt (INT)*, page 77.

Usage **NLS = x** Set Negative Limit Switch to level 1 = high or 2 = disabled.
NLS Show Negative Limit Switch level.

3.4 Command Description

Desired function		Command
Limit switch	Transition from logic 1 to logic 0 stops motor	NLS=1, CB25=1
Limit switch	Transition from logic 0 to logic 1, stops motor	NLS=1, CB25=0
Normal input	Use NL as normal input	NLS=2. Default
Normal input	Use NL as interrupt input to INT9	NLS=2, INT9
Normal input	Use NL input as interrupt controlled smooth stop of motor	NLS=2, NSTOP=9

3.4.56 Interrupt controlled start (NSTART)

Command NSTART

Modes Programming, Running

Range 0 - 15

Default 0 (disabled)

Description This command must be used if timing is very critical for a certain motor start sequence. The *NSTART* command enables an interrupt feature that will detect a start signal at the same moment it occurs. All the inputs can be used, including the analogue inputs and limit inputs. The advantage of using this command instead of the *WAIT* command is the fast response time. In addition, precise repetition timing is obtained with this command. The typical response time for this command is 500 to 650 μ s. The *WAIT* command requires 0-5 ms. To obtain high noise immunity, the *NSTART* command always measures 16 times at the input signal before a start is realized. Using the CB18 flag it is possible to select between 4 trigger conditions. See *CB18 NSTART Trigger level flag*, page 113. If ultra high-speed Start/stop is required, see *CB20 High speed start trigger at IN1*, page 114, and *CB21 High speed stop trigger at IN2*, page 114. The following inputs can be used:

The following start conditions are available		
Input	Command	Trigger level
Disabled	NSTART=0 (default)	-
I1 (Input 1)	NSTART = 1	See CB18 CB18=0 logic 0 to1 CB18=1 logic1 CB18=2 logic 0 CB18=3 logic 1 to 0
I2 (Input 2)	NSTART = 2	
I3 (Input 3)	NSTART = 3	
I4 (Input 4)	NSTART = 4	
I5 (Input 5)	NSTART = 5	
I6 (Input 6)	NSTART = 6	
I7 (Input 7)	NSTART = 7	
I8 (Input 8)	NSTART = 8	
NL (Negative limit)	NSTART = 9	
PL (Positive limit)	NSTART = 10	
HM (Home input)	NSTART = 11	
AI1 (Analogue input)	NSTART = 12	
AI2 (Analogue input)	NSTART = 13	
SOK	NSTART = 14	
OE (Output error)	NSTART = 15	

3.4 Command Description

Usage **NSTART=1** Set input 1 as start input.

NSTART Show the actual start input condition.

Example

```

R1=10000      ;Register 1 equal to 10000
NSTART=6      ;Set input 6 as start input
:LOOP        SR=R1      ;The program will stop here until the
                    start input is activated
                WAIT RS=0 ;Wait until motor has stopped
R1=0-R1      ;Register1 = -Register1
J:LOOP       ;Jump to loop
    
```

The program will always stop at the loop line because of the SR statement, and will only continue if input IN6 is activated, i.e. goes from inactive to active.

3.4.57 Interrupt controlled stop (NSTOP)

Command NSTOP

Modes Programming, Running

Range 0 - 10

Default 0 (disabled)

Description This command must be used if the timing is very critical for a certain motor stop sequence. The *NSTOP* command enables an interrupt feature that will detect a stop signal at the same moment it occurs.

The command makes it possible to stop the motor as a function of an input condition.

Note that only Input 1 - 3 and NL and PL can be used in a stop condition.

The advantage of using this command instead of the *WAIT* or the *IF* command is the fast response time. In addition, precise repetition timing is obtained with the *NSTOP* command.

The typical response time for this command is 500 to 650 μ s. The *WAIT* or *IF* command requires 0-5 ms. If ultra high-speed Start/stop is required, see *CB20 High speed start trigger at IN1*, page 114, and *CB21 High speed stop trigger at IN2*, page 114.

The following stop conditions are available		
Input	Command	Trigger condition
disabled	NSTOP=0	none
Input 1 (I1)	NSTOP = 1	Transition from logic 1 to 0. (CB12=0). Default Logic 0. (CB12=1).
Input 2 (I2)	NSTOP = 2	Logic 0. (CB13=1). Transition from logic 1 to 0. (CB13=0). Default
Input 3 (I3)	NSTOP = 3	Transition from logic 0 to 1. (CB14=1). Default Transition from logic 1 to 0. (CB14=0).
Counter 1 (CN1)	NSTOP = 7	Overflow in counter 1 from 65535 to 65536
Counter 2 (CN2)	NSTOP = 8	Overflow in counter 2 from 65535 to 65536
Input NL	NSTOP = 9	Transition from logic 0 to 1. CB25=0. Default Transition from logic 1 to 0. CB25 =1.
Input PL	NSTOP = 10	Transition from logic 0 to 1. CB26=0. Default Transition from logic 1 to 0. CB26=1.

3.4 Command Description

Usage **NSTOP=1** Set input 1 as stop input.
NSTOP Show the actual stop input condition.

Example 1

```

VM=1000 ; SET VELOCITY TO 1000 RPM
NSTART=5 ; SET INPUT 5 AS START INPUT
NSTOP=2 ; SET INPUT 2 AS STOP INPUT
:START SR=10000 ; MOVE MAX 10000 PULSES WHEN INPUT 5 IS ACTIVATED
; WHEN INPUT 2 IS ACTIVATED THE MOTOR IS STOPPED
WAIT RS=0 ; WAIT UNTIL MOTOR HAS FINISHED RUNNING
OUT1=1 ; ACTIVATE OUTPUT 1
OUT1=0 ; DEACTIVATE OUTPUT 1
J:START

```

Example 2

```

NSTOP=1 ; SET INPUT 1 AS STOP INPUT
SR=1000 ; THE MOTOR WILL START IMMEDIATLY AND DECELERATE
; TO STOP WHEN THE STOP INPUT IS ACTIVATED
OUT1=1 ; NEXT COMMAND IS EXECUTED (SET OUTPUT 1 = 1)

```

3.4.58 Read/Set Status of Outputs O1 - O8 (OUT)

Command OUT

Modes Stand by, Programming, Run

Description The Indexer has 8 outputs. The status of these outputs can be read or set using the following *OUT* commands. When the status of the Outputs O1 - O8 is read, information is also given about the status of the 8 control LEDs.

Bit no.	Output	Special function
0	O1	
1	O2	
2	O3	
3	O4	
4	O5	Error output if there is an error in ES0, ES1 or ES2. See command CB7
5	O6	This output is active if the position is within the interval given in RX1 and RX2
6	O7	
7	O8	If CB17 is activated, then output 8 is active parallel with the Run LED

Usage **OUT** Read status of outputs

OUT n Read status of output n (n=1 to 8)

OUT n=x Set output n to x (0 or 1)

OUT# =xxxxxxx Set all outputs to x, where x is 0 or 1. (To be used in Standby mode only)

OUT = x Set all 8 outputs to decimal value x (x= 0-255)

3.4 Command Description

<u>Examples</u>	Sent to Indexer	Out#=1010	Sets outputs to 00001010
	Received from Indexer	Y	All digits to the left of msb will be set to 0
	Sent to Indexer	OUT#	Read outputs
	Received from Indexer	OUT#=00001010	Note bit 0 is the rightmost digit (LSB)
	Sent to Indexer	OUT	
	Received from Indexer	OUT=10	
	Sent to Indexer	OUT=255	Sets all outputs to 1
	Received from Indexer	Y	
	Sent to Indexer	OUT3=1	Sets O3 to 1
	Received from Indexer	Y	

3.4 Command Description

3.4.59 Pulse Input Format (PIF)

<u>Command</u>	PIF
<u>Modes</u>	Stand by, Programming, Running
<u>Range</u>	1 = Normal or 2 = Encoder
<u>Default</u>	1 (normal)
<u>Description</u>	This command determines the input format of inputs 7 and 8. If the format is set to 1 (PIF = 1) inputs 7 and 8 function as normal inputs. If the format is set to 2 (PIF = 2) inputs 7 and 8 are dedicated as encoder inputs. See also <i>Counter mode (CTM2)</i> , page 65

3.4.60 Positive Limit Switch (PLS)

<u>Command</u>	PLS
<u>Modes</u>	Stand by, Programming, Running
<u>Range</u>	1=enabled or 2=disabled
<u>Default</u>	2 (disabled)
<u>Description</u>	<p>The PL and NL Inputs function as end-of-travel limits. If the motor is moving in a positive direction and PL is activated, the motor is stopped at once. The NL Input is the negative end-of-travel input. The PL input can be set to active high (PLS=1 and CB26=1), active low (PLS=1 and CB26=0), or disabled (PLS=2).</p> <p>For connection of the end-of-travel inputs, see <i>End-of-travel Limit Inputs</i>, page 16.</p> <p>Please note the following limitations when using the limit switch (PLS=1)</p> <p>If PLS= 1, it is not possible to use the PL input as a stop input in connection with the <i>NSTOP</i> command. Neither is it possible to use the input as an interrupt input. If one of these 2 modes is chosen, the limit function will have the highest priority.</p> <p>If the limit input is activated when a motor run command is being executed, the command will be ignored and a bit will be set in the error status register.</p> <p>Control bit 4 (CB4) will go high when the limit input is activated. The bit will first go low when a new motor run command is executed without the limit switch being activated.</p> <p>If the user wants to make a controlled limit with proper deceleration, this should be done with CB5=1 or with an interrupt program and the <i>INT</i> command. See <i>Interrupt (INT)</i>, page 77.</p>

Usage **PLS = x** Set Positive Limit Switch to level 1=high or 2=disabled.

PLS Show Positive Limit Switch level.

Desired function		Command
Limit switch	Transition from logic 1 to logic 0, stops motor	PLS=1. CB26=1
Limit switch	Transition from logic 0 to logic 1, stops motor	PLS=1. CB26=0
Normal input	Use PL as normal input	PLS=2. Default
Normal input	Use PL as interrupt input to INT10	PLS=2, INT10
Normal input	Use PL input as normal interrupt controlled smooth stop of motor	PLS=2, NSTOP=10

3.4 Command Description

3.4.61 Pulses per revolution (PR)

Command PR

Modes Stand by, Programming, Running

Range (50 - 20000) - see text

Description **SMI30 and SMC31:** To achieve correct velocity of the motor, the number of encoder pulses per revolution must be programmed. The value specified must be the resolution specified for the encoder.

Note that many drivers internally multiply the number of encoder pulses by a factor 4, so that, for example, an encoder/motor with a resolution of 500 pulses per revolution effectively has a resolution of 2000 pulses per revolution. If the motor is to rotate 1 revolution, the positioning command must be based on the effective resolution of 2000 pulses.

Note that in a typical step motor system, *PR* must be set to 200 for fullstep, and 400 for halfstep operation. (See also *Connection to Yaskawa servo drives*, page 140). Default value is *PR*=8192.

SMC35: The following table shows the step resolutions that are available. The values "per revolution" are based on a standard motor with 200 steps per revolution. If motors different from 200 steps per revolution are used see RX 16

Type	Mini steps/full step	Mini steps/revolution when 200 steps/rev. motor is used
SMC35x	1, 2, 4, 8	200, 400, 800, 1600

Use of higher step resolution minimises mechanical resonances and thus provides optimum motor torque throughout the entire range of velocity. Note that the motor's resonances/torque are also heavily determined by the supply voltage. At high supply voltages, optimum torque is achieved at high rates of revolution.

It is not recommended to change step resolution when the motor is running. Default value is 8 ministeps/full step or *PR*=1600. Contact JVL if a different step resolution is required.

Usage **PR** = x Set pulses per revolution

PR Show pulses per revolution.

3.4.62 Print to external module (PRINT) - Only SMI31 and SMC35B

Command PRINT

Modes Stand by, Programming, Running

Range Address 0-31, Register 0-65535, Value 0-65535 (or text)

Description The Print command is used to print out the contents of registers to external modules. At present, print-out to 4 external modules is possible: to a PC via the RS232 interface and to JVL DIS10, KDM10 and IOM11 Modules via the JVL bus RS485 interface. Remember that all modules should have their own unique address.

Command Format : PRINTn1.n2.n3

3.4 Command Description

- n1 Specifies the address of the module to be printed to (1-31). Address 255 is reserved for a PC.
- n2 Specifies the register or cursor position to be printed to in the external module.
- n3 Specifies the register, numeric value or text string in the Indexer to be printed.

Example 1: PRINT1.0.R23

Prints the contents of register R23 to the module whose interface address is 1. Since transmission via the RS485 interface is balanced, it is possible to locate external modules up to 500 metres from the Indexer.

Example 2: PRINT255.0."TEST"

Prints the text "TEST" to a PC via the RS232 interface. Address 255 is reserved as the address for PCs. Note that the *PRINT* command can be used to print out register contents at run-time. It is especially well-suited for debugging a program. If JVL's *MotoWare* program is used, once the Indexer program has been transferred, the online feature can be used to display when a *PRINT* command is executed at run-time.

Example 3: PRINT3.41."Key in Value: "

When a Keyboard-Display Module KDM10 is incorporated in a system, it is often desirable to display information to the user. The above example illustrates how text can be written to the module's LCD display. In the example, the address of the module is 3. The second parameter value is cursor position 41, which is the first character on line 2 of the display.

Example 4:

```
R1=5555           ;ASSIGN A VALUE OF 5555 TO REGISTER R1
R30=333          ;ASSIGN A VALUE OF 333 TO REGISTER R30
PRINT5.41.R1     ;PRINT THE CONTENTS OF REGISTER R1 TO CURSOR
                 ;POSITION 41 OF A KDM10 MODULE WITH ADDRESS 5
PRINT2.0.R30     ;PRINT THE CONTENTS OF REGISTER R30 TO THE DISPLAY
                 ;OF A DIS10 MODULE WITH ADDRESS 2
```

When external modules DIS10 or KDM10 are used in a system, it is often necessary to print out the contents of register on the displays of the modules. As illustrated in the above example, this is best accomplished using the *PRINT* command to print the contents of a register either to a cursor position or directly to the LED display of the DIS10 module. Note that the figure printed must be within the interval 0 to 65.535.

3.4.63 Set the Indexer in programming mode (PROGRAM)

Command PROGRAM

Modes Stand by

Description The *PROGRAM* command sets the Indexer to Programming Mode, i.e. so that the Indexer is ready to receive programming instructions. Each time the program command is used, the content of the Indexer's working memory are reset, erasing any existing instructions. It is recommended that the command *RESET* is used before the *PROGRAM* command. If the *MotoWare* software is used for programming, this command should not be included in the program as *MotoWare* sends it automatically.

3.4 Command Description

3.4.64 User Register (R)

Register R

Modes Standby, Programming

Selection 0-220

Range -2147483648 to +2147483647

Description The Indexer includes 220 32-bit registers which can be used freely in a program. A register can be assigned a value, be included in an equation, etc.

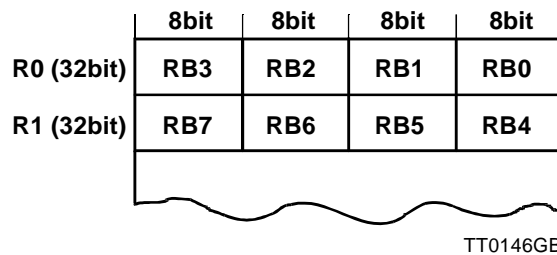
Usage **Rx=v** Set register x to the value v
Rx Show the value of register x

Examples R1=VM+100
R67 ;show the value of register 67 on the RS232

3.4 Command Description

3.4.65 User Register (RB)

<u>Register</u>	RB
<u>Modes</u>	Standby, Programming
<u>Selection</u>	0-880
<u>Range</u>	-127 to +128
<u>Description</u>	The Indexer include 880 8-bit registers which can be used freely in a program. The registers can be assigned a value, included in an equation, etc. The RB register uses the same memory space as the R register.



Example R1=65535 results in
RB4 and RB5 each has the value -127. RB6 and RB7 each has the value 0.

3.4.66 Reset Indexer (RESET)

<u>Command</u>	RESET
<u>Modes</u>	Stand by
<u>Description</u>	If a system overload occurs, the system must be reset before motor control is possible again. The <i>RESET</i> command has the same effect as turning the Indexer off and then on again. No communication with the Indexer until 2 sec. after the Reset command has been sent is allowed.
<u>Reset</u>	RESET Reset Indexer.

3.4.67 Return from subroutine (RET)

<u>Command</u>	RET
<u>Modes</u>	Programming, Running
<u>Description</u>	If the <i>JS</i> (jump subroutine) command has been used, the program will jump to a sub-routine. When this routine is finished, the <i>RET</i> command must be executed so that program execution can continue at the line just after the <i>JS</i> command. See also the <i>Jump-Sub command (JS)</i> command, page 78.
<u>Usage</u>	RET Return from subroutine.

3.4 Command Description

3.4.68 Return from interrupt (RETI)

Command RETI

Modes Programming, Running

Description If the *INT* command has been used, the program will jump to the INT routine when a specified input is activated. When this routine is finished, the *RETI* command must be executed so that program execution can continue at the line just after it was interrupted. See also the *Interrupt (INT)* command, page 77.

Usage **RETI** Return from interrupt routine.

Example

```

VM=1000      ; SET VELOCITY TO 1000 RPM
:START WAIT I8=1 ; WAIT HERE FOR START SIGNAL AT INPUT 8
SR=100000   ; RUN A RELATIVE DISTANCE OF 100000 PULSES
WAIT RS=0   ; WAIT HERE UNTIL MOTOR HAS STOPPED
J:START     ; JUMP TO START AND REPEAT PROGRAM
.
INT2        ; THIS INTERRUPT ROUTINE IS EXECUTED IF INPUT 2
            ; IS ACTIVATED
WAIT RS=0   ; WAIT HERE UNTIL MOTOR HAS STOPPED
VM=2000     ; SET SPEED TO 2000 RPM
RETI        ; TERMINATE INTERRUPT ROUTINE AND CONTINUE FROM
            ; THE PROGRAM LINE WHERE THE INTERRUPT
            ; OCCURED
    
```

3.4.69 User Register (RI)

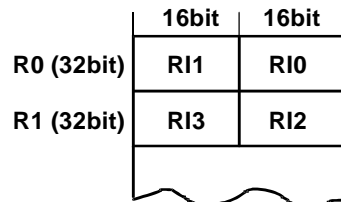
Register RI

Modes Standby, Programming

Selection 0-440

Range -32768 to +32767

Description The Indexer include 440 16-bit registers which can be used freely in a program. The registers can be assigned a value, be included in an equation, etc. The registers use the same memory space as the R registers.



TT0147GB

Usage **RIx=v** Set register x to the value v
RIx Show the value of register x

Example R1=65535 results in
 RI2 has the value 65536 and RI3 has the value 0

3.4 Command Description

3.4.70 Report Motor Status (RS)

<u>Command</u>	RS
<u>Modes</u>	Stand by
<u>Range</u>	0 - 7
<u>Description</u>	<p>During operation, the system can report information about the status of the motor (stationary, running, etc.) using the RS command.</p> <p>The command can be used to ensure that a new positioning command is not executed while a motor movement is in progress.</p>
<u>Usage</u>	<p>RS Motor Status:</p> <ul style="list-style-type: none">0=stationary1=accelerating2=Running at max. speed3=decelerating4=Motor not in position. (COIN=1)5=Fatal error in connected servo/step driver. (SALA=1)6=Motor zero searching7=Motor stationary but NL or PL active. See CB 38.
<u>Example</u>	<pre>:START VM=1000 ; SET TOPSPEED TO 1000 RPM WAIT RS=0 ; WAIT HERE UNTIL LAST POSITIONING IS FINISHED SP=10000 ; RUN MOTOR TO POSITION 10000 J:START ; RETURN TO START</pre>
<u>Notes</u>	<p>If RS=4 or RS=5 occurs while the motor is running, the Indexer will continue to generate pulses as if nothing has happened. These pulses will be lost as the driver returns an error message. If the RS=4 or RS=5 error lasts repeatedly up to 500 times, the Indexer will give an E39 or E46 error message respectively.</p> <p>If RS=4 or RS=5 occurs before a motor command is executed, the Indexer will give an E39 or E46 error message respectively. If RS=5 the program will also stop.</p> <p>If the servo alarm is activated while the motor is running, pulse generation will cease first when an attempt to start the motor again is made. This is due to the fact that RS is updated when RS is read or prior to executing a motor command. If it is required that RS is updated once for each line, the CB35=8 command is used.</p>
<u>Example</u>	<p>Execution of a special program is required when error conditions occur. The program below can be used.</p> <pre>:START VM=1000 ;SET VELOCITY TO 1000 RPM SR=10000 ;RUN A RELATIVE DISTANCE OF 1000 pulses :WAIT IF RS=4 ;IF SERVO DRIVE NOT IN POSITION J:ERR39 ;... JUMP TO ERROR ROUTINE IF RS=5 ;IF FATAL ERROR IN DRIVER J:ERR47 ;...JUMP TO ERROR ROUTINE IF RS>=1 ;IF MOTOR ERROR OR RUNNING J:WAIT ;...JUMP TO TRY AGAIN D=200 ;NO ERROR. PROGRAM CONTINUE HERE J:START ;JUMP BACK TO START :ERR39 PRINT255.1."ERROR 4" ;Print ERROR4 on RS232 J:WAIT :ERR47 PRINT255.1."ERROR 5" ;Print ERROR5 on RS232 J:WAIT</pre>

3.4 Command Description

3.4.71 Report Motor/Program Status in text (RST)

Command RST

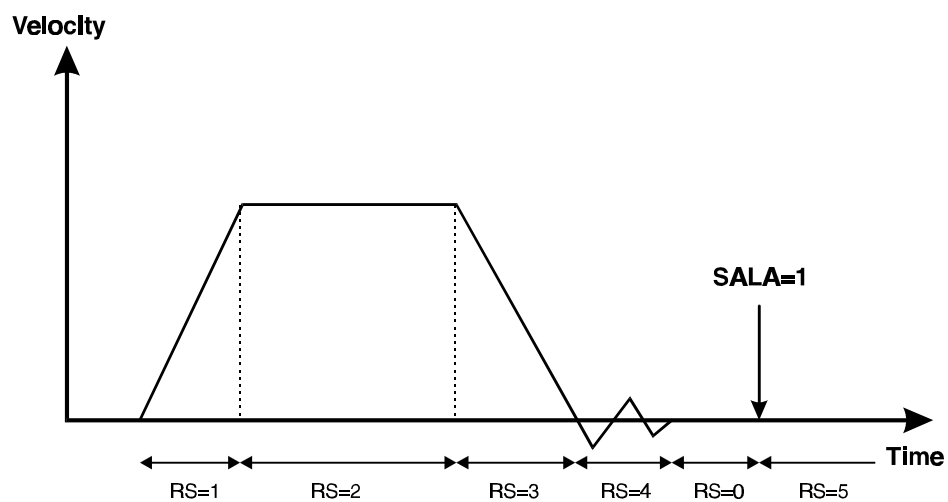
Modes Stand by, Programming, Running

RS	Description	RST Motor Status
0	Motor stopped	Stationary
1	Motor accelerates	Accelerating
2	Motor running at max. speed	Running at max. speed
3	Motor decelerating	Decelerating
4	Servo not in position. COIN=1	Motor not in position
5	Servo alarm. SALA=1	Fatal error in connected servo/step driver
6	SZ command execution	Motor zero seek in progress
7	Limit inputs active	Motor stationary but NL or PL active

RST Program mode	Explanation
Standby	Program not running. Keyed-in commands will be executed immediately
Programming	The Indexer is in programming mode
Running	The Indexer is executing a program

Notes: If *RST* is included in a program, a response will be returned on the RS232 interface when commands are executed in the program

Example Sent to Indexer: RST
 Received from Indexer: Motor Status: Accelerating
 Program Mode: Running



TT0172GB

3.4 Command Description

3.4.72 Special user registers RX

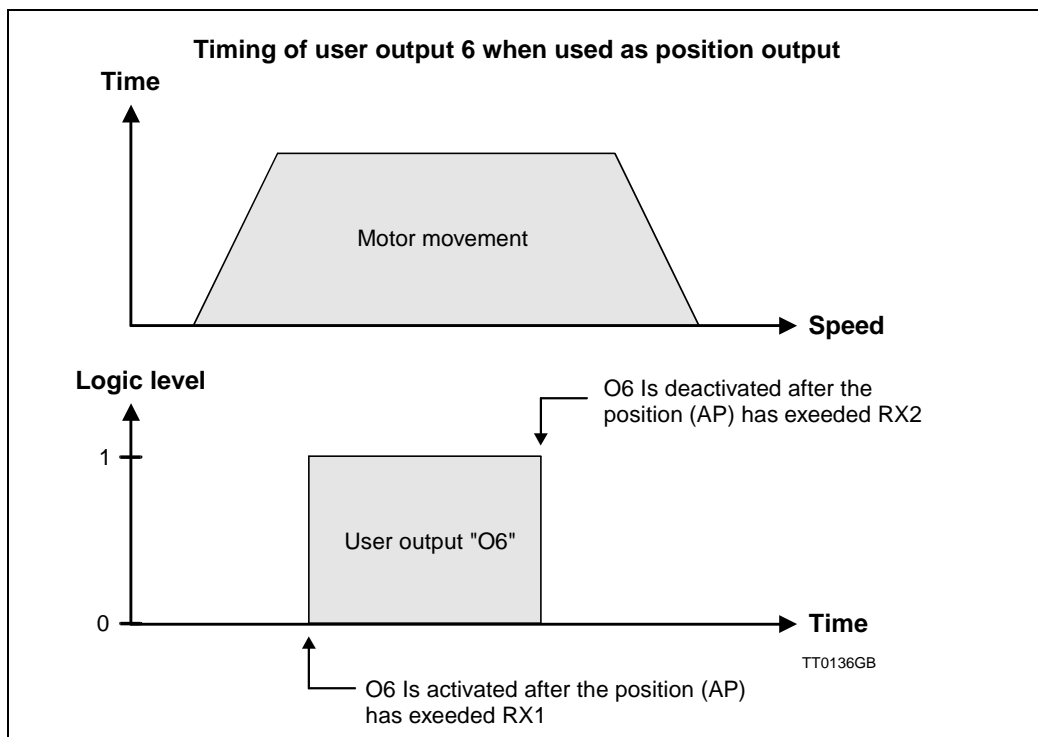
Command RX

Modes Stand by, Programming, Running

Selection 1-16

Description **RX1 / RX2** High-speed interval output at O6.
 For purposes where an external signal is required when the motor reaches a certain distance, these 2 registers can be used.
 The RX1 register specifies when user output 6 (O6) is set high and RX2 specifies when the output is set low. The function is only enabled when RX2 is higher than RX1.
 The delay time from the internal position counter to the output is less than 100µs.

