SMC23A / SMC23B
SMC24A / SMC24B
SMC25A / SMC25B
SMC26A / SMC26B

Step Motor Controller
User's Manual

JVL Industri Elektronik - Stepmotor Controller SMC25A
JVL Industri Elektronik
Power
Overload
RS232C
SMC26B Step Motor Controller

JVL Industri Elektronik A/S - September 1995

Revised 14.5.96

CONTENTS

1.1 Introduction

Stepper Motor Controllers Types SMC23-26 provide a range of easy-to-use, cost-effective controllers for stepper motors. They combine an advanced motion-control indexer and motor drive in a single unit.

The Controllers can be used as stand-alone units or connected to a terminal or personal computer (PC) via the RS232C/V24 interface. They are equipped with inputs and outputs which provide the user with a high degree of flexibility for tailoring configuration to the specific application. The Controllers are ideal for controlling small milling machines and drills, handling- and feeder units, etc., where quick and precise motion control is required without the use of components which are large or costly.

The SMC23-26 Controller series provides the following features:

	SMC23	SMC24	SMC25	SMC26
230V Power Supply		●		●
15-45V Power Supply	●		●	
Storage Registers	●	●		
External User Supply		●		●
Module Interface	●	●		
Extended Memory	●	●		

All Controller Types are available in two versions with 3A and 6A motor drives respectively. All Controllers are equipped with 3 digital User Inputs and 3 digital User Outputs for general use. 6 analogue inputs can be used for example when a pressure transducer is used to transmit measurement or control values to the Controller.

All inputs and outputs for fully overload protected. The motor driver is further equipped with short-circuit protection which disconnects current to the motor in the event of a short-circuit.

Types SMC24 and SMC26 are equipped with an integral power supply which enables direct operation from a mains supply.

Types SMC23 and SMC25 cannot be powered directly from a mains supply, but are primarily intended for use in larger systems where a central power supply is used for powering 2 or more controllers.

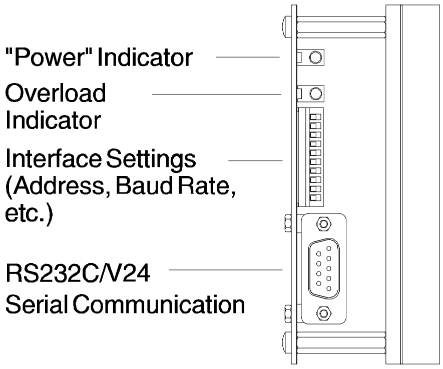
Controller Types SMC23 and SMC24 are advanced models which include an extended set of program commands and User Registers for storing parameters and intermediate results. In addition, these models include a Module Interface which enables connection of extension modules such as a keyboard/display module and additional user inputs and outputs. The Module Interface enables connection of up to 31 external modules including other controllers.

The SMC Series provides the following basic features:

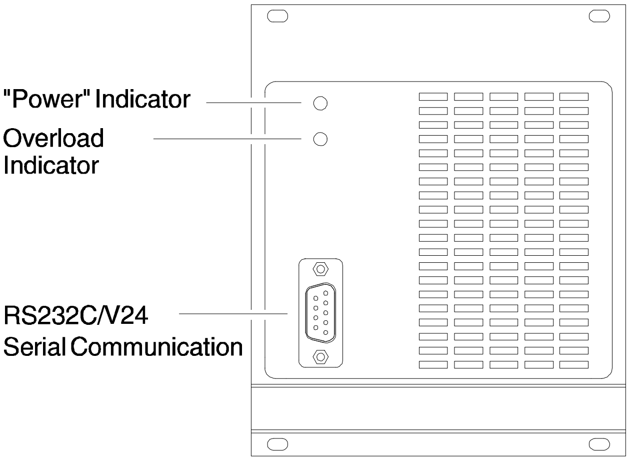
- 15-45V DC supply (Types SMC23/SMC25)
  115/230V AC supply (Types SMC24/SMC26).
- RS232C/V24 communication.
- Simple programming.
- Max. stepping frequency 15kHz.
- Connection of up to 7 controllers on the same RS232 interface bus.
- Baud Rates: 110 - 9600.
- Small physical dimensions:
  SMC23/25: bxhxd: 46.5 x 100 x 160 mm
  SMC24/26: bxhxd: 106.5 x 111 x 171 mm
- Thermal protection.
- 3 User Inputs.
- 6 Analogue Inputs.
- 1 Stop Input.
- 3 User Outputs (each 500mA).
- Connection via DIN41612 socket or connector board CON10/CON10P.
- Mounting in 19" rack or flush mounting.

1.2 Controller Connections

Front Panel SMC23 and SMC25 :


"Power" Indicator
Overload Indicator
Interface Settings (Address, Baud Rate, etc.)
RS232C/V24 Serial Communication

Front Panel SMC24 and SMC26 :



Interface :

The RS 232C Interface enables the Controller to be connected to a computer or terminal. Up to 7 Controllers can be connected on the same interface bus.

Motor Output :

Enables connection of a 2-phase or 4-phase stepper motor. The output is short-circuit protected. The motor can be controlled with a speed of 15000 Full/Half-steps per second.

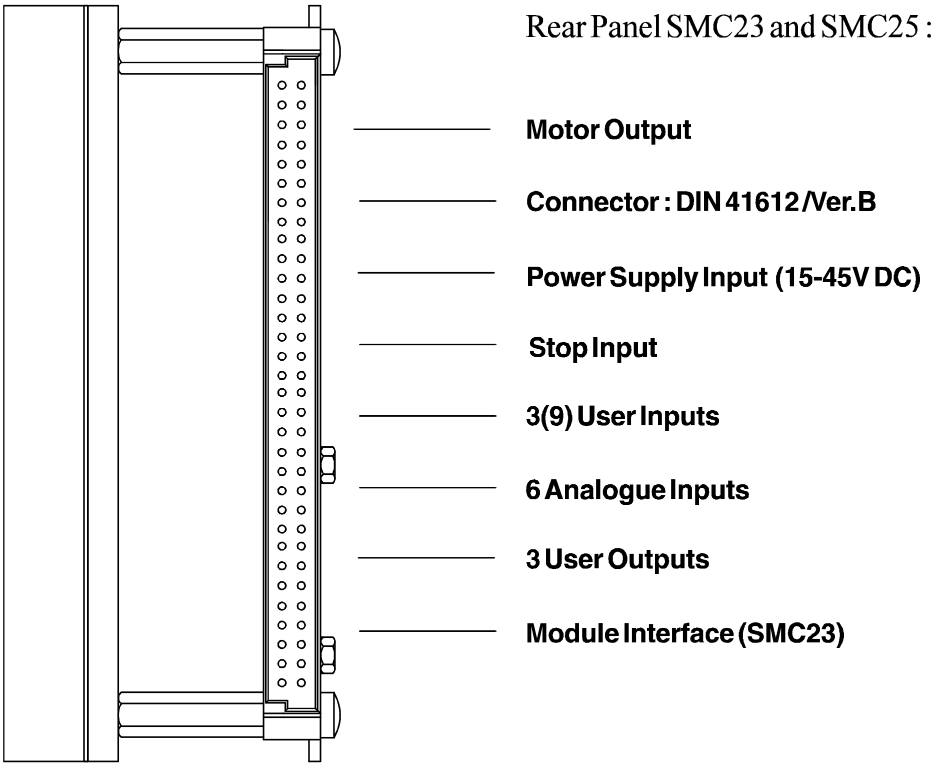
Power Supply :

The Controllers are operated from a single supply voltage: from 15 to 45 V DC for Types SMC23 and SMC25, and 115/230V AC for Types SMC24 and SMC26.

User Inputs :

The Controllers are equipped with 4 noise-suppressed inputs, one of which is reserved for the Stop function. The remaining 3 User Inputs can be used, for example, for connecting inductive sensors or for synchronization with other controllers. The inputs are equipped with a Schmidt-trigger function and operate in the range 5-30 V. The User Inputs are all optically isolated from other Controller circuitry.

1.2 Controller Connections



User Outputs :

The Controllers are equipped with 3 User Outputs which for example can be used to drive small DC motors, or to synchronise the unit with other controllers.

Each output can supply up to 500mA and operates in the range 5-30 V.

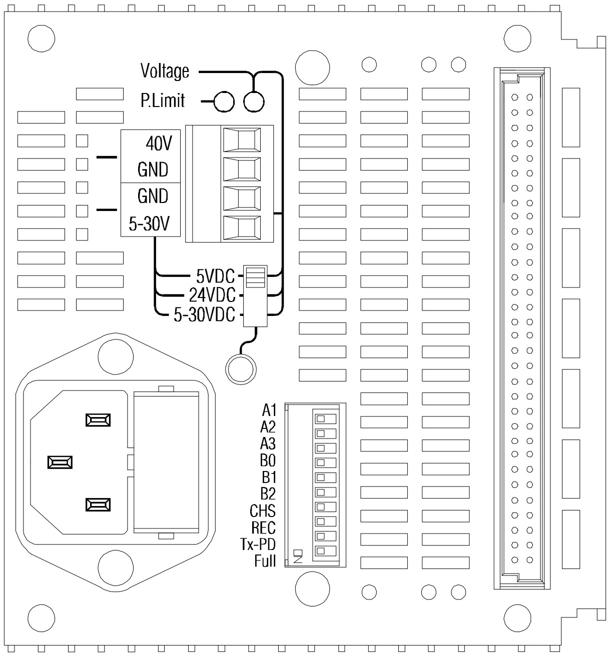
In addition, the User Outputs are short-circuit protected and optically isolated from other Controller circuitry.

Analogue Inputs :

The voltages at the Controller's 6 Analogue Inputs can be read using a set of program commands, thus enabling control of a motor's maximum speed, absolute or relative distance, etc., by the application of a voltage to one of the 6 Analogue Inputs.

The inputs accept voltages in the range 0 - 5.10V, and are protected against short-duration overloads up to 45V.

1.2 Controller Connections



Module Interface (Types SMC23/24 only)

The Module Interface fitted to Controllers Type SMC23 and SMC24 consists of 2 optically isolated terminals.

These are used for connection to all external modules such as keyboard/display modules, input/output modules, etc.

CW/CCW Limit Inputs (Types SMC23/24 only)

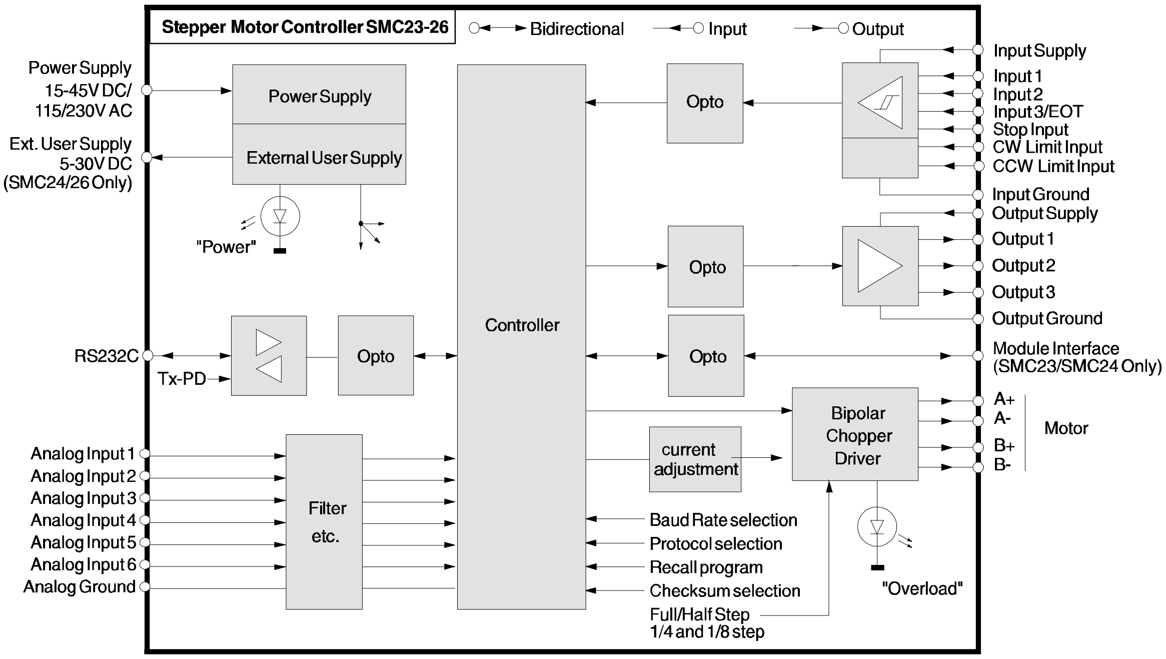
The CW (clockwise) and CCW (counter-clockwise) Limit Inputs are used in applications where it is crucial that the motor does not advance beyond some predefined mechanical limits. On activation of a Limit Input, the motor is immediately halted. Together with the User Inputs, the Limit Inputs are optically isolated from other circuitry in the Controllers.

User Power Supply (Types SMC24/26 only)

To enable power to be supplied to external sensors, etc., Types SMC24 and SMC26 are equipped with a User Power Supply on the rear panel. This supply can be adjusted to one of 3 settings to give a voltage of either: 5V DC; 24V DC; or continuously adjustable from 5 to 30V DC.

The User Power Supply output supplies a current of 0.5A regardless of voltage level.

2.0 Block Diagram of the Controllers



2.1 Power Supply (Types SMC23 and SMC25).

To ensure powering of the Controllers is as simple as possible, the Controllers are supplied from a single 15 to 45V DC supply. The Controllers' internal circuitry ensures the correct supply for the interface, control circuitry, etc.

In the event of incorrectly connected polarity or overload of the power supply voltage, the Controllers are fuse-protected. If an overload occurs, the power should be disconnected and the fuse replaced. To ensure correct operation of the Controller, it is recommended that a capacitor (min. 5000µF) is connected across the positive and negative terminals of the external supply.

It is also recommended that the cables used to connect the Controller to the external power supply are minimum 0.75mm².

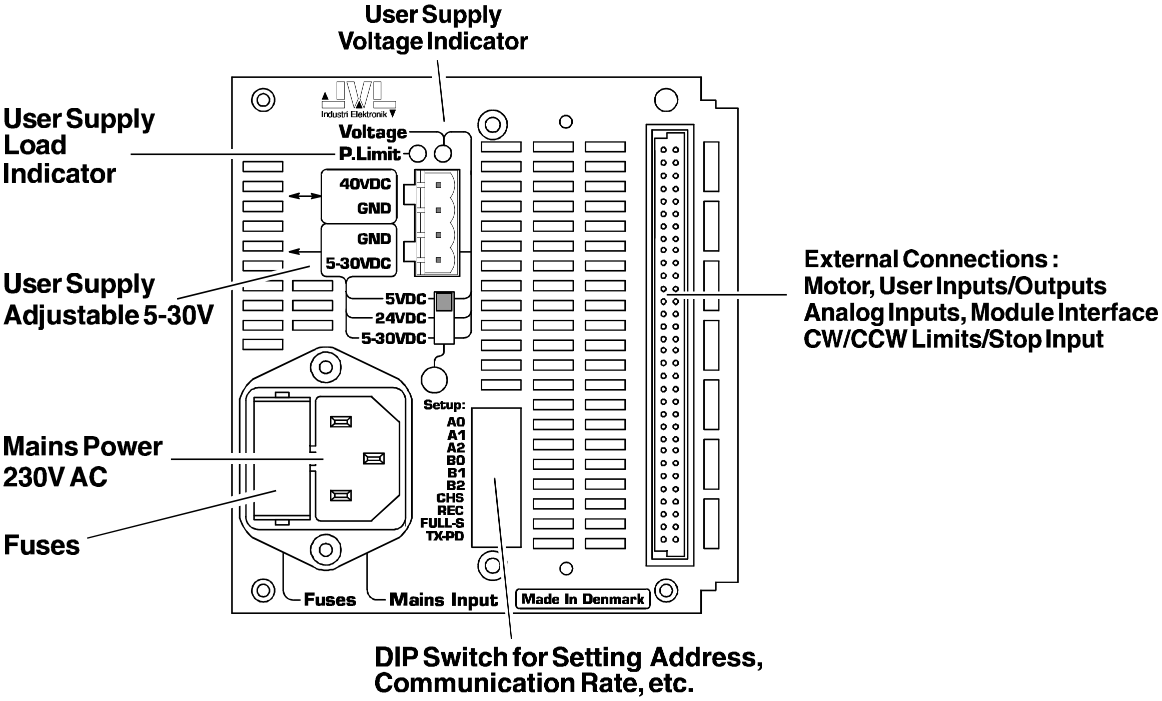
If the voltage used to supply the Controller falls below a level of 10V, the Controller will automatically be reset and any program instructions, etc., will be lost. Provision should therefore be made to ensure that the supply voltage does not fall below a minimum of 15V, even in the case of mains voltage drop.

The program in the permanent memory (E^{2}PROM) will not be lost.

2.1 Power Supply

(Types SMC24 and SMC26 only)

Rear Panel SMC24 and SMC26 :


User Supply Voltage Indicator
User Supply Load Indicator
User Supply Adjustable 5-30V
Mains Power 230V AC
Fuses
External Connections : Motor, User Inputs/Outputs Analog Inputs, Module Interface CW/CCW Limits/Stop Input
DIP Switch for Setting Address, Communication Rate, etc.
Industri Elektronik
Voltage
P.Limit
40VDC
GND
GND
5-30VDC
5VDC
24VDC
5-30VDC
Setup:
A0
A1
A2
B0
B1
B2
CHS
REC
FULL-S
TX-PD
Fuses
Mains Input
Made In Denmark

Controllers Types SMC24 and SMC26 are equipped with an integral supply for powering from an AC mains supply.

This supply also provides a User Supply Output which can be used for powering external equipment. The User Supply can be adjusted to 1 of 3 settings (see above) as follows:

Position 1 provides a fixed supply voltage of 5V DC. Position 2 provides a fixed supply voltage of 24V DC. Position 3 provides a continuously adjustable supply voltage in the range 5 to 30V DC.

Regardless of the voltage setting, the User Supply Output provides a continuous current of 0.5A. If this current is exceeded, the supply will automatically reduce the supply voltage to ensure overload does not occur.

The external supply is thus protected against short-circuiting. Depending on the selected supply voltage, the “Voltage” LED (see drawing) will vary in light intensity.