RX1 / RX2	Status
RX1 = RX2	High speed output disabled - O6 used as normal output
RX1 > RX2	High speed output disabled - O6 used as normal output
RX1 < RX2	High speed output enabled - Active when RX1 < (APP) < RX2



Note that the specified positions in RX1 and RX2 refer to where the motor started and not the AP register

Example 1

An application requires that user output 6 must be activated after 500 pulses and deactivated after 1000 pulses. Therefore the following program is made:

```
:START AP=12345 ;SET POSITIONCOUNTER EQUAL TO 12345
;NOTE THAT THE POSITIONCOUNTER (AP) IS NOT
```

3.4

Command Description

```

;USED BY THE RX1/RX2 FEATURE IT IS ONLY THE PULSE COUNT
;FROM THE POINT WHERE THE MOTOR STARTS.
RX1=500 ;SET THE DISTANCE WHERE O6 MUST BE SET HIGH
RX2=1000 ;SET THE DISTANCE WHERE O6 MUST BE SET LOW
SR=5000 ;LET THE MOTOR RUN 5000 PULSES
;NOW THE MOTOR WILL START RUNNING AND AFTER
;RUNNING 501 PULSES, O6 WILL BE SET HIGH
;WHEN THE MOTOR HAS MOVED 1001 PULSES FROM THE
;START, O6 WILL BE SET LOW.
```

RX3 Delay of High Speed Start

For purposes where the high speed start must be delayed until a certain distance is achieved, the RX3 register can be used. RX3 can have values from 0 to 65535.

A 2-channel encoder must be connected to IN7 and IN8. If RX3 is set to a value higher than 0, the encoder pulses will be connected to a counter that increments each time a transition happens at IN7 or IN8.

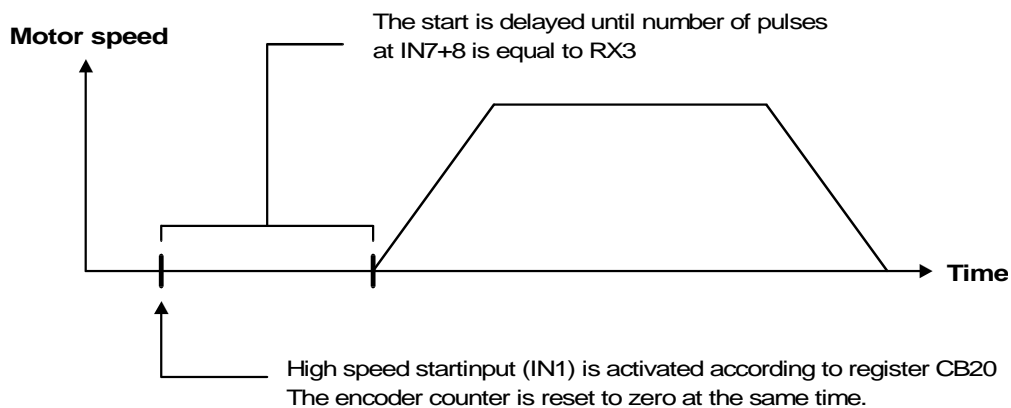
When a high speed start is recognized at IN1 according to CB20, the counter is reset and will start counting. When the counter reaches the value specified in RX3, the motor is started and is running.

Note that RX3 can be changed at any time. A change of RX3 will not have any effect on the counter.

Only a stop will reset the counter to zero. A stop could be one of following conditions.

1. Motor reaches final position
2. The halt (H) command or the soft halt (SH) command is executed.
3. The stop (STOP) command is executed.

Note that the RX3 feature only can be used together with high speed start !



TT0155GB

RX4 IN1 high speed start digital filter

When IN1 is used as high speed start together with CB21, RX4 is used to specify a debounce period. If $RX4 = 5$, the input should be stable for $5 \times 102 \mu s = 510 \mu s$ before the motor is started.

RX5 IN2 high speed stop digital filter.

When IN2 is used as high speed stop together with CB22, RX5 is used to specify a debounce period. If $RX5 = 10$, the input should be stable for $10 \times 102 \mu s = 1.02 \text{ms}$ before the motor is stopped.

RX7, RX8, RX9, RX10 Monitoring of distance travelled

This feature can be used in situations where "High speed start and stop" is used and the distance travelled by the motor is to be monitored. Typical applications include label dispensing. See appendix for a flowchart and program example.

Note that Input 4 (IN4) is used as a secondary start input when the function is active.

The function monitors 2 parameters:

1. How far the motor runs before it sees the stop input (IN2).
This is determined by the sum of the pulses in RX7 + RX8.
If this sum is exceeded, output 6 is activated for the duration in mS determined by RX10.
2. How long the stop signal is active.
This duration is determined by the sum of the pulses in RX8 + SR2 (SR2 = run-length after stop signal).
If this sum is exceeded, output 7 is activated for the duration in mS determined by RX10.

Register description:

RX7 Contains the nominal length of operation.

RX7 can be specified in the range 0 - 16000000.

RX8 Contains the nominal length that the stop sensor must be active.

RX8 can be specified in the range 0 - 16000000.

RX9 Specifies whether the monitoring function will be active or not.

RX9=0 (default) Monitoring inactive. RX7, RX8 and RX10 have no effect.

RX9=1 Monitoring active.

RX10 Specifies the duration of the pulse when an error occurs on either output 6 or output 7. The default duration is 1 msec.

The value assigned to RX10 is specified in steps of 0.1024 ms. For example, if a duration of 10 ms is required, RX10 is assigned a value of $10 / 0.1024 = 98$. RX10 can be specified in the interval 0 - 65000 (ca. 6.5 seconds)

Automatic adjustment of RX7 and RX8

If IN4 is activated, the motor will start and measure the length of operation until the stop sensor is activated the first time. This value is assigned to RX7. The motor will continue to run and begin measurement of the length that the stop sensor is active. When the stop sensor is again passive, the measured value is assigned to RX8. See appendix for further description, time history and program example.

RX11 Frequency multiplier in CVI command.

Used in CVI command as frequency multiplier. See CB28 flag in *Current Frequency (CVI)*, page 67.

RX12 Max. pulses/rev. in turntable mode.

Used in turntable mode as max. number of pulses per revolution. See CB30 flag in *CB30 Modula Mode (Turntable mode)*, page 116.

RX13 Multicontrol (only applicable from version 1.8)

Using this mode it is possible to perform advanced multicontrol of up to 32 axes via a single program. The MotoWare program is transferred to the master SMI31 and when executed, commands are sent via the RS485 interface to other JVL slave controllers, e.g. SMI3x, DMC10, AMCxx etc. The slave controller addressed is selected via the RX13=x command. Slave-controllers must be able to interpret JVL language, e.g. SR=100, VM and RS commands.

3.4

Command Description

In multicontrol mode, the master must always have address 0 and each slave must be assigned its own unique address in the range 1-31.

The master is set in multicontrol mode by specifying RX13 = address of the "slave". In this way a selected number of registers and commands on the slave controller become accessible to the master controller, with the same names as on the slave controller. The various slave controllers are selected by changing the slave address, e.g. RX13=4 to select slave controller with address 4. In order to transfer registers from the slave and perform calculations on the master controller, registers R1-R99 are reserved on the master. If a register value is to be retrieved from a slave controller, it must be from a register greater than R99. Communication is carried out via the standard RS232 interface and it is therefore possible to monitor communication between the master and slave controllers via MotoWare's on-line window.

To use the master controllers own 8 user outputs and a motor, RX13 must be set to 0, which is the factory default setting. Any slave controller can be addressed by using the corresponding address and RX13=1 to 31.

Example 1 RX13=1 ;Select slave with address 1 (Addr=1)
 R1=R110 ;Transfer the value of R110 from the slave with
 address 1 to master register R1

Example 2 RX13=3 ;Select slave with address 3 (Addr=3)
 SR=R1 ;The value of master register R1 is sent to
 slave with address 3 and the slave runs length R1
 OUT=255 ;The value 255 is transferred from the master to
 the 8 user outputs on the slave with address 3.
 All user outputs are set to "1" (255)

Example 3: An x-y-z table is controlled using a master SMI30 with master address 0, and 2 servo controllers with slave addresses 1 and 2 that are connected via the RS232 interface. Operating commands are sent to slaves 1 and 2 so that they operate simultaneously. When these operations are complete, the master indexer runs.

```
RX13=1           ;Select axis 1. Address 1
VM=500           ;Send velocity value to axis 1
SR=1000          ;Send relative operation length to axis 1. The
                  motor operates immediately.
RX13=2           ;Select axis 2. Address 2
VM=2000          ;Send velocity value to axis 2
SR=1000          ;Send relative operation length to axis 2
Wait RS=0        ;Wait until axis 2 operation is completed
RX13=1           ;Select axis 1. Address 1
Wait RS=0        ;Wait until axis 1 operation is completed
RX13=0           ;Select master indexer. Address 0
SR=-1000         ;Move relative 1000 in reverse
Wait RS=0        ;Wait until master motor operation is complete.
```

Care must be taken in transmitting data from a PC to a master or slave controller since communication errors may arise. If a PC attempts to transmit to a master indexer and for example RX13=1, the master will send the transmission to the slave (address 1) which will reply to the master, which in turn will reply to the PC. This should normally be avoided since there is a lot of communication on the RS232 and a risk of simultaneous transmission and reception.

A new program can be transmitted to the master, since MotoWare stops a program with repeated H (Halt) commands.

If during transmission of data, an error arises or unexpected commands are returned,

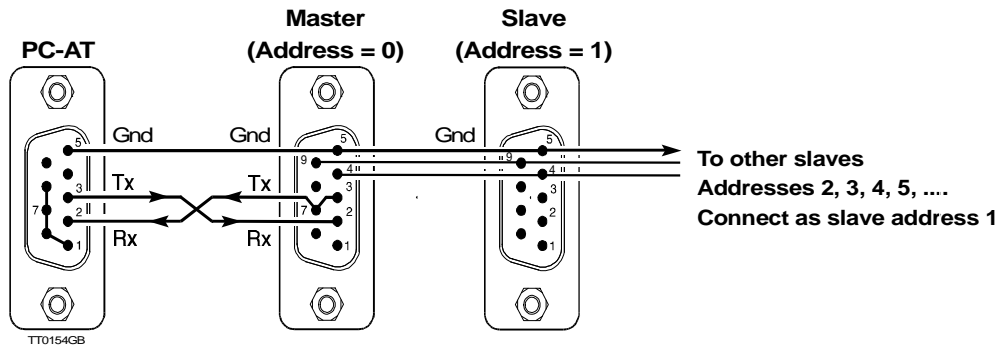
3.4

Command Description

re-transmission is attempted up to a maximum of 4 times. If incorrect data is received by all 4 attempts, error E22 is reported, the ERROR light is lit constantly and program execution stops. The error can be a parity error, no receipt of "Y" confirmation, or a time-out error.

If the master controller tries to transmit to a slave and the slave does not respond, the master controller will timeout after approximately 1 sec. Error E23 is reported, the ERROR lamp is lit constantly and program execution stops.

Note that the following commands cannot be used when RX13 is not 0 (i.e. when in multicontrol mode): RX, Makro, CON, all JVL-bus commands, SPT, MS, etc.



RX14 Lower limit for position check. See CB40 Default value is -2 147 483 648

RX15 Upper limit for position check. See CB40. Default value is +2 147 483 648

RX16 Selection of step/revolution on the motor. (SMC35 only)

Since all velocities and acceleration are specified in terms of r.p.m., it is important to specify the step/revolution of the motor. Normally this is 200 step/revolution, but 400 or 24 step/revolution for example are also very common. Specify RX16 in accordance with the following table.

RX16 Steps/rev	Degrees/ step	PR=			
		Steps/rev. in 1/1 step	Steps/rev. in 1/2 step	Steps/rev. in 1/4 step	Steps/rev. in 1/8 step
24	15	24	48	96	192
48	7.5	48	96	192	384
100	3.6	100	200	400	800
200	1.8	200	400	800	1600
400	0.9	400	800	1600	3200

The default setting is RX16=200 and PR=1600, corresponding to 1/8 step

Example

A HWI motor with 24 step/revolution is connected. and 1/2 step operation is required.

RX16 = 24

PR = 48

RX17 Diverse register

If RX17 is preset with a velocity and the LL1 command is executed, the speed will be changed faster than if VM=x is executed.

3.4

Command Description

3.4.73 System Default (SD)

Command SD

Modes Stand by

Description The *SD* command is used to recall the Indexer's factory default set-up. All user registers will be set to 0. Note that address, checksum, memory recall and Baud rate registers will not be changed. These can only be reset using of the ## command.

The factory default set-up is as follows:

AC=100	CB9, CB14, CB18=1	CTM2=1	SR=0
ACS=0	CB15, CB16=2	ES0, ES1, ES2=0	SR2=0
AP=0	CN1=0	NLS=2	VS=10
CB2, CB3, CB6, CB7	CN2=0	OUT=#00000000	VM=100
CB8, CB10, CB11, CB12,	CND1=1	PIF=1	
CB13, CB17, CB19, CB20,	CND2=8	PLS=2	
CB21, CB23, CB24, CB25,	CON=1.0000	PR=8192	
CB26, CB27, CB28, CB29,	CTM1=1	R1-R220=0	
CB30, CB31, CB32, CB33,	CB36=1	RX1, RX2=0	
CB34, CB35, CB37, CB38,		RX11=921600	
CB39, CB40=0		RX12= 8192	

Usage **SD** Recall factory default set-up

3.4 Command Description

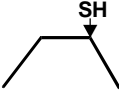
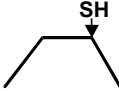
3.4.74 Smooth Halt of Motor (SH)

Command SH

Modes Stand by, Programming, Running

Description This command is used to perform a controlled halt of the system. The motor is stopped in accordance with the pre-programmed deceleration (acceleration).
If *SH* is used in a program, it will only stop the motor and not the program execution. Use the *H* command if the program must be stopped. See also the *Halt of Motor and Program (H,K)* command, page 73, and the *Stop motor (STOP)* command, page 106.

Usage **SH** Smooth halt of motor.

	Command given in Standby mode	Command given in Program Running
Motor		
Program	Stopped	Running

TT0131GB

3.4.75 Serial Number (SN)

Command SN

Modes Standby, Programming, Running

Range 0 - 65536

Description The serial number of the Indexer is shown using this command. The number cannot be changed. (Available from version 2.0 of the firmware only)

3.4.76 Servo on (SON)

Command SON

Modes Stand by, Programming, Running

Range 0 - 1

Description This command is used to enable or disable the connected servo or step motor driver. The command activates or deactivates the *SON* output placed at the 9 pole D-SUB connector called *Driver*. The *SON* output is a NPN output and is set low if the command *SON = 1* is executed.
ON the SMC35, *SON = 1* must be set in order to make the internal driver active. When the SMC35 is switched on, the motor will be completely without current and no voltage will be applied. If sensitive measuring equipment is located close to the motor, *SON*

3.4 Command Description

can be set to 0 while measurements are made. SON can also be set to 0 if for example the motor is to be rotated by some external force and therefore should not yield stationary torque.

Usage **SON=1** Set servo on (activate driver)
 SON=0 Set servo off (deactivate driver - motor is without current)

3.4.77 Set new absolute Position (SP)

Command SP

Modes Stand by, Programming, Run

Range -2147483648 to +2147483647 units or pulses

Description In Standby Mode and Programming Mode, the motor can be set to move to a new absolute position specified in terms of pulses or units if a conversion factor is used. (See also the *Conversion factor (CON) - Only SMI31 and SMC35B* command, page 60.)

Usage **SP = x** Move to new Position.
 SP Show new position.

Example Sent to Indexer SP=-1000 Move to absolute position -1000
 Received from Indexer Y
 Sent to indexer SP = 0-R1 Move to absolute position -R1

3.4.78 Set new global absolute position (SPT)

Command SPT

Modes Standby, Programming, Run

Range -2147483648 to +2147483647 units or pulses

Description When it is required to start several motors at the same time, the *SPT* is set to the absolute position. When the *E* command is received via the RS232 interface, all the motors will run to the position given by the *SPT* command. As the *E* command is received independently of address, several motors can thus be started simultaneously.

3.4 Command Description

3.4.79 Relative Positioning (SR)

<u>Command</u>	SR+, SR-, SR=n, SR=-n
<u>Modes</u>	Stand by, Programming, Running
<u>Range</u>	-2147483648 to +2147483647 units or pulses
<u>Description</u>	In Standby Mode and Programming Mode, the motor can be set to move a specified number of pulses in a positive or negative direction. For movement in a negative direction, the parameter value is specified with a minus sign.

The *SR* command is also used to move the motor continuously in a specified direction. The command is then followed by a + or - parameter which specifies the direction of the movement. To stop the motor once the *SR* command has been issued, a *SH* (Smooth Stop) or *H* (Halt) command must be used.

If the *NSTOP* (input set-up) command is used before the *SR* command, the conditions specified by the *NSTOP* or *CB21* command can also stop the motor. See the description of the Input Setup commands (*NSTOP*, page 85, and *NSTART*, page 84) for further details.

The Position Counter *AP* is updated while the *SR* command is executed. If an *SP* command is executed, the *SR* will hold the value that is calculated for the run.

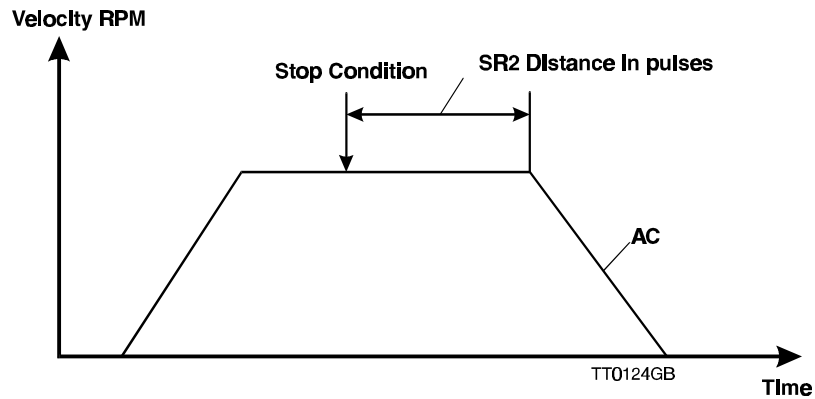
Usage **SR = x** Set relative position

<u>Example</u>	Sent to Indexer SR=5000 Move 5000 pulses in positive direction
	Received from Indexer Y
	Sent to Indexer SR=-5000 Move 5000 pulses in negative direction
	Received from Indexer Y
	Sent to Indexer SR- Move continuously in negative direction
	Received from Indexer Y

3.4.80 Relative Offset Positioning (SR2)

<u>Command</u>	SR2
<u>Modes</u>	Stand by, Programming, Run
<u>Range</u>	0 to 16.777.215 pulses
<u>Description</u>	The <i>SR2</i> command has the same function as <i>SR</i> but will first be executed after a Soft-halt has occurred. This soft halt (<i>SH</i>) can occur by sending the command <i>SH</i> via the RS232/RS485 interface or if it is implemented in a program. A soft halt can also be executed if the <i>NSTOP</i> command is used in a program, which means the motor is soft halted when a certain input is activated. When the soft halt has been detected, the motor continue running at the same speed (no deceleration) and moves the distance specified by <i>SR2</i> . Once the distance has been reached, the motor decelerates and stop.
<u>Usage</u>	SR2 = x Set relative offset distance
<u>Example</u>	Sent to Indexer SR2=5000 Move 5000 pulses in positive direction
	Received from Indexer Y

3.4 Command Description



Example 1

```
VM=1000      ;Velocity 1000 RPM
AC=1000      ;Acceleration 1000 RPM/sec.
SR2=10000    ;Relative offset distance
SR=1000000   ;Run maximum distance
WAIT IN1=1   ;Wait for stop input on IN1
SH           ;Run relative distance before stop
WAIT RS=0    ;Wait for motor to stop
```

Example 2

```
CB21=1      ;Enable the high speed stop on IN2
SR2 =1000   ;Relative offset distance
SR=10000    ;Run maximum distance
WAIT RS=0   ;Wait for motor to stop
```

3.4 Command Description

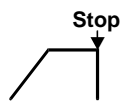
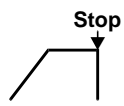
3.4.81 Stop motor (STOP)

Command STOP

Modes Stand by, Programming, Running

Description This command will cause the motor to stop. If the motor was running, it will be halted as when using the *H* command, but the program will not be stopped. See also the *Smooth Halt of Motor (SH)* command, page 102, and the *Halt of Motor and Program (H,K)* command, page 73.

Usage **STOP** Stops the motor.

	Command given in Standby mode	Command given in Program Running
Motor		
Program	Running	Running

TT0144GB

3.4.82 Seek Zero Point (SZ)

Command SZ+ or SZ-

Modes Stand by, Programming, Running

Description This command is used to reset the motor position to a known zero point.

The Seek Zero command enables an electrical and mechanical reset of the system to a pre-defined reference position. As soon as the Indexer receives the Seek Zero command, the motor will move in the specified direction, either SZ+ or SZ-.

As soon as the HM (End of Travel) input becomes low, the motor will stop. The motor is then at its reference position. The speed at which a reset occurs is determined by the VS (Start Rate) command.

During and after execution of a Seek Zero command, the Position Counter is reset to zero (AP=0). See also *Home Input*, page 17

Usage **SZ+** Begin zero point seek in positive direction.

Example

```

SZ+           ;Begin zero point seek
:WAIT IF CB4=1 ;If limit switch active before home
              ;switch...
J: ERRORPL   ;..jump to error handler
IF RS<>0     ;If motor is running and HM not active..
J: WAIT      ;...jump and test again
    
```

3.4 Command Description

3.4.83 Temperature (TP) (SMC35 only)

<u>Command</u>	TP
<u>Modes</u>	Stand by, running, running
<u>Description</u>	The TP command shows the actual temperature of the SMC35 Controller.
<u>Usage</u>	TP Show actual temperature
<u>Example</u>	Send to controller TP Received from controller TP=37 Indicating that the temperature of the controller is 37°C

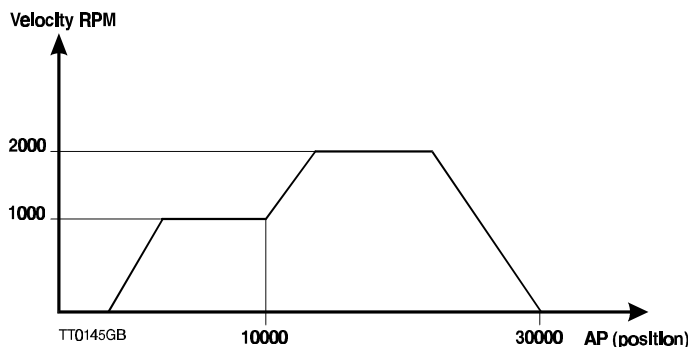
3.4.84 Firmware Version (VE)

<u>Command</u>	VE
<u>Modes</u>	Stand by
<u>Description</u>	The VE command provides information about the Indexer firmware version and date.
<u>Usage</u>	VE Show version and date.
<u>Example</u>	Send to Indexer VE Received from Indexer 19-01-1998 . V1.45/MCP2.0 Indicating that the firmware is from 19 January 1998, version 1.45, and the motion processor firmware is version 2.0 For example, R1=VE used in a program will transfer the number 19011998 to R1

3.4.85 Maximum Velocity (VM)

<u>Command</u>	VM
<u>Modes</u>	Stand by, Programming, Run
<u>Range</u>	0 - 65535 RPM
<u>Description</u>	The VM command is used to set the maximum velocity. The speed can be changed at any time regardless of whether the motor is running or not.
<u>Usage</u>	VM = x Set maximum velocity in rpm. VM Show current max. velocity
<u>Example</u>	AC=10000 ;set acc/dec to 10000 RPM/sec SP=30000 ;run to absolute position VM=1000 ;set velocity to 1000 RPM WAIT AP>=10000 ;wait until position reached VM=2000 ;change velocity to 2000 RPM WAIT RS=0 ;wait until motor has stopped

3.4 Command Description



3.4.86 Supply Voltage (VOL)

Command VOL

Modes Stand by

Range 8 - 100

Description The VOL command is used to check the voltage applied to the Indexer. Note that the supply should be in the range 10-32VDC for the SMI3x and 10-85VDC for the SMC35.

Usage VOL Show voltage in Volts

3.4.87 Start Rate (VS)

Command VS

Range 1- 10000 RPM

Modes Stand by, Programming, Running

Description The Start Rate is the speed at which the motor is started. The command is intended for use in step motor systems. In servo systems the start speed should normally be set to 10 RPM.

In a step motor system, the motor will simply stop at an arbitrary position if the start speed is set too high. The default Start Rate is 10 RPM.

3.4.88 Wait for condition (WAIT)

Command WAIT

Modes Programming, Running

Description Using this command it is possible to wait at a specific program line until a condition is fulfilled. It is possible to use all user registers, predefined registers and control bits.

Example Program execution is halted until input 1 is activated. Then the motor must run 100000 pulses with the velocity of 1000 RPM. When the position has passed the first 8000 pulses,

3.4

Command Description

the motor should accelerate up to 2000 RPM. After the motor has reached the final position (100000), it must return to zero position.

```
VM=1000          ; Velocity 1000 RPM
AP=0             ; Zero positionscounter
:START          WAIT IN1=1      ; Wait here for input 1
                SP=100000     ; run motor to position 100000
                WAIT AP>=8000 ; Wait here until position 8000 is
                                ; passed - then change velocity to
                                ; 2000 RPM
                VM=2000       ; Accelerate to velocity 2000 RPM
                WAIT RS=0     ; Wait here until motor is stopped
                SP=0          ; Return motor to zero position.
                WAIT RS=0     ; Wait here until motor is stopped
J:START          ; Jump to label START
```

The WAIT command cannot be recommended for time-critical tasks such as a rapid stop of a motor using WAIT IN1=1 followed by the SH command. Instead use NSTOP or CB21. The WAIT command functions in the way that the condition is checked first time and thereafter the program line is executed again and again until the condition is fulfilled. Up to 1ms to 4ms may therefore elapse before the next line, e.g. the SH command, is executed.

3.5 Control Flags

3.5.1 Control flags in general

In addition to the 221 user registers, the Indexer contains a number of control flags. These flags control some basic parameters in the Indexer.

For example, a flag can control whether a certain input should be activated at logic 1 or logic 0.

Also the resolution at the analogue inputs can be set using control flags.

Some of the flags can only be read. These flags show the status of different conditions during program execution — for example, in which direction the motor is moving or whether errors have been signalled in the error registers.

The following Control Flags are available.

3.5.2 CB1 Direction level flag. (Status flag)

By reading this flag the actual level of the direction output can be verified.

CB1=1 Positive direction signal to driver is logic 1 (5V nominal)

CB1=0 Negative direction signal to driver is logic 0 (0V nominal)

3.5.3 CB2 / CB3 Analogue conversion flags

Using these two flags it is possible to change the resolution of the two analogue inputs AI1 and AI2. The A/D converter of the analogue inputs is 8-bit, but by using an integration technique the resolution can be much higher. It is possible to select up to 14-bit resolution.

Note that at high resolution the execution time is much longer for commands that use the analogue inputs. The default is 0 for both flags.

CB3	CB2	Resolution	Measurements	Interval	Time (ms)
0	0	8 Bit (default)	16	0-16383	4.5
0	1	10 Bit	64	0-16383	18.0
1	0	12 Bit	256	0-16383	55.7
1	1	14 Bit	1024	0-16383	66.0

3.5.4 CB4 End of travel flag. (Status flag)

This flag is set to 1 if the end-of-travel inputs (*NL* or *PL*) have been activated.

The flag will be 1 until the motor has been running without any of the end-of travel inputs activated. Note that the function is active only if NLS and PLS is different from 2.

3.5.5 CB5 End of travel, deceleration

Using this flag it is possible to select whether the motor should decelerate in H or SH mode when a limit switch is activated (i.e. if NL or PL is activated)

0: Deceleration as if a halt (H) command is used. (Default).

1: Deceleration as if a smooth stop (SH) command according to AC is used.

3.5.6 CB6 Error at user output flag

This flag is set (CB6=1) if a fault occurs at the user outputs *O1-O8*.

The default for this flag is 0. The program and motor are stopped if the output is short-circuited. The CB6 flag will be set to 0 again if the EST or GO command is executed or a motor command SR and SP is executed.

3.5 Control Flags

3.5.7 CB7 Output error flag

A message in the error register EST will cause activation of output 5 (*O5*)
This feature can be used to give, for example, a PLC a message that something has failed.
Default for this flag is 0 (function is disabled).
CB7=0 *O5* is used normally. Nothing can set the output 5 except the *OUT* command.
CB7=1 *O5* is set by error (also the *ERROR* LED lights).

3.5.8 CB8 General error flag. (Status flag)

This flag is set to 1 if an error message appears in the error registers EST, ES0, ES1 and ES2. If the error message(s) is read using the *EST* command, the flag is cleared after the register is empty (No errors). The flag is intended to be used as a quick reference to see if an error has occurred. Default for this flag is 0 (no errors).

3.5.9 CB9 RS232 communication flag

CB9=1 Enable RS232 communication (transmit) when using address. (addr>0) (default).
CB9=0 Disable RS232 communication.(transmit).

3.5.10 CB10 Default direction flag

CB10=1 The direction output is inverted. The motor will run in reverse direction.
CB10=0 Direction output is high if SR+ and low if SR- (default).

3.5.11 CB11 Disable error E43-E47 flag

Disable fatal error E43 to E47.

Error LED	RS232 Error Message	Running LED	CB11=	bit	Description
Lit	Yes	Off	0		E43 to E47 will stop program execution and motor (Default) Error message will be sent to the RS232 and Error LED will be lit.
Flash	Yes	On	+1	0	E43 to E47 are treated as common errors. Motor and program will not stop. Error LED will flash once. Error message will be sent to the RS232.
Flash	No	On	+2	1	E43 to E47 are treated as common errors. Error message will not be sent to RS232 and Error LED will flash once.

3.5

Control Flags

3.5.12 CB12 Trigger level flag for INT1

Trigger level at input 1 (I1) when interrupt routine 1 (INT1) is used

Default for this flag = 0

CB12=0 Interrupt at the falling edge - when input 1 goes from logic 1 to 0.

CB12=1 Interrupt level, logic 0. Pulse interrupt. If the level remains 0, the system will remain interrupted.

CB12=2 Interrupt level, logic 0. Interrupt is disabled by the first measurement of 0 on Input 1. Interrupt enables at the next *NSTOP* command.

3.5.13 CB13 Trigger level flag for INT2

Trigger level at input 2 (I2) when interrupt routine 2 (INT2) is used.

Default for this flag = 0.

CB13=0 Interrupt at the falling edge - when input 2 goes from logic 1 to 0.

CB13=1 Interrupt level logic 0. Pulse interrupt. If the level remains 0, the system will remain interrupted.

CB13=2 Interrupt level, logic 0. Interrupt is disabled by the first measurement of 0 on Input 1. Interrupt enables at the next *NSTOP* command.

3.5 Control Flags

3.5.14 CB14 Trigger level flag for INT3

Trigger level at input 3 (I3) when interrupt routine 3 (INT3) is used.

Default for this flag = 1

CB14=1 Interrupt at the rising edge - when input 3 goes from logic 0 to 1.

CB14=0 Interrupt at the falling edge - when input 3 goes from logic 1 to 0.

3.5.15 CB15 Servo alarm signal (SALA) flag

Active level for servo alarm, SALA. Pin 7 SALA at driver/control connector.

CB15=0 SALA signal disabled. (RS register can never be 5)

CB15=1 SALA active high

CB15=2 SALA active low (default)

If the servo alarm is activated while the motor is operating, pulse generation will first cease when an attempt is made to start the motor again. This is due to the fact that RS is updated when RS is read or just before execution of a motor command. If it is required that RS is updated once for each line, the CB35=8 command is used.

3.5.16 CB16 Motor in position (COIN) flag

Active level for COIN (motor in position) signal. Pin 8 at driver connector.

CB16=0 COIN signal disabled. (RS register can never be 4)

CB16=1 COIN input active high

CB16=2 COIN input active low (default)

3.5.17 CB17 Enable running output (O8) flag

Running status at user output 8 (O8).

The CB17 flag can enable a status function at output 8 indicating when the motor is moving. When enabling this feature, O8 will be set / reset synchronously with the MOTOR LED at the front of the Indexer.

CB17=0 Function is disabled - O8 works as normal (default).

CB17=1 Function enabled - O8 is high when the motor is running.

CB17=2 Function enabled - O8 is low when the motor is running.

3.5.18 CB18 NSTART Trigger level flag

Trigger level when *NSTART* command is used.

CB18=0 Transition from logic 0 to 1

CB18=1 High logic 1 (default)

CB18=2 Low logic 0

CB18=3 Transition from logic 1 to 0

3.5.19 CB19 Digital filtering on user inputs (from version 1.7)

Disable average measurement on inputs I1-I8, PL, NL and HM. Normally the input is measured 16 times to detect and verify a valid level. If CB19=1, the input is only measured once. This can be done if the program is executing some time-critical operation.

CB19=1 1 time measurement

CB19=0 16 times measurement. Default.

If CB19 is preset to a value between 1 and 255, this will be the number of times the input is measured. Each measurement takes 5.4 microseconds, which means that the time cycle can be improved by 86 microseconds by choosing CB19=1 instead of CB19=0. This can be significant when using *NSTART* for example.

3.5

Control Flags

3.5.20 CB20 High speed start trigger at IN1

This flag can be used in applications where the motor must be started extremely fast (no delay time) when an external event happens — e.g. when a sensor is activated. The high speed start offers a reaction time of less than 100 μ s regardless of other activities in the Indexer.

The distance must be specified by the *SR* (set relative distance) command before the start input will react. The distance only need to be specified once to be remembered by the high speed function, but it can be changed at any time.

CB20=0 High speed start disabled (default at power up)

CB20=1 High speed start enabled, triggering at the rising edge at IN1

CB20=2 High speed start enabled, triggering at the falling edge at IN1

Note ! If the CB20 flag is enabled, the motor will not react on any movement commands (*SR*, *SP*, *SR+*, etc.). Only a valid signal at input 1 will start the motor.

Note also that the only commands that disable the high speed start input are *H* (halt), *RESET*, *SD* (set default), or simply setting CB20=0. The *H* (halt) and *STOP* instruction will disable the high speed start (CB20=0) permanently in the same manner as *RESET* or *SD* (set default)

Example

```
CB20=1      ;ENABLE THE HIGH SPEED START (RISING EDGE)
SR=1000     ;SET THE MAXIMUM DISTANCE. THE MOTOR WILL NOT
            ;MOVE BY THIS COMMAND ITSELF; ONLY WHEN THE INPUT
            ;1 IS ACTIVATED.
:START OUT1=1 ;SET O1 - THE MAIN PROGRAM STARTS HERE.
-        -   ;THE PROGRAM IS NOW EXECUTED AND THE HIGH SPEED
-        -   ;START IS WORKING IN THE BACKGROUND.
```

3.5.21 CB21 High speed stop trigger at IN2

This flag can be used in applications where the motor must be stopped extremely fast (no delay time) when an external event happens — e.g. when a sensor is activated. Note that a stop is performed with the specified deceleration. The high speed stop offers a reaction time of less than 100 μ s regardless of other activities in the Indexer.

The high speed stop functions exactly as the *SH* (soft halt) command

CB21=0 High speed stop disabled (default at power up)

CB21=1 High speed stop enabled, triggering at the rising edge at IN2

CB21=2 High speed stop enabled, triggering at the falling edge at IN2

Example

```
CB21=1      ;ENABLE THE HIGH SPEED STOP (RISING EDGE)
:START SR=1000 ;SET THE DISTANCE. THE MOTOR WILL START MOVING
            ;BUT STOP WHEN INPUT 2 IS ACTIVATED
OUT1=1      ;SET O1
WAIT RS=0   ;WAIT UNTIL MOTOR IS STOPPED. THIS WILL
            ;HAPPEN IF THE MOTOR IS STOPPED BY ACTIVATING
            ;INPUT 2 OR IF ALL 1000 PULSES ARE EXECUTED.
OUT1=0      ;RESET O1
J:START    ;JUMP TO START AND MAKE A NEW RUN
```

3.5.22 CB22 Diverse flag

The flag is set to CB22=0 after use.

CB22=1 deletes the last 15 error messages that can be shown with the *ESTG* command.

CB22=2 resets the driver and sets *SON* as it was before. (SMC35 only)

CB22=3 deletes all user registers R0 - R220

CB22=4 test of user registers R0-R220 and the EEPROM where R0-R220 are saved with *MS2* command. Errors are written to *RS232* and CB22 is set to 255 if there are errors.

3.5

Control Flags

3.5.23 CB23 Used for electronic gear

If the motor must be completely synchronised with another moving part, this feature can be used. Inputs I7 and I8 are in encoder format. Output is pulse and direction signal. The Indexer does not perform gearing on the signal. This must be done in the connected servo driver. Note that it is possible to activate and deactivate the gearing by use of the commands CB20 and CB21. Thus the gearing is active when input I1 is active and inactive when input I2 is active.

CB23=0 Disable the gear function (default)

CB23=1 Enable the gear function all the time.

CB23=2 Enable the gear function except when decelerating before target. Deceleration will start on VM, follow ACP and stop at VS.

3.5.24 CB24 Position reached. (Status flag)

The CB24 flag indicates if a positioning is completed without any stop.

CB24=0 The last positioning was finished with a *SH*, or highspeed stop.

CB24=1 Last positioning was finished without any stop.

The flag will be cleared if there has been a soft halt command (*SH*) or high speed stop (see *CB21 High speed stop trigger at IN2*, page 114).

The basic idea behind this command is to prevent the motor from keeping running if e.g. a stop sensor is faulty or similar.

Example

```
CB20=1      ;ENABLE HIGH SPEED START
CB21=1      ;ENABLE HIGH SPEED STOP
SR=1000     ;SET MAXIMUM POSITIONING LENGTH
            ;FROM NOW ON THE MOTOR WILL START AND STOP ACCORDING TO
            ;THE HIGH SPEED START AND STOP INPUTS.
            ;EVERY TIME A START IS APPLIED, MAX.1000 PULSES WILL BE
            ;EXECUTED.
:START     IF CB24=1 ;CHECK IF THE TOTAL LENGTH HAS BEEN EXECUTED
           J:ERROR  ;IF YES (THE TOTAL LENGTH HAS BEEN EXECUTED), JUMP TO
                   ;THE ERROR ROUTINE SINCE NO STOP SIGNAL WAS RECEIVED.
           J:START   ;IF NO CHECK AGAIN
:ERROR     OUT1=1    ;SET O1 AS AN ERROR INDICATION.
           ....
```

3.5.25 CB25 Trigger level flag for NL input

CB25=1 Transition from logic "1" to "0"

CB25=0 Transition from logic "0" to "1". Default.

This flag works when *NSTOP*, *INT* and limit switch *NL* is used.

3.5.26 CB26 Trigger level flag for PL input

CB26=1 Transition from logic "1" to "0"

CB26=0 Transition from logic "0" to "1". Default.

This flag works when *NSTOP*, *INT* and limit switch input *PL* are used.

3.5 Control Flags

3.5.27 CB27 Zero search flag. (Status flag)

CB27=0 (default)

CB27=1

CB27 is set high as soon as the *SZ+* or *SZ-* command is executed. If zero seek is interrupted by a *HALT* command while the program is being executed, the flag will still be 1.

CB27 can then be set manually to 0 by CB27=0

3.5.28 CB28 CVIx Frequency range

CB28=0 Used when measurement at low frequencies is desired in the *CVIx* command. (typically 1Hz to 7kHz). Default.

CB28=1 Used when measurement at high frequencies is desired in the *CVIx* command. (typically 14 to 60kHz)

Note that timer 1 (CN1) cannot be used if CB28=1.

3.5.29 CB29 RS232 activity

If CB29=1, then the Power LED will blink concurrently with the reception and transmission of data on the RS232 (RS485) interface. The LED will have reduced light intensity and blink to indicate activity. This is used when one is in doubt whether the Indexer is receiving data correctly, for instance if problems arise with the cable or port set-up. Note that the data transmission rate will be reduced.

3.5.30 CB30 Modula Mode (Turntable mode)

This flag is used for example when a turntable is used, and it is desired to move to a fixed positive position/angle and not a certain number of pulses. The position counter will always contain a number between 0 and RX12. RX12 indicates how many pulses are needed for one revolution. If a servomotor with 8192 pulses is used, a gear has a ratio of 10:1, and CON = 1.0000, RX12 must be set to 81920. The position range is between 0 and 81919. Positioning must take place using the SP command. If SP is set to SP=81920 and AP=0, the motor will not run as it is already at the desired position.

Using the CB30 flag, it is possible to select whether operation should be clockwise, counter clockwise or the shortest possible distance to the desired position.

CB30=0 Function disabled. (default)

CB30=1 Counter clockwise

CB30=2 Clockwise

CB30=3 Shortest possible distance

See section 4.9 for program example.

3.5

Control Flags

3.5.31 CB31 For internal use, reserved

3.5.32 CB32 Set/Read Interrupt status (Flip/Flop) (from version 1.7)

Using this register it is possible to activate/simulate an interrupt, even though it has not occurred. This can be used to test programs or to initiate an interrupt routine from a program-dependent condition (see the INT command). The register can be read to examine if an input has been active, even if only for a short duration. Inputs NL, PL, IN3, I2, I1 can in this way be used as a form of flip/flop inputs since a bit is set when the input is activated. Note that the bit is only reset when a corresponding INT routine is executed or if INT is not used, by manual reset via the software command.

CB32	Reserved	CN2	CN1	PL	NL	IN3	IN2	IN1
Bit no.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Triggerlevel Flag				CB26	CB25	CB14	CB13	CB12
Interrupt		INT8	INT7	INT10	INT9	INT3	INT2	INT1

Example 1:

```
CB32=0 ; Reset any interrupt bit
IF CB32=#xxxxxxx1 ; If IN1 has been activ-
; ated...
JS:RUN ; .. jump to RUN routine
.
.
.
:RUN CB32=CB32 ANDL 254 ; Reset bit so input is not
; doubly detected

SR=1000
WAIT RS=0
RET
```

Example 2

```
CB32=0 ; Reset any Interrupt bit
CB32=#xxxxxxx1 ; Activate bit 0 corresponding
; to interrupt INT1 will be
; executed

.
.
INT1 ; When bit1 in CB32 activated
; program jumps here OUT=255
; and outputs set active.

; RETI; Jump back to
; line after
; CB32=#xxxxxxx1
```

3.5

Control Flags

3.5.33 CB33 User output Error mode (from version 1.7)

This flag can be used to determine what happens when an error occurs at user outputs 01-08. When a user output is shorted to ground or overloaded, the output will close down automatically after a few seconds and the Error LED and Output Error LED will be lit. CB33 can be used to determine what happens to program execution and the outputs when this happens.

	Output 1-8	Error LED	Output Error OE LED	RS232 Error 43 text	Motor and Program	Outputs after short circuit is remove
CB33=0	No change	Lit	Flashing	Yes	Stopped (SH)	Value as before short circuit
CB33=1 (Default)	No change	Lit	Flashing	Yes	Running	Value as before short circuit
CB33=2	00000000	Lit	Flashing once	Yes	Stopped (SH)	All outputs will be logic 0. OUT=0
CB34=3	No change	Lit	Flashing	No	Running	Value as before short circuit

3.5.34 CB34 Reserved

3.5.35 CB35 Acknowledge from the indexer to a PC that a command has been executed and that RS is updated continuously (available from version 1.7)

Using this flag, it is possible for response commands to be automatically sent via the RS232 dependent on a change of user inputs, or the motor has completed a reset or has moved to the required position. The following is based on a program is being executed and the check will only be carried out once per line execution. If the Indexer is in stand-by mode and the program is there not being executed, the function will be executed approximately 1000 times/sec.

3.5

Control Flags

Trigger	Bit no.	Function	CB35=0	Examples of re-sponse commands
SR+, SR- SR=x SP=x SZ+, SZ-	0	Position transferred when the motor is stopped. AP transferred when RS goes from a value greater than 0 to 0. Normally requires that bit no. 3 is also set.	+1	AP=1000
Active level on IN 1-3	1	Once for each program line execution inputs IN1-IN8 are read and if changes have occurred, transmission is made via the RS232. Note that level changes must be active for a period greater than the time taken to execute a line	+2	IN=127
Level shift on IN 1-3	2	When interrupt trigger level for IN 1-3 (CB12-14) is fulfilled, the output level IN1-8 is transmitted on the RS232. Data are written even though only a short pulse is applied to the input.	+4	IN=7
	3	Updates RS and AP automatically once for each program line execution	+8	

This function is often used when a PC or PLC is used in a system. The function avoids having to spend time continuously querying the Indexer whether a task is complete or an input is set for example. The Indexer automatically responds as soon as an event occurs. CB35 can be used both when a program is executed and if commands are sent directly via the RS232/RS485 interface. If addressing is used, the command response will be prefixed by the corresponding device address. For example 1AP=1000, for the device with ADDR=1.

Several monitoring function can be used in parallel and are selected by writing the summed value to CB35. For example, if bit 1 and 3 are required, 2 + 8 are summed and the command is thus CB35=10.

3.5 Control Flags

3.5.36 CB36 1 ½ axis control and selection of chopper frequency (SMC35B only)

Internal or external axis:

Using a 1 ½ axis controller it is possible to control 2 motors using the same program. Both motors are controlled independent of one another, but not simultaneously. In addition, it is possible to operate both motors fully synchronously. The advantage of this method over 2 individual 1-axis controllers is the cost saving in having only 1 program.

Since a 1½ axis controller is used in this case, the motors cannot be controlled inter-dependently. For example, linear or circular interpolation is not possible. By changing CB36 throughout a program, an internal axis, external axis or both axes be selected.

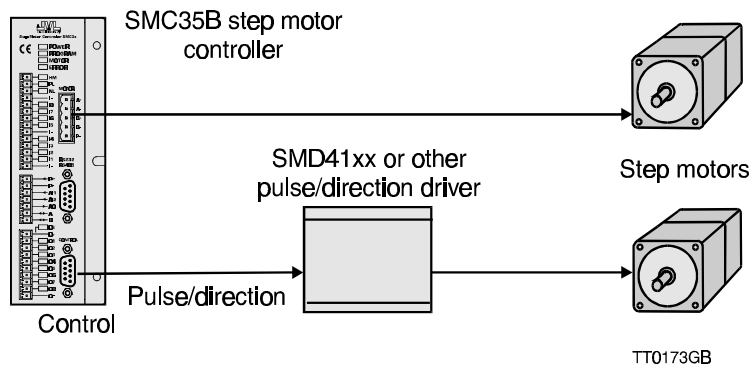
If only the internal axis is selected, the built-in step motor driver is used. No pulse/direction signals will be output to the CONTROL connector external axis. Conversely, if the external axis is selected, pulse/direction signal will only appear at the CONTROL connector. In this way an external step or servo driver can be controlled. With both axes active simultaneously, it is possible to operate 2 motor synchronously.

Chopper frequency: (from version 1.8)

Motors with very little inertia, e.g. disc motors, can operate more smoothly if the chopper frequency is doubled. Doubling the chopper frequency means that the current is regulated more precisely and thus results in better regulation of the motor. Noise levels can also be reduced slightly by operating at double chopper frequency. The disadvantage is that the motor and controller become warmer, so normally a chopper frequency of 20Khz must be used. Operating with double chopper frequency is not recommended is the current is set higher than 2.5-3 Amp.

	Chopper frequency internal	External axis	Internal axis
CB36=0	Normal 20kHz	No	No
CB36=1 Default	Normal 20kHz	No	Yes
CB36=2	Normal 20 kHz	Yes	No
CB36=3	Normal 20kHz	Yes	Yes
CB36=128	Double 40kHz	No	No
CB36=129	Double 40kHz	No	Yes
CB36=130	Double 40kHz	Yes	No
CB36=131	Double 40kHz	Yes	Yes

See section 4.9.6 for program example.