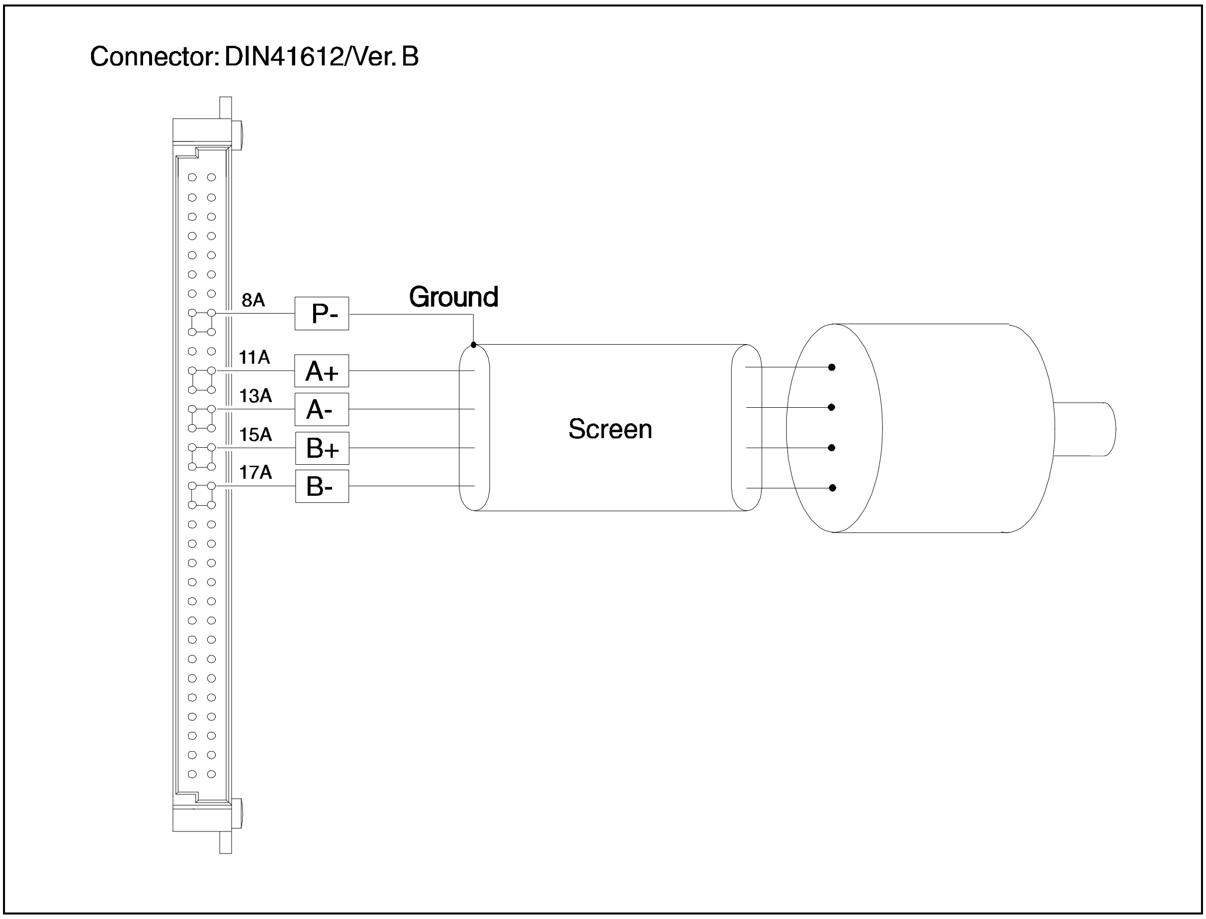
The User Supply Output and the Controller's internal supply (40V DC) are available at terminals on the rear panel of the Controllers.

The Controller's internal supply can thus be used for powering other controller's in the motion control system. The red LED named *P. Limit* (see drawing) will light up if the power consumption from the power supply of the controller exceeds the 120W maximum it can supply.

In the event of voltage overload in the mains supply, the Controller's secondary or primary fuse will be blown. If this occurs, the Controller should be disconnected from the mains supply and the fuse(s) replaced.

Primary and secondary fuses are located in the fuse holder in the mains socket.

2.2 Motor Driver



The Controller is intended for use with 2- or 4-phase stepper motors. Controller Types SMCxxA provide a motor phase current of up to 3A, while Types SMCxxB provide a phase current of up to 6A. The phase current is continuously adjustable. The motor standby current, acceleration/deceleration current and constant speed current can be set individually. The current is controlled via a set of software commands as described in Section 4.4.

The Controller Driver consists of a 2-phase bipolar chopper driver. This type of driver results in optimum utilization of the motor since current is continuously supplied to both phases of the motor.

The chopper-driver regulates the current at a frequency of 22kHz (nominal) thus ensuring that the motor control does not produce audible noise.

The switching time of the Driver is very small (<200nS), which can result in high-frequency noise components in the connection between the driver and the motor.

In some cases, this high-frequency noise can result in unwanted interference of other electronic equipment close to the stepper-motor system. To avoid this problem, screened cable should be used to connect the Controller to the stepper motor, as illustrated above.

2.2 Motor Considerations

It should be noted that the lower the self-inductance of a motor the better, since self-inductance greatly influences the motor driving torque at high speeds. The torque is also affected by the current supplied to the motor.

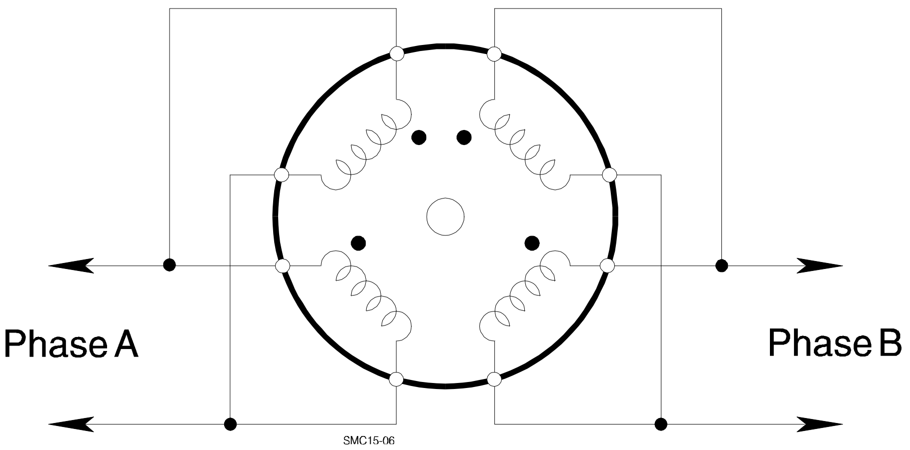
This is also illustrated by the following equation:

$$\text{Phase Current} = \frac{\text{Applied Voltage}}{\text{Phase Inductance * Driving Frequency}}$$

It should be noted that the phase inductance of a motor is dependent on the other phases during operation. Individual motor manufacturer's specifications of phase induction are normally measured statically.
The applied voltage is regulated by the driver so that the phase current is adjusted to the required level. In practice this means that if a motor with a large phase inductance, e.g. 100mH, is used, the driver cannot supply the required phase current at high speeds (high rotational frequencies), since the output voltage is limited.

If a 4-phase motor is used, it should be connected as shown below. The motor phases should be connected in parallel to result in as low a value of self-inductance as possible. The phases can also be connected in serial, but this will limit the top speed of the motor. If the phases are connected in serial, the motor will typically provide greater torque (at low speeds).

Connection of 4-Phase Step Motors



2.2 Motor Driver

Selection of Step Resolution

The driver can be configured to operate with either full-, half-, 1/4 or 1/8 motor steps. It is often an advantage to operate with fractional steps since this increases the resolution per motor rotation. Operating the motor with half-, 1/4 or 1/8 step resolution often eliminates the need for mechanical gearing.

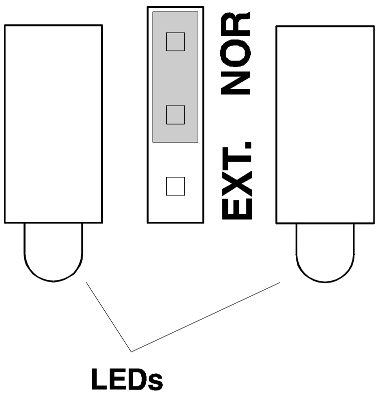
Another advantage is that resonance problems, which are almost unavoidable with full-step operation, can normally be avoided.
A stepper motor always has a resonance frequency which can vary depending on the motor load and results in loss of torque.

The Controller can be set to normal (NOR) or extended (EXT) step resolution. For normal step resolution, the jumper is placed in position “NOR” which makes it possible to switch between Full- and Half- step operation using the dipswitch. If the jumper is placed in position “EXT”, 1/4 or 1/8 step operation can be selected using the dipswitch. The jumper is placed between the “Power” and “Overload” LEDs.
You have free access to this jumper on Controllers SMC23 and SMC25, whereas the front panel has to be removed to gain access to the jumper on Controllers SMC24 and SMC26.

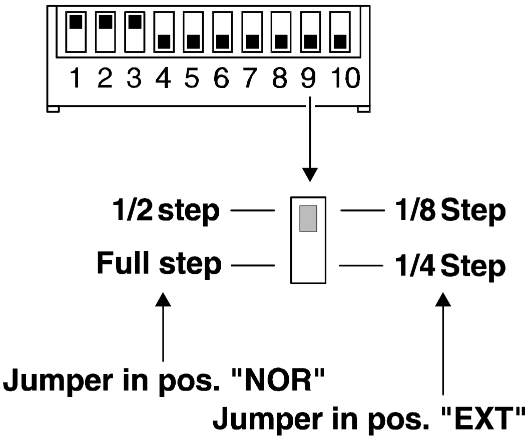
On delivery the step resolution is set to 1/2 step.

The drawing below shows how the step resolution can be set:

Selection of extended step resolution



Selection of step resolution



Overload Protection

The Motor Driver is short-circuit protected. If the motor current exceeds 3.2A (6.4A) for more than 2ms, the voltage is disconnected from the outputs to prevent overload or damage to the motor. An instantaneous short-circuit of any two arbitrary output terminals has no effect and will not damage the Controller, although an Overload indication will occur. To reset the Controller, the power must simply be disconnected for a minimum of 5 seconds, after which normal operation can be resumed.

() Valid for Types SMCxxB

2.3 User Inputs and Outputs

To ensure flexibility and ease of use, the Controller is equipped with 3 digital User Inputs and 3 digital User Outputs which can be used for a variety of purposes. A fourth input can be used as a Stop input.

All inputs are optically isolated.

Equipment connected to the User Inputs and Outputs must be powered from external supplies.

User Outputs.

The Controller's User Outputs can be used for control of secondary functions such as actuators, small motors, etc.

This enables the stepper motor to be synchronized with the surrounding environment.

The 3 User Outputs are controlled via software commands and each provide a current of up to 500mA.

They are protected against inductive-load transients and are short-circuit protected.

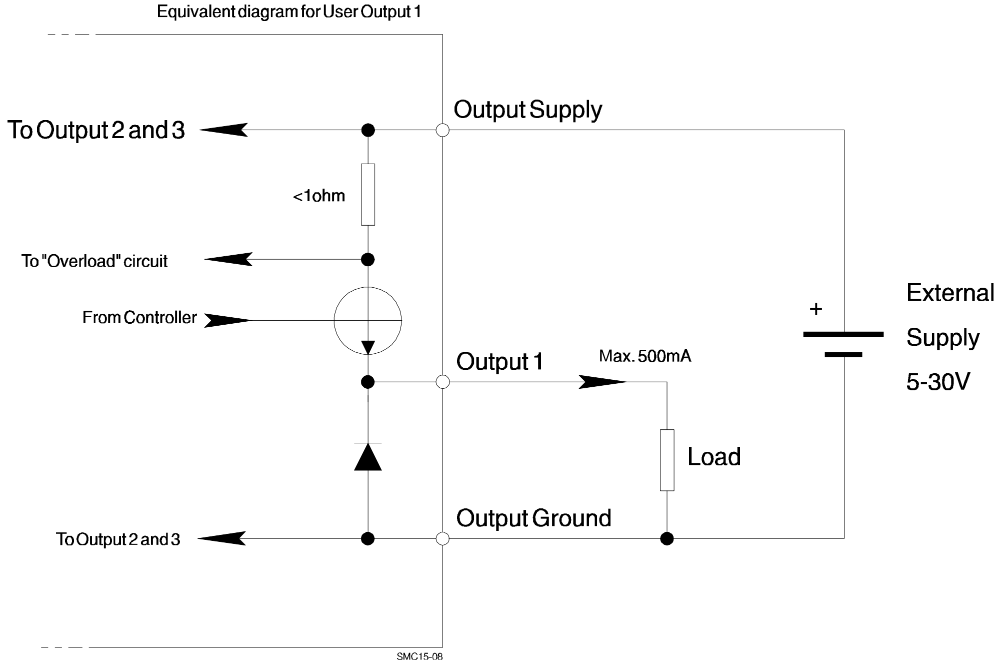
If one of the User Outputs is short-circuited or the output current exceeds 700mA, the "Overload" indicator blinks. In addition the voltage is disconnected from the User Output at which the overload has occurred.

To reset the Controller, the output-supply should be disconnected for a minimum of 5 seconds, after which normal operation can be resumed.

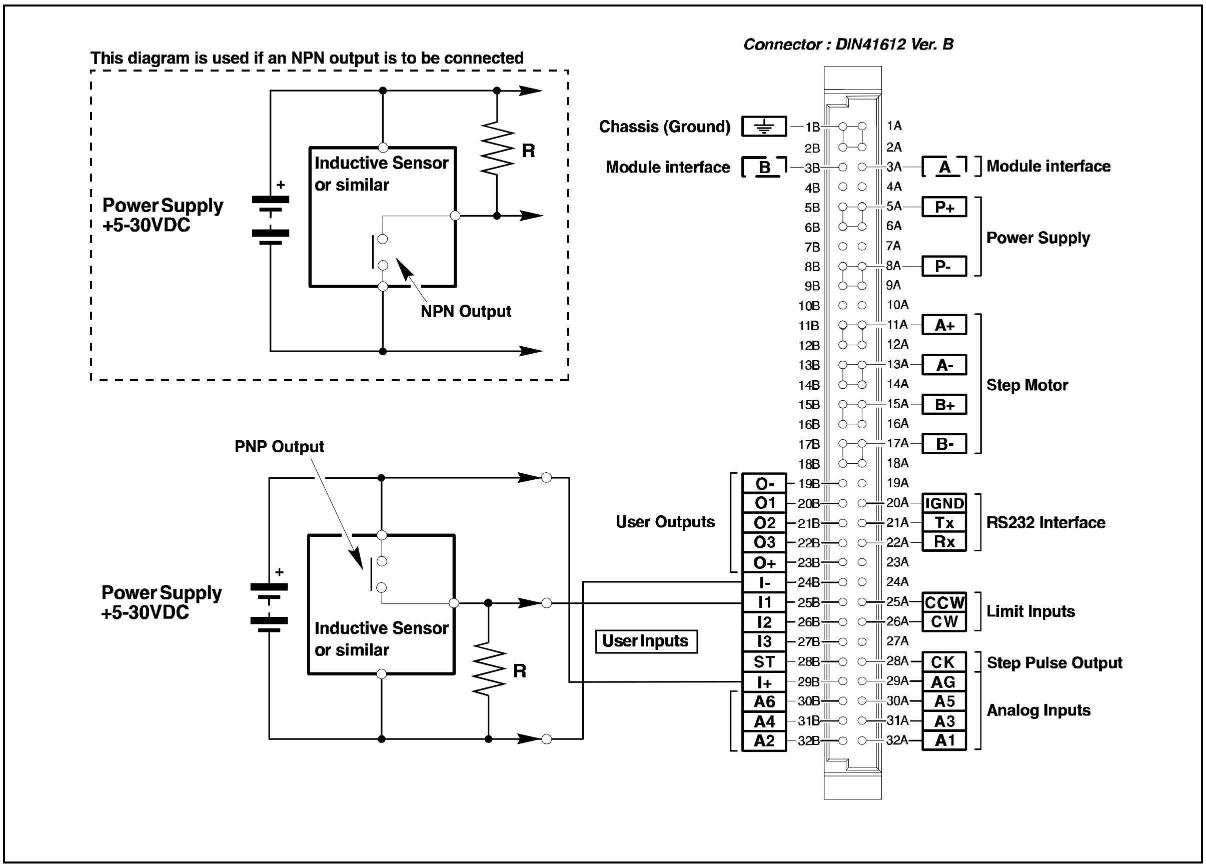
Each of the output terminals is the + switch terminal, i.e. that the load must be connected between the output and ground (see figure below).

To enable compatibility with logic circuitry, a "pull down" resistor should be connected between the output terminal and ground. For TTL logic, a 1kOhm resistor should be used; for CMOS logic a resistor of approximately 10kOhm should be used.

Connection of User Outputs



2.4 User Inputs


This diagram is used if an NPN output is to be connected
Connector : DIN41612 Ver. B
Power Supply +5-30VDC
+
Inductive Sensor or similar
R
NPN Output
PNP Output
Power Supply +5-30VDC
+
Inductive Sensor or similar
R
Chassis (Ground)
Module interface
B
A
Module interface
1B 2B 3B 4B 5B 6B 7B 8B 9B 10B 11B 12B 13B 14B 15B 16B 17B 18B 19B 20B 21B 22B 23B 24B 25B 26B 27B 28B 29B 30B 31B 32B
1A 2A 3A 4A 5A 6A 7A 8A 9A 10A 11A 12A 13A 14A 15A 16A 17A 18A 19A 20A 21A 22A 23A 24A 25A 26A 27A 28A 29A 30A 31A 32A
P+
P-
Power Supply
A+
A-
B+
B-
Step Motor
User Outputs
O-
O1
O2
O3
O+
I-
I1
I2
I3
ST
I+
A6
A4
A2
User Inputs
IGND
Tx
Rx
RS232 Interface
CCW
CW
Limit Inputs
CK
AG
Step Pulse Output
A5
A3
A1
Analog Inputs

Each of the Controller's User Inputs is equipped with a 1st. order low-pass filter with a cut-off frequency of 1kHz. This ensures that electrical noise from start motors, etc., does not influence the input signal.

It should be noted that the state of each of the 3 User Inputs is undefined if no connection is made to the input.

All User Inputs are optically isolated from other Controller circuitry.

Some inductive sensors have an open collector-output. If sensors with an NPN output are used, a resistor must be connected between the Input and the positive supply terminal. If a PNP sensor is used, the resistor must be connected between the Input and Ground. A resistor of 500Ohm to 5kOhm should be used, depending on the supply voltage.

Stop Input

If it is required to stop the motor movement immediately, the Stop Input can be connected to ground. If the ground connection is re-moved, the motor will continue operation and the value of the Position Counter (step counter) will be retained. However, an instantaneous stop of the motor in this way will normally imply that the motor position is undefined since activation of the Stop Input does not take account of the pre-defined acceleration/decele-ration ramp (see Motor Commands, Section 4.4).