3.5

Control Flags

3.5.37 CB37 Digital filtering on interrupt inputs (only from version 1.7)

Normally inputs are digitally filtered (see CB19) but when an input is used as an interrupt input where fast response is expected, only a pulse or a level shift is sufficient to activate the input. When using interrupt input, the input is only measured once and a noise pulse can unwantedly trigger the input. Therefore it is possible to perform digital filtering on the interrupt input using this CB flag. If CB37=0, the interrupt routine is directly executed. If CB37 is preset to a value between 1 and 255, this value specifies the number of times the input is measured. Each measurement takes approximately 5.4 μ s. Note that digital filtering will not work together with high speed start/stop trigger commands CB20 and CB21 because the input is connected directly to the signal processor and cannot be filtered by the microprocessor. It is possible to perform digital filtering on inputs: IN1, IN2, IN3, NL and PL.

3.5.38 CB38 Motor status (RS) when limit is activated (only from version 1.7)

Using this flag it is possible to determine whether the RS value is set to 0 or 7 when the limit input is activated. Often a program is built up with the motor being started by an SR command and execution waits for the motor to finish with "WAIT RS=0". If the limit input is activated, the motor has not completed its run, but RS becomes 0. This can be properly regarded as an error condition and therefore RS should have a value other than 0. Using CB38 this can be changed so that RS is assigned the value 7 and it is thus possible to control program execution in this case. Regardless of which mode is selected, the motor will always stop when the limit inputs are activated. Note that E11 is erased when the EST command is executed or when a new run command is executed. RS will thus be set to 0 again.

CB38 mode	RS	Description
0 (Default)	RS=0	When NL or PL is activated, the motor will stop and RS is 0
1	RS=7	RS is 7 when error flag "E11:limit switch activated" is set.
2	RS=7	RS is 7 when "E12: limit switch active" is set.
3	RS=7	RS is 7 when E11 or E12 is set.
4	RS=7	RS is 7 if operating in a positive direction and PL is activated, or if operating in a negative direction and NL is activated.

3.5.39 CB39 Diverse flag (only from version 1.7)

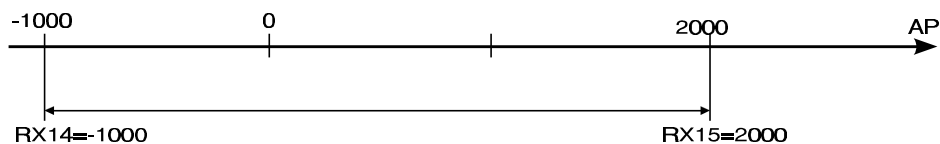
CB39	Bit	Description
+1	0	Normally a motor alarm will not stop the Indexers running LED or program. Even the ERROR LED will first be lit when a new motor running command is executed. This is because the motor status (RS) is only read before a motor run command. If the CB39 bit 0 is set and CB35 bit 3 is set, then the motor status (RS) will be tested continuously so that a motor alarm will stop the motor and program immediately. Motor and program will also stop if CB39 bit 0 is set and an RS command is executed. Note that the motor will stop immediately as if the SH command was executed.

3.5 Control Flags

3.5.40 CB40 Software position limits (only from version 1.7)

CB40	Description
0	Default. No test for position limits.
1	If the required position (AP) is calculated to be out of the interval specified in RX14 and RX15, the motor will stop at the position specified in RX14 or RX15. The error bit 13 will be set and the ERROR LED will flash once.
2	Same as CB40=1 but when the position is calculated to exceed the position limit, the run is not executed. The error bit 13 is set.
3	Same as CB 40=2 but the program will also stop when the software position limit is exceeded. When the position is calculated to exceed the limits, the program will stop and therefore the last motor run will not be executed.

Example:



TT0170GB

```

CB40=01      ; Activate software position lim-
RX14=-1000  ; its, check lower position limit
RX15=2000   ; Higher position limit
:Start      SR=100
            Wait RS=0
            J:Start
            INT 113      ; If software limits exceeded
            SP=0         ; run back to position 0
            Wait RS=0   ; Wait until motor stopped
            RET          ; Return to start loop

```

3.6

Error Messages

When an error occurs in communication with the Indexer or when an internal error occurs, the Indexer transmits an error message. The error message consists of an 'E', followed by an error number, followed by a colon ':', followed by a descriptive English text. The following illustrates an example of an error message:

Example: E2: Out of range

If error E43 to E47 occurs, a fatal error has occurred and motor and program execution stop immediately. The ERROR LED will light constantly as an indication of a fatal error. The LED will switch off after a program is started, or an EST command or a motor command is executed. If required, the error can be handled by setting CB11 to 1, thus enabling one's own error routine to be run to handle the fatal error.

When an error occurs, the corresponding ERR flag will be set, and by introducing some special interrupt routines using the INT command, provision can be made for determining how an error should be handled. The INT routine that should be introduced in the program is given by: 100 + the error number. For example: INT139 for *E39 Warning! Motor drive running*.

3.6.1 Description of Error Messages

E0: No errors

No errors have occurred since the last request.

E1: Error

The command string is not understandable or not allowed in the controller

Example:

VXUSADF

Results in error E1.

Correction:

Carefully check the command sent to Indexer and compare with the description of the command given in this manual.

E2: Out of range

The parameter value specified with the command is out of the allowable range.

Example:

VM=999999

The above command attempts to set the velocity to 999999 rpm, which is outside the allowable range. The Indexer therefore reports an E2 error.

Correction:

Specify a parameter value within the allowable range for the actual command.

E3: Number of parameters is wrong

The number of parameters specified with the command is incorrect.

Example:

EST66 or SH9

Both of the above command examples will produce an E3 error.

Correction:

The *EST* command has only 1 register associated with it and can therefore only be called by specifying *EST*.

The *SH* command is only used to make the motor decelerate and therefore specifying a parameter has no meaning.

E4: Instruction does not exist

The command given does not exist

Example *ABCDEF*

Correction:

Use a valid Indexer command. See the description of the command for details of the required command syntax.

E5: It is not an instruction

The Indexer has not received a proper command.

Example:

4R

If the Indexer is not using addressing, this example will result in error E5.

Correction:

Use a proper command.

E6: Parameter error or out of range

There is an error in the specified parameter or the parameter value is out of the allowable range.

Example:

SP=11111111111 or *VM=8G7*

Correction:

The Indexer cannot handle values as great as *11111111111* in the first example. Use a value within the allowable range.

In the second example: parameter values must not contain alphabetic characters.

E6: Parameter error or out of range

The Indexer has received a parameter value which must be an integer.

Example:

VM=1000.8

Correction:

Send the command specifying an integer value *VM=1000*.

E7: Register number error or out of range

Error in register number.

Example:

XP7777 or *XP4F*

Correction:

In the first example: use a register number in the allowable range.

In the second example: register numbers must not contain alphabetic characters.

E8: Data can not be saved in EEPROM

The data cannot be saved in the EEPROM, the data read from the EEPROM are illegal, a hardware or software error has occurred that prevents the CPU from communicating with the EEPROM or an illegal value is stored in the EEPROM caused by the ESTG command. Correction: Try to use the delete EEPROM command *##*.

E9: Checksum error

The Indexer's (receiver's) calculated checksum is not the same as the transmitted checksum.

Example:

25VM=2000Q3.

Correction:

Send the command as *25VM=2000A2*.

E10: Value out of range for CS and CT command

Correction: Select a value within the specified range

E11: Limit switch activated

If the negative (NL) limit switch is activated, motor movement in the negative direction is stopped. Only positive movement is now possible.

If the positive (PL) limit switch is activated, motor movement in the positive direction is stopped. Only negative movement is now possible. This error message will not be written to the RS232 interface. It will be placed in the error register only.

E12: Limit switch active. Motor run command ignored

If a limit switch is activated when a motor-run command is executed, the motor command will be ignored and the program will continue to the next command.

E13: Position limit exceeded. Position truncated**E14: Odd parity error**

A command with wrong parity has been received because of noise on the RS232 interface. Send the command again. Can also arise because the baud rate is not set correctly.

E15: Reserved**E16: Command not allowed in standby mode**

The command is an illegal command in standby mode and can only be used in a program.

Example.

The *PX* command is used to change mode from programming mode to standby mode.

Correction:

Do not use the command in standby mode.

E17: Memory full, (EEPROM)

The memory is full and cannot contain any more commands.

Correction:

Reduce the size of the program, erase blank lines and try to use commands that use little memory. See *Alphabetical Overview of Commands*, page 129 for command list.

E18: Command not allowed. Program is running.

A command has been used which is not allowed because the program is running.

Example

WAIT INI=1

Correction:

Use a valid command. See *Alphabetical Overview of Commands*, page 129 for a complete list of valid commands.

E19: Command not allowed in programming mode

A command has been used which is not allowed because the indexer is in programming mode.

Example

PE

Correction:

Use a valid command. The *PE* command changes mode from standby mode to programming mode. It is not allowed to change from programming mode to programming mode. See *Alphabetical Overview of Commands*, page 129 for a complete list of valid commands.

3.6

Error Messages

E20: Command not allowed. Motor is running

A command has been used which is not allowed because the motor is running.

Example *SR=4000*

Correction:

Use a valid command. The SR command runs the motor and is not allowed when the motor is already running. See *Alphabetical Overview of Commands*, page 129 for complete list of which commands can be used.

E21: No program in RAM

If the temporary memory does not contain a program and the *GO* command is executed, this error will occur.

Correction:

Load program and remember to store it permanently in the EEPROM. Set MR=1 so that the program is started at power up, or 'check' the 'run at power up' item in the program flow dialogue. (AMCxx or SMIxx).

E22: Command ignored. Multicontrol or JVL bus communication error

An error in the communication between the SMI Indexer and a connected module on the JVL bus has occurred or an error occurred between 2 SMI Indexers in multicontrol mode.

Correction:

Check cable and make it shorter if possible. Use a screened cable.
Check that all units have their own unique address.

E23: Command ignored. Multicontrol or JVL bus timeout

The SMI Indexer has tried to send a command but has not received a reply.

Correction:

Check if the module is connected correctly and has been given the correct address corresponding to the command sent.
Check that all units have their own unique address

E24: Unknown register/flag on JVL Bus

E25: Reserved

E26: AC lower than 1. AC value changed to 1

E27: AC higher than 1250000. AC value changed to 1250000

E28: Reserved

E29: AOUT parameter out of resolution range

E30: Warning, wrong supply voltage

If the supply voltage is less than 10 VDC or higher than 32 VDC for the SMI3x or higher than 85 VDC for the SMC35 this error message will be given.

Correction: Change the supply voltage to typically 24 VDC.

E31: Warning. SMI3x/SMC35 conflict solved

E32: VM specified lower than VS. VM value changed to VS

The Indexer have received a VM value lower than VS.

Correction: Change the VM value to a value higher than VS

E33: Error in CTM parameter

The number of parameters specified with the command is incorrect.

Example:

CTM1=86

Correction:

Use a parameter within the limits.

E34: Too many gosub, max 32

A maximum of 32 *GOSUB* or *JS* levels are allowed in a program. This error message will be given if the program detects more than 32 *JS* levels. The error will be given when the Indexer is in running mode and detects the error.

Correction: Simplify the program with less *GOSUB* or *JS* commands.

E35: Program end

This error message will be given if the program ends without a jump command.

The Indexer will be in standby mode if this happens. The error will be given when the Indexer is in running mode and detects the error.

Correction: Place a jump command to a label above in the end of the program.

E36: Too many else after if (ELSE)

This error message will be given if there is a greater number of *ELSE* statements than *IF* statements in the program. The error will be given when the indexer is in running mode and detects the error.

This error can also be given if there are over 300 *IF* statements in the program.

Correction: Use less than 300 *IF* statements in the program

E37: Too many interrupts (INT)

A maximum of 32 interrupt programs running at the same time is allowed. If the Indexer is servicing an interrupt routine and another interrupt appears, it jumps to the new interrupt routine. It will first continue with the first interrupt routine when the last is finished. This error message will be given if the program detects more than 32 interrupt levels. The error will be given when the Indexer is in running mode and detects the error.

Correction: Simplify the program with less interrupt routines.

E38: Return without jump (RET, RETI)

This error message will be given if the program detects a *RET* or *RETI* command without a *J* or *JS* command first. When a return command is executed, it jumps to a line number and if the Jump command is missing, the Indexer does not know which line to return to.

E39: Warning !. Motor drive running

This error message will be given if a motor command (*SR,SP,SZ*) is under execution and the motor driver is still running or busy after a deceleration. Note that this error is given if the motor 'in position signal' (*COIN*) is active.

Correction: This signal must be tied if not used or disabled by the *CB16* flag - see *CB16 Motor in position (COIN) flag*, page 113

E40: Address not allowed. Max 255

E41: Timeout on IN7/IN8. (CVI command)

E42: Motor processor fault

E43: 01-08 Output error

One of the outputs O1-O8 has been short circuited.

Correction:

Turn power off and fix the problem.

E45: Fatal case error. Contact JVL

This error will be given if there is a problem in the Indexer's firmware or a problem with the hardware.

Correction: Check program or use the delete EEPROM command ##.

E46: ERROR !. Alarm signal from motor drive

This error message will be given if the motor driver reports an error by activating the SALA input.

Correction:

Turn power off and fix the problem in the motor driver. Check the cable.

Check if the SALA signal is connected properly or, if not used, disable the signal using the CB15 flag - see *CB15 Servo alarm signal (SALA) flag*, page 113

E47: Unknown error. Contact JVL

This error will be given if there is a problem in the Indexer's firmware or a problem with the hardware.

Correction: Check program or use the delete EEPROM command ##.

Note:

E43 - E47 are fatal errors. If they occur, motor and program execution will stop immediately and the ERROR LED will light continuously.

If CB11=1 the ERROR LED will be switched off and program execution will continue. This can be used when it is required to handle errors which occur in the motor driver for instance. The error routine can for instance activate the output connected to the driver reset input.

3.7 Alphabetical Overview of Commands

Com- mand	Description	Defaults	Limits		Mode		Units	page
			Min.	Max.	S	P		
?	Show set-up	-	-	-	x	x	-	48
!	Show Indexer type and address	-	-	-	x	x	-	48
##	Delete EEPROM	-	-	-	x		-	41
AC	Acceleration	100	1	100000	x	x	Rpm/s	49
ACP	Acceleration defined in pulses	-	1	100.000	x	x	Pulses	49
ACT	Acceleration defined in time	-	1	10000	x	x	ms	50
ADDR	Address	0	0	255	x	x		43
AI1	Analogue input 1	-	0	16383	x	x	5/16383 V	52
AI2	Analogue input 2	-	0	16383	x	x	-	52
(AO)[n1,n2]	Activate flag in external module	-	-	-	x	x	-	53
AOUT	Analogue output	0	0	65535	x	x	5/65535 V	54
AP	Motor's Actual Position in units	0	-2.147.483.648	2.147.483.647	x	x	Units	55
APP	Actual position in pulses	-	-2.147.483.648	2.147.483.647	-	-	Pulses	47
BAUD	Baud rate on RS232/ RS485	2	1	8	x	x	bit/sec	48
CB [n]	Control Flag	-	1	40	x	x	-	95
CHS	Use checksum at RS232/RS485	0	0 = No	1 = Yes	x	x		56
CN1	Counter / timer 1	0	0	2.147.483.647	x	x	Pulses	57
CN2	Counter / timer 2	0	0	2.147.483.647	x	x	Pulses	57
CND1	Divider for CN1	1	1	2.147.483.647	x	x	-	58
CND2	Divider for CN2	8	1	2.147.483.647	x	x	-	58
(CO)[n1,n2]	Clear flag in external module	-	-	-	x	x	-	59
COMP	Compare memory	-	0	1000	x	x	-	51
CON	Conversion factor for distance	1.0000	0.0001	9999.9999	x	x	-	60
CTM1	Counter / timer 1 mode	1	1	7	x	x	-	63
CTM2	Counter / timer 2 mode	1	1	10	x	x	-	65
CV	Show current Velocity	0	0	65.535	x	x	Rpm	66
CVI [n]	Show velocity	-	0	65.535	x	x	-	57
D	Delay	-	1	32.000	-	x		58
E	Global execute	-	-	-	x	x		58
ELSE	Used together with IF command	-	-	-	-	x	-	68
ERR [n]	Error bit (n=0-47)	-	0	47	x	x	-	59
ES [n]	Error status (n=0-2)	-	0.....0	1.....1	x	x	-	69
EST	Error status in text	-	-	-	x	x	Text	72
ESTG	Error status text	-	-	-	x	x	Text	62
EXIT	Exit programming mode	-	-	-	-	x	-	62
GO	Start program execution	-	-	-	x	x	-	63
H,K	Halt of motor and program	-	-	-	x	x	-	73
(I)[n1.n2]	Read flag from external module	-	-	-	x	x	-	67
IF	Used to control program flow	-	-	-	-	x	-	74
IN [n]	Read input port status (n=1-8)	-	00000000	11111111	x	x	Bit	75
(INPUT) [n1,n2]	Read data from external module	-	-	-	x	x		75
INT[n]	Interrupt control (n=1-10)	-	1	10	x	x	-	77

() = Only available on indexer type SMI31 or SMC35B

(Continued on next page)

3.7 Alphabetical Overview of Commands

Com- mand	Description	Defaults	Limits		Mode		Units	page
			Min.	Max.	S	P		
J	Makes an unconditional jump	-	0	2000	-	x	Line no.	78
JS	Makes an unconditional jump to subroutine	-	0	2000	-	x	Line no.	78
LINE	Verify actual program line number	-	0	2000	x	x	Line no.	79
(MAKRO)	Used for custom routines	-	-	-	x	x	-	79
MCHS [n]	Memory checksum (n=0-3)	-	0	2147483647	x	x	-	80
MEM	Show used memory	0	0	100	x	x	Text	80
MR1	Recall program from EEPROM		0	3	x	x		81
MR2	Recall registers from EEPROM	-	-	-	x	x		81
MS1	Save program in EEPROM	-	-	-	x	x		81
MS2	Save registers in EEPROM	-	-	-	x	x		83
NLS	Negative Limit Switch	2	1	2	x	x		83
NSTART	Define trigger condition for motor run	0	0	15		x	-	84
NSTOP	Define trigger condition for motor run	0	0	10		x	-	85
OUT [n]	Show/set levels at User Outputs (n=1-8)	00000000	00000000	11111111	x	x	Bit	86
PIF	Set format at input 7 and 8	1	1 = Normal	2 = Encoder	x	x	-	88
PLS	Positive Limit Switch	2	1	2	x	x	-	88
PR	Encoder pulses per revolution	8192	50	20000	x	x	Pulses/rev.	89
(PRINT) [n1,n2]	Print data to external module	-	-	-	x	x	-	89
PRO- GRAM	Programming mode	-	-	-	x	-	-	78
R [n]	User register, 32-bit	-	0	220	x	x	-	78
RB [n]	User register, 8-bit	-	0	880	x	x	-	79
RESET	Reset Indexer	-	-	-	x	x	-	92
RET	Return from subroutine in a program	-	-	-	-	x	-	92
RETI	Return from interrupt	-	-	-	-	x	-	93
RI[n]	User register, 16-bit	-	0	440	x	x	-	82
RS	Status: 0=stop,1=acc.,2=max.,3=dec....		0	7	x	x	-	94
RST	Motor/program status in text	-	-	-	x	x	Text	82
RX1/RX2	High speed interval output at O6	0	0	2147483647	x	x	Pulses	85
RX [n]	Special user register	-	-2147483648	2147483647	x	x	-	85
SD	Default set-up	-	-	-	x	x	-	101
SH	Smooth Halt of motor	-	-	-	x	x	-	102
SN	Serial number	-	0	65535	x	x	-	85
SON	Servo on	0	0 = Servo off	1 = Servo on	x	x	-	102
SP	Set new absolute position	0	-2147483648	2147483647	x	x	Pulses	103
SPT	Set new global absolute position	0	-2147483648	2147483647	x	x	Units/Puls.	88
SR	Set relative position	0	-2147483648	2147483647	x	x	Pulses	104
SR2	Set relative offset distance	0	-16777215	16777215	x	x	Pulses	104
STOP	Stop motor immediately	-	-	-	x	x	-	106

() = Only available on indexer type SMI31 or SMC35B

(Continued on next page)

3.7 Alphabetical Overview of Commands

Com- mand	Description	Defaults	Limits		Mode		Units	page
			Min.	Max.	S	P		
SZ	Seek zero-point	-	-	-	x	x	-	106
VE	Show firmware version and date	-	-	-	x	x	Text	107
VM	Maximum velocity	100	0	65535	x	x	RPM	107
VOL	Show supply voltage	-	8	45	x	x	Volt	108
VS	Start speed	10	1	10000	x	x	Rpm	94
WAIT	Waits for a specified condition	-	-	-	-	x	-	108

() = Command only available on indexer type SMI31 or SMC35B

4.1 Technical Data SMI30/31 and SMC35

Description	Min.	Typical	Max.	Units
Supply(P+, P-) : SMI30/31 only				
Supply Voltage	10		32	V DC
Power Consumption (unconnected I/O) @ 24VDC supply		8		W
Supply (P+, P-): SMC35 only				
Supply Voltage	10		85	VDC
Power Consumption (unconnected I/O) @ 80VDC supply		?		W
Motor Connector, Current: SMC35 only				
Running and standby current SMC35A	0		3	A
Running and standby current SMC35B	0		6	A
Resolution 6 bit			64	levels
Driver/Control Connector:				
Output level (CLK+, CLK-, DIR+, DIR-)	0		5	V
Output level at SON (NPN output) @ 50mA			1.3	V DC
Frequency for CLK output	0		2	MHz
Input level SALA	0	5	30	V
Input level COIN	0	5	30	V
SALA/COIN logic 0	0		2.5	V
SALA/COIN logic1	4.5		2.5	V
PS+ (Only SMI30/31)		24		VDC
+14V Pin 6 (Only SMC35)	13	14	15	VDC
+14V maximum current			50	mA
+5V Pin 8 (Only SMC35)	4.5	5	5.5	VDC
+5V maximum current			50	mA
User Inputs (I1 - I6 / PL, NL, HM) :				
Input Impedance	3.2		3.6	kOhm
Logic "0"	-1		2.5	V DC
Logic "1"	4.5		30	V DC
Logic "0"	-		1.0	mA DC
Logic "1"	2.0		-	mA DC
Max. frequency on input I7 and I8			100	kHz
User Inputs (I7-I8)				
Input Impedance	?		?	
Logic "0"	?		?	
Logic "1"	?		?	
User Outputs (O1 - O8) :				
Supply Voltage (0+, 0-)	8		28	VDC
Power Consumption (unconnected I/O) @ 24VDC supply			?	
Power Consumpt. All outputs activated @ 24VDC supply			?	
Loaded Current per Output			700	mA DC
Overload Current			2	A

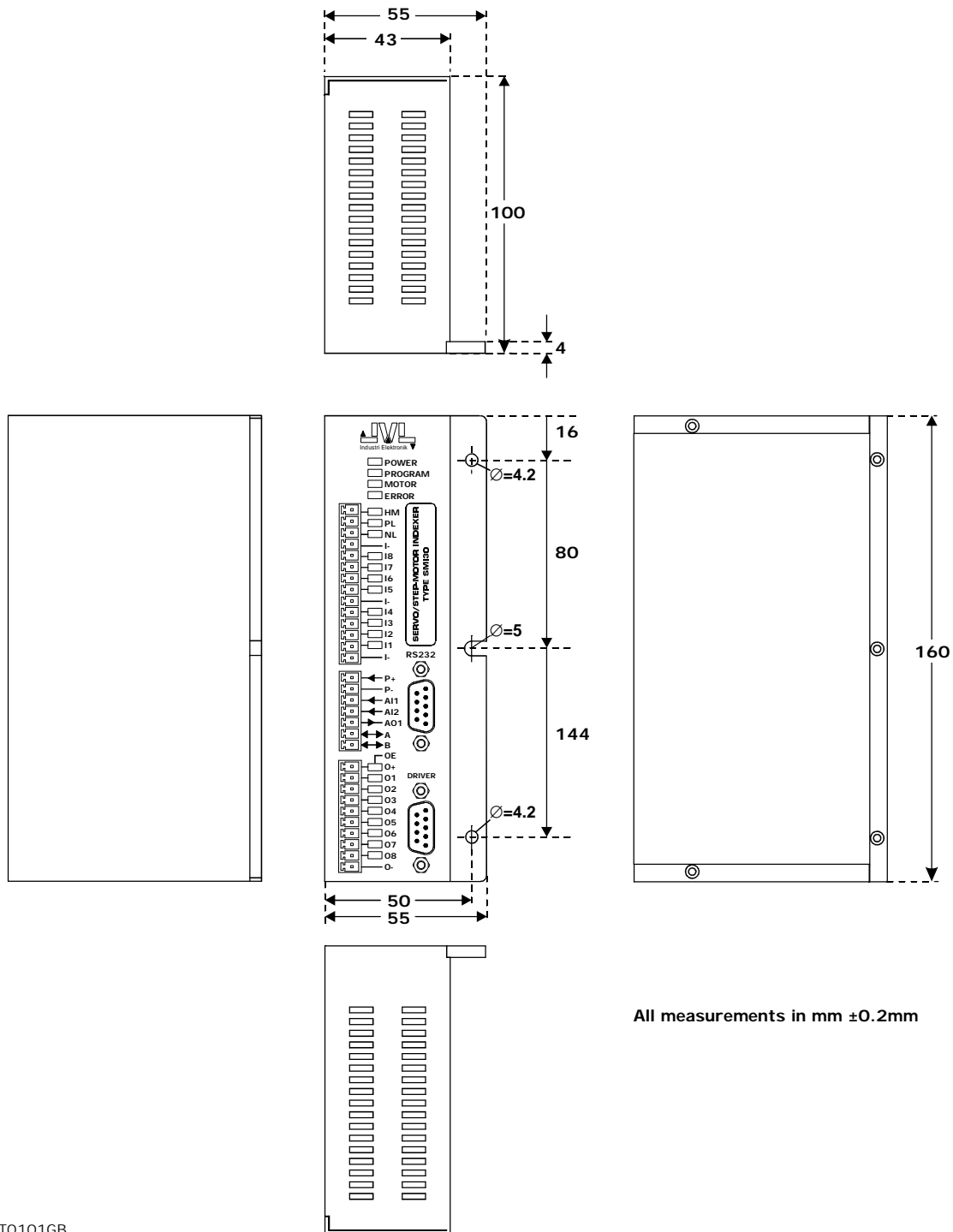
Continued next page

4.1 Technical Data SMI30/31 and SMC35

Description	Min.	Typical	Max.	Units
Analogue Output (AOUT) :				
Output voltage	0.00		5.00	V DC
Output current	0		5.0	mA
Output resistance at 5V out		10		Ohm
Analogue Inputs (AI1, AI2) :				
Input Voltage (nominal)	0		5.0	V DC
Input Impedance		10		kOhm
Time constant		1.0		ms
Precision			5	%
Diverse :				
Operating Temperature Range	0		45	°C
Storage Temperature	-20		70	°C
Humidity non condensing	0		90	%
Weight SMI30		500		grams
Weight SMC35		585		grams