2.4 User Inputs

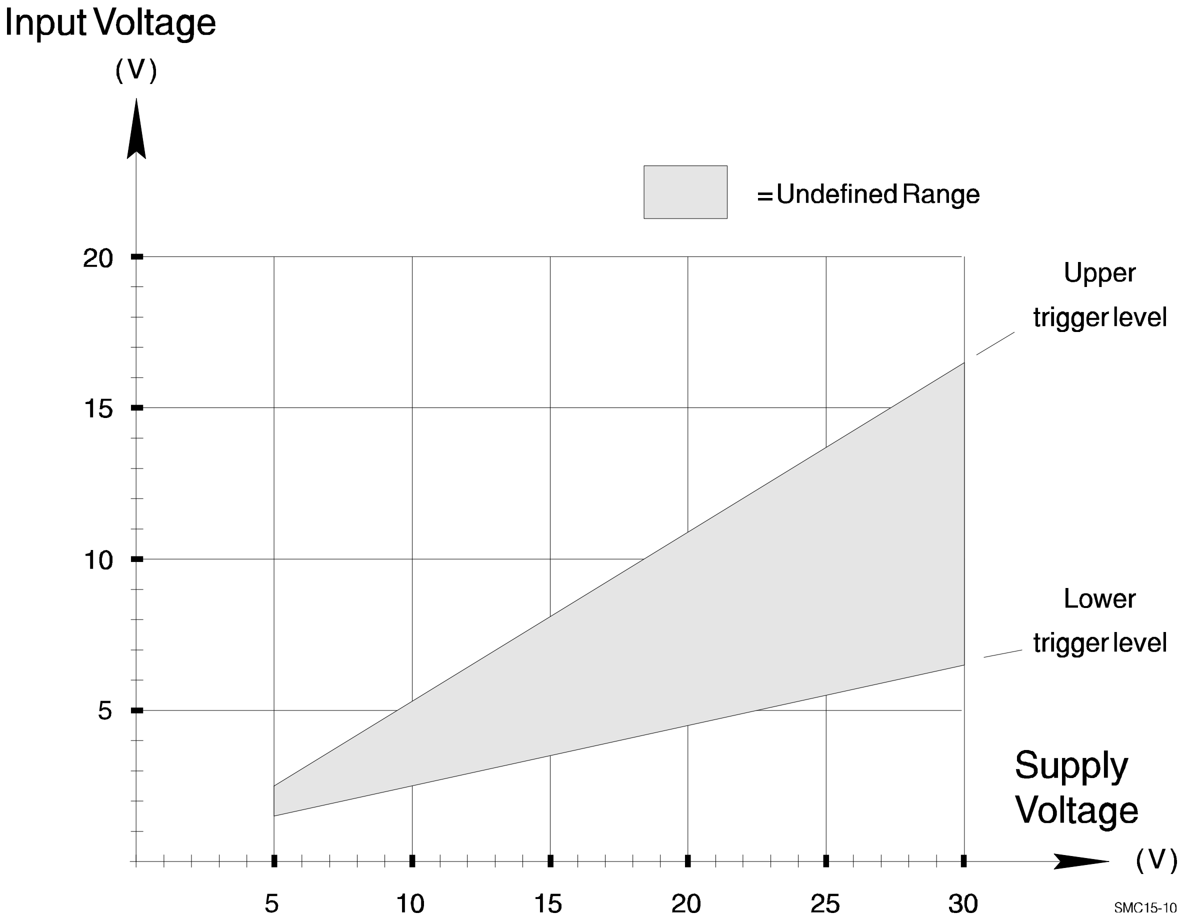
Input Hysteresis.

All User Inputs are noise-protected and are compatible with commonly used logic types: CMOS, TTL, etc.
The hysteresis of the Inputs is dependent on the connected supply voltage, as illustrated in the figure below.

Example:

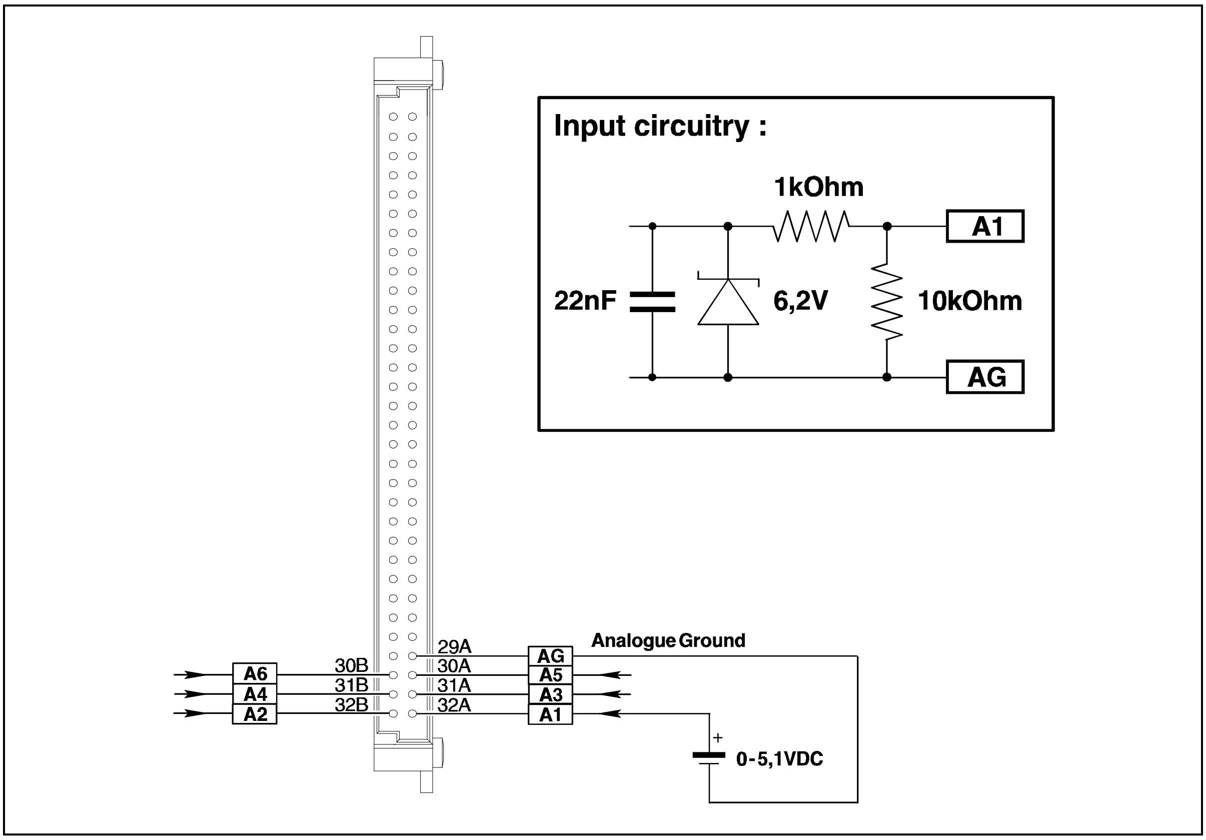
User Input 1 is used, and the supply voltage is 24V.
As can be seen from the figure, the Input is logic "1" if an input voltage greater than 13.3V DC is applied. To set logic "0", the applied input voltage must fall below approximately 5.3V DC.

The trigger tolerances on the input voltage is ±10%.



See also Section 4.5 (User Interface) for further details of User Inputs and Outputs.

2.5 Analogue Inputs



The Controller is equipped with 6 Analogue Inputs which can be scanned and read using software commands as described in Chapter 4.

The Analogue Inputs can be used for example to control the speed of the stepper motor using an analogue input voltage.

The inputs are protected against short-duration overloads up to 45V.

Each time the controller measures the signal at an Analogue Input, a total of 16 samples are made. These are then averaged to minimise the possibility that an instantaneous noise impulse, for example from the motor driver, influences the measurement.

The Analogue Inputs can also be used as conventional User Inputs (digital inputs), although without input hysteresis or optical isolation.

No special requirements are necessary to use the Analogue Inputs in this way. At any time, an Analogue Input can be used as either a true analogue input or as a standard User Input (digital input).

For further details, see Chapter 4 - commands ±*A, DA, G±A, JCA, NA, r, s, t, U, VA, W.*

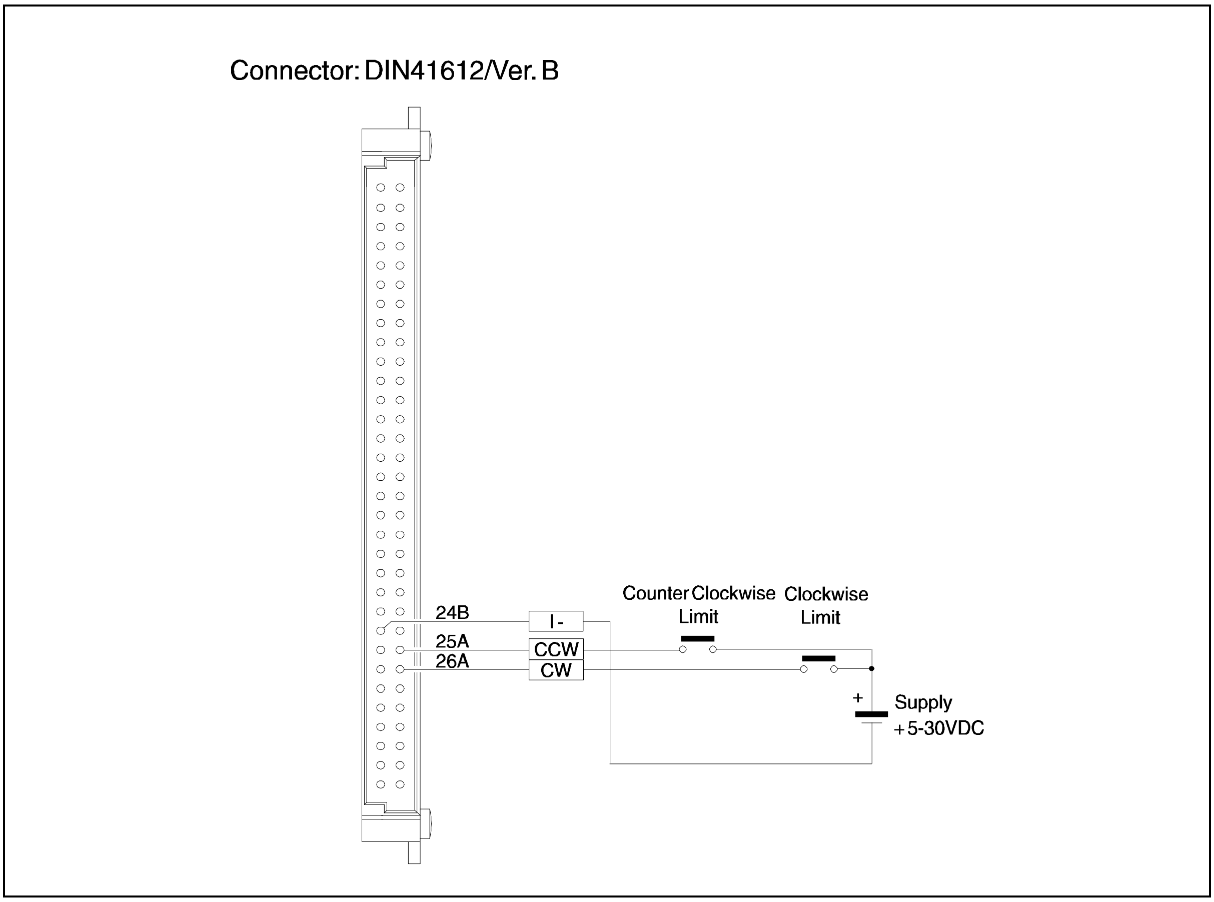
The Analogue Inputs accept voltages in the range 0V to 5.10V. The Controller uses an 8-bit A/D converter which results in a resolution of 256 steps. E-ach step therefore corresponds to 20.0mV at the Input.

To avoid incorrect measurements, the Analogue Ground AGND (see Section 2.7) must be used with the 6 Analogue Inputs.

See also the Electrical Specifications (Section 5.1) for further information.

Each of the Analogue Inputs is equipped with a 1st. order low-pass filter which suppresses frequencies above 10kHz.

2.6 CW/CCW Limit Inputs



In stepper motor systems it is often necessary to set up certain mechanical limits which the motor must not exceed under any circumstances.

To enable these Limits to be set up, the Controller is equipped with 2 inputs: *CW* (Clockwise Limit) and *CCW* (Counter-clockwise Limit).

One of these 2 inputs, depending on the actual direction of rotation of the motor, will stop the motor when the input is activated.

CCW Limit.

If the motor is rotating counter-clockwise and the CCW Limit Input is activated (logic "1"), the motor will be stopped. The CW Input has no effect during counter-clockwise rotation.

CW Limit.

If the motor is rotating clockwise and the CW Limit Input is activated (logic "1"), the motor will be stopped. The CCW Limit Input has no effect during clockwise motor rotation.

Note that activation of either of the CW or CCW Limit Inputs will result in instantaneous stop of the motor, regardless of any pre-set deceleration ramp.

Inactivation of the motor

A special feature of the Controllers enables the driver current to the motor to be completely disconnected so that the motor is free to rotate without any mechanical resistance.

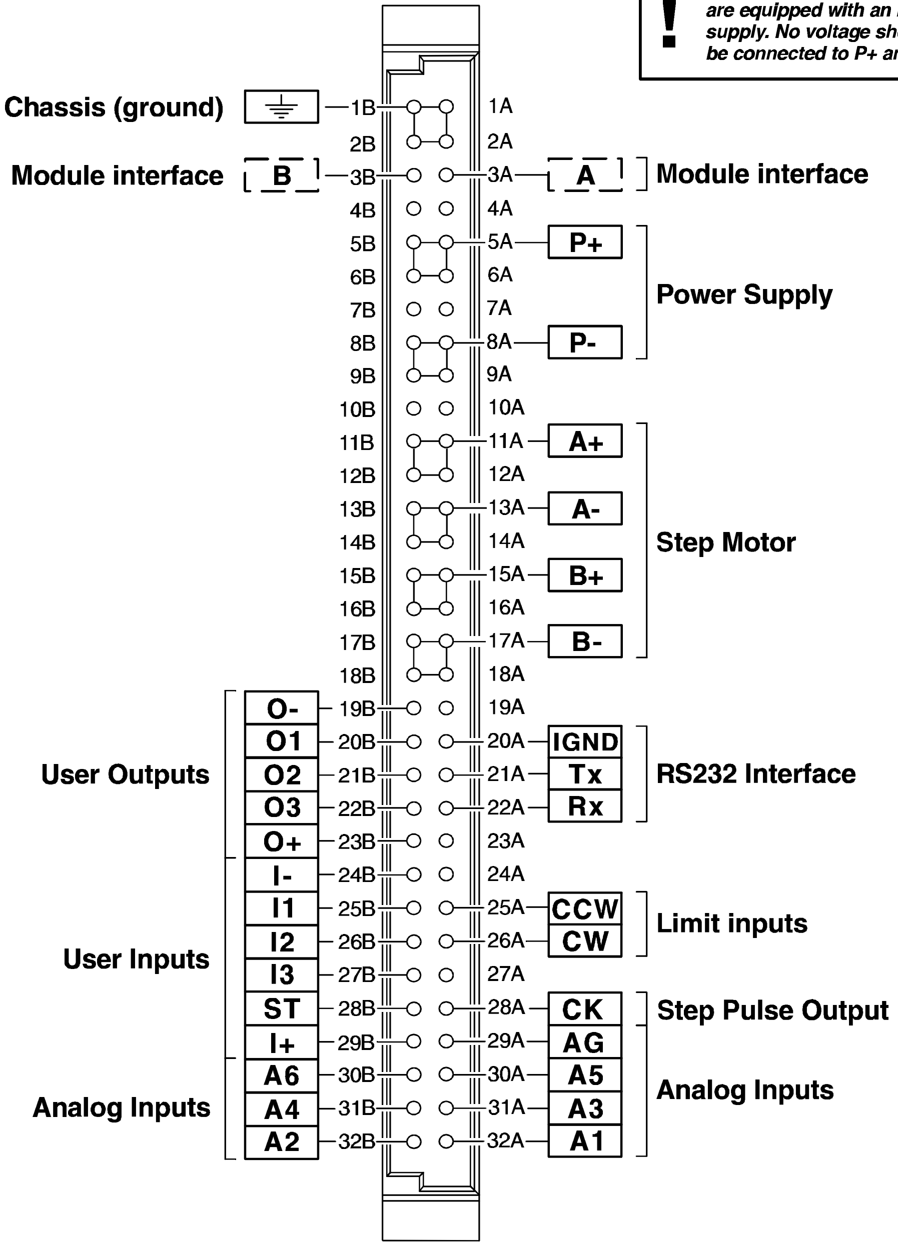
This is done by activating both end-of-travel inputs, CW and CCW, simultaneously.

2.7 Connections

Connections for SMC23, SMC24, SMC25 and SMC26

Connector : DIN41612 Ver. B

! ***Note that Types SMC24 and SMC26 are equipped with an internal mains supply. No voltage should therefore be connected to P+ and P-.***

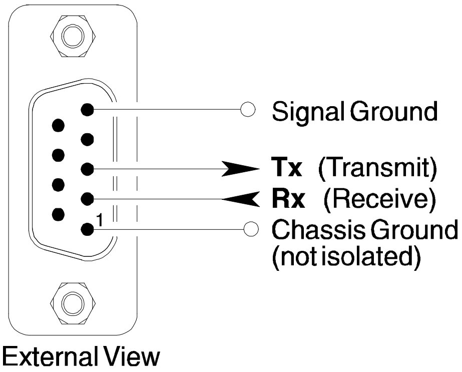


⌐ ¬ = Terminals used only on Controller Types SMC23 and SMC24

3.1 Interface Connections

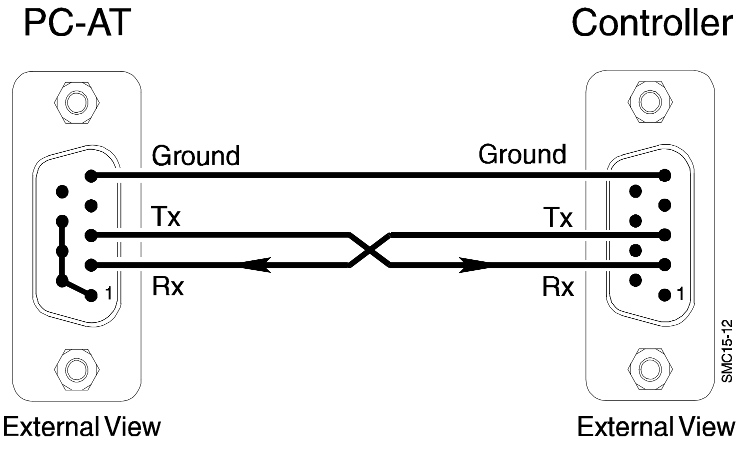
The Controller Interface uses the widespread RS232C standard, which provides a great degree of flexibility since all Personal Computers and standard terminals have provision for using this communication standard.
For operation of the Controller via an RS232C interface, the 3 standard RS232 connections Rx, Tx and Ground are used.
In general, interface cables should not exceed 10 metres in length but if a longer cable is required, the interface checksum facility should be used to ensure data integrity. See Section 3.5.