4.2 Physical Dimensions

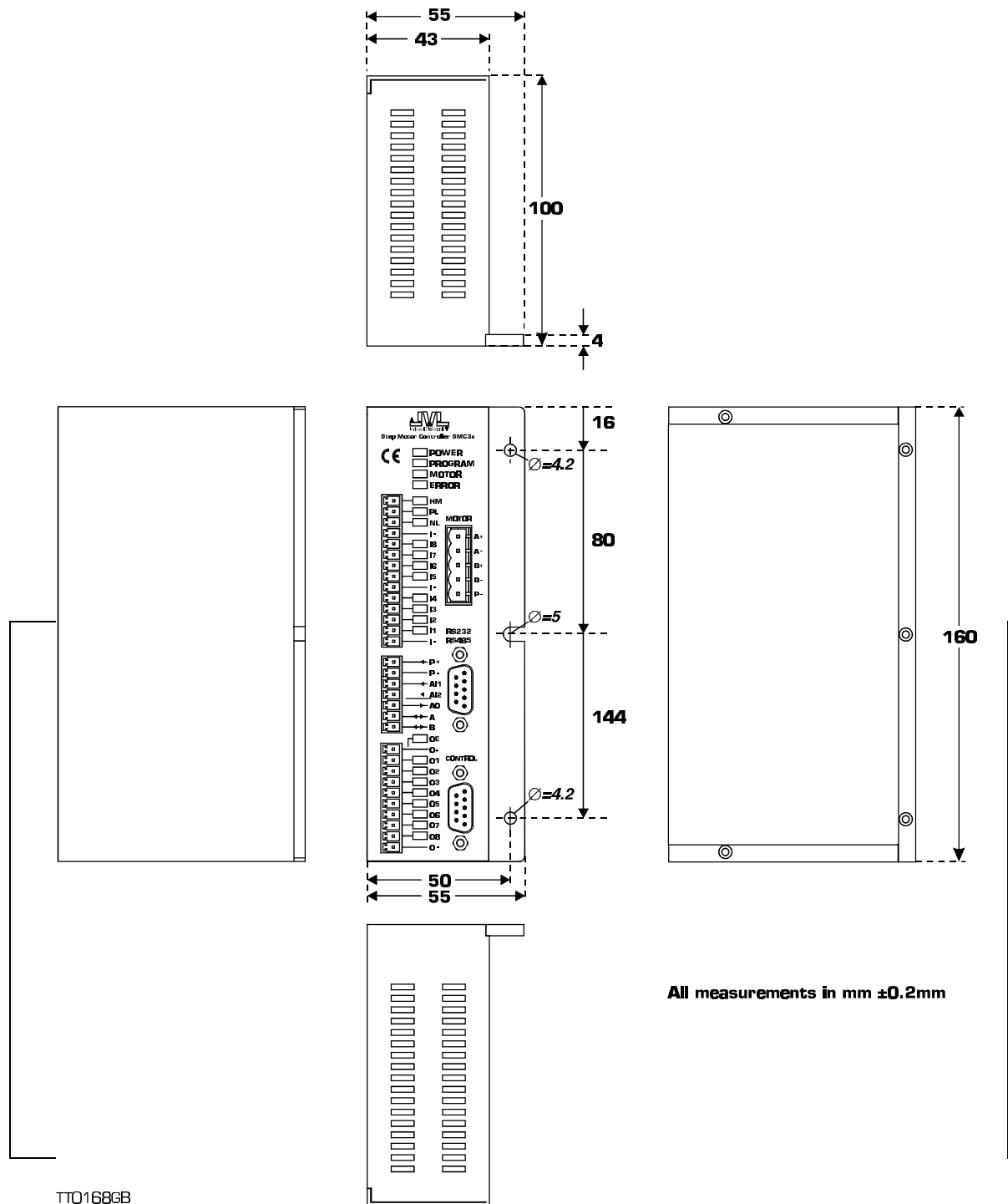
4.2.1 Physical Dimensions of SMI30 / 31



TT0101GB

4.2 Physical Dimensions

4.2.2 Physical Dimensions of SMC35A and SMC35B



4.3 Status and error indication

In addition to their normal function, the Indexer LEDs are also used to indicate vital error conditions. The following describes the normal functions of the LEDs and their additional functions. See also *Error Status Text (EST)*, page 72, concerning Indexer error messages.

The 4 LED's on the front of the Indexer can show different status and error conditions.

4.3.1 Power LED (Green)

The Power LED is lit when there the power is on and supplied to the Indexer. The LED is controlled by the internal microprocessor. If the power is applied to the *P+* and *P-* terminal and the LED is not lit, there may be a defect fuse inside the Indexer or a malfunction in the Indexer's switch mode or processor circuit. Check the internal 5x20 fuse.

4.3.2 Program LED (Green)

The Program LED is lit when the Indexer executes a program. If a program is transferred to the Indexer and the LED not is lit, check the MotoWare setup and select "*Run Program*". It is also possible to start a program from the *On line editor* using the *GO* command.

4.3.3 Motor LED (Green)

The Motor LED is lit when the motor is running. This is done if a motor command *SR*, *SP* or *SZ* command is executed in run or standby mode.

4.3.4 Error LED (Red)

The Error LED is lit when there is an error in the user output circuit, an error in the external driver or an error in the program.

LED flashing once.

This indicates an error in a program when this was transferred to the Indexer. When using the MotoWare software this will normally give an error message to the screen. If no error message is shown on the screen, look at the internal SMI30-31 error handling systems by typing *EST* in the *Online editor*. See *Error Messages*, page 123 for an error list and correction possibilities.

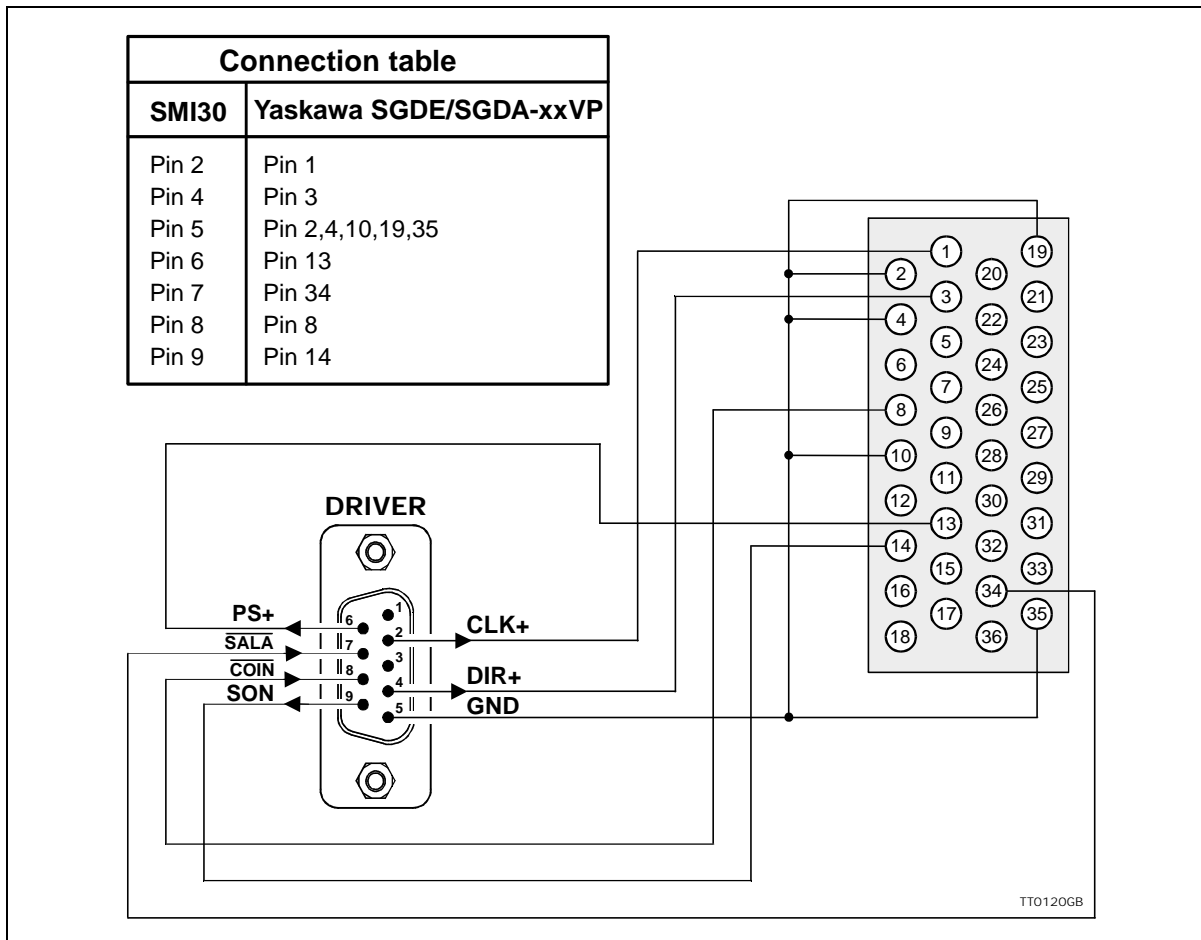
LED continuously lit.

The LED can be lit continuously if a fatal error has occurred. See *Error Messages*, page 123 for an error list.

- 1: If a user output O1-O8 is short circuited.
Correction: Remove power from the Indexer and fix the problem.
Turn power on again.
- 2: If a servo driver reports an error to the Indexer. This is done via the SALA pin in the driver/control connector. If the SALA is active when a motor command is executed, the SMI30 will stop the motor/program and turn the LED on.
Correction: Turn power off and fix the problem in the servo driver.
- 3: Fatal software error in the firmware. A command has been transferred to the Indexer and the Indexer has not been able to understand the command.
Correction: Use the *RESET* command to restart the Indexer or turn power off and on. Check the program carefully for syntax errors, etc. and send it again. Please fax or e-mail a description of the error to JVL.

During installation and use of the Indexer, various errors may occur. Information about many of these can be obtained from the Indexer itself using the *EST* command. (see *Error Status Text (EST)*, page 72). Some error conditions are similar to other errors. The following describes some of the most common errors and possible solutions.

4.5 Connection to Yaskawa servo drives

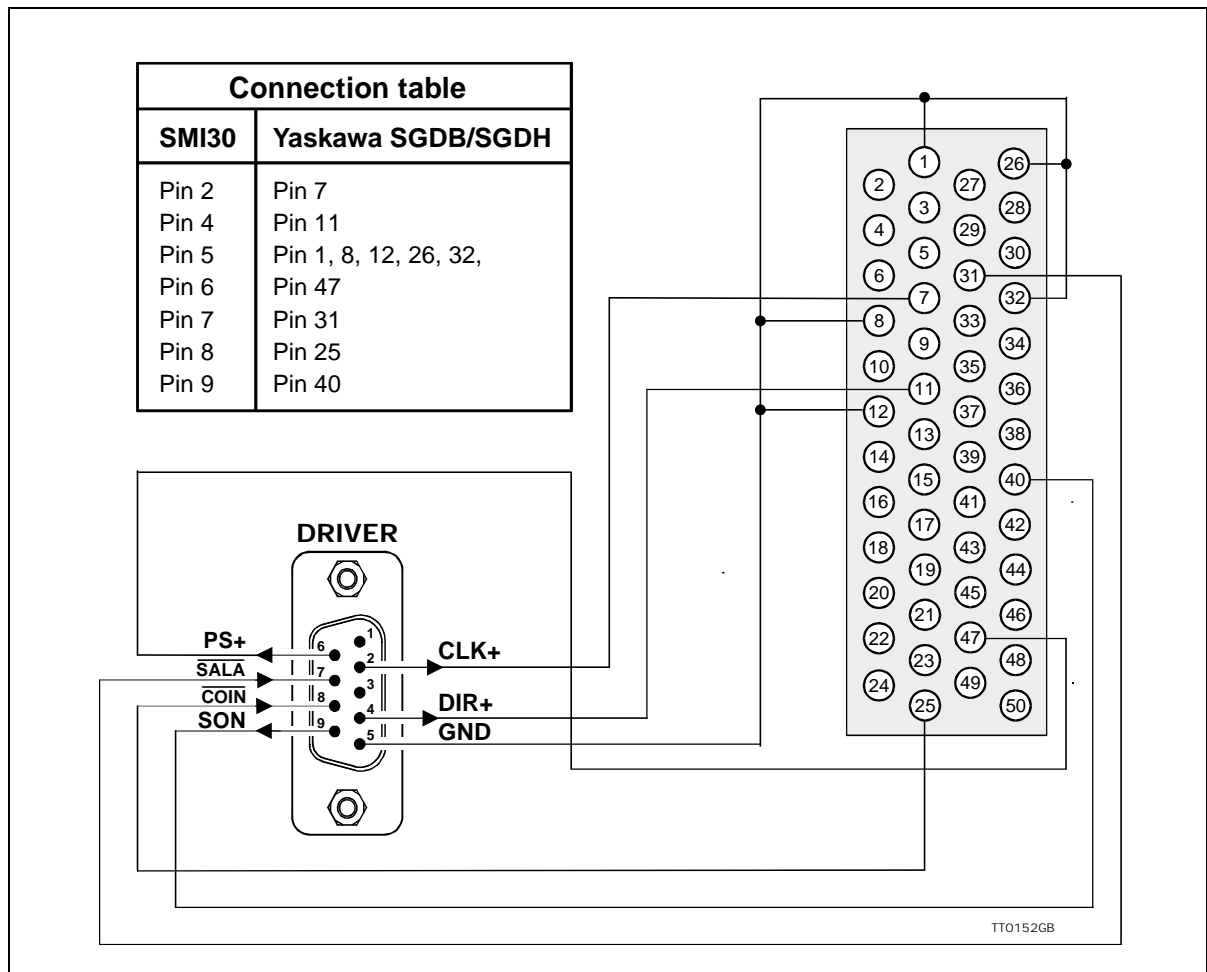


4.5.1 Connecting a Yaskawa SGDE/SGDA driver

The illustration above shows how to connect a Yaskawa sigma driver type SGDE/SGDA-xxVP to the Indexer. It is recommended that shielded cable is used, with the shield connected to GND (Pin 5) on the SMI30-31. Do not use cable lengths longer than 2 meters.

Cable order no.: YASK-SMI30-SGDA

4.5 Connection to Yaskawa servo drives

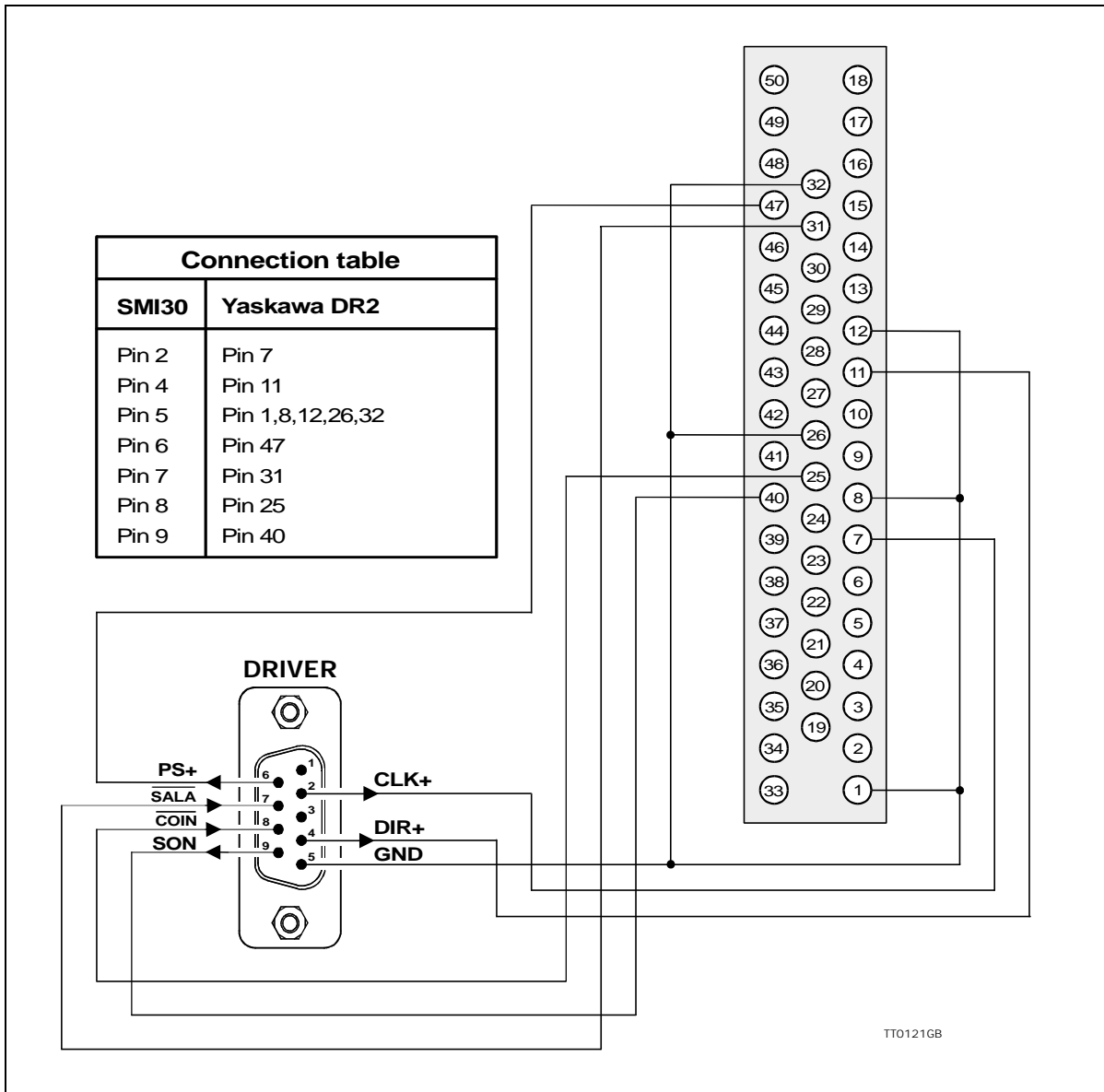


4.5.2 Connecting a Yaskawa SGDB/SGDH driver

The illustration above shows how to connect a Yaskawa sigma driver type SGDB/SGDH to the Indexer. It is recommended that shielded cable is used, with the shield connected to GND (Pin 5) on the SMI30-31. Do not use cable lengths longer than 2 meters.

Cable order no.: YASK-SMI30-SGDB/H

4.5 Connection to Yaskawa servo drives



4.5.3 Connecting a Yaskawa DR2 driver

The illustration shows how to connect a Yaskawa driver type DR2 to the Indexer. It is recommended that shielded cable is used, with the shield connected to GND (Pin 5) on the SMI30-31. Do not use cable lengths longer than 2 meters.

Cable order no.: YASK-SMI30-DR2

4.5 Connection to Yaskawa servo drives

4.5.4 Velocity and Angle

In both the Yaskawa servo drivers and in the Indexer SMI30, there are parameters which influence the velocity of the motor and the angle it moves.

1. Yaskawa

In all Yaskawa drives, 3 registers influence the velocity and the angle (Cn-11, Cn-24 and Cn-25)

The number of encoder pulses from the motor encoder is multiplied by a factor of 4 in the Yaskawa driver, so that a motor encoder with for instance 2048 pulses will indicate 8192 pulses/rev. in the driver. If parameters 24 and 25 are both 1, the motor will require 8192 pulses to turn 1 revolution. At low velocities this gives the best angular resolution, but it will cause problems at high speeds as the Yaskawa driver has a built-in 450 kHz filter which corresponds to a velocity of max. 3295 rev./min. with 8192 pulses per rev. If the velocity must be higher, Cn 24 can be set e.g. to 2.

2. SMI3x

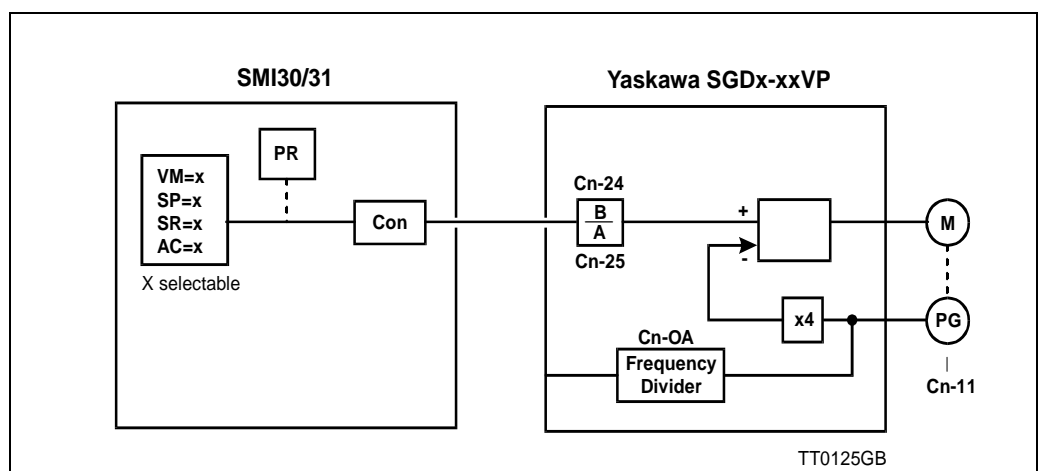
In the SMI Indexer there are also some parameters which influence the velocity and acceleration of the motor and the angle it turns.

The *PR* command defines the number of pulses required to make the motor turn 1 revolution. *CON* is a scaling factor which is used for example to rescale a given length (e.g. in mm) to a number of pulses.

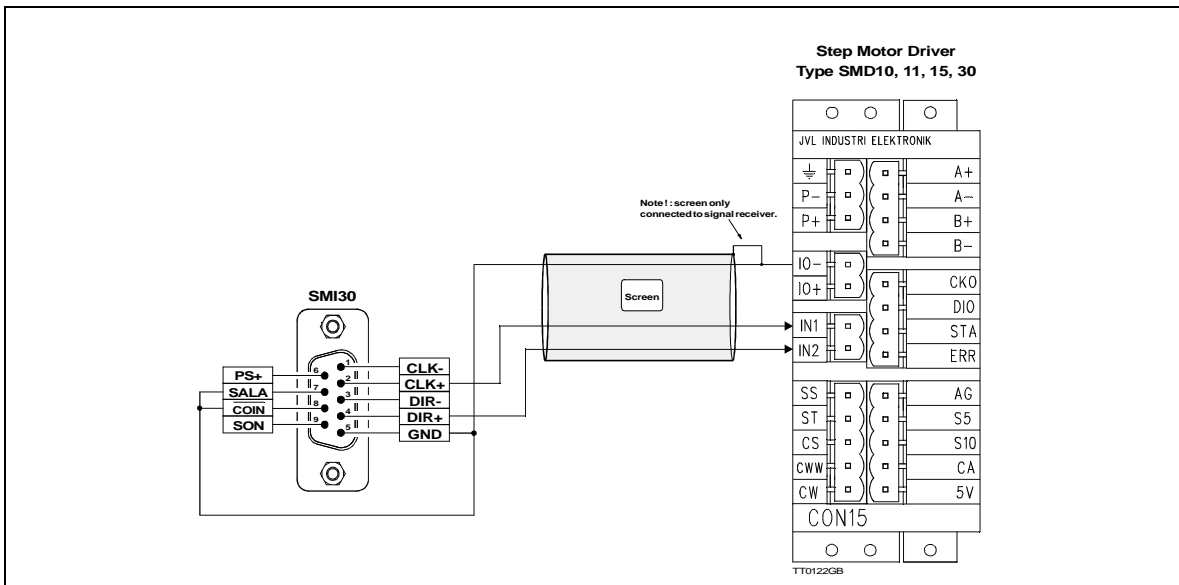
Example:

	SMI3x				SGD(A/B)			Motor (with 8192 p/rev.)		
	PR	VM	SR	CON	Cn-11	-24	-25	Velocity	Position	Max. Vel.
1	2048	1000	2000	1	2048	4	1	1000	8000	
2	2048	1000	2000	1	2048	1	1	250	2000	3295
3	8192	1000	2000	1	2048	1	1	1000	2000	3295
4	8192	1000	2000	2	2048	1	1	2000	4000	3295
5	8192	1000	2000	1	2048	4	1	4000	8000	

Combinations 1 and 3 should be preferred as in these cases the motor will run with the velocity and run the distance corresponding to VM and SR.



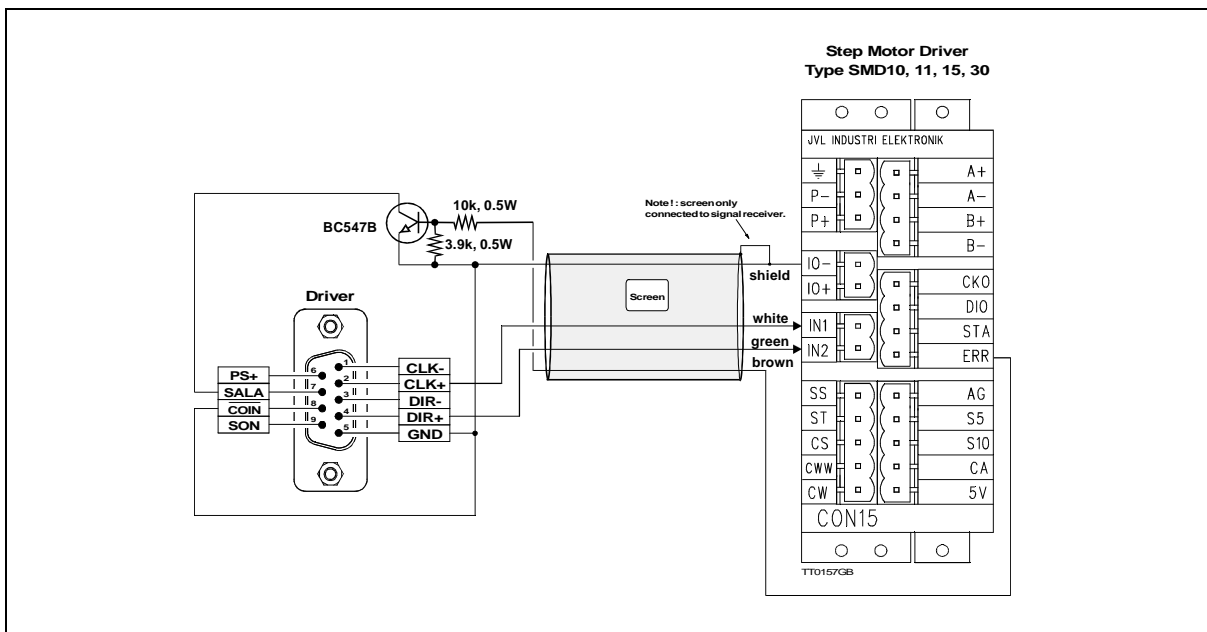
4.6 Connection to JVL step motor driver



4.6.1 Connecting a JVL step motor driver SMD10,11,15,30

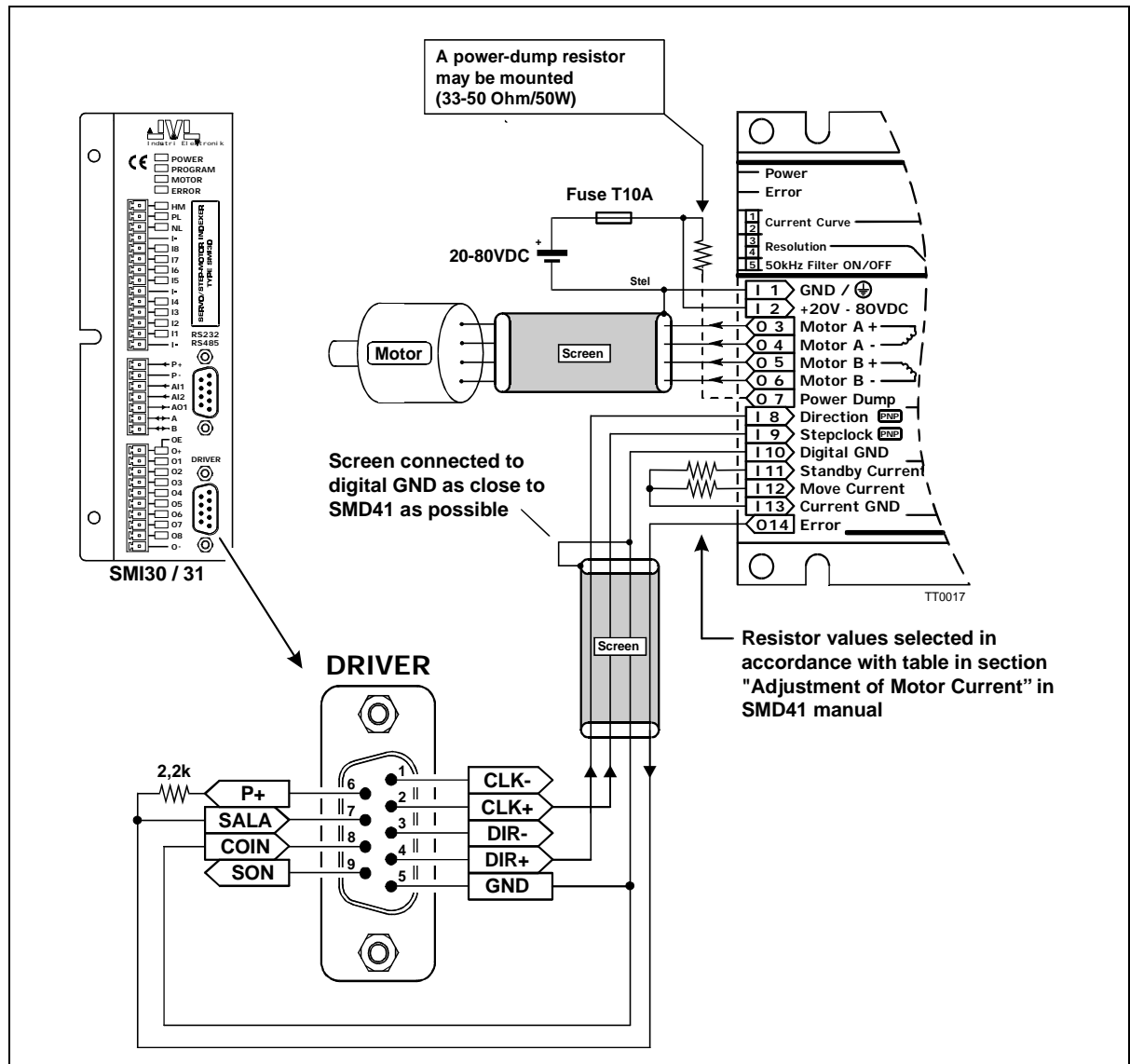
The illustration above shows how to connect a JVL step motor driver SMD10/11/15/30 to the Indexer.

It is recommended that shielded cable is used, with the shield connected to GND (Pin 5) on the SMI30-31. Do not use cable lengths longer than 2 meters. Remember to set the Pulses/rev. command (PR) according to the driver's pulses/rev. settings.



If it is required that the ERR (Error) output is used so that the SMI30 can report if the driver produces an error, special cable WI0020 (2m) must be used. The Error signal is inverted to that it matches the level at the SALA input. When using the WI0020 cable, remember to set CB15=0.

4.6 Connection to JVL step motor driver



4.6.2 Connecting a Ministep Driver SMD41

The above illustration shows how a typical connection is made between JVL Indexer SMI30 or SMI31 and a Ministep Step Motor Driver SMD41. It is recommended that screened cable is used for connecting the motor and the logic signals to the SMI3x in order to avoid spurious noise problems and to fulfil the requirements of CE conformity for the complete system.

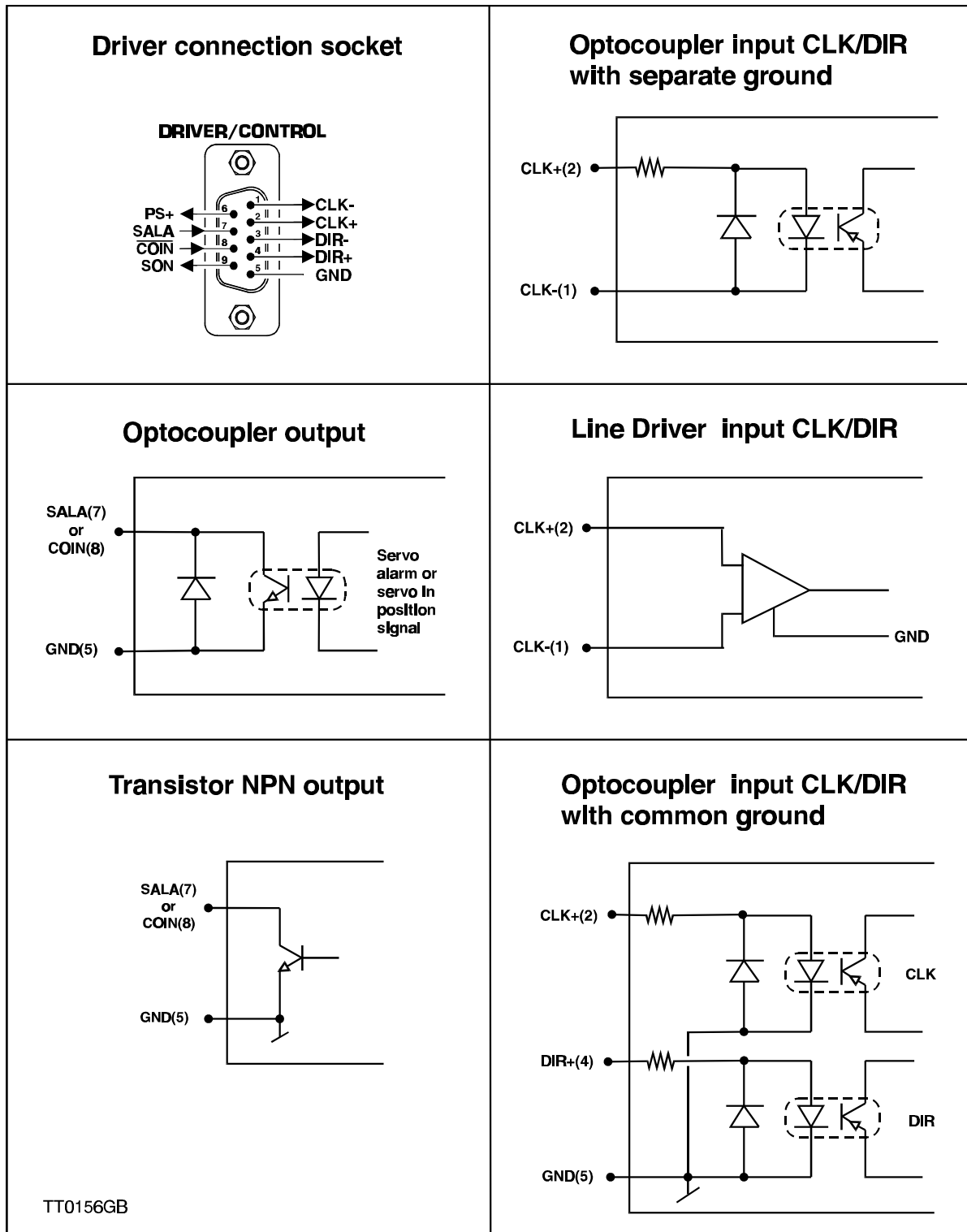
It is recommended that the cable length between the SMI3x and SMD41 does not exceed 2 m.

The following SMI3x registers must be set:

PR Pulses per motor revolution. This register must be set according to the number of steps per revolution selected on the SMD41.

CB15 Control flag for SALA (servo alarm). This flag is set to CB15=1 so that the SMI3x accepts logic 0 as the active level for the SALA input. This means that the SMI3x detects any error from the SMD41 when the "Error" Output goes to logic 0.

4.7 Connection to other selected drivers





Accessories etc. for SMI30/31 and SMC35A/B:**Software:**

MotoWare Programming and setup SoftWare for Windows 95,98,2000,NT

Cables:

RS232-9-1 Standard 3m RS232 programming cable
RS232-9-a-b RS232 cable for multipoint. a = number of units. b= length of cable (3 or 5 meter).
Yask-SMI30 SGDA Cable for Yaskawa SGDE/SGDA
Yask-SMI30-SGDB Cable for Yaskawa SGDB/SGDH
SMI30CON D-Sub connector for driver or controller connection

Extension Modules:

KDM10T/KDM10D Keyboard/Display module for 19" rack (D version) or panel mounting (T version)
IOM11 Input/Output module. 16 extra inputs and 8 extra outputs for mounting in 19" rack
DIS11 Panel Mounting LED display

Power Supplies:

PSU24-1 24VDC/100W Power Supply for Indexer/controller and In- and Outputs
PSU80-2 80VDC/200W power supply for SMC35x controller.(115 or 230 VAC input)
KITPSU80-2 Transformer, capacitor, diodes for low cost 80VDC power supply

Step motors for SMC35x:

(SMC35B is used if the motor is connected in parallel. 3Amp controller SMC35A can be used if the motor is connected in series.

MST171A01	High torque step motor. 200 step/rev. 0.3Nm	1.9Amp
MST172A01	High torque step motor. 200 step/rev. 0.56Nm	1.9Amp
MST230B01	High torque step motor. 200 step/rev. 0,6Nm	5.4Amp.
MST231B01	High torque step motor. 200 step/rev. 1.06Nm	5.5Amp.
MST232B01	High torque step motor. 200 step/rev. 1.87Nm	6.7Amp.
MST340B01	High torque step motor. 200 step/rev. 3.2Nm	6.0Amp.
MST341B01	High torque step motor. 200 step/rev. 4.6Nm	6.5Amp.
MST342B01	High torque step motor. 200 step/rev. 8.5Nm	6.3Amp.

External Drivers for SMIx or SMC35 1½ axis operation:

SMD41xx series Ministep Drivers

Connectors

CS0107 7-pole connector
CS0110 10-pole output connector
CS0002 14-pole input connector
CS0205 5-pole connector to motor (only SMC35)

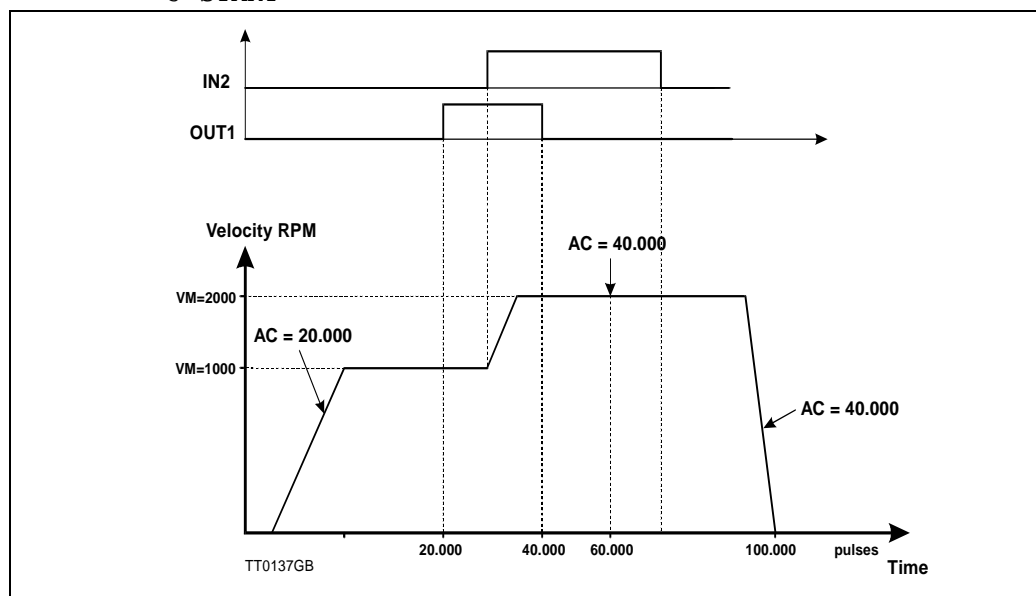
4.9

Program examples

4.9.1 Multi-tasking

The Indexer has a built-in signal processor to handle the actual motor run. Thus the Indexer has the capability for operating the motor while the main program continues execution in the background. This takes place without decreasing the speed of program execution. When a run command is executed (SR+, SZ+, SP+), the length is transferred from the main processor to the signal processor, which then handles the entire motor run. The main processor continues execution of its program and can continuously change the velocity, acceleration ramps, lengths, and, for instance, set outputs, etc. This gives the possibility for performing very advanced motion profiles and thus solving all kinds of tasks.

```
OUT=0
OUT4=1
SON=1
PR=8192           ;Set pulses per revolution
:START  WAIT RS=0
        OUT7=0
        AP=0      ;set position.
        AC=20000  ;set acc/dec to 500 RPM/sec.
        VM=1000  ;Set velocity to 1000 RPM.
        SR=100000 ; run max. length 100000 pulses.
:LOOP   IF AP>20000 ;when position > 20000
        OUT1=1     ;set output1
        IF AP>30000 ;when position > 30000
        OUT7=1     ;OUT7 connected to IN2
        IF AP>40000 ;when position > 40000
        OUT1=0     ;reset output1.
        IF IN3=1   ;if IN3 is activated
        J: BREAK   ;jump out of LOOP
        IF IN2=1   ;if IN2 is activated shift
        VM=2000   ;velocity to 2000 RPM.
        IF AP>60000 ;when position > 8000 shift
        AC=40000  ;deceleration to 10000 RPM/sec.
        IF RS<>0  ;If motor is still running
        J: LOOP    ;20000 pulses jump to LOOP
        WAIT IN1=1
        J: START
```



(continued)

The above illustration shows how the motion profile can be changed according to external events or internal positions. The more events are introduced, the longer the loop scan time will be. It is therefore recommended that the number of "IF" conditions is kept to a minimum. The example above for instance requires that IN2 be activated during the whole loop program in order that the function is reliable, as measurement only takes place once within the loop program. (See *Command timing*, page 159. See also *Monitoring of inputs and errors*, page 46, and the *Interrupt (INT)* command, page 77, for information on how to control program flow using interrupts.)

4.9

Program examples

4.9.2 Use of analogue input

```
; This program shows how to use the analogue inputs for variable speed
; in +- direction via a joystick. 0-5 Volt connected to AI1.
; Automatic mode via joystick and manual mode via inputs and preset distance +-
; Components used: SMI30, SMD41 ministep driver, step motor, analogue joystick
; 5kOhm.
:INIT      PR=1600          ; Ministep resolution. 1600 pulses/rev
           R6=3            ; Use R6 to scale the analogue input AI1.
                               ; (VM=AI1/R6)
           OUT=0          ; Clear all 8 outputs
           VS=10          ; Start speed
           VM=300         ; Maximum velocity
           AC=1000        ; Acc/deceleration in RPM/s
           R4=2200        ; Analogue value where motor stands still
           R5=300         ; Dead band +- where motor must stand
                               ; still
           R2=2700        ; Upper limit value
           R3=1700        ; Lower limit value
           CB2=0          ; 8-BIT resolution with oversampling
           CB3=0          ; Value 0-16360
:MAIN      IF IN1=1        ; If manual mode selected
           J:MANUAL       ; Jump to MANUAL
           R1=AI1/R6      ; Read DA converter and divide by 3.
           IF R1>R2       ; If analogue value > upper limit ..
           J:MOTOR+      ; .. run the motor in + direction
           IF R1<R3       ; If analogue value < upper limit ..
           J:MOTOR-      ; .. run the motor in - direction
           SH             ; If in dead band, smooth stop the motor
           WAIT RS=0      ; Wait until motor stands still
           J:MAIN        ; Jump to main, test again
:MOTOR+    R10=R1-R2      ; Adjust analog value
           IF R10<=VS     ; If calculated speed < start speed
           VM=VS          ; .. set speed to start speed
           ELSE           ; .. else
           VM=R10         ; Set speed to calculated speed
           IF RS=0        ; If motor not running
           SR=10000000    ; .. start motor in positive direction
           J:MAIN        ; Jump to main
:MOTOR-    R10=R3-R1      ; Adjust analogue value
           IF R10<=VS     ; If calculated speed < start speed
           VM=VS          ; .. set speed to start speed
           ELSE           ; .. else
           VM=R10         ; Set speed to calculated speed
           IF RS=0        ; If motor not running
           SR=-10000000   ; .. start motor in positive direction
           J:MAIN        ; Jump to main
:MANUAL    IF IN2=1       ; If manual mode and manual run in posi-
                               ; tive direction
           J:MANPLUS     ; .. jump
           IF IN3=1       ; If manual mode and manual run in nega-
                               ; tive direction
           J:MANMINUS    ; .. jump
           J:MAIN        ; Jump to main and check again
:MANPLUS   SR=100        ; Run distance in positive direction
           WAIT RS=0      ; Wait for motor stopped
           J:MANUAL      ; Jump to main and check again
:MANMINUS  SR=-100       ; Run distance in negative direction
           WAIT RS=0      ; Wait for motor stopped
           J:MANUAL      ; Jump to main and check again
```


4.9 Program examples

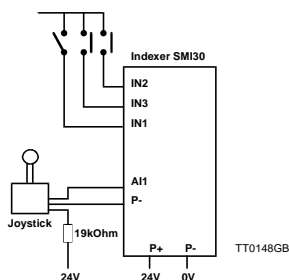
4.9.3 Use of analog output and input

This program shows how it is possible to control the speed in both directions using two potentiometers.

5 volts for the supply of two 5kOhm potentiometers is generated by the internal 5V output which is set to supply 5V constantly. Input 1 and 2 are mounted with 2 switches that determine whether the motor runs in one direction or the other.

```

R1=2          ; Use R1 to scale the analogue Input
              ; AI1
R2=2          ; Use R2 to scale the analogue input
              ; AI2
OUT3=1        ; Possibly use OUT3 to control IN1
OUT4=1        ; Possibly use OUT4 to control IN2
CND2=7        ; Set analogue output to 8-bit
CTM2=7        ; Set TIMER2 as analogue output
AOUT=255      ; Set 5V out
:INIT         AC=1000      ; Set ACC/DEC slope to 1000RPM/SEC
              VS=100      ; Set Min. Speed to 100 RPM
              SON=1       ; Servo On
:START        IF IN1=1    ; If GO-Positive switch is activated
              J:GOPOS     ; .. then Jump to subroutine
              IF IN2=1    ; If Go-negative switch is activated
              J:GONEG     ; .. then jump to subroutine
              J:START     ; Test inputs again
:GOPOS        VM=AI1/R1   ; Set Velocity to value of Analogue
              ; input1
              SR=10000000 ; Move max. distance, while ..
:WAIT1        VM=AI1/R1   ; .. changing the speed
              IF IN1=1    ; Move until STOP is signalled
              J:WAIT1     ; .. on input 1
              SH          ; If STOP Decelerate motor speed
              WAIT RS=0   ; Wait until motor has reached its
              ; destination point
              J:START     ; Test inputs again
:GONEG        VM=AI2/R2   ; Set Velocity to value of Analogue
              ; input 2
              SR=-10000000 ; Move max. distance, while ..
:WAIT2        VM=AI2/R2   ; .. changing the speed
              IF IN2=1    ; Move until STOP is signalled
              J:WAIT2     ; .. on input 2
              SH          ; If STOP, Decelerate motor speed
              WAIT RS=0   ; Wait until the motor has reached its
              ; destination point
              J:START     ; Test inputs again
```



4.9

Program examples

4.9.4 Test-program, quick start

```

R1=0 ;Set register 1 to Zero
:START OUT=#00000000 ;Reset all outputs
OUT1=1 ;Turn on OUT1
D=30 ;Wait 0.3 sec.
OUT1=0 ;Turn off OUT1
OUT2=1 ;Turn on OUT2
D=30 ;Wait 0.3 sec.
OUT2=0 ;Turn off OUT2
OUT3=1 ;Turn on OUT3
D=30 ;Wait 0.3 sec.
OUT3=0 ;Turn off OUT3
OUT4=1 ;Turn on OUT4
D=30 ;Wait 0.3 sec.
OUT4=0 ;Turn off OUT4
OUT5=1 ;Turn on OUT5
D=30 ;Wait 0.3 sec.
OUT5=0 ;Turn off OUT5
OUT6=1 ;Turn on OUT6
D=30 ;Wait 0.3 sec.
OUT6=0 ;Turn off OUT6
OUT7=1 ;Turn on OUT7
D=30 ;Wait 0.3 sec.
OUT7=0 ;Turn off OUT7
OUT8=1 ;Turn on OUT8
D=30 ;Wait 0.3 sec.
SR=8000 ;Move 8000 pulses forward
WAIT RS=0 ;Wait until motor has stopped
SR=-8000 ;Move 8000 pulses backward
Wait RS=0 ;Wait until motor has stopped
R1=R1+1 ;Inc. Register1
IF R1 < 5 ;If register R1 is less than 5
J:START ;Jump to START
ELSE
J:STOP
:STOP
```

4.9 Program examples

4.9.5 TurnMaster Mode

Example

```
VM=100
AC=100
SON=1
CB30=1           ;Can also be 2 or 3
PR=1600         ;Ministep mode 1600 puls/rev selected
RX12=4800       ;Gear of 3 is used
:START If IN=1
SP=0
WAIT RS=0
IF IN2=1
SP=2400
WAIT RS=0

IF IN3=1
SP=1200
WAIT RS=0
J:START
```

4.9

Program examples

4.9.6 1½ axis mode

This program shows how 1 stepmotor and 1 servo motor can be controlled from a single SMC35B.