Controller Interface:



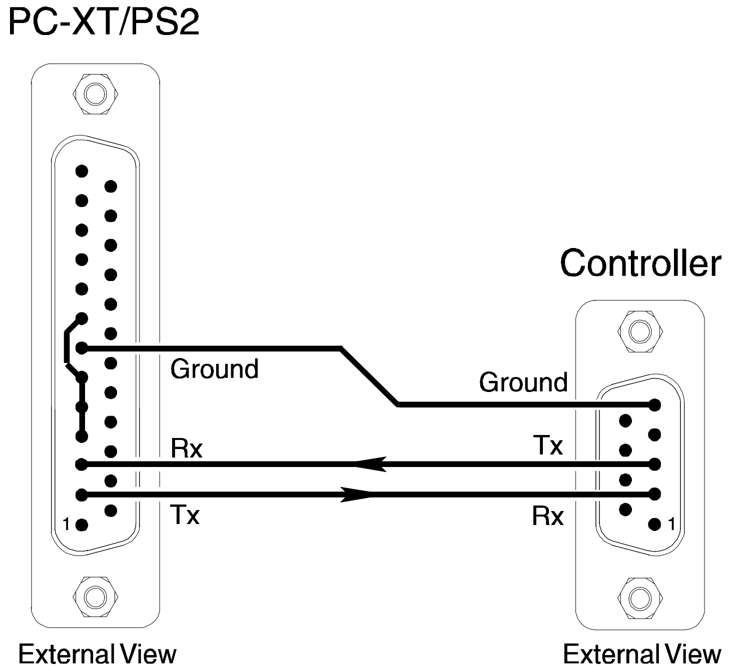
For communication with the Controller via a PC, the interface connections are illustrated in the following figures.

Connection between the Controller and an IBM AT or compatible :



3.1 Interface Connections

Connection between the Controller and an IBM XT/PS2 or compatible :



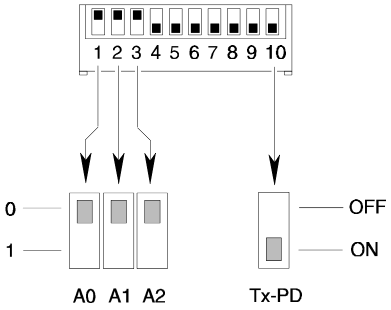
3.2 Interface Addressing

The Controller can be configured to respond to all communication on the interface ("point-to-point" communication). In addition, it is possible to connect up to 7 Controllers on the same interface bus, i.e. "multipoint" communication. For multipoint communication the Controller DIP switches must be set to assign a unique address to each controller on the interface bus. In this way, each controller only responds to interface commands which are preceded by its preset address. To configure Controllers for multipoint addressing, the DIP switch marked *Tx-PD* on one (and only one) of the Controllers must be set to *ON*. For the remaining Controllers in the multipoint configuration, *Tx-PD* must be set to *OFF*. For point-to-point communication, *Tx-PD* is set to *ON* on the Controller.
In order to ensure data integrity during communication, it is recommended that the interface checksum facility is used. Checksum is enabled by setting the CHS DIP-switch to *ON*. For further details, see Section 3.5.

The DIP switch settings for configuring the interface address on Controllers are given in the following table:

A0	**A1**	**A2**	**Address**	**Protocol**
0	0	0	-	Point to point
1	0	0	1	Multipoint
0	1	0	2	Multipoint
1	1	0	3	Multipoint
0	0	1	4	Multipoint
1	0	1	5	Multipoint
0	1	1	6	Multipoint
1	1	1	7	Multipoint

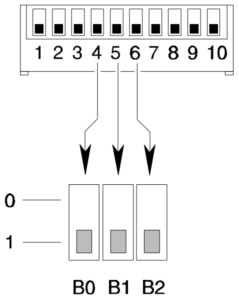
IMPORTANT! : If the address switch settings are changed, the Controller must be reset by switching off and on the power for the new address to take effect.



3.3 Communication Rate

Baud Rates of 110 to 9600 Baud can be selected for communication using the Controller's RS232C interface. The Baud Rate is configured by setting 3 DIP switches, as shown in the following table.

B0	B1	B2	Baud Rate
0	0	0	110
1	0	0	150
0	1	1	300
1	1	1	600
0	0	0	1200
1	0	0	2400
0	1	1	4800
1	1	1	9600



IMPORTANT! : If the Baud Rate setting has been changed, the Controller must be reset by switching the power off and on again for the new Baud Rate to take effect.

Note that the Baud Rate must also be set to the selected value on the PC or terminal used for communication with the Controller. In addition, the communication protocol should be set as follows:

(1 start bit) **7 data bits** **odd parity** **1 stop bit**

() A start bit is always used with RS232C/V 24 protocol.

3.4 Command Syntax

Interface communication with the Controller must ful fill the following command syntax:

Address **Command** **Argument** **Checksum** **Return**

Address : It is only necessary to specify the Controller Address if more than 1 Controller is connected to the interface (multipoint configuration). The specified Address is a value in the range 1 - 7.

Command : The command character(s) to be transmitted to the Controller. See the Software Description in Chapter 4 for details of the Controller commands.

Argument : The command arguments if any. Certain commands such as the *K* (Kill) or *Z* (Smooth Stop) commands have no argument. (See Software Description).

Checksum : The checksum can be used if long communication lines are used between the Controller and PC or terminal. The checksum ensures integrity of data transmission on the interface. If an error occurs, an error message (*E1*) will be received. It is then necessary to re-transmit the command string (see following page).

Return : ASCII character 13. The return character tells the Controller that the command string is complete and interpretation of the command can be initiated.

3.5 Checksum Facility

Electrical noise from sources such as electrical motors is a common occurrence in industrial applications. This noise can be completely random in nature and despite effective electrical filtration electrical noise cannot be eliminated completely. In applications where it is vital to ensure that the system operates precisely as required, it is therefore essential to select a communication rate (Baud Rate) that is not too high. Moreover, the interface cable used to connect the Controller to a PC or terminal should not exceed 10m in length. A typical command string used for interface communication with the Controller will be of the following form:

1A3%

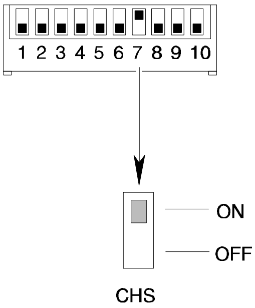
In the above example, multipoint communication is used and the command is being transmitted to a Controller which has address 1. The command is being sent to activate output number 3 (A3). The communication checksum for transmission of a command is determined as follows. First, the ASCII value of each character in the command string is determined. The ASCII values are then summed and the result divided by 128. The integer-result of the division is discarded, while the remainder is used as the checksum. Calculation of the checksum for the above example is thus as follows:

Address character 1 = ASCII 49
Command character A = ASCII 65
Argument character 3 = ASCII 51

Sum + remainder = (49+65+51)/128   Checksum = Remainder * 128

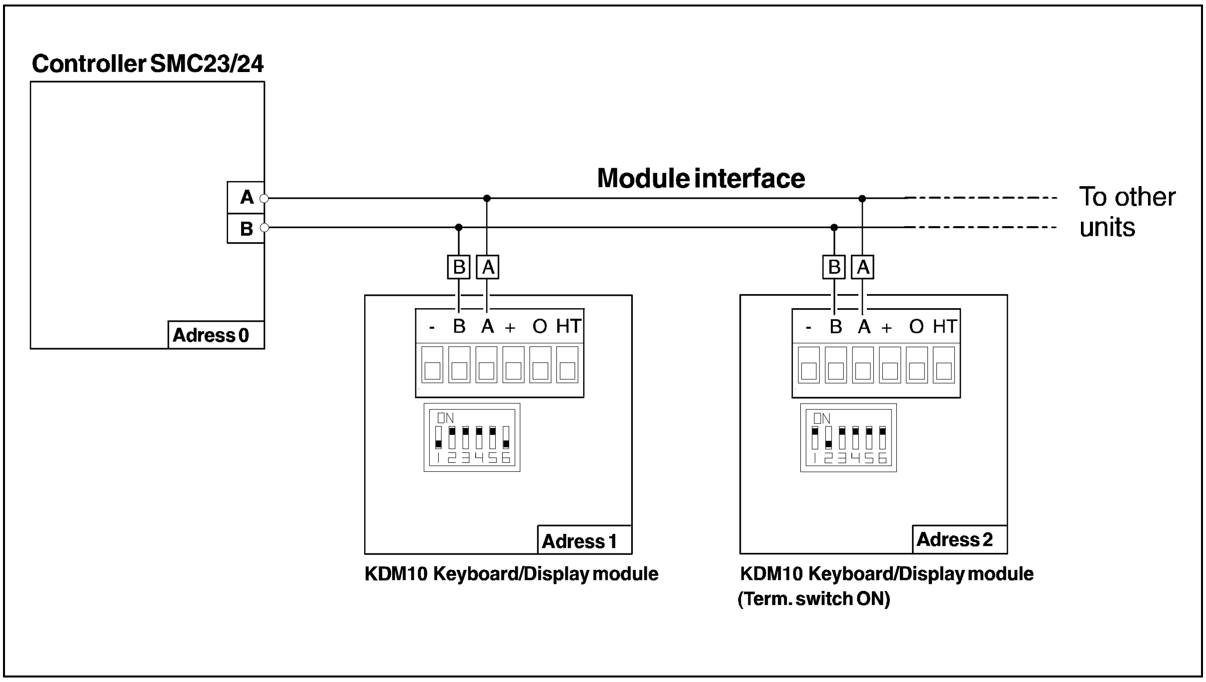
If an error occurs during transmission of the command string, the checksum will be incorrect and the Controller will return the error message *"E1"* indicating that the Controller could not interpret the received command string. The command must then be re-transmitted. If the Controller continues to transmit *"E1"*, the interface Baud Rate should be reduced or a shorter cable used to connect the computer to the Controller.
The Checksum facility is activated by setting the CHS DIP-switch to *ON*.

IMPORTANT! : If the checksum switch setting is changed, the Controller must be reset by switching the power off and then on for the new setting to take effect.



3.6 Module Interface

(Types SMC23 and SMC24 only)



Controllers Type SMC23 and SMC24 can be connected to external modules such as an31, input/output-module, keyboard/display-module etc.

Connection to external modules is made via the Controller's RS485 serial interface using the two terminals marked "A" and "B".

All external module functions are controlled via this interface. Up to 31 modules (and at least 1 motor - controller) can be connected to the interface bus.

The RS485 Interface offers several advantages in that the interface operates with a balanced output and has low impedance. In addition, the Controller's RS485 interface is optically isolated from other Controller circuitry.

The RS485 Interface is protected against transients on the cable connecting the Controller to external modules. These factors enable communication at long distances despite the presence of electrical noise.

It is recommended that twisted-pair cable is used for connection between the Controller and other modules on the interface.

If the communication distance between 2 units in a system exceeds 25 metres, the DIP switch marked *TERM* must be set to the *ON* on those units which are located more than 25 metres apart.

See the User Manual for the module in question for details of DIP switch settings.

Module Addresses:

In communication systems where several modules are connected together, each unit must be assigned a unique address in the range 1 to 31.

The above illustration shows how addresses in a typical system are set.

Note that care must be taken to ensure no two modules use the same address. If the module addresses are not unique, the Controller will terminate program execution and an error message will occur.

Note that the Controller's address is the same as that used for RS232 communication. (See Section 3.2.)

The address of each module should be set in accordance with the instructions given in the respective module's User Manual.

4.1 General Aspects of Controller Software

Before the individual software commands are described in detail, it is necessary to describe some general aspects of the Controller software structure.

The Controller is equipped with 2 types of storage memory, both of which are accessible to the user. These are used for storing programs and operational parameters sent from a computer or terminal.

The first of these storage memories is referred as the *"working memory"* in the following pages. The Controller's working memory is used during connection to a computer or terminal. The working memory is a volatile memory; its contents are deleted when the Controller is switched off. The working memory can also be used for storing instructions during programming.

The second of the Controller's storage memories is an E²PROM, i.e. a non-volatile memory which retains its contents when the Controller is switched off. This is referred to as the Controller's *"permanent memory"* in the following description. The Controller's permanent memory is intended for use when the Controller is used as a *Stand alone unit*, i.e. is not connected to a computer or terminal. Use of the permanent memory enables the Controller to begin execution of pre-programmed instructions without requiring connection to an external PC or terminal.
Permanent memory can also be used if the Controller is connected to a PC or terminal. In this case it is typically used to store frequently used program sequences which can be pre-programmed and downloaded to the permanent memory.

Position Counter.
The Position Counter is a storage register which keeps track of the motor's current position during operation. The Position Counter can be reset by the *I* - (Initialise) or *H* (Home)command (see Sections 4.3 and 4.4). The Position Counter's contents can also be read or changed using the commands *V1* and *f[±n]*.
When the Position Counter reaches its maximum value of +8,388,607 or -8,388,608, the motor stops automatically.

Command Descriptions.
The following pages (Sections 4.3 to 4.6) describe each of the commands used for programming the Controller.
To avoid any misunderstanding regarding the use of the commands and command syntax, the following text convention should be noted:

Each command is described by one or more command characters followed by a word in parentheses. The actual command used to program the Controller using the command syntax consists of the characters, not the word which is included in the description as a mnemonic. Note that many of the command descriptions include examples of the command string.
Almost all commands are followed by one or more parameters: either a value, or a plus (+) or minus (-) sign. It is important that the speci-fied numeric value is within the permitted range since the Controller will not interpret parameters outwith the allowable range.

See also Section 3.4 for details of the Controller command syntax.

4.1 General Aspects of Controller Software

Operating Modes.
The Controller can be operated in 1 of 3 modes:

1) *Standby Mode.*
Standby Mode occurs after a *K* (Kill), *Z* (Smooth Stop) or *PX* (Program Exit) command, and after execution of a program.

2) *Programming Mode.*
This mode is used when a program is read in to the Controller or to edit an existing program. Use the *PO* (Program) or *PE* (Program Enter) command to set the Controller to Programming Mode.

3) *Execute Mode.*
The *E* (Execute) command is used to execute the program currently in the Controller's working memory.
Program execution stops when all commands have been executed or if interrupted by a *K* or *Z* command. Thereafter, the Controller returns to Standby Mode.

Programming.
When creating a new program, the first command is always *PO*, i.e. the Controller is set to Programming Mode. The actual program commands can then be keyed-in.
Once all the required commands have been programmed, the E command is used to switch the Controller to Execute Mode and the program is executed. To store the program in permanent memory, the *M* (Memory save) command can be used once program execution is complete. Programming Mode can also be interrupted using the *PX* (Program Exit) command, in which case the Controller will be set to Standby Mode.

A typical sequence for programming the Controller is as follows:

		Mode:
1)	Controller switched on.	*Standby*
2)	*PO* (Program) command keyed-in.	*Program*
3)	Required sequence of program commands keyed-in.	*Program*
4)	*PX* (Program Exit) command keyed-in, after which the Controller is set to Standby Mode.	*Standby*
5)	*E* (Execute) command keyed-in * The entire program is then executed, unless an interrupt occurs via *K* (Kill) or *Z* (Smooth Stop) command.	*Execute*
6)	The program can be stored in permanent memory by keying-in the *M* - (Memory Save) command.	*Standby*

It should be noted that at power-up, the error message E1 will probably be received when communication is first established between the Controller and a PC/terminal. This is due to transients which arise on the interface cable when the computer or Controller is switched on.

4.1 Controller Response

Each time the Controller receives a command or query via the interface, it responds to the PC or terminal with a short response string. The syntax of the response string is as follows:

Reply Code **Argument** **Checksum** **Carriage-Return**

Reply Code - The Reply Code is the actual response to the received command and is one of the following:

Y = (Yes) The command has been received and will be, or has been, complied with.

B = (Busy) The Controller is busy with program execution and is not ready to receive the command or query.

R = (Ready) The Controller is ready to execute a command or respond to a query.

V = (Verify) Position or User Input/Output status. This message will only occur if the Controller is queried about the status. See the descriptions of the V1 and V2 commands in Sections 4.3 and 4.5 for further details.

E = (Error) An error has been found in the received command and the Controller is not able to comply with the command. This response returns an argument which indicates the type of error as follows:

***E1* -** Parity Error after receiving one or more characters. Checksum Error. The received command string was too long.

***E2* -** The command argument is too long or is unnecessary.

***E3* -** The working memory is full.

***E4* -** Unknown command or the Controller is unable to comply with the received command.

***E5* -** The Position Counter has exceeded its maximum of -8,388,607 or +8,388,607 steps, the motor has been stopped. Error in Parameters (R, S, T).

***E6* -** An error occurred during transmission to or from the Controller's Permanent Memory.

Argument - An argument to the response will only occur with E (Error) or V (Verify) messages. The argument consists of 1 to 7 characters.

Checksum - A checksum value is only included in the response string if the checksum facility is enabled via the DIP switch setting on the Controller (set to *ON*).
See the description of the checksum facility in Section 3.5 for further details.

Carriage-Return - Terminates the response string. ASCII-value 13.

4.2 Command Overview

System Commands :

E	(Execute)	Starts program execution.
f [±nnnnnnn]	(Forcing Pos.)	Reads in new position.
F	(Feedback)	Status query to the Controller.
I [n]	(Initialize)	Resets Controller Registers (software reset).
K	(Kill)	Stops execution of current program.
M	(Memory)	Saves working program in Permanent Memory.
PE	(Program Enter)	Sets the Controller to Programming Mode without erasing existing program in working memory.
PO	(Program)	Sets the Controller to Programming Mode. This command erases any program instructions already in working memory.
PX	(Program Exit)	Exits Programming Mode. Returns to Standby Mode.
Q	(Query)	Displays the program currently stored in Working Memory.
TP	(Temperature)	Returns the current temperature of the Controller.
V1	(Verify)	Returns the Position Counter value.
X	(Recall)	Loads program from Permanent Memory into Working Memory.
Z	(Smooth Stop)	Stops program execution slowly taking account of deceleration ramp.