Using 1 program it is possible to control 2 motors but not at the same time. (2 motors can be run at the same time, but synchronously and with the same step-pulse rate).

The step motor is controlled from the internal driver and the servomotor and driver are controlled via step-pulse and direction signals on the connector "setup".

```

:INIT_STEP PR=1600; Pulses/rev
CS=1000 ; X-axis motor current at standstill
CT=2000 ; X-axis motor current when accelera-
; tion and running
VS=100 ; Start speed in RPM
AC=800 ; Acceleration in RPM/s
VM=300 ; Top speed in RPM
:INIT SON=1 ; Activate internal driver. X axis.
R1=0 ; position for x-axis. Step motor
R2=0 ; position for y-axis. Servo motor
R3=1000 ; relative length X and Y should move
AP=0 ; Zero position counter
:MAIN JS:STEP_ON ; activate X-axis
VM=400 ; Change speed
SR=3000 ; Run X-axis relative position forward
WAIT RS=0 ; Wait until position
R1=AP ; Set R2 = Actual position
D=10 ; wait 100ms
JS:SERVO_ON ; activate Y-axis
VM=3000 ; change speed
SR=20000 ; Run Y-axis relative position forward
WAIT RS=0 ; Wait until position
R2=AP ; Set R2 = Actual position
D=10 ; wait 100ms
JS:STEP_ON ; activate X-axis and Y-axis
VM=40 ; Change speed for X and Y axes
AC=800 ; change acceleration for both axes
SR=R3 ; Run X and Y axes at the same time.
WAIT RS=0 ; wait until position reached
R1=R1+R3 ; update position counter for X-axis
R2=R2+R3 ; update position counter for Y-axis
D=100 ; wait 100ms
J:MAIN ; jump to main
:STEP_ON PR=1600 ; Pulses/rev for step motor axis
AC=800 ; General acceleration for Y-axis step motor
AP=R1 ; Preset position with last value for X-axis
CB36=1 ; Switch to X axis active
RET ; Return
:SERVO_ON PR=8192 ; Pulses/rev for servo axis
AC=10000 ; general acceleration for Y-axis servo-
; motor-step motor
AP=R2 ; Preset position with last value for Y-axis
CB36=2 ; Switch to Y axis active
RET ; Return
:STEP_ON PR=1600 ; Pulses/rev for step motor axis. Speed and
; acc for servo
; axis will not be valid.
AC=800 ; Acceleration for X-axis stepmotor
CB36=3 ; Enable X and Y axes to run at the same time
```

4.9

Program examples

4.9.7 Use of KDM10 and registers

This program shows the use of absolute registers, with an SMI31/SMC35 at address 1, and a KDM10 at address 3. The program uses the registers 100-199. The register pointer calculation is done at the "REG_POINT" label.

At the first menu, the user has the possibility to change program number, change data, or run the motor using the specified data. If <F1> is pressed, the change program routine is run and the corresponding menu displayed; if <F2> is pressed, the data change menu is shown; and if <F6> is pressed, the motor runs the specified length and speed.

```
;R1=CHOOSEN PROGRAMNUMBER
;R2=ABSOLUTE REGISTERPOINTER
;R3=ABSOLUTE REGISTERPOINTER (WORK)

MR2                ;READ ALL REGISTERS
SON=1              ;SET SERVO SIGNAL ON
R1=1               ;SET PROGRAMNUMBER
:MENU              AO3.1                ;CLEAR DISPLAY
PRINT3.1."PRGNO=" ;PRINT MENU AND ACTUAL PRGNO.
PRINT3.0.R1
PRINT3.10."<F1>CHANGE PRG"
PRINT3.41."<F2>CHANGE DATA <F6>MOVE"

:READ_MENUKB      R99=INPUT3.222        ;READ KEYBOARDINPUT
IF R99=0           ;IF F1 IS PRESSED
J:CHANGE_PRGNO    ;...THEN JUMP TO CHANGE_PRGNO
IF R99=1           ;IF F2 IS PRESSED
J:CHANGE_DATA     ;...THEN JUMP TO CHANGE_DATA
IF R99=5           ;IF F6 IS PRESSED
J:MOVE_MOTOR      ;...THEN JUMP TO MOVE_MOTOR
J:READ_MENUKB     ;JUMP TO READ_MENUKB

:CHANGE_PRGNO     AO3.1                ;CLEAR DISPLAY
PRINT3.1."ACTUAL PRGNO="
PRINT3.0.R1
PRINT3.41."NEW PRGNO="
R99=INPUT3.255    ;READ INPUT FROM KEYBOARD
IF R99=0           ;IF "NOTHING" ENTERED
J:NO_PRG_CHANGE   ;...THEN JUMP NO_PRG_CHANGE
IF R99<1           ;IF ENTERED VALUE IS BELOW 1
J:PRGNO_MINMAX    ;...THEN JUMP PRGNO_MINMAX
IF R99>10         ;IF ENTERED VALUE IS ABOVE 10
J:PRGNO_MINMAX    ;...THEN JUMP PRGNO_MINMAX
R1=R99            ;SET R1 = R99
MS2               ;SAVE ALL REGISTERS IN EEPROM

:NO_PRG_CHANGE    JS:ACT_DATA          ;JUMP TO SUBROUTINE ACT_DATA
J:MENU            ;JUMP TO MENU

:PRGNO_MINMAX     PRINT3.1."ONLY DIGITS BETWEEN 1 AND 10"
D=400             ;WAIT 4 SECONDS
J:CHANGE_PRGNO
```

4.9

Program examples

```
:CHANGE_DATA JS:ACT_DATA          ;JUMP TO SUBROUTINE ACT_DATA
AO3.1        ;CLEAR DISPLAY
PRINT3.1."<F1>CHANGE LENGTH <F6>RETURN"
PRINT3.41."<F2>CHANGE SPEED"

:DATA_MENUKB R99=INPUT3.222      ;READ KEYBOARDINPUT
IF R99=0     ;IF F1 IS PRESSED
J:CHANGE_LENGTH ;...THEN JUMP TO CHANGE_LENGTH
IF R99=1     ;IF F2 IS PRESSED
J:CHANGE_SPEED ;...THEN JUMP TO CHANGE_SPEED
IF R99=5     ;IF F6 IS PRESSED
J:MENU       ;...THEN JUMP TO MENU
J:DATA_MENUKB ;JUMP TO DATA_MENUKB

:CHANGE_LENGTHR3=R2             ;SET R3 = R2
AO3.1        ;CLEAR DISPLAY
PRINT3.1."ACTUAL LENGTH="
PRINT3.0.R[R3] ;PRINT REGISTER R[R3]
PRINT3.41."NEW LENGTH="
R99=INPUT3.255 ;READ INPUT FROM KEYBOARD
IF R99=0     ;IF "NOTHING" ENTERED
J:CHANGE_DATA ;...THEN JUMP CHANGE_DATA
R[R3]=R99    ;SET REGISTER R[R3] TO ENTERED
; VALUE
MS2         ;SAVE ALL REGISTERS TO EEPROM
J:CHANGE_DATA ;JUMP TO CHANGE_DATA

:CHANGE_SPEED R3=R2+1          ;SET R3 = R2 + 1
AO3.1        ;CLEAR DISPLAY
PRINT3.1."ACTUAL SPEED="
PRINT3.0.R[R3] ;PRINT REGISTER R[R3]
PRINT3.41."NEW SPEED="
R99=INPUT3.255 ;READ INPUT FROM KEYBOARD
IF R99=0     ;IF "NOTHING" ENTERED
J:CHANGE_DATA ;...THEN JUMP CHANGE_DATA
R[R3]=R99    ;SET REGISTER R[R3] TO ENTERED
;VALUE
MS2         ;SAVE ALL REGISTERS TO EEPROM
J:SKIFT_DATA ;JUMP TO CHANGE_DATA

:ACT_DATA JS:REG_POINT         ;JUMP TO SUBROUTINE REG_POINT
AO3.1        ;CLEAR DISPLAY
PRINT3.1."ACTUAL LENGTH="
PRINT3.0.R[R3] ;PRINT REGISTER R[R3]
R3=R2+1     ;SET R3 = R2 + 1
PRINT3.41."ACTUAL SPEED="
PRINT3.0.R[R3] ;PRINT REGISTER R[R3]
D=200      ;WAIT 2 SECONDS
RET        ;RETURN
```

4.9

Program examples

```
:REG_POINT  R2=R1-1           ;SET R2 = R1 - 1
             R2=R2*10         ;EVERY RECIPE CONTAINS 10 REGIS
                                     ;TERS
             R2=R2+100        ;THE RECIPE STARTS AT REGISTER 100
             R3=R2             ;SET R3 = R2
             RET              ;RETURN

:MOVE_MOTOR JS:REG_POINT      ;JUMP TO SUBROUTINE REG_POINT
             AO3.1            ;CLEAR DISPLAY
             PRINT3.1."MOVING "
             PRINT3.0.R[R3]    ;PRINT REGISTER R[R3]
             PRINT3.0." STEP WITH"
             R3=R2+1          ;SET R3 = R2 + 1
             PRINT3.41.R[R3]   ;PRINT REGISTER R[R3]
             PRINT3.0." RPM"
             VM=R[R3]         ;SET VM TO REGISTER R[R3]
             SR=R[R2]         ;SET SR TO REGISTER R[R2]
             WAIT RS=0        ;WAIT UNTIL MOTOR HAS STOPPED
             PRINT3.0."OK!"    ;PRINT "OK"
             D=200            ;WAIT 2 SECONDS
             J:MENU           ;JUMP TO MENU
```

4.10

Command timing

In critical-timing applications it can be important to know the time for each command. Execution time is the time a command will use in a program. Interpretation time is the time a command will use if it was sent to the Indexer/Controller via the RS232/RS485. The time is from receipt of <CR> until <CR> is sent again to acknowledge command received ok. All measurements are based on 9600 bit/sec.

The table below shows the execution time for the most typical Indexer commands.

Command	Description	Execution time (msec.) typical	Interpretation time (msec)	Bytes in EE-PROM
AC=x	Set acceleration	10.6		13
AO 1.21	Set output on module	6.0		6
AP	Send actual position to RS232 (9600 baud)	9.2		7
CB2=1	Control flag	3.0		13
IF INx=x	Statement	3.2		14
If IN=#xxxxxxx	Statement	4.5		14
If R1=R2	Statement	3.2		13
INT1....RETI	Interrupt + Return INT	2.1		2+1
J:LABEL	Jump to label	0.54		3
JS:SUB....:SUB RET	Jump to subroutine and back again	1.23		3+1
OUTX=x	Set output x = x	2.6		13
PRINT 1.1."AB"	Print text on display	11.0		9
R1	Send R1 to RS232 (9600 baud)	9.8		7
R1=AI1	Mathematic. Read analog input in 8-bit	4.5		12
R1=AI1	Mathematic. Read analog input in 14-bit	66.0		12
R1=R2 / 15	Mathematic	5.8		18
R1=R2 / R3	Mathematic	5.8		17
R1=R2 -100.000.000	Mathematic	4.3		18
R1=R2+100.000.000	Mathematic	4.3		18
R1=R2 * 100.000.000	Mathematic	4.4		18
RX=x	Mathematic	2.7		13
R1=INPUT 1.201	Read mailbox on display	6.8		
SH	Soft Halt	0.8		1
SON=x	Servo On/Off	3.7		13
SR=0	Run motor relative	6.6		13
SR=x	Set relative distance	6.6		13
VM=x	Set top speed	9.5		13
VS=x	Set start speed	9.4		13
WAIT IN1=1	Wait statement	3.3		14
WAIT R1=R2	Wait statement	3.3		13

If the Indexer/Controller is controlled from a PC or PLC that sends commands to be executed immediately, it is often desirable to know how much time will elapse.

The total time can be divided into 4 separate stages:

- 1) Transmission time from the PC/PLC to the Indexer/controller via the RS232.
- 2) Interpretation time of the command. This is the time the Indexer/Controller uses to check whether the command is valid and any values are within allowable ranges. The command interpretation time is given in the table on the previous page.
- 3) Execution of the command. This is time taken for the command itself to execute, in the same way as if it were part of a program. The execution time is given in the table on the previous page.
- 4) Transmission time of the response from the Indexer/Controller to the PLC/PLC via the RS232.

Example:

SR=10000 is to be sent to indexer/controller with address 1. The baud rate is 9600 bit/s. The command to be sent is therefore 1SR=10000<CR>, in all a total of 10 characters. Each character comprises approximately 10 bits, so the transmission rate is 960 characters/s. 10 characters therefore take 10.4 ms to transmit. The Indexer responds with the reply Y<CR>, i.e. 2 characters. This takes 2.08 ms. In accordance with the table on the previous page, the total calculation is as follows: Total time = 10.4ms + 5.9ms + 7ms + 2.08ms = 25.38ms.

If a baud rate of 19200 bit/sec was selected, the transmission time would be halved, corresponding to a total time for transmission, interpretation and execution of 19.14ms.

4.11

Connection tables

The connection tables below make it easier to see the connections to PLCs, other connectors, etc., so that fault finding will be easier. For example, the tables can be used to indicate the numbers of the individual connectors on a PLC in the "Connector no." column and to specify what the connection is used for under the "Application" column (e.g. "Start signal from PLC", "Vacuum on", "Air cylinder up", etc.).

Connector no,	Designation	Description	Application
	HM	Home Input	
	PL	Positive limit	
	NL	Negative limit	
	I-	Gnd for Input	
	I8	Input 8	
	I7	Input 7	
	I6	Input 6	
	I5	Input 5	
	I-	Gnd. for Input	
	I4	Input 4	
	I3	Input 3	
	I2	Input 2	
	I1	Input 1	
	I-	Gnd. for Input	
	P+	Supply +10 - 35V DC	
	P-	Supply Ground	
	AI1	Analog Input1	
	AI2	Analog Input2	
	AO	Analog Output (0 - 5VDC)	
	A	RS485 Terminal A	
	B	RS485 Terminal B	
	O+	Output Supply +5 - 30VDC	
	O1	Output 1	
	O2	Output 2	
	O3	Output 3	
	O4	Output 4	
	O5	Output 5	
	O6	Output 6	
	O7	Output 7	
	O8	Output 8	
	O-	Gnd. for Outputs	

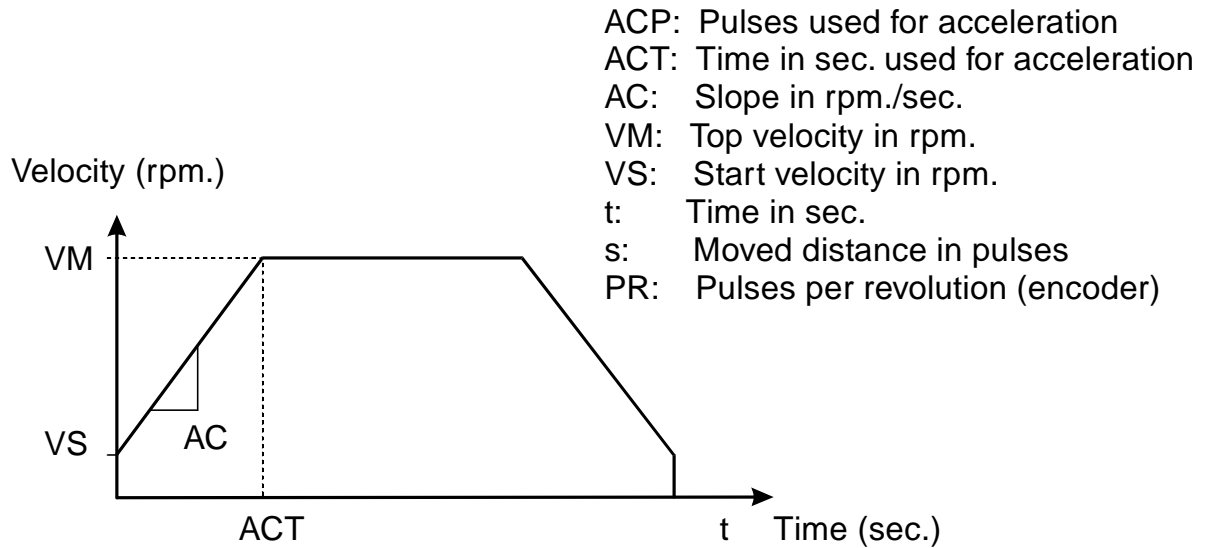
4.11

Connection tables

Connection of SMI30, RS232/RS485 SUB D Connector (female)			
Pin no.	Designation	Description	Application
1	Common	Chassis	
2	Rx	RS232 Receive	
3	Tx	RS232 Transmit	
4	A	(A RS485)	
5	Gnd.	Signal Gnd.	
6	Not used	Not used	
7	Tx-PD	Tx Pull Down	
8	Term.	RS485 Term.	
9	B	(B RS485)	

Connection of SMI30, Driver SUB D Connector (male)			
Pin no.	Designation	Description	Application
1	CLK-	Clock pulse-	
2	CLK+	Clock pulse+	
3	DIR-	Direction signal-	
4	DIR+	Direction signal+	
5	GND.	Signal Gnd.	
6	PS+	Voltage output	
7	SALA	Alarm input	
8	COIN	Servo-in position input	
9	SON	Servo on/off output	

4.12 Calculation of motor movement



$$ACP = \frac{(VM + VS) * ACT * PR}{120} \quad [\text{Pulses}]$$

$$ACP = \frac{(VM^2 - VS^2) * PR}{120 * AC} \quad [\text{Pulses}]$$

$$AC = \frac{VM - VS}{ACT} \quad [\text{RPM/sec.}]$$

$$s = \frac{AC * t^2 * PR}{120} \quad [\text{Pulses}]$$

(Valid if VS=0)

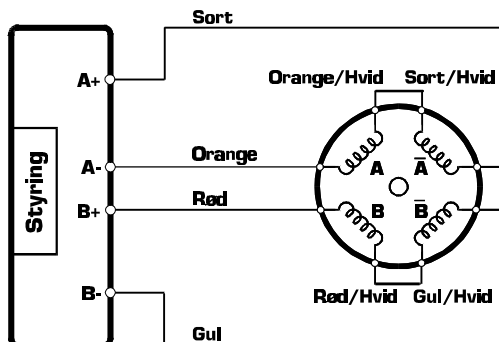
$$s = \frac{VM * t * PR}{60} \quad [\text{Pulses}]$$

(Valid if velocity is constant)

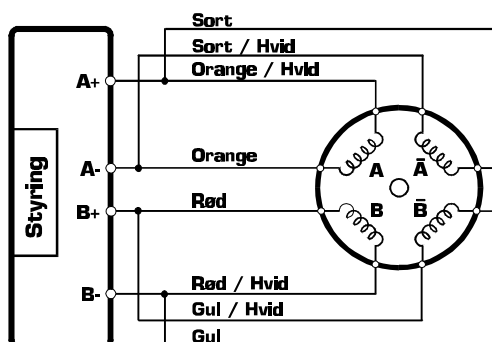
TT0151GB

4.13 Motor Connections (SMC35 only)

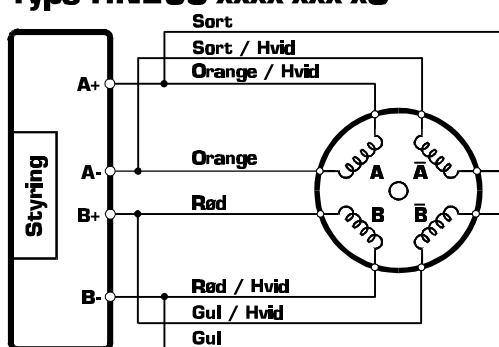
Forbindelse af JVL motor (serie) Type MST230/31/32 og MST340/41/42



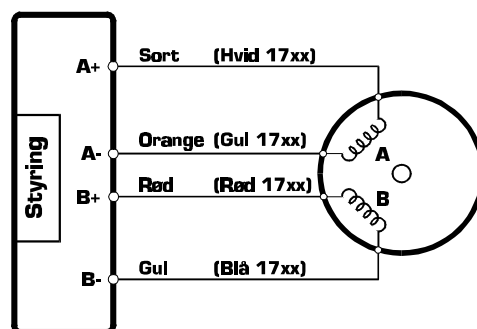
Forbindelse af JVL motor (parallel) Type: MST230/31/32 og MST340/41/42



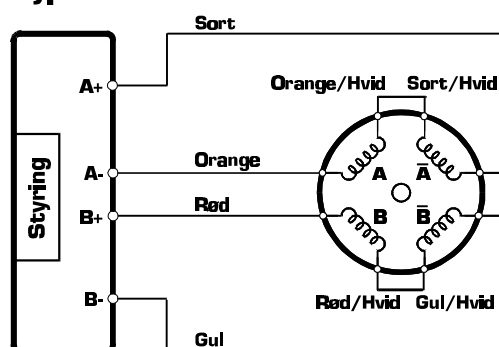
Forbindelse af MAE motor (parallel) Type HY200-xxxx-xxx-x8 Type HN200-xxxx-xxx-x8



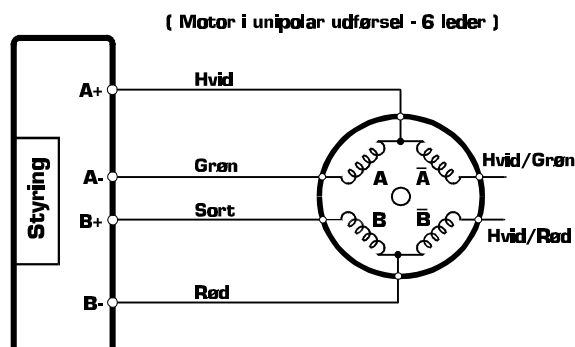
Forbindelse af MAE motor Type HY200-xxxx-xxx-x4



Forbindelse af MAE motor (serie) Type HY200-xxxx-xxx-x8



Forbindelse af MAE motor (unipol.) Type HY200-1xxx-xxxxx6

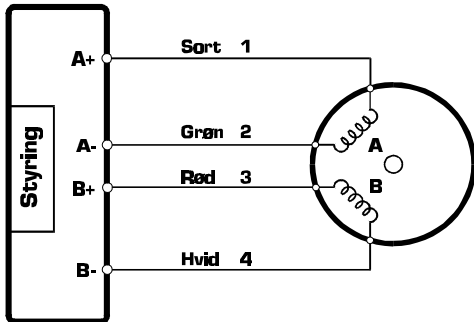


Bemærk: Omdrejningsretningen kan vendes ved at ombytte A+ og A-

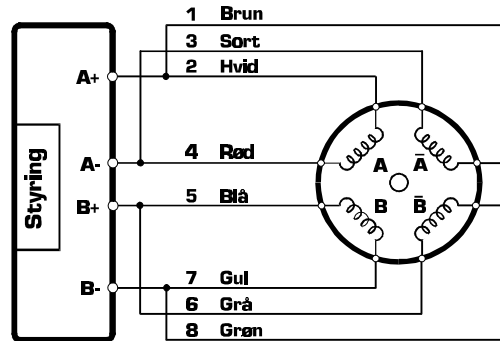
TT0162GB

4.13 Motor Connections (SMC35 only)

**Forbindelse af Zebotronics motor
Type : SMxxx.x.xx.x (4 terminaler)**

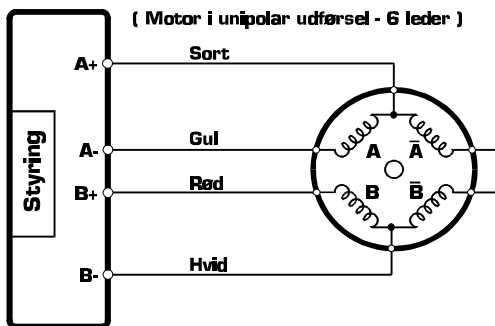


**Forbindelse af Zebotronics motor
Type : SMxxx.x.xx.x (8 terminaler)**

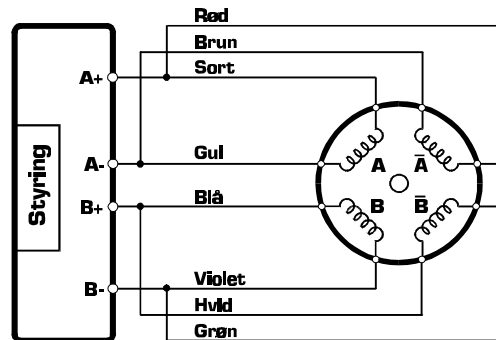


SM87/SM107/168.x.xx ↑ SM56.x.xx

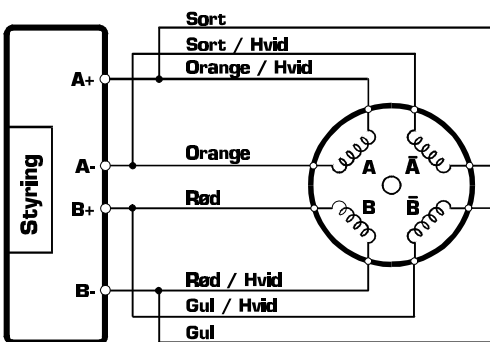
**Forbindelse af Vexta motor
Type PH2xx.xxx/PK2xx.xxx**



**Forbindelse af Phytron motor
Type ZSx.xxx.x,x**



**Forbindelse af Vexta stepmotor
Type : PH2xx-xxx**



TT0163GB

Bemærk: Omdrejningsretningen kan vendes ved at ombytte A+ og A-

Manufacturer's declaration of conformity

The undersigned hereby declares that the following equipment has been designed and manufactured in accordance with the protection requirements in EU directive 89/336/EEC concerning electromagnetic compability (EMC)

Identification of equipment:

Category: Motor Controller

Manufacturer: JVL Industri Elektronik A/S

Type: SMI30, SMI31 Servo/Step Motor Indexer

Manufacturer data:

JVL Industri Elektronik A/S

Blokken 42

DK-3460 Birkerød

Denmark

Tel. +45 45 82 44 40

Fax. +45 45 82 55 50

E-mail: jvl@jvl.dk

Internet: <http://www.jvl.dk>

The following standards are used as the basis for this declaration:

Emission - EN50081-2 1993

Immunity - EN50082-2 1994

Remarks:

The CE-mark only indicates conformity with the EMC-directive 89/336/EEC

Date: 1. January 1996



Bo V. Jessen

Head of Development

TT0153GB

LX0007-01GB

Revised 14.2.1996

EU - Declaration of Conformity

The undersigned hereby declare that the following unit fulfils the protection requirements of EU-directive 89/336/EEC concerning electromagnetic compatibility (EMC).