Motor Commands :

[±nnnnnnn]		Relative positioning given by direction of rotation (+/-) and number of steps.
±A [n].[n1-n2]		Relative positioning controlled by voltage at Analogue Input.
CR [nnnn]	(Current Ramp)	Determines motor current during acceleration.
CS [nnnn]	(Current Start)	Determines motor current when stationary.
CT [nnnn]	(Current Top)	Determines motor current at top speed.
g [±]	(Velocity)	Continuous operation forward/reverse.
G [±nnnnnnn]	(Goto)	Absolute positioning.
G±A [n].[n1-n2]	(Goto)	Absolute positioning controlled by voltage at Analogue Input.
H [±]	(Home)	Resets motor and electronic circuitry.
N [n1n2.n3n4]	(Input Setup)	Starts/stops motor in accordance with User Inputs.
NA [p1.p2]	(Input Setup)	Starts/stops motor in accordance with Analogue Inputs.
R [nnnnn]	(Ramp)	Acceleration/deceleration parameter (1-10000 steps).
RT [nnnn]	(Ramp Time)	Acceleration/deceleration parameter (0.01-10 seconds).
RS [nnnnn]	(Ramp Slope)	Acceleration/deceleration parameter (10-30000 step/s^2).
S [nnnn]	(Start Rate)	Minimum speed.
T [nnnnn]	(Top Rate)	Maximum speed.
r [n1.n2]	(A/D Ramp)	Same as *R*, controlled by voltage at Analogue Input.
s [n1.n2]	(A/D Start Rate)	Same as *S*, controlled by voltage at Analogue Input.
t [n1.n2]	(A/D Top Rate)	Same as *T*, controlled by voltage at Analogue Input.
VR	(Verify Ramp)	Returns the current acceleration/deceleration parameter.
VS	(Verify S. Rate)	Returns the current Start Rate parameter.
VT	(Verify T. Rate)	Returns the current Top Rate parameter.

4.2 Command Overview

(continued)

User Interface :

A [n]	(Activate)	Activates one of the outputs.
C [n]	(Clear)	De-activates one of the outputs.
U [n]	(Until)	Repeats program(segment) until a specified input is activated.
VA [n]	(Verify Ainput)	Returns (measures) voltage at one of the 6 Analogue Inputs.
VA	-	Returns (measures) the logic levels of the Analogue Inputs.
V2	(Verify)	Returns status of user inputs and outputs.
W [n]	(Wait for)	Pauses program execution until a specified input is activated.

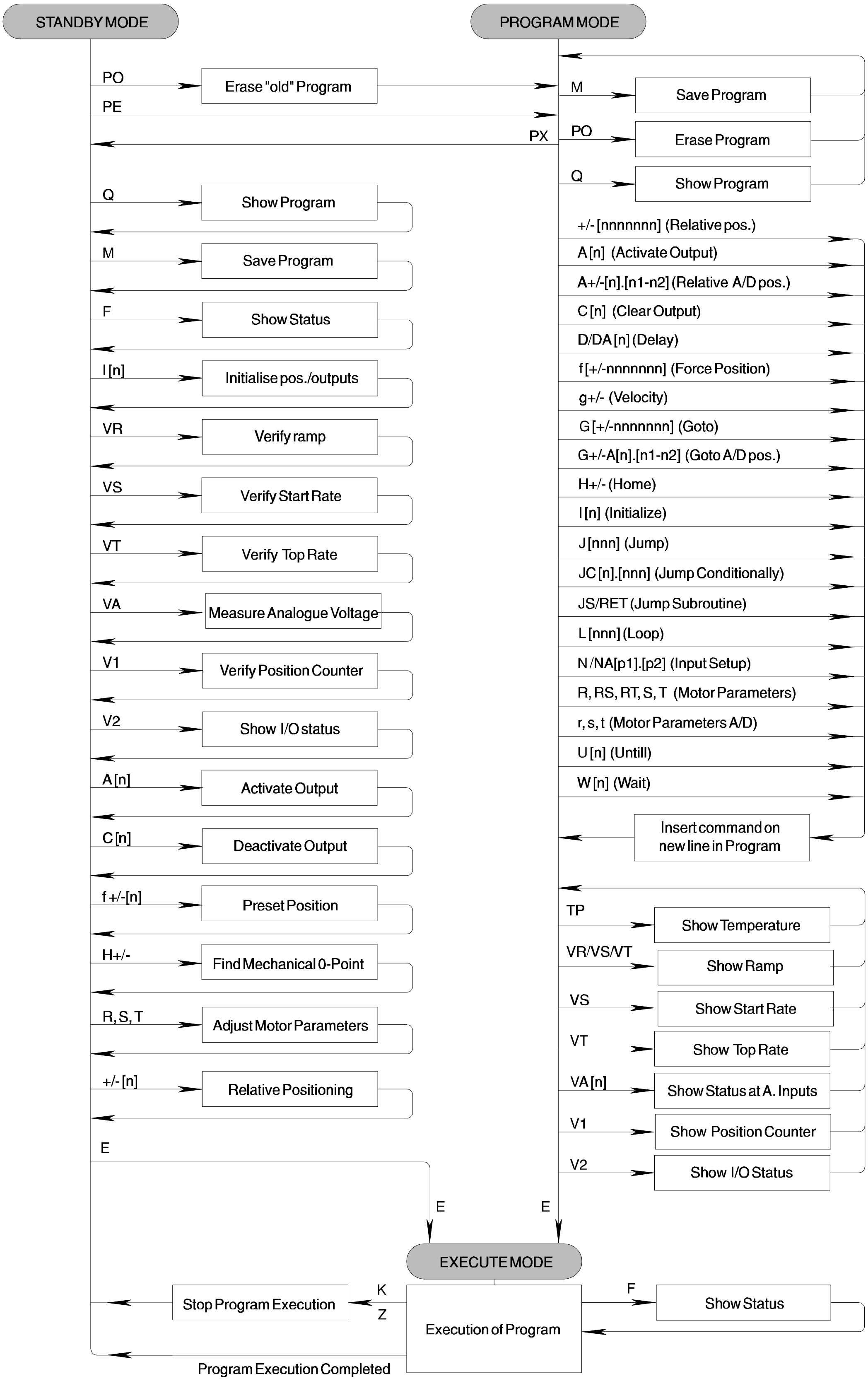
Flow Commands :

D [nnn]	(Delay)	Wait a specified time.
DA [n].[n1-n2]	(Analog Delay)	Wait a specified time controlled by analogue voltage.
J [n1]	(Jump)	Unconditional jump to a specified program line.
JC [n].[n1]	(Jump Cond.)	Conditional jump to a specified program line.
JCA [p].[n1]	(Jump Cond.)	Conditional jump to a specified program line when specified voltage is applied to Analogue Input.
JS [n1]	(Jump Sub)	Unconditional jump to sub-routine.
RET	(Return)	Return from sub-routine.
L [nnn]	(Loop)	Repeat program segment a specified number of times.

Command Overview - Extended Command Set (Valid only for Types SMC23/24)

+ - / *		Arithmetic operators: addition, subtraction, division, multiplication.
VR[0-510]	(Verify)	Return contents of User Register.
I[5-7]	(Initialize)	Initialize User Registers.
con=[n]	(Convert)	Specify conversion between current and a user-specified measurement unit.
PRINT[n1.n2.n3]	(PRINT)	Print Register contents to external module.
IF [p1 m p2]	(IF)	If expression specified by command arguments is true, execute next line.
INPUT[n1.n2.n3]	(INPUT)	Read in data from external module to Register.
AO[a].[o]	(Activate)	Activate flag in external module.
CO[a].[o]	(Deactivate)	Deactivate flag in external module.

4.2 Command Overview



4.3 System Commands

E
(Execute)

Starts program execution. The Execute command can also be used to complete a programming sequence. The command can be used when the Controller is either in Standby Mode or Programming Mode.

f+/-[nnnnnnn]
(Forcing pos.)

Assigns a specified value to the Position Counter.
The position can be specified in the range -8,388,607 to +8,388,607, both values included. The command can be used when the Controller is in Standby Mode and in Programming Mode.

Example:
f+100 assigns a value of +100 to the Position Counter.

F
(Feedback)

Status Query to the Controller. 1 of 3 responses will occur.

1) If the Controller is ready to receive and execute commands, the Status Query response is R (Ready).

2) If the Controller is busy, the Status Query response is B (Busy).

3) If the motor has been stopped automatically because of an overflow in the Position Counter, the Status Query response is E5 (Error 5).

I [1-3]
(Initialize)

The Initialize command is used to reset either the Position Counter and/or User Outputs.

I1 = Resets the Position Counter only.

I2 = Resets the User Outputs only.

I3 = Resets both the Position Counter and User Outputs.

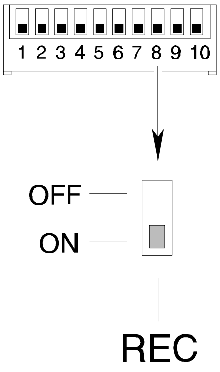
K
(Kill)

The Kill command has the highest priority since it stops program execution regardless of motor movement. The Kill command is effective immediately, i.e. as soon as the command is issued, the Controller 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 H (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 Kill command.

4.3 System Commands

M
(Memory Save)

To enable completed programs to be permanently stored after the power has been switched off, the Controller is equipped with a permanent, non-volatile memory. The Memory Save command is used to store the contents of the Controller's volatile working memory in the non-volatile permanent memory. Only 1 program can be stored in permanent memory at a time. If the *REC* DIP switch is set to "ON", the program stored in permanent memory is automatically recalled and executed when the Controller is switched on.



PE
(Program Enter)

The Program Enter command is used to set the Controller to Programming Mode without erasing any existing instructions in the working memory. This command is primarily used when editing a program during development.

PO
(Program)

The Program command sets the Controller to Programming Mode, i.e. so that the Controller is ready to receive programming instructions. Each time the Program command is used, the contents of the Controller's working memory are reset, erasing any existing instructions. (See also the Program Enter command above and the description of the Program command at the beginning of this Chapter.)

PX
(Program Exit)

The Program Exit Command is used to exit Programming Mode and set the Controller to Standby Mode. A program can then be executed or a new program keyed-in.

Q
(Query)

The Query command returns the program currently stored in working memory, including run-time parameters. If a printout of the program in permanent memory is required, the X (Recall Program) command should be used prior to the Query command. Note that use of the Recall Program command will erase the contents of the Controller's working memory. Note that the Q (query) command is only implemented in Types SMC25 and SMC26, it is **not** available in Types SMC23 and SMC24.

4.3 System Commands

TP
(Temperature)

Returns the current temperature of the Controller.
The Temperature command can be used to verify that the Controller is operating within its specified temperature range of 0-50°C. If, under "worst-case" conditions, a temperature greater than approximately 60°C is registered, ventilation of the Controller must be improved.

V1
(Verify Pos.)

The Verify Position command is used to read the contents of the Position Counter.
The value returned is relative to 0, the Home position. (See also the Home command).

X
(Recall Prog.)

The Recall Program command is used to read the program (if any) stored in the Controller's non-volatile, permanent memory and load the program into the working memory. This command can be used advantageously if, for example, a program is to executed at regular intervals. A Recall Program command is then followed by an Execute command, thus starting program execution immediately.
Note that each time the Recall Program is used to load a program from permanent memory in to working memory, any instructions in the Controller's working memory will be erased.

Z
(Smooth Stop)

The Smooth Stop command has the same function as the Kill command.
except that the motor is decelerated in accordance with the specified R, S, T parameters.
The Smooth Stop command is thus used to ensure that the motor does not stop at an undefined position. (See also the *K* (Kill) command).

4.4 Motor Commands

+/- [nnnnnnn]
(Relative)

The Relative command is similar to the Goto command. Instead of positioning the motor relative to the 0 (Home) position, the Relative command positions the motor relative to its current position. The command specifies the direction (+ or -) and the number of steps the motor is moved.

The number of steps can be specified in the range 1 to 8,388,607 steps.

Example :

+15 , A Relative positioning command of +15 will advance the motor 15 steps relative to its current position.

±A [n].[n1-n2]

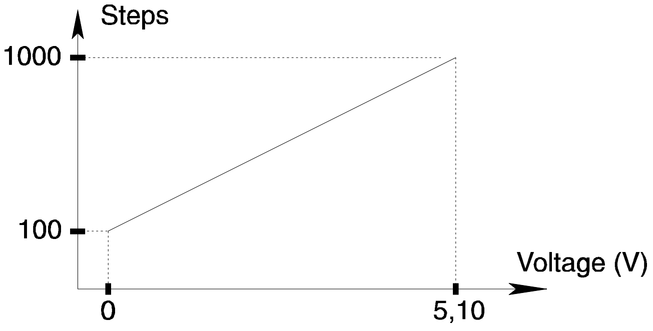
This command is used to move the motor a specified number of steps, using the value of an analogue voltage.

The command parameter "n" specifies which Analogue Input (1-6) is used for controlling the movement. The parameters "n1" and "n2" specify the step interval, where n1 indicates the number of steps corresponding to an input voltage of 0V, and n2 indicates the number of steps corresponding to an input voltage of 5.1V. n1 and n2 can be specified from 1 to 65000 steps.

Example :

+A1.100-1000

The above command advances the motor (indicated by +) 100 steps if a voltage of 0V is applied to Analogue Input 1, and 1000 steps if a voltage of 5.1V is applied. For voltages between 0V and 5.10V a linear interpolation is used to determine the number of steps the motor is moved (see figure below).



4.4 Motor Commands

The current supplied to a stepper motor can be adjusted to specified values for standby, acceleration/-deceleration, and top speed. Normally only a small current is required when the motor is stationary since the static inertia of a typical stepper motor is much less than the inertia while the motor is rotating, depending on the speed range of the motor.
The torque of a stepper motor is directly proportional to the applied current, up to the specified phase current (see the specifications for a given motor).
In the nominal current is exceeded, the motor will overheat and only very little increase in torque will result.

The following 3 commands are used to specify the current supplied to the motor. The commands can be used at any point in a program. All 3 commands can be specified and changed continuously throughout a program.
If any of the commands is omitted, the respective parameter assumes a default value of 1000mA.

CS [0-6000] (Current Standby)
Determines motor current when the motor is stationary.

CR [0-6000] (Current Ramp)
Determines motor current during acceleration/deceleration.

CT [0-6000] (Current Top)
Determines motor current at maximum speed.

Note that all 3 parameters can only be specified in the range 0 to 3000 mA for Controllers Types SMC23A/SMC24A/SMC25A/SMC26A (3Amp versions).

Example:

(Program)

.

CS500	Sets motor Standby Current to 500mA (0.5A).
CR6000	Sets motor Ramp Current during acceleration/deceleration to 6000mA (6A).
CT4000	Sets motor Top Current to 4000mA (4A) at top speed.
+100	Advances the motor 100 steps.
CT5500	Sets new motor Top Current to 5500mA (5.5A).

.

.

4.4 Motor Commands

g+/-
(Velocity Mode)

The Velocity Mode command is used to move the motor continuously in a specified direction.

The command is followed by a + or - parameter which specifies the direction of movement.

To stop the motor once the Velocity Mode command has been used, a *Z* (Smooth Stop) or *K* (Kill) command must be used.

If the *N* (Input Setup) command is used before the *g*± command, the conditions specified by the *N* command can also stop the motor. (See the description of the Input Setup (N) command for further details.)

It should be noted that the Position Counter is updated while the Velocity Mode command is executed. The command can only be used when it is included in a program.

G+/- [nnnnnnn]
(Goto)

The Goto command is used for absolute positioning of a stepper motor.

The specified parameter value refers to the Position Counter and can be specified in the range -8,388,607 and +8,388,607.

G±A [n].[n1-n2]
(Analog Goto)

The Analog Goto command is used for absolute positioning of a motor motor (similar to the G±[n] (Goto) command) but the required position is determined by the analogue voltage applied to a specified Analogue Input. The "n" command parameter specifies which Analogue Input (1-6) is used for the control signal. Parameters "n1" and "n2" specify the required positions corresponding to applied voltages of 0V and 5.10V respectively. The specified position can be set in the range +0 to +65000.

See also the ±A command.

Example:

G±A2.0-800

The above example moves the motor to position +0, if a voltage of 0V is applied to Analogue Input 2, and to position +800 if the applied voltage is 5.10V.

For applied voltages between 0 and 5.10V a linear interpolation between positions +0 and +800 is made.

H+/-
(Home)

The Home command enables an electrical and mechanical reset of the system to a pre-defined reference position. As soon as the Controller receives the Home command, the motor will move in the specified direction (either *H+* or *H-*).

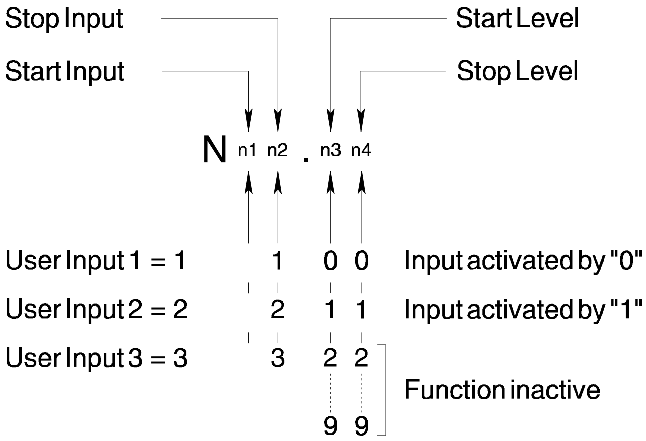
As soon as the *EOT* (End of Travel) input becomes low, the motor will stop. The motor is then at its reference position. The speed at which a reset occurs is determined by the *S* (Start Rate) command.

After execution of a Home command, the Position Counter is reset to "+*0*".

N [n1n2.n3n4] (Input setup) The Input Setup command enables a motor to be started or stopped using control signals at the User Inputs.

The Input Setup command itself does not start or stop the motor. It only determines how the next motor command ±*[n]* / *g[±]* / *G±[n]* will be interpreted and executed. Thereafter the Input Setup command is inactive until a new Input Setup command is executed. To subsequently start or stop the motor using the control signal at a User Input, a new Input Setup command must precede the new motor command.

Command Syntax:



n1: Specifies the User Input (1-3) used to start the motor.

n2: Specifies the User Input (1-3) used to stop the motor.

n3: Refers to n1 in that n3 determines the logic level to be applied to the specified User Input in order to start the motor. If *n3* is set to 0, the motor will start when logic level 0 is applied to the specified User Input.
If n3 is set to a value between 2 and 9, the start function will be inactive and the motor will start immediately.

n4: Refers to n2, in that n4 determines the logic level to be applied to the specified User Input in order to stop the motor. If *n4* is set to 1, the motor will stop when logic level 1 is applied to the specified User Input.
If n4 is set to 0, the motor will stop when the level changes from logic 1 to logic 0.
If n4 is set to a value from 2 to 9, the stop function will be inactive and the motor will only be stopped a motor command requires it.

(continued on following page)

(continued)

N [n1n2.n3n4]
(Input setup)

When the Input Setup command is used for a movement sequence, the Position Counter is updated normally.

While the motor is moving, a *Z* (Smooth Stop) or *K* (Kill) command can be used to stop the motor. Program execution can also be halted using a *K* (Kill) or *Z* (Smooth Stop) command while the Controller is waiting for a start signal from a User Input.

Example 1:

The command N13.01 followed for example by a *g+* command will start the motor when User Input 1 attains a voltage of logic 0. Note that it is a logic level 0, and not a change from "1" to "0" that activates a start.