Identification of the product

Category: *Motor Controller*

Manufacturer: *JVL Industri Elektronik A/S*

Type: *SMC35A and SMC35B Step Motor Controller*

Manufacturer's name and address:

JVL Industri Elektronik A/S

Blokken 42

DK-3460 Birkerød

Tlf. +45 45 82 44 40

Fax. +45 45 82 55 50

E-mail: jvl@jvl.dk

Internet: <http://www.jvl.dk>

Conformance to the following standards :*

Emission - EN50081-2 1993

Immunity - EN50082-2 1994

Supplementary Information :

The product complies only with the requirements of the EMC-Directive 89/336/EEC.*

July 2000



Bo V. Jessen

Head of Development

Index

- Subtraction operator 41

Symbols

! (Indexer type) command 48, 51

Binary notation 41

(Delete EEPROM) command 49

* Multiplication operator 41

+ Addition operator 41

/ Division operator 41

; Semi Colon, Command Delimiter 29, 34

< Less than operator 41

<= Less than or equal to operator 41

<> Not equal to operator 41

= Equal to (value assignment) operator 41

> Great than operator 41

>= Greater than or equal to operator 41

? (Show set-up) command 48

Numerics

1½.axis mode, program example 155

1½-axis Control Flag, CB36 120

A

AC (Acceleration) command 49, 159

Acceleration (AC) command 49, 159

Acceleration pulses (ACP) command 49

Acceleration time (ACT) command 50

ACP (Acceleration pulses) command 49

ACT (Acceleration time) command 50

Activate flag in external module (AO) command 53, 159

Actual Position (AP) command 55, 159

Actual Position Pulses (APP) command 55

Addition operator, +, 41

ADDR (Address) command 51

Address 28

Address (ADDR) command 51

AI1, AI2 (Analogue Input) commands 52

Analogue Inputs 26, 161

AI1, AI2 commands 52

Analogue Conversion Flags CB2/CB3
110, 159

Program Examples 151–152

Analogue Output 27, 161

AOUT command 54

Program Examples 152

AND operator 41

ANDL 42

AO (Activate flag in external module) command 53, 159

AOUT (Analogue Output) command 54

AP (Actual Position) command 55, 159

APP (Actual Position Pulses) command 55

Argument, Command argument 28

Arithmetic Expressions 40

Array functions 45

B

Baud Rate 28

Baud Rate (BAUD) command 56

Binary Notation 41

Bipolar motors 20

Bit operations 42

C

Cabling 19, 140–142, 144–145, 148, 161–162, 164–165

Capacitor 22–23

Carriage Return (Command Delimiter) 29

CB1 Direction Level Flag 110

CB10 Default Direction Flag 111

CB11 Disable Error E43-E47 Flag 111

CB12 Trigger Level Flag for INT1 112

CB13 Trigger Level Flag for INT2 112

CB14 Trigger Level Flag for INT3 113

CB15 register (SMI3x Indexer) 145

CB15 Servo Alarm Signal Flag 113

CB16 Motor in Position Flag 113

CB17 Enable Running Output (08) Flag 113

CB18 NSTART Trigger Level Flag 113

CB19 Disable Input Averaging Flag 113

CB2/CB3 Analogue Conversion Flags 110, 159

CB20 High-speed start trigger at IN1 114

CB21 High-speed start trigger at IN2 114

CB22 Diverse Flag 114

CB23 Electronic Gearing Flag 115

CB24 Position Reached Flag 115

Status Flags 115

CB25 Trigger Level Flag for NL Input 115

CB26 Trigger Level Flag for PL Input 115

CB27 Zero Search Flag 116

CB28 CVIx Frequency Range Flag 116

CB29 RS232 Activity Flag 116

CB30 Turntable Mode 116

CB31 Reserved, for internal use 117

CB32 Set/Read Interrupt status 117

CB33 User Output Error Mode 118

CB34 Reserved 118

Index

- CB35 Command acknowledge flag 118–119
- CB36 1½-axis control/chopper frequency flag 120
- CB37 Digital filtering on interrupt inputs flag 121
- CB38 Motor status (RS) when limit activated flag 121
- CB39 Diverse flag 121
- CB4 End-of-travel flag 110
- CB40 Software position limits flag 122
- CB5 110
- CB6 Error at User Output Flag 110
- CB7 Output Error Flag 111
- CB8 General Error Flag 111
- CB9 RS232 Communication Flag 111
- CE requirements 19
- Checksum 28–29
 - CHS (Checksum) command 56
 - Memory Checksum (MCHS) command 80
- CHS (Interface Checksum) command 56
- Clear Flag in external module (CO) command 59
- CLK Outputs 24–25
- CN1 (Counter/Timer 1) command 57
- CN2 (Counter/Timer 2) command 57
- CND1 (Counter/Timer 1 Divider) command 58
- CND2 (Counter/Timer 2 Divider) command 58
- CO (Clear Flag in external module) command 59
- COIN Output 24–25
 - Motor in Position Flag CB16 113
- Command 28
 - Argument 28
 - Checksum 28–29, 56
 - Command acknowledge flag, CB35 118–119
 - Command Description 47–109
 - Command Overview 129–131
 - Command Timing 159
 - Delimiter 28–29, 34
 - Line, max. no. of characters 29
 - Syntax 28
- Common Errors 139
- Communication 28–30
 - Address 28
 - Checksum 28–29, 56, 80
 - Command 28
 - Command syntax 28
 - Errors, see ES0 command 69–71
 - Protocol 28
 - Rate 28, 56
 - RS232 Activity Flag CB29 116
 - RS232 Communication Flag CB9 111
 - Synchronisation 29
 - Use of RS232/485 Interface Commands 34
- COMP (Compare Memory) command 59
- Compare Memory (COMP) command 59
- CON (Conversion Factor) command 60
- Connection
 - Analogue Inputs 26
 - Analogue Output 27
 - Connection Tables 161–162
 - Driver Outputs 24
 - End-of-travel Limit Inputs 16
 - Home Input 17
 - Indexer Front Panel Overview 8
 - of Indexer to a PC 30
 - Overview 14
 - Power Supply 22
 - RS232/RS485 Interface connection 28, 30
 - User Inputs 15
 - User Outputs 18
- Connection of motor 20–21, 164–165
- Connection of motor phases 21
- Control Flags 110–122
- Control of program flow 42–44, 68, 74, 78, 92–93
- Conversion Factor (CON) command 60
- Counter Mode (CTM1) command 63
- Counter Mode (CTM2) command 65
- Counter/Timer 1 (CN1) command 57
- Counter/Timer 1 Divider (CND1) command 58
- Counter/Timer 2 (CN2) command 57
- Counter/Timer 2 Divider (CND2) command 58
- CR (Command Delimiter) 28–29
- Critical Errors 69, 128

Index

- CS (Motor Current at Standby) Command 61
- CT (Motor Current when Running) command 61–62
- CTM1 (Counter Mode) command 63
- CTM2 (Counter Mode) command 65
- Current Frequency (CVI) command 67
 - CVIx Frequency Range Flag CB28 116
- Current Velocity (CV) command 66
- CV (Current Velocity) command 66
- CVI (Current Frequency) command 67
 - CVI Frequency Range Flag CB28 116
- D**
- D (Delay) command 46, 68
- Deceleration 49–50
- Default
 - Default Direction Flag CB10 111
 - System Default (SD) command 101
- Delay (D) command 46, 68
- Delete EEPROM command ## 49
- Digital Inputs 15–17, 161
 - Disable Input Averaging Flag CB19 113
 - Encoder inputs
 - see PIF command 88
 - High-speed Start Trigger at IN1, Flag CB20 114
 - High-speed Start Trigger at IN2, Flag CB21 114
 - Read Status of Inputs (IN) command 75
- Digital Outputs 18
 - Enable Running Output (08) Flag CB17 113
 - Error at User Output Flag CB6 110
 - Read/Set Status of Outputs (OUT) command 86–87
- Dimensions 136–137
- DIR Outputs 24–25
- Direction
 - Default Direction Flag CB10 111
- Direction Level Flag CB1 110
- Disable Error E43-E47 Flag CB11 111
- Disable Input Averaging Flag CB19 113
- Diverse Flag CB22 114
- Division operator, /, 41
- Driver
 - Outputs 24
 - Yaskawa Servo Drivers 140–143
- E**
- E (Global Execute) command 68
- EEPROM, Delete EEPROM (##) command 49
- Electronic Gearing Flag CB23 115
- ELSE 42, 68
- Enable Running Output (08) Flag CB17 113
- Encoder inputs
 - see PIF command 88
- Encoder Pulses (PR) command 89
- End-of-travel Limit Inputs 16, 161
 - Disable Input Averaging Flag CB19 113
 - End of travel Flag CB4 110
 - Negative Limit Switch (NLS) command 83
 - Positive Limit Switch (PLS) command 88
 - Trigger Level Flag for NL Input CB25 115
 - Trigger Level Flag for PL Input CB26 115
- Equal to (value assignment) operator, =, 41
- ERR (Error Bit) command 69
- Error
 - Output Error Flag CB7 111
- Error Messages 43, 124–127
- Errors
 - Common Errors 139
 - Communication errors, see ES0 command 69–71
 - Critical Errors 69, 128
 - Disable Error E43-E47 Flag CB11 111
 - Diverse Flag CB22 114
 - Error at User Output Flag CB6 110
 - Error Bit (ERR) command 69
 - Error Messages 124–127
 - Error Status Text (EST) command 72
 - Error Status Text (ESTG) command 72
 - General Error Flag CB8 111
 - Monitoring inputs and errors 46
 - Read-out of Error Status (ES) command 69–71
 - Status and Error Indication 138
 - Testing error routines using the ERR command 69
- ES (Read-out of Error Status) command 69–71

Index

- EST (Error Status Text) command 72
- ESTG (Error Status Text) command 72
 - Diverse Flag CB22 114
- Exclamation mark
 - See Indexer type command 48, 51
- Execute. Global Execute command E 68
- Execution time of commands 159
- EXIT (Exit Programming Mode) command 72
- Exit Programming Mode (EXIT) command 72
- External modules 53, 59, 75–76, 89–90, 159
- F**
- Factory default settings (## command) 49
- Factory Default Set-up, SD command 101
- Firmware Version (VE) command 107
- Flags
 - Control Flags 110–122
- Front Panel, Overview 8
- G**
- Galvanic isolation 15–18, 26–27
- Gearing Flag CB23 115
- General Error Flag CB8 111
- Global Execute (E) command 68
- GND Output 24–25
- GO (Start Program Execution) command 73
- Grammatical errors 43
- Greater than operator, >, 41
- Greater than or equal to operator, >=, 41
- Ground 15–17, 161
- H**
- H (Halt) command 73
- Halt (H) command 73
- Hardware 13–18, 22, 24, 26–30
- High-speed Start Trigger at IN1, Flag CB20 114
- High-speed Start Trigger at IN2, Flag CB21 114
- Home Input 17, 161
 - Disable Input Averaging Flag CB19 113
- I**
- I (Read Flag from External Module) command 76
- IBM AT/IBM-XT/PS2 30
- IF statement 42, 74, 159
- IN (Read Status of Inputs) command 75
- Indexer type command ! 48, 51
- INPUT (Read Data from External Module) command 75
- Inputs
 - Analogue Inputs 26, 52, 151–152, 161
 - Disable Input Averaging Flag CB19 113
 - Encoder inputs
 - see PIF command 88
 - End-of-travel Limit Inputs 16, 161
 - High-speed Start Trigger at IN1, Flag CB20 114
 - High-speed Start Trigger at IN2, Flag CB21 114
 - Home Input 17, 161
 - Monitoring inputs and errors 46
 - Read Status of User Inputs (IN) command 75
 - User Inputs 15–17, 161
- INT (Interrupt) command 77
 - Trigger Level Flag for NL Input CB25 115
 - Trigger Level Flag for PL Input CB26 115
- Interface
 - see RS232/RS485 Interface
- Interrupt
 - Return from Interrupt (RETI) command 93
- Interrupt (INT) command 77
 - Trigger Level Flag for NL Input CB25 115
 - Trigger Level Flag for PL Input CB26 115
- Interrupt Controlled Start (NSTART) command 84–85
 - NSTART Trigger Level Flag CB18 113
- Interrupt Controlled Stop (NSTOP) command 85–86
 - Trigger Level Flag for NL Input CB25 115
 - Trigger Level Flag for PL Input CB26 115
- Interrupt status 117
- INV 42
- J**
- J (Jump) command 44, 78, 159
- JS (Jump Subroutine) command 44, 78, 159
- Jump command 44, 78, 159

Index

Jump Subroutine command 44, 78, 159

JVL Step Motor Driver 144

L

Labels

See the J and JS commands and MotoWare

LEDs

Status and Error Indication 138

Less than operator, <, 41

Less than or equal to operator, <=, 41

Limit Inputs 16, 161

LINE (Verify Line Number) command 79

Line Number

Verify Line Number (LINE) command 79

Logic Operations 42

Logical Operators 41

M

Macro Functions (MAKRO) command 79

MAKRO (Macro Functions) command 79

Maximum Velocity (VM) command 107, 159

MCHS (Memory Checksum) command 80

MEM (Show Used Memory) command 80

Memory Checksum (MCHS) command 80

Memory, Show Used Memory (MEM) command 80

Ministeps 89

Mode 47

Motor Connection 20–21, 164–165

Motor Current at Standby (CS) commands 61

Motor Current when Running (CT) command 61–62

Motor in Position (COIN) Flag CB16 113

Motor Phases 20

MotoWare 37–39

MR1 (Recall Program) command 81

MR2 (Recall Registers) command 81

MS1 (Save Program) command 81

MS2 (Save User Registers) command 83

Multiplication operator, *, 41

Multi-tasking 149–150

N

Negative Limit Switch (NLS) command 83

Trigger Level Flag for NL Input CB25 115

NLS (Negative Limit Switch) command 83
Trigger Level Flag for NL Input CB25 115

Noise 19

Noise emission 19

Not equal to operator, <>, 41

NPN output 15–17

NSTART (Interrupt Controlled Start) command 84–85

NSTART Trigger Level Flag CB18 113

NSTOP (Interrupt Controlled Stop) command 85–86

Trigger Level Flag for NL Input CB25 115

Trigger Level Flag for PL Input CB26 115

O

Operators

Arithmetic Operators 40

Logical Operators 41

Precedence 41

Optical isolation 15–18, 26–27

OR operator 41

ORL 42

OUT (Read/Set Status of Outputs) command 86–87, 159

Outputs

Analogue Output 27, 54, 152, 161

CLK 24–25

COIN 24–25

COIN Motor in Position Flag CB16 113

DIR 24–25

Driver Outputs 24

Enable Running Output (08) Flag CB17 113

Error at User Output Flag CB6 110

GND 24–25

Output Error Flag CB7 111

PS+ 24–25

Read/Set Status of User Outputs (OUT) command 86–87

SALA 24–25

SALA Signal Flag CB15 113

SON 24–25

User Outputs 18

Overload

Voltage 22–23

Index

Overview of Commands 129–131

P

P- terminal 22–23, 161

P+ terminal 22–23, 161

Parallel connection of motor phases 20–21

Pausing program execution

See the Delay (D) command

PC 28, 30–31, 35, 81, 89–90, 99, 118–119, 160

Connection of Indexer to a PC 30

Phases 20

PIF (Pulse Input Format) command 88

PLC 7, 75–76, 111, 119, 160–161

PLS (Positive Limit Switch) command 88

Trigger Level Flag for PL Input CB26
115

PNP output 15–17

Pointer functions 45

Position Reached Flag CB24 115

Positive Limit Switch (PLS) command 88

Trigger Level Flag for PL Input CB26
115

Power Supply 22

Capacitor 22–23

Supply Voltage (VOL) command 108

PR (Encoder Pulses) command 89

Prefix 47

PRINT (Print to External Module) command 89–90, 159

Print to External Module (PRINT) command 89–90, 159

Programming & Programs 36–46

Command Description 28–29, 47–109
Alphabetical Overview of
Commands 129–131

Control Flags 110–122

Error Messages 124–127

Execution time of commands 159

PROGRAM (Set Indexer to Programming Mode) command 90

Program Examples 149–158

Multi-tasking 149–150

Test-program, quick start 10, 153

Use of analogue input 151–152

Use of analogue output 152

Program execution, GO command 73

Program Exit (PX) command 72

Program flow 42–44, 68, 74, 78, 92–93

Programming the Indexer using Motoware 37–39

Recall Program (MR1) command 81

Save Program (MS1) command 81

Set Indexer to Programming Mode (PROGRAM) command 90

PS+ Output 24–25

Pull-up resistor 15–17

Pulse Input Format (PIF) command 88

PX (Program EXIT) command 72

Q

Question mark

See Show set-up command 48

Quick Start, Test program 10, 153

R

R (User Register) command 91

R0-R220

See User Registers

Range 47

RB (User Register) command 92

Read data from external module (INPUT) command 75

Read Flag from external module (I) command 76

Read Status of Inputs (IN) command 75

Read/Set Status of Outputs (OUT) command 86–87, 159

Recall Program (MR1) command 81

Registers 36

Recall Registers (MR2) command 81

Save User Registers (MS2) command 83

Special User Registers (RX) command 96–100, 159

User Register (R) command 91

User Register (RB) command 92

User Register (RI) command 93, 159

Registers, SMI3x 145

Relative Offset Positioning (SR2) command 104–105

Relative Positioning (SR) command 104, 159

Report Motor Status (RS) command 94

Report Motor/Program Status in Text (RST) command 95

Reset factory default settings, (## command) 49

Index

- Reset Indexer (RESET) command 92
- Resonances 89
- RET
 - See Subroutines 44, 92
- RETI (Return from Interrupt) command 93
- Return from Interrupt (RETI) command 93
- RI (User Register) command 93, 159
- RS (Report Motor Status) command 94
 - Motor status (RS) when limit is activated flag, CB38 121
- RS232/RS485 Interface 28–30
 - Address 28
 - Checksum 28–29, 56, 80
 - Command 28
 - Command Syntax 28
 - Communication (Baud) Rate 28, 56
 - Communication Protocol 28
 - Connection 28
 - Errors, see ES0 command 69–71
 - RS232 Activity Flag CB29 116
 - RS232 Communication Flag CB9 111
 - Synchronisation 29
 - Use of commands 34
- RST (Report Motor/Program Status in Text) command 95
- Runtime errors 43
- RX (Special User Registers) command 96–100, 159
- S**
- SALA (servo alarm) 145
- SALA Output 24–25
 - Signal Flag 113
- Save Program (MS1) command 81
- Save User Registers (MS2) command 83
- Screened cable 19, 140–142, 144–145, 148, 161–162, 164–165
- SD (System Default) command 101
- Seek Zero Point (SZ) command 106
 - Zero Search Flag CB27 116
- Semi-colon (Command Delimiter) 29, 34
- Serial connection of motor phases 20–21
- Serial Number (SN) command 102
- Servo Alarm Signal (SALA) Flag CB15 113
- Servo On (SON) command 102, 159
- Set New Absolute Position (SP) command 103
- Set New Global Absolute Position (SPT) command 103
- Set/Read Interrupt status 117
- Set/Read Status of Outputs (OUT) command 86–87, 159
- Set-up (Show set-up) command ? 48
- SH (Smooth Halt of Motor) command 102, 159
- Show set-up command ? 48
- Show user memory (MEM) command 80
- Smooth Halt of Motor (SH) command 102, 159
- SN (Serial Number) command 102
- Software 33
- SON Output 24–25
 - Servo On (SON) command 102, 159
- SP (Set New Absolute Position) command 103
- Special User Registers (RX) command 96–100, 159
- Specifications 134–135
- SPT (Set New Global Absolute Position) command 103
- SR (Relative Positioning) command 104, 159
- SR2 (Relative Offset Positioning) command 104–105
- Standby Mode 35
 - Motor Current at Standby (CS) command 61
- Start program execution, GO command 73
- Start Rate (VS) command 108, 159
- Status and Error Indication 138
 - Motor Status (RS) when limit is activated flag, CB38 121
- Status Flags 110–111, 113, 115–116
- STOP (Stop Motor) command 106
- Stop Motor (STOP) command 106
- Subroutines 44, 92
- Subtraction operator, -, 41
- Supply Voltage (VOL) command 108
- Synchronisation 29
- Syntactic errors 43
- Syntax
 - Command Syntax 28
- System Default (SD) command 101
- SZ (Seek Zero Point) command 106

Index

Zero Search Flag CB27 116

T

Technical Data 134–135

Temperature (TP) command 107

Test program, Quick Start 10, 153

Testing error routines using the ERR command 69

Timing, Command Timing 159

Torque 21

TP (Temperature) command 107

Trigger Level Flag for INT1 CB12 112

Trigger Level Flag for INT2 CB13 112

Trigger Level Flag for INT3 CB14 113

Trigger Level Flag for NL Input CB25 115

Trigger Level Flag for PL Input CB26 115

Turntable Mode CB30 116

U

Unipolar Motors 20

User Inputs 15–17, 161

 Disable Input Averaging Flag CB19 113

 Encoder inputs

 See PIF command 88

 High-speed Start Trigger at IN1, Flag CB20 114

 High-speed Start Trigger at IN2, Flag CB21 114

 Read Status of User Inputs (IN) command 75

User Outputs 18

 Enable Running Output (08) Flag CB17 113

 Error at User Output Flag CB6 110

 Read/Set Status of Outputs (OUT) command 86–87

 User Output Error Mode Flag, CB33 118

User Registers 36

 Recall Registers (MR2) command 36, 81

 Save User Registers (MS2) command 36, 83

 Special User Registers (RX) command 96–100, 159

 User Register (R) command 91

 User Register (RB) command 92

 User Register (RI) command 93, 159

V

Value assignment (equal to) operator, =, 41

VE (Firmware Version) command 107

VM (Maximum Velocity) command 107, 159

VOL (Supply Voltage) command 108

Voltage Overload 22–23, 26

VS (Start Rate) command 108, 159

W

Wait for Condition (WAIT) command 108, 159

Y

Yaskawa Servo Drivers 140–143

Z

Zero Search Flag CB27 116