The motor will move according to the specified parameters and run at normal speed until a voltage corresponding to logic "1" is applied to User Input 3. Thereafter the motor will decelerate until it stops, and the next program command is executed.

Example 2:

The command N21.10 followed for example by a *+10000* command will start the motor when User Input 2 is logic "1" and operate at normal speed for 10000 steps (including deceleration ramp), or until a change from logic 1 to logic 0 occurs at User Input 1.

The motor will then decelerate and the next program command is executed.

Example 3:

The command N11.19 followed for example by a *G+3500* command will start the motor when User Input 1 becomes logic "1" and stop when position +3500 is reached. The motor will operate in accordance with the parameters specified by the R, S and T commands.

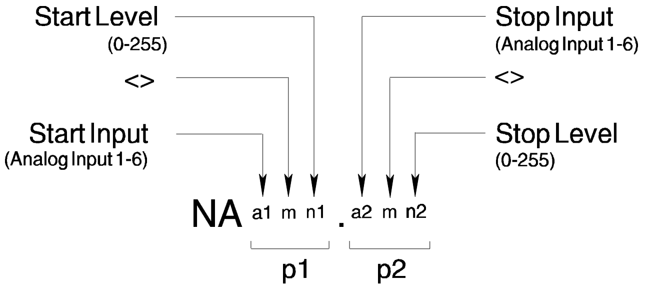
4.4 Motor Commands

NA [p1.p2]
(Analog Input setup)

This command enables start/stop control of the motor via control signals at the analogue inputs. The principle of the Analog Input Setup command is the same as the Input Setup command, N [n1n2.n3n4].

The 2 command parameters (p1 and p2) specify start and stop conditions respectively. The Analog Input Setup command itself does not start or stop the motor. It only influences how the next motor command *±[n]* / *g[±]* / *G±[n]* will be interpreted and executed. Thereafter the command is inactive. To subsequently start the motor again using control signals at an Analogue Input, a new Analog Input Setup command must precede the new motor command.

The complete syntax for the Analog Input Setup command is as follows:



p1: If the start conditions are fulfilled, the motor is started. If p1 is assigned the character *X*, an unconditional start is defined, and the motor will start immediately the motor command is executed.

p2: If the stop conditions are fulfilled, the motor will be stopped. If p2 is assigned the character *X*, an unconditional stop is defined and the motor will operate until the specified motor command *(±[n], g±, G±[n])* requires it to stop.

a1: Specifies the Input used for the start control signal. a1 can be specified in the range A1 to A6, corresponding to Analogue Inputs 1-6.

a2: Specifies the Input used for the stop control signal. a2 can be specified in the range A1 to A6, corresponding to Analogue Inputs 1-6.

n1: Specifies the Start Reference Value. The Reference Value is compared with the measured voltage at the specified start Input (a1). The Start Reference Value can be set in the range 0 to 255.

n2: Specifies the Stop Reference Value. The Reference Value is compared with the measured voltage at the specified stop Input (a2). The Stop Reference Value can be set in the range 0 to 255.

m: The operator for comparison between the Reference Value and the measured value at the input(s). To start/stop the motor when the voltage at the respective analogue input is less than the Reference Value, the operator should be specified as "<". To start/stop the motor when the applied voltage is greater than the Reference Value, the ">" operator is specified.

(continued)
NA [p1.p2]
(Analog Input setup)

Since n1 and n2 are specified as values in the range 0-255 and the voltage measured at the Analogue Inputs is in the range 0-5.10V, a conversion must be made when specifying n1 and n2, either by converting the Reference values to a voltage or vice versa. The conversion is made as follows:

Vref = 0.02 x n or n = 50 x Vref

Example:

If a Reference Value of 1.20V is required, n should be specified as:

n = 50 x 1.2 = 60

Program Example 1:

```
.
.
NAA1<60.A6>100
+10000
.
```

In the above program example, the motor is started if the applied voltage at Analogue Input 1 is less than 1.2 Volts (see conversion example above). Otherwise nothing occurs. When the motor is running, it is stopped either after completion of the 10000 steps or when the voltage at Analogue Input 6 is greater than or equal to 2V (n2=100).

Program Example 2:

```
.
.
NAX.A5>220
G+100000
.
```

The parameter specification X indicates that the start condition is inactive and the motor will therefore start immediately. Thereafter the motor will stop when position +100000 is reached or if the applied voltage at Analogue Input 5 is greater than or equal to 4.4V (n2=220).

ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

The K (Kill) and Z (Smooth Stop) commands can be used to interrupt program execution.

In contrast to a normal DC motor (which is *"Self-commutating")*, a stepper motor is electrically commutated. That is, a stepper motor is driven by magnetic fields which are controlled electronically. When the motor is loaded, the magnetic fields will eventually not be powerful enough to continue to turn the rotor. The motor will stop, but the electronics will continue to move the magnetic fields at the same speed. It is therefore important that a motor is accelerated and decelerated at appropriate rates, in order for the magnetic fields to drive the rotor.
Similarly a stepper motor has a maximum speed and if this is exceeded the motor can no longer provide the same power and will simply stop.
There are 3 basic parameters which should be considered:

S [16-2000] (Start Rate steps/second)

The Start Rate is the speed at which the motor is started. If it is set too high, the motor will simply stop at an arbitrary position. The Start Rate can be set in the range 16 to 2000 steps/second. The default Start Rate is 100 steps/second.

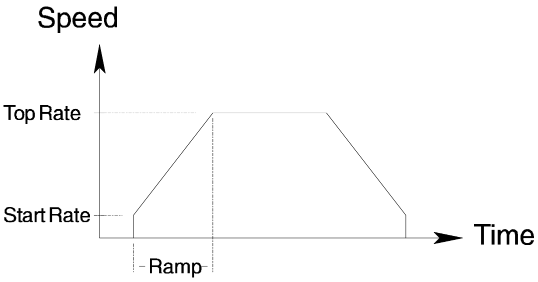
T [16-15000] (Top Rate steps/second)

The Top Rate specifies the maximum speed of the motor. If it is set too high, the motor will be unable to provide enough power and will stop at an arbitrary position. The Top Rate can be set in the range 16 to 15000 steps/second. The default Top Rate is 1000 steps/second.

R [1-10000] (Ramp) / **RT [1-1000]** (Ramp time) / **RS [10-30000]** (Ramp slope)

This value specifies how the motor is accelerated and decelerated. The value can be specified in 1 of 3 forms. R (Ramp) is used if the required acceleration/deceleration is specified in steps. RT (Ramp Time) is used if the required acceleration/deceleration is specified in terms of time, and RS (Ramp Slope) is used if the required acceleration/deceleration is specified in steps/second². Ramp Slope can be used advantageously if the Top Rate or Start Rate are repeatedly changed in a program, since the acceleration per unit of time remains the same. If too high an acceleration/deceleration rate is selected, the motor will stop.
R [n] can be specified in the range 1 to 10000 steps. The default Ramp is 100 steps.
RT [n] can be specified in the range 1 to 1000, corresponding to a range from 0.01 to 10 seconds.
RS [n] can be specified in the range 10 to 30000 steps/second².

Examples:
R100 specifies an acceleration/deceleration of 100 steps. RT50 results in an acceleration/deceleration time of 0.5 seconds. RS900 gives an acceleration/deceleration rate of 900 steps/second².



All 3 parameters must be specified in a program and can be adjusted at any point in the program. If T (Top Rate) is set to a value less than S (Start Rate), the motor will operate at the Top Rate specified by T without accelerating or decelerating.

4.4 Motor Commands

r [n1.n2] These 3 commands are used to determine the motor parameters R, S
(A/D Ramp Step) and T using the analogue voltage applied at one of the 6 Analogue Inputs.
s [n1.n2] The r [n1.n2] command determines the Ramp Step parameter.
(A/D Start Rate) The s [n1.n2] command determines the Start Rate parameter
The t [n1.n2] command determines the Top Rate parameter.

t [n1.n2]
(A/D Top Rate) n1 specifies the Analogue Input used for controlling a given step/frequency. n1 can be specified in the range 1-6 corresponding to the required Analogue Input 1 to 6. The voltage applied to the specified input must be in the range 0 to 5.10V.

n1	Analog Input
1	AN1
2	AN2
3	AN3
4	AN4
5	AN5
6	AN6

n2 specifies the value corresponding to full-scale input voltage (5.10V) either in terms of steps for the *r [n1.n2]* command, or in terms of frequency for the *s [n1.n2]* or *t [n1.n2]* command.
n2 can be specified as a value in the range 1 to 10 according to the following table for Ramp, Start Rate and Top Rate.

n2	r[n1,n2]	s[n1,n2]	t[n1,n2]
	step	step/sec	step/sec
1	100	100	1000
2	200	200	2000
3	300	300	3000
4	400	400	4000
5	500	500	5000
6	600	600	6000
7	700	700	7000
8	800	800	8000
9	900	900	9000
10	1000	1000	10000

A voltage of 0V at a specified Analogue Input always corresponds to either 16 steps for the *r [n1.n2]* command, or 16 steps/second for the *s [n1.n2]* or *t [n1.n2]* commands.
The *VR*, *VS* and *VT* commands can be used to verify current parameter settings. See the description of these commands for further details.

(continued on following page)

(continued)

r [n1.n2]
(A/D Ramp Step)

s [n1.n2]
(A/D Start Rate)

t [n1.n2]
(A/D Top Rate)

Example:

The command t1.4 is used in a program. When the program is executed, the Controller measures a voltage of 2.5V at Analog Input 1 (AN1). This voltage is converted to a frequency of:

$$\frac{2.5 \text{ x } 4000}{5.10} = 1960 \text{ Hz} = 1960 \text{ steps/second}$$

This frequency is then used for the next motor movement. The specified value of n2 in the example results in 0V corresponding to a frequency of 16Hz and full-scale 5.10V corresponding to 4000Hz.

VR
(Verify Ramp)

VS
(Verify Start Rate)

VT
(Verify Top Rate)

The Verify Ramp, Verify Start Rate, and Verify Top Rate commands can be used to verify the current values of the motor parameters R, S and T. VR returns the value of the Ramp Step *"R"* in steps. VS returns the value of the Start Rate *"S"* in steps/second. VT returns the value of the Top Rate in steps/second.

Example:

A verification of the current Top Rate is required. The following command string is therefore sent to the Controller:

VT (carriage return)

The Controller responds:

T1000 (carriage return)

Indicating the current Top Rate is 1000 steps/second.

VR, VS and VT can also be used to check the value of the motor parameters determined by the *r [n1.n2]* , *s [n1.n2]* and *t [n1.n2]* commands.

4.5 User Interface Commands

A [1-3]
(Activate Output)

The Activate Output command sets a specified User Output to logic "1".
The command character is followed by a parameter value of 1 to 3 which specifies which output is to be activated.

Example: A2 sets Output 2 to logic "1".

C [1-3]
(Clear Output)

The Clear Output command sets a specified User Output to logic "0".
The command character is followed by a parameter value of 1 to 3 which specifies, which output is de-activated.

Example: C1 sets Output 1 to logic "0".

U [1-3]
U [A1-A6]
(Until)

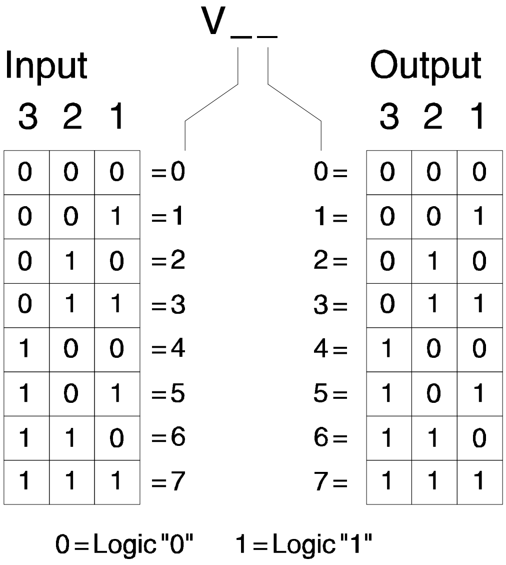
The Until command is used to repeat a program segment until logic "0" is applied to a specified input (see Electrical Specifications).
Either the entire program or only a specified segment can be repeated. The User Inputs are specified by the parameter values 1 to 3. The Analogue Inputs are specified by the parameter values A1 to A6.

e.g.:

```
      .
      .
      .
      G+50
      A2
*     U3     -    Repeats program segment from the beginning until logic
                  "0" is applied to User Input 3.
      A1
      D25
      C1
**    UA1    -    Repeats program segment between * and **, until logic "0"
                  is applied to Analogue Input 1.
      .
      .
      .
```

V2
(Verify I/O)

The Verify I/O command enables the status of User Inputs and Outputs to be determined. When the *V2* command is sent to the Controller, it responds with a *V* followed by two parameter values between 0 and 7.
The first parameter indicates the voltage levels at the User Inputs. The second parameter indicates the voltage levels at the User Outputs.
The status of the Inputs and Outputs is determined from the returned parameters according to the following table:



V_ _

Input 3	Input 2	Input 1			Output 3	Output 2	Output 1
0	0	0	=0	0=	0	0	0
0	0	1	=1	1=	0	0	1
0	1	0	=2	2=	0	1	0
0	1	1	=3	3=	0	1	1
1	0	0	=4	4=	1	0	0
1	0	1	=5	5=	1	0	1
1	1	0	=6	6=	1	1	0
1	1	1	=7	7=	1	1	1

0=Logic "0"   1=Logic "1"

Example:
The Verify I/O query returns a response *V25*, which indicates that Input 2 is logic "1", and Outputs 1 and 3 are logic "1".

The Verify I/O command is used exclusively when there is a constant interface connection between a computer/terminal and the Controller. It cannot be used in a program.

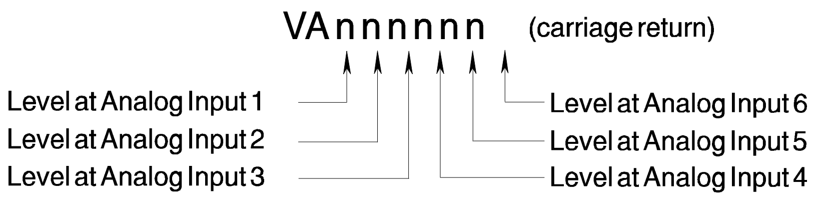
VA [1-6]
(Verify ainput)

The Verify AInput command is used to determine the voltage at a specified AnalogueInput. 8-bit resolution is used for the measurement, which results in the measured value being given in 20mV steps with 5.10V as the maximum value. The Verify AInput command can be used when the Controller is in Standby Mode. The 3 User Inputs can, for example, be connected to 3 Analogue Inputs and thus verify that the voltages are as expected.

4.5 User Interface Commands

VA
(Verify ainput)

The Verify AInput returns the digital levels at the Analogue Inputs. The response string has the following format:



Example:

The command string *"VA"* is sent to the Controller.
The following response is returned:

VA101001 (Carriage return)

indicating that Analogue Inputs 1, 3, and 6 are logic 1 (>2.5V), and Analogue Inputs 2, 4, and 5 are logic "0" (<2.5V).

W [1-3]
W [A1-A6]
(Wait For)

The Wait For command stops program execution until logic "1" is applied to a specified input. If the command character is followed by a number from 1 to 3, the specified input is one of the 3 User Inputs. If the command parameter is A1 to A6, the specified input is one of the 6 Analogue Inputs.

Example:

```
.
.
A3
G+372
W1    -   Pauses program execution until User Input 1 is logic "1".
G+46
C1
D20
WA5      -   Pauses program execution until Analogue Input 5 is logic "1".
.
.
```

4.6 Flow Control Commands

D [1-32000]
(Delay)

The Delay command pauses program execution. The command character must be followed by a parameter value between 1 and 32000 which specifies the Delay duration in 1/100 second.

Example:

D27 - results in a delay of 0.27 seconds.

DA[n].[n1-n2]
(Analog delay)

The Analog Delay command is used to set a delay in program execution which is determined by one of the Analogue Inputs and can vary from 0.01 to 320 seconds.

Command Format:

n: Specifies which Analogue Input is used to control the delay.

n1-n2: Specify the required delay duration. n1 is the lower limit and corresponds to the delay when a voltage of 0V is applied to the specified input. n2 is the upper limit and corresponds to the delay when a voltage of 5.10V is applied.

Example:

DA2.10-100 - Enables a delay of between 0.1 and 1.0 seconds, controlled by a voltage of between 0 and 5.10V applied to Analogue Input 2.

J [n1]
(Jump)

The Jump command is used to make an unconditional jump to a specified line number in the program.
The program line number "n1" can be specified in the range 0-255.

Example:

```
Line no.:
   0        A1
   1        +1000
   2        A2
   3        G+5
   4        C2
   5        J2
```

The Jump command at line 5 in the above example causes the A2 , G+5 , and C2 commands (lines 2 to 4) to be continuously repeated. The program can only be interrupted using the *Z* (Smooth Stop) or *K* (Kill) command.

4.6 Flow Control Commands

JC [0-7].[0-255] (Jump Cond.)

In contrast to the *J* (Jump) command, the *JC* (Jump Conditional) command is used to make a conditional jump to a specified line number in a program, depending on the levels at all 3 User Inputs. The line number can be specified in the range 0-255. The condition for the jump to occur is specified according to the following table:

Example:

The command JC5.10 results in a jump to program line 10 if User Inputs 1 and 3 are logic "1".

JC _ . [nnn]

Input

3	2	1	
0	0	0	=0
0	0	1	=1
0	1	0	=2
0	1	1	=3
1	0	0	=4
1	0	1	=5
1	1	0	=6
1	1	1	=7

0 = Logic "0"
1 = Logic "1"

Example:

Line no.:

```
0     S450
1     R200
2     A1
3     +1200
4     JC5.4        -   Jumps to line 4 if User Inputs 1 and 3 are logic "1".
5     G+0
6     C1
7     JC3.2        -   Jumps to line 2 if User Inputs 1 and 2 are logic "1".
```

4.6 Flow Control Commands

JC [p].[n1]
(Jump con.)

This conditional jump command is similar to JC[0-7] - if an input level condition is fulfilled, a jump is made to a specified program line. In contrast to the JC[0-7] command however, the JC[p] jump condition is determined by the level at a specific Input and not by the pattern at all 3 User Inputs.
The JC[p] command can be used both with User Input levels and with the Analogue Inputs.

Command Format:

JC [i]=[n].[n1]

i: Specifies the input used for the jump condition.
A value of "i" between 1 and 3 specifies the corresponding User Input 1 to 3. A value of "i" between A1 and A6 specifies the corresponding Analogue Input 1 to 6.

n: Specifies the Reference Level to be compared with the measured level at the specified input. n can assigned a value of 0 or 1. If the level at the specified input is equal to the Reference Level, the jump is made.

n1: Specifies the program line to jump to if the jump condition is fulfilled.

Program Example:

```
Line no.:
   0         S750
   1         R700
   2         C3
   3         +500
   4         JC2=0.3    -    Jumps to line 3 if User Input 2 is logic "0".
   5         G+0
   6         T1000
   7         A2
   8         JCA4=1.5   -    Jumps to line 5 if Analogue Input 4 is logic "1".
```

When the Analogue Inputs are used as logic inputs, logic "1" corresponds to voltages greater than or equal to 2.5V, and logic "0" corresponds to voltages less than 2.5V.

4.6 Flow Control Commands

JCA [p].[n1]
(Jump Cond.)

This conditional jump command is similar to the JC command - if the level at a specified input fulfils the jump condition, a jump is made to the specified program line.

In contrast to the JC command however, the JCA jump condition is determined by an analogue voltage level at one of the Analogue Inputs 1 to 6, and not by a logic level.

Command Format:

JCA [a1mn].[n2]

a1: Specifies the Analogue Input used for evaluating the jump condition.

mn: Specifies the Reference Level (n) with which the measured input voltage is compared and the operator for the comparison (m).
If "m" is specified as ">", the jump is made if the measured input voltage is greater than or equal to the Reference Value. If m is specified as "<", the jump is made if the measured voltage is less than the Reference Value.
The Reference Level "n" can be specified in the range 0 to 255.

n1: Specifies the program line number to jump to if the jump condition is fulfilled.

Since *n* is specified as a value in the range 0-255 and the measured voltage is a value in the range 0 to 5.10V, a conversion of voltage to a valid Reference Level value must be made when specifying the jump condition. The conversion is made as follows:

Vref = 0.02 x n   or   n = 50 x Vref

Example:

A Reference Value of 3.00V is required.

n = 50 x 3.00 = 150

Program Example:

```
Line no.:
   0         S450
   1         R600
   2         D2
   3         +342
   4         JCA3>150.2    -    Jumps to program line 2 if the voltage at Analogue Input 3 is
                                greater than or equal to 3.00V.
   5         G+0
```

4.6 Flow Control Commands

JS [n1]
(Jump Sub.)

In contrast to the *J* (Jump) command which jumps to a specified program line number,the *JS* (Jump Sub) command makes an unconditional jump to a program sub-routine.

When a *JS* command is executed, the Controller first stores the number of the next line after the *JS* command and then jumps 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 execution from there.

The JS command be used up to 32 times in a program, corresponding to 32 nested sub-routines.

L [0-255]
(Loop)

The Loop command is used to repeat execution of a specified program segment. The command parameter specifies the number of times the Loop is executed and can be specified in the range 1 to 255. The segment to be repeated must be delimited by a pair of Loop commands such as *L0* and *L5*, as illustrated in the following example.

Example:.

```
    L0
     .
     .
     .
(Program)
     .
     .
     .
    L5
     .
```

The program segment between *L0* and *L5* will be repeated 5 times. If the initial Loop delimiter *L0* is omitted, the entire program will be repeated from line 1.

4.7 Extended Command Set (Types SMC23/24 only)

The following pages describe the Extended Command Set available only with Stepper Motor Controllers Types SMC23 or SMC24. Note that the Q (Query) command is not implemented in Controller Types SMC23 and SMC24.

4.7 Extended Command Set (Types SMC23/24 only)

Register description.

Types SMC23 and SMC24 are equipped with 510 User Storage Registers which can be used for storing intermediate results, etc. These are designated R1 - R510. In addition, the Controllers are equipped with 7 predefined registers which can only be used for specific purposes. Register T for example is used to determine the motor Top Rate. The user- and predefined registers enable parameters such as lengths, speeds, acceleration, delay times, program loops, etc., to be continuously changed and controlled during program execution.
In addition, the User Register contents can also be stored permanently in the Controller's EEPROM memory. For example, parameters which have been set via Keyboard/Display Module KDM10 can thus be stored permanently in the Controller and recalled when the system is started up.

The following registers are available in Types SMC23 and SMC24:

R : Predefined register for the number of steps used to accelerate/decelerate the motor (Ramp). Note that this register is completely independent of, and should not be confused with, User Registers R1 to R510.
S : Predefined register for Start Rate.
T : Predefined register for Top Rate.
D : Predefined register for program Delay.
L : Predefined register for program Loop Counter.
n : Predefined register for Position Counter in steps.
A1-A6 : Predefined registers for Analogue Inputs' level from 0-255 corresponding to voltages from 0-5.10V.
R1-R510 : 510 User Registers for intermediate results, etc.

Example 1:

```
R2 = 3000          ; Sets the value of register R2 to 3000
T  = R2 + 100      ; Sets the Top Rate to 3100 steps/second for the next motor
                   ; operation.
```

Example 2:

```
R34 = 400
D   = R34 + A1      ; Waits (400 + A/D conversion of Analogue Input 1 level) x 10ms
```

Example 3:

```
R1 = n + R2         ; Sets the value of R1 to value of the Position Counter in steps + R2
```

Example 4:

```
R1 = 350 + 700      ; Sets the value of R1 to 1050
+(R1)               ; Advance the motor 1050 steps.
```

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
 G+(R34)            ; Goto (move motor to) position R34
```

In addition, register arrays can be defined by using one register to point to the contents of other registers. This enables, for example, the contents of a block of registers to be copied to a different array.

Example 6: The contents of ARRAY1 (registers 100-199) are copied to ARRAY2 (registers 300-399)

```
R1=100           ; Set Array 1 pointer
R2=300           ; Set Array 2 pointer
L0               ; Loop delimiter for copying 100 registers
R(R2)=R(R1) ; Copy contents of a register in array1 to register in array2
R1=R1+1     ; Increment the Array 1 pointer to the next register
R2=R2+1          ; Increment the Array 2 pointer to the next register
L99              ; Repeat Loop until the contents of all 100 registers have been copied.
```

4.7 Extended Command Set (Types SMC23/24 only)

Rules for Register Operations:

1: An "is equal to" sign "=" is used to assign the contents of a register. A maximum of 3 registers may be used in an assignment expression:

e.g. R3=R23+T is legal, but R3=R23+T-100 is illegal.

2: The following 4 arithmetic operators can be used in register operations:

+	: Addition
-	: Subtraction
*	: Multiplication
/	: Division

3: All values and register contents must be integers in the range 0 to 65535. For division however, all operands must be max. 32767 or an error will result.

4: The result of division is always rounded down.
Example:

```
R2=289/10     ; Calculates the result of 289 divided by 10.
PRINT(5.R2)  ; The result is rounded down, the printed result is 28.
```

5: The R, S, T, L and D registers can be assigned values using an "equals" sign, but a quicker method, both in terms of programming and at run-time, is to assign values directly to these registers. If the required value is known and a constant, it can be directly assigned to the register.

Example:

```
T2000  is equivalent to T=2000
D200   is equivalent to D=200
S500   is equivalent to S=500
```

6: When "*if*" expressions are used for numeric comparison, the values to be compared must be maximum 32767. This is valid both for register contents and numeric values.

4.7 Extended Command Set (Types SMC23/24 only)

VR[0-510]
(Verify)

The Verify Register command is used to verify the contents of a register.
SMC23A / SMC23B / SMC24A / SMC24B contain extended memory and there are 510 user registers that can be verified.

Example:

VR9 ; Returns the contents of register R9.

I[5-7]
(Initialize)

The Initialize command is used to save or recall the contents of all user registers to or from the Controller's permanent EEPROM memory. The command is also used to reset (erase) all user register contents.
The Initialize command can be used both in Standby Mode and Programming Mode.

Example:

I5 ; Resets (erases the contents) of all User Registers.

I6 ; Stores the contents of all 510 User registers in EEPROM.
; The predefined registers (R,S,T etc,) are not stored in EEPROM

I7 ; Recalls the contents of all 510 User Registers from EEPROM.

4.7 Extended Command Set (Types SMC23/24 only)

con=n The Conversion command is used to set a conversion factor between the number of steps a motor moves and a unit of measurement such as length, volume, position, etc. "n" specifies the number of steps per unit length, volume (mm, ml, cm, dl, etc). The conversion factor can be specified as a real number in the range 0.0001 to 1600.0000, with up to 4 decimal points.

The conversion factor 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 steps.

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) steps. The motor will then move 1055 steps and the remainder (0.52 steps) will be stored. The remainder is used in the next motor operation to correct for the 0.52 dosage steps.

Example:

A system requires motor operation of 14.654 steps to dose a volume of 1 ml.

```
 con=14.654     ;  The conversion factor is set to 14.654 steps per millilitre.
R1=290          ;  A value of 290 is assigned to register R1.
 +(R1)          ;  The motor is moved to provide a dose of 290 ml.
                ;  The number of steps run is 290 * 14.654=4249. The step
                ;  remainder is 0.66
D100            ;  Delay of 1 second.
 +18            ;  Dose 18 ml. The number of steps is
                ;  18*14.654+step_remainder=264.
                ;  The new step remainder is 0.432
```

Note that after a "Home" operation using the H command, the step remainder is reset to 0 since the motor is set to its absolute reference point.

PRINTn1.n2.n3 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 DIS10, KDM10 and IOM10 Modules via the RS485 interface.

Command Format :

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 Controller 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 Controller.

Example 2:

PRINT255.0.R2

Prints the contents of register R2 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 *"Editor2"* program is used, once the Controller program has been transferred using the F5 function key, the F6 function key can be used to switch to the communication window where register contents will be displayed 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.

4.7 Extended Command Set (Types SMC23/24 only)

IF [p1 m p2]

The IF command is used for comparison of 2 numeric values, p1 and p2. These values may be the contents of registers such as R1, A1, T etc., or simply integer values such as 10500, 420, etc. All registers described in the "Register Description" section at the beginning of this Chapter can be used in IF expressions. The comparison operator "m" may be one of the following :

m	Condition fulfilled if:
<	Less than
>	Greater than
=	Equal to
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to

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.

Example 1:

```
:START  IF R10 < 9800     ; If the content of register R10 is less than
        J:PROG1           ; 9800, jump to label PROG1.
        J:PROG2       ; else jump to label PROG2.
```

Example 2:

```
        T=100         ; Set Top Rate to 100 steps/second
:START  T=T+50            ; Increase Top Rate by 50
        IF T>4000         ; If the Top Rate is greater than 4000 steps/s
        J:SPEEDOK     ; Jump to label SPEEDOK
        +1000             ; else move 1000 steps clockwise
        J:START           ; Jump to label START, where the speed is increased
```

The above program moves the motor 1000 steps at a speed of 150 steps/second and increases the Top Rate by 50 steps/second until a Top Rate of 4000 steps/second is reached.

INPUTn1.n2 The INPUT command is used to read-in data from external modules connected to the 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 registers whose contents can be read into the Controller'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.

Command Format :

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 1:

An IOM10 module has 16 inputs and 8 outputs are 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 IOM10 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_COUNTER
                  J:READINP         ; else read Counter value and continue
                                    ; program

:READ_COUNTER   R30=INPUT5.3      ; Read Counter and transfer to R30
                  R(R1)=R30         ; Transfer Counter value to an array
                                    ; register using R1 as array pointer.
```

4.7 Extended Command Set (Types SMC23/24 only)

AO[a].[o] (Activate) 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 IOM10 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 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. 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 IOM10 module and SMC23/24 are connected together in a system. The IOM10 module address is 10. Output 4 is to be activated. The following command is used:

```
AO10.4
```

CO[a].[o]
(Clear)

The Clear command is used to clear a flag in an external module. The number of flags which that can be cleared in different external modules varies, but each module has at least 1 flag. For the 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.

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

Example 1:

Controller Type SMC23 and a KDM10 module are connected in a system via the RS485 interface. The address of the SMC23 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:

Controller Type SMC23 and an IOM10 module are connected in a system via the RS485 interface. The IOM10 module's address is 5. The IOM10's output 7 is to be deactivated. The command is as follows:

```
CO5.7 ; Deactivate output 7 on IOM10 module with address 5.
```

5.1 Electrical Specifications

			Min.	Typical	Max.	Units
Power Supply (Types SMC 23/25):						
Supply voltage		15		45	V DC	
Power Consumption (unloaded, no motor)				4		W
Power Supply (Types SMC24/26):						
Supply Voltage (Nom. 230V)			207		242	VAC
Supply Voltage (Nom. 115V)			100		125	VAC
Power Consumption (unloaded, no motor)				10		W
User Supply :						
Output Voltage (adjustable)			5.1		30.2	V DC
Rated Current					500	mA DC
Motor Driver :						
Output Current (per phase)			0.2		3/(6)	A DC
Output Voltage			15		45	V DC
Chopper Frequency				22		kHz
Interface :						
Rx mark position			-1		-12	V
Rx space position			2.5		12	V
Tx mark position			-3		-12	V
Tx space position			5		12	V
Communication Rate			110		9600	Baud
Isolation Voltage					* 500	V (Max.)
Module Interface (SMC23/24):						
Communication Distance		0		500	m	
Communication Rate				50		kbit/sec
Isolation Voltage					* 500	V (Max.)
User Inputs 1 -3 & Stop Input:						
Input Impedance			10			kohm
Supply :	Voltage		5		32	V DC
	Current	at 5V	-		10	mA DC
		at 12V	-		20	-
		at 30V	-		45	-
Logic "0"		at 5V	<1.7		-	VDC
		at 12V	<3.0		-	-
		at 30V	<6.7		-	-
Logic "1"		at 5V	-		>2.7	-
		at 12V	-		>6.5	-
		at 30V	-		>16.2	-

* Measured from Supply ground to Interface ground

() Values valid for Type SMCxxB.

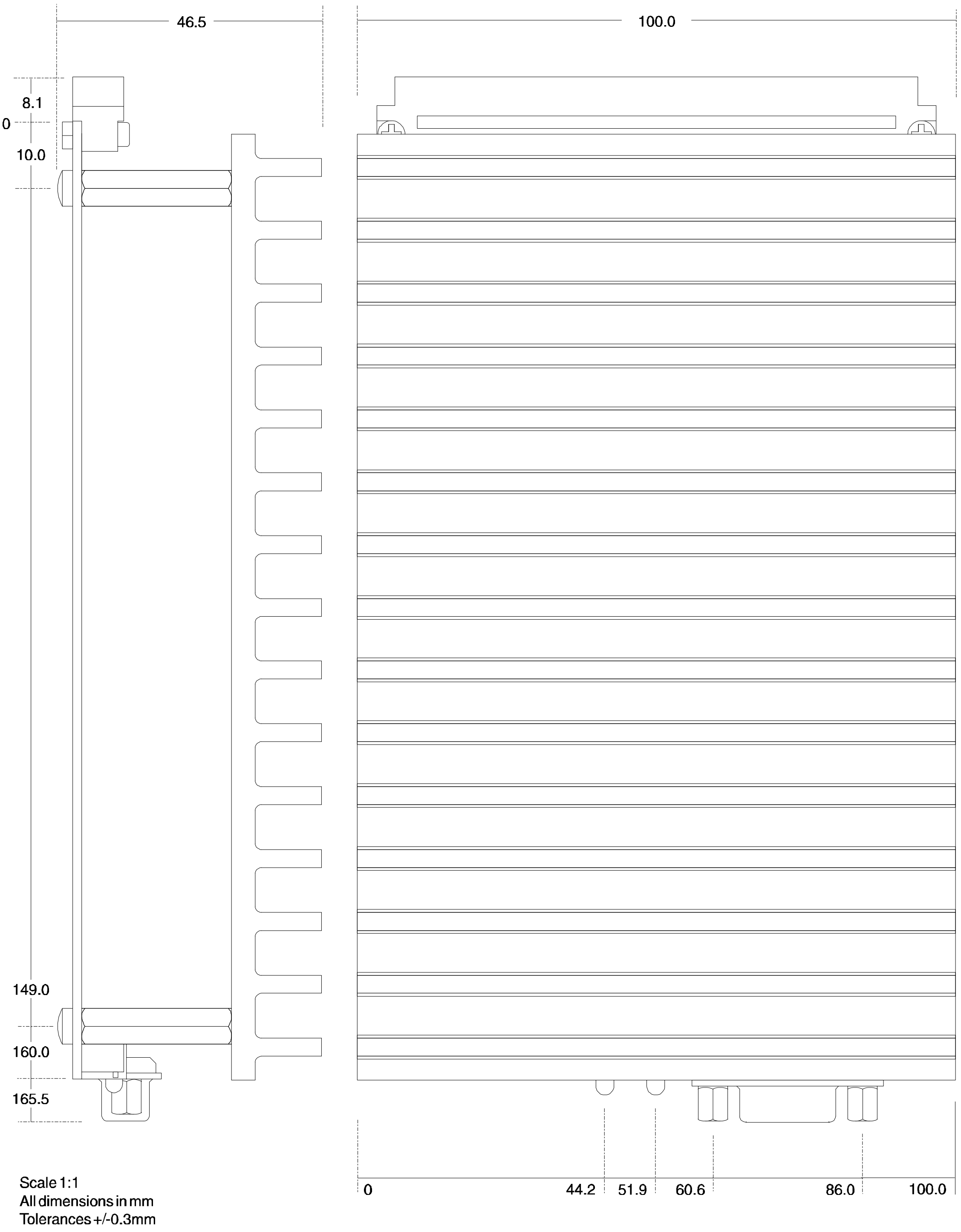
(Continued on following page)

5.1 Electrical Specifications

	Min.	Typical	Max.	Units
Analog Inputs :				
Resolution	-		8	Bit
Input Voltage (Max allowable)	-20		** 45	V DC
Input Voltage (Nominal)	0.00		5.10	V DC
Offset Error	-	±½	±1	LSB
Gain Error	-	±½	±1	LSB
Temperature Drift @ 0-50°C	-	±¼	±½	LSB
Logic "0"	<2.5		-	V DC
Logic "1"	-		>2.5	-
CW/CCW Inputs (SMC23/24):				
Input Impedance	2.4		3.3	kOhm
Logic "0" (active)	(-30)		3.5	V DC
Logic "1" (inactive) 3.0		30	V DC	
User Outputs :				
Supply Voltage	5		30	V DC
Rated Output Current per output				
1 output activated @25°C		700	mA DC	
2 outputs activated @25°C		460	mA DC	
3 outputs activated @25°C		300	mA DC	
Operating Temperature Range :	0		50	°C

** Absolute max. time < 1 sec.

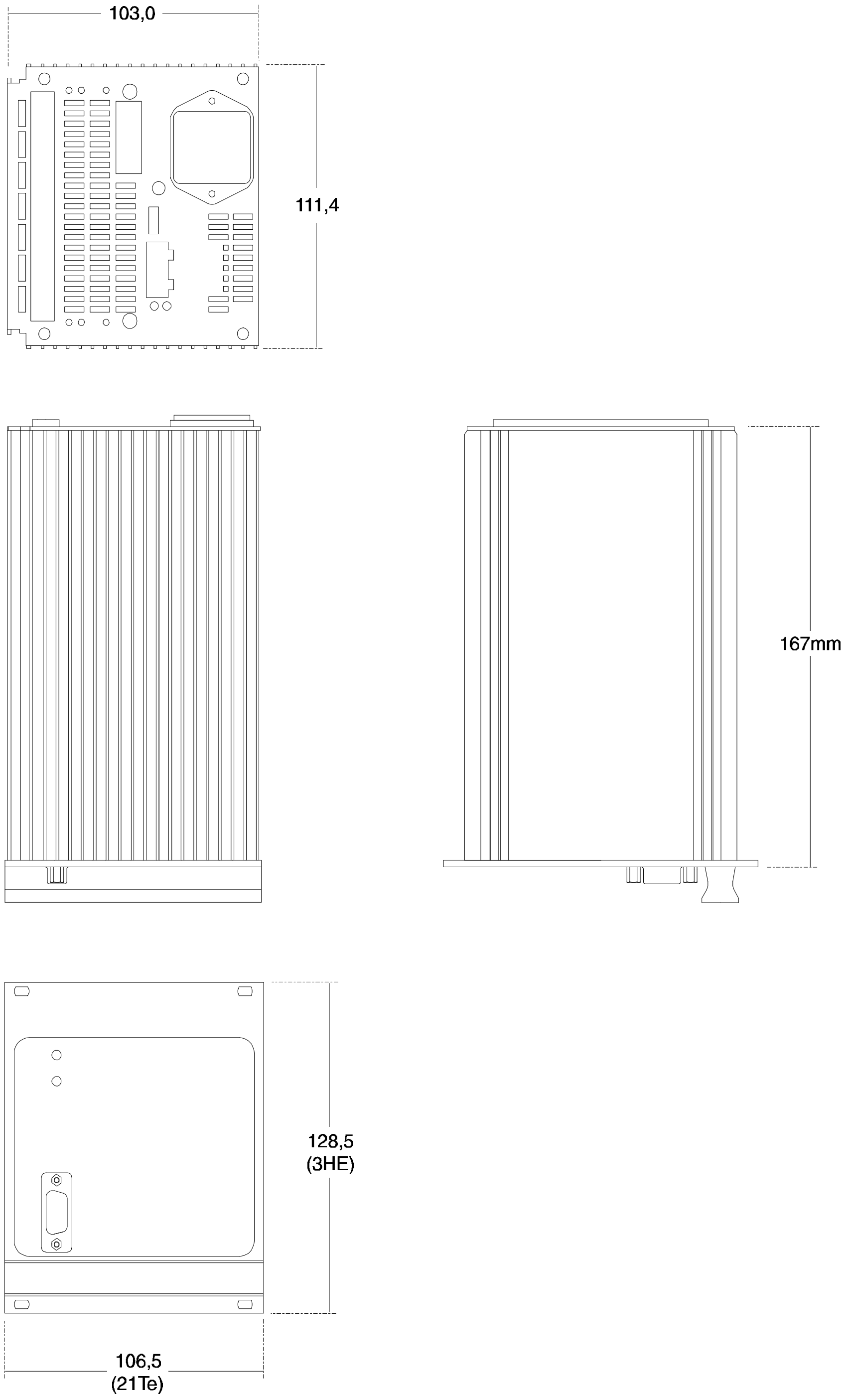
5.2 Physical Dimensions (Types SMC23 and SMC25)



Scale 1:1
All dimensions in mm
Tolerances +/-0.3mm

If the Controller is mounted in a closed cabinet, a ventilation fan or other form of cooling should be installed. The Controller is however protected against overheating by a built-in thermo-switch which disconnects the driver stages at a temperature of approximately 80 °C.

5.2 Physical Dimensions (Types SMC24 and SMC26)



If the Controller is mounted in a closed cabinet, a ventilation fan or other form of cooling should be installed. The Controller is however protected against overheating by a built-in thermo-switch which disconnects the driver stages at a temperature of approximately 80°C.

5.3 Memory Utilization

The permanent memory of the Controller consists of a E²PROM. The memory is 0.5 kbyte (512 bytes) in SMC25, SMC26, and 8kbyte in SMC23, SMC24.
4 bytes are used to correct the A/D converter offset-error and for adjustment of the Controller's thermometer function (TP).
In order to enable optimum utilization of the available memory, the following table gives the memory requirements of each program command. The total size of a program must not exceed 508 bytes in SMC25/26 and 7500 bytes in SMC23/24.
If an attempt to store a program greater than 508/7500 bytes is made, the Controller will issue an error message "E3".

1 byte :	g±
	H±
	RET
	(Blank line)
2 bytes :	A[n]
	C[n]
	I[n]
	r,s,t[n.n]
	U[n]
	W[n]
	L[nnn]
	J[nnn]
	JS[nnn]
3 bytes :	AO[n.n]
	CO[n.n]
	CR[nnnn]
	CS[nnnn]
	CT[nnnn]
	R[nnnnn]
	RS[nnnn]
	RT[nnnnn]
	S[nnnn]
	T[nnnn]
	N[nn.nn]
	D [nnnnn]
4 bytes :	f [±nnnnnnn]
	± [nnnnnnn]
	G [±nnnnnnn]
	JC[n.nnn]
	JCA[p].[n1]
5 bytes :	con=[nnnn.nnnn]
6 bytes :	DA[n].[n1-n2]
	±A[n].[n1-n2]
	G±A[n].[n1-n2]
7 bytes :	NA[p1-p2]
10 bytes :	PRINT(typ.)
	R[n]=x
12 bytes :	INPUT[n.n.n]
14 bytes :	R[n]=x+x
17 bytes :	IF(max.)

5.4 Connectorboard for the controller

Connections
Connector Board for SMC23, SMC24, SMC25 and SMC26.
Type: "CON10P"

Selection of Input Type
Each of the 4 User Inputs IN1, IN2, IN3 and ST, has a jumper.
If a given input receives a signal from a PNP (Source) output, the associated jumper is set to "0".
If a given input receives a signal from an NPN (Sink) output, the associated jumper is set to position "1".
Note that Input "ST" (Stop Input) is activated if a signal which is not logic "1" is applied to the input. If the Input is not used, the "ST" jumper must therefore be set to position "1".

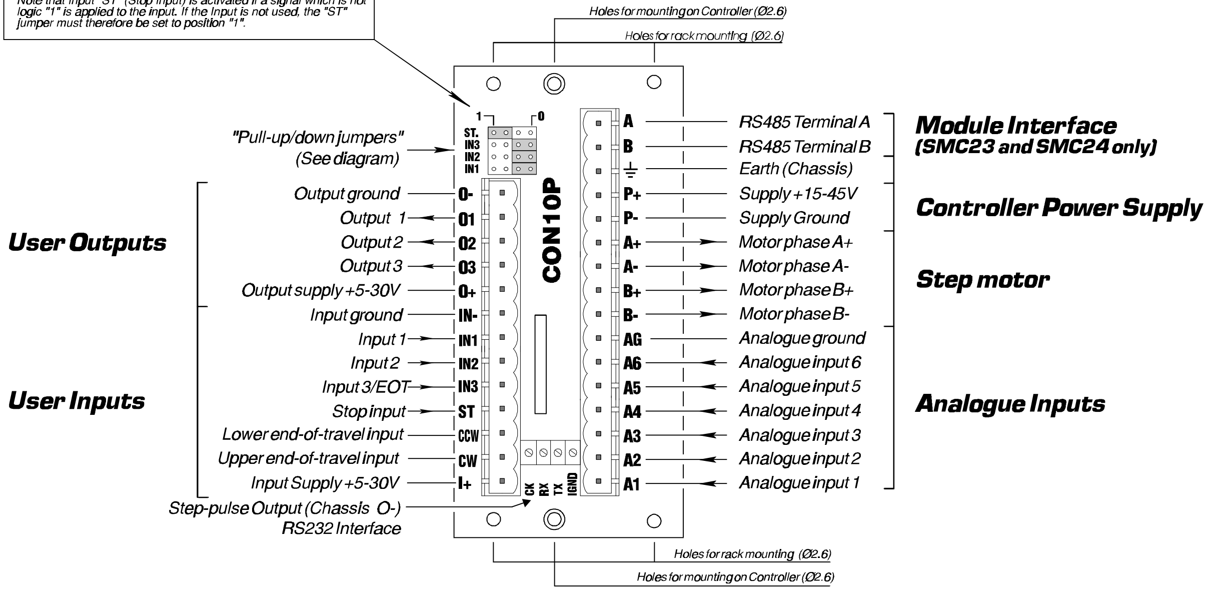
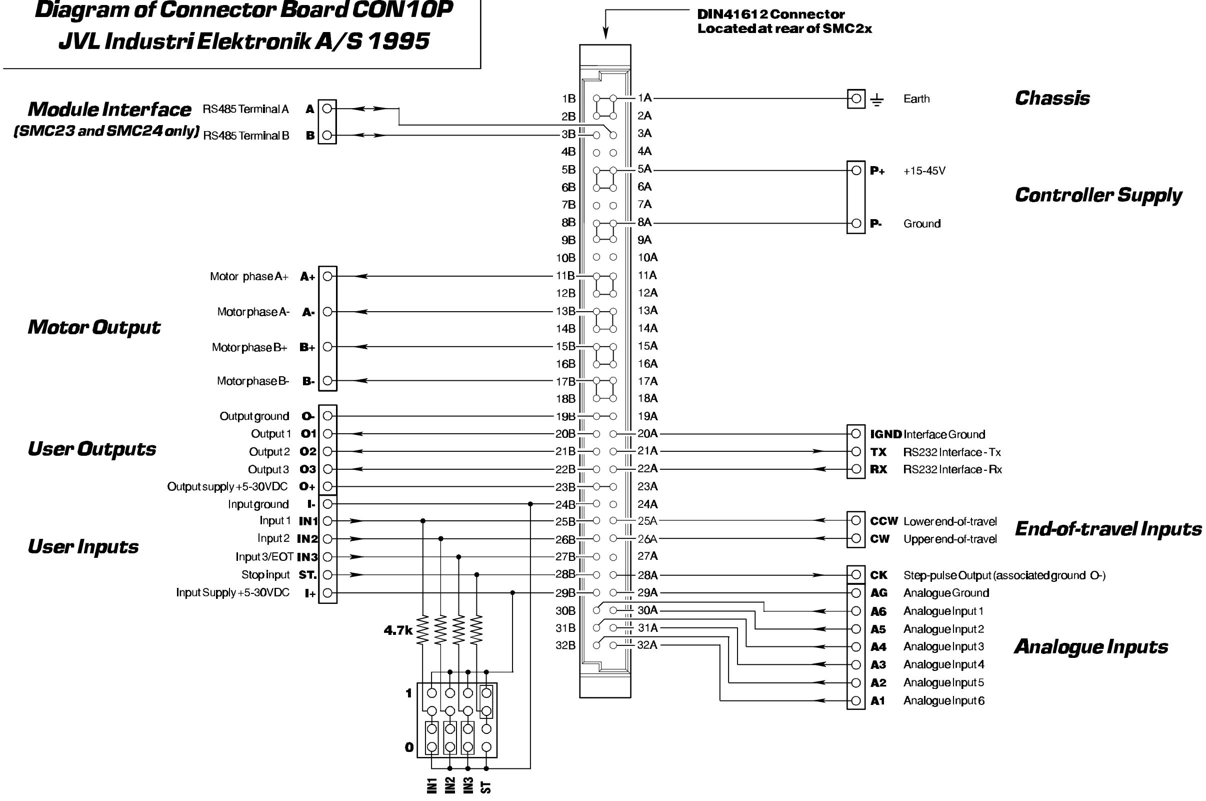
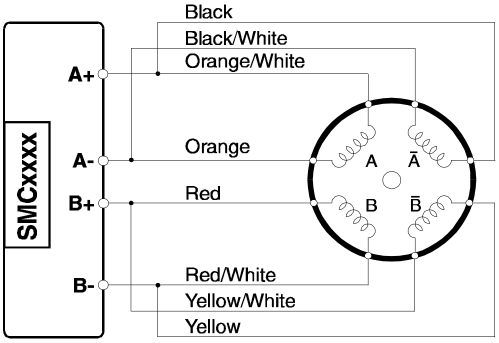


Diagram of Connector Board CON10P
JVL Industri Elektronik A/S 1995

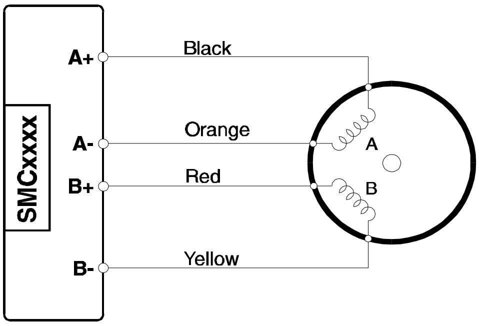


5.5 Motor Connections

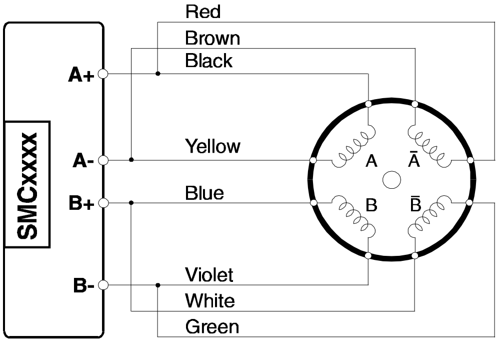
Connection of MAE motor Type HY200-xxxx-xxx-x8



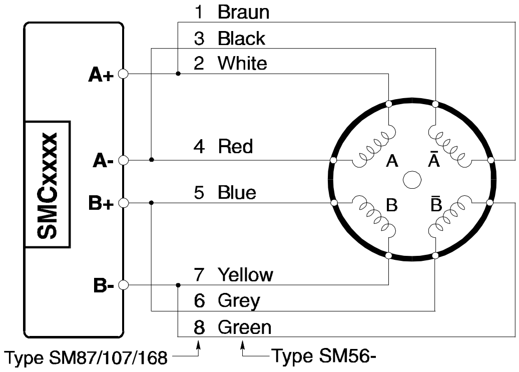
Connection of MAE motor Type HY200-xxxx-xxx-x4



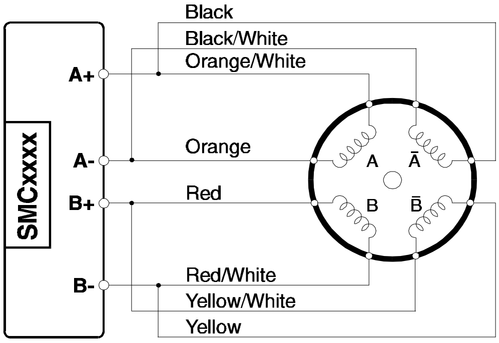
Connection of Phytron motor Type ZSx . xxx.x,x



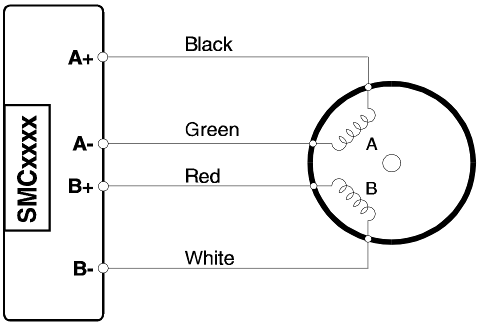
Connection of Zebotronics motor Type: SMxx.x.xx.x



Connection of Vexta motor Type: PH2xx-xxx



Connection of Teco motor Type: 4Hxxxx

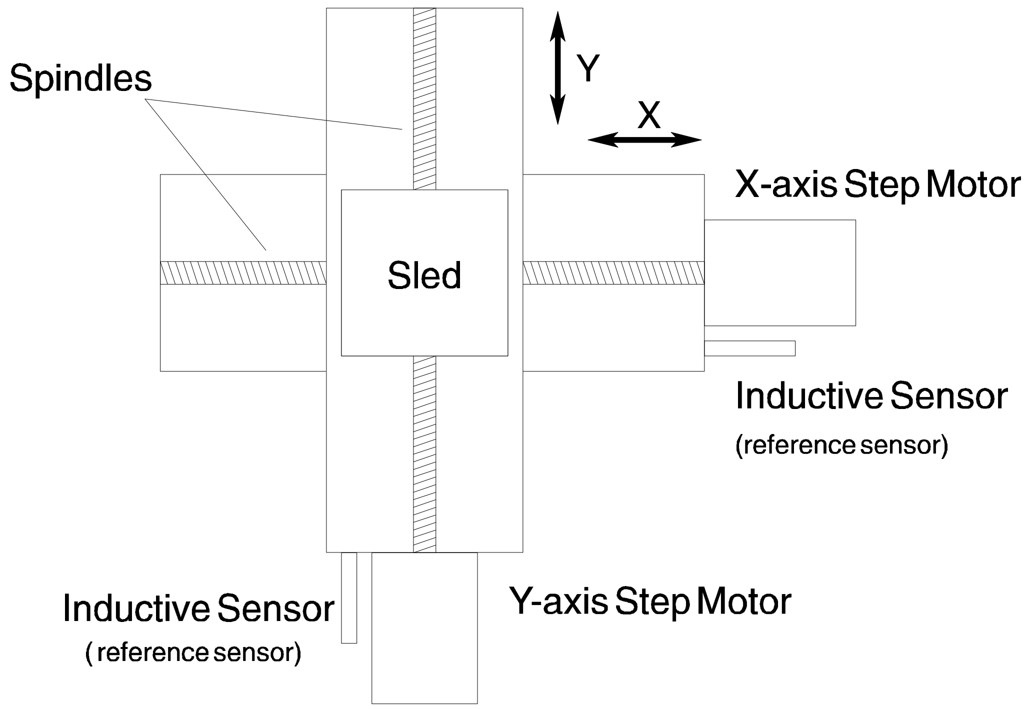


5.6 Application Example for Controller Type SMC25

The following example illustrates how 2 Controllers can be used for motion control of an X-Y Table. The Table is used to drill 2 holes in an aluminium block. Two stepper motors are mounted. In addition, an inductive sensor is used for each axis to register when the respective axis has reached its mechanical reference position (0 position). The feeler gauge distance should be as small as possible (typically 1-2mm) since it greatly determines the repetition accuracy.

This is important because the inductive sensors determine the reference points for the X and Y axes and thus the accuracy of the entire system. A sensor with a gauge distance of 1mm will typically result in a repetition accuracy of ±1/100mm, which should be sufficient for the majority of applications. The drill itself is not described here, but could for example consist of a DC motor vertically driven by a hydraulic/pneumatic cylinder.

Plan View of X-Y Table:



5.6 Application Example for Controller Type SMC25

Electrical Connections.

Before the actual task of programming the Controllers is started, it is recommended that the input and output signals of the system are clearly defined.

A Start button is connected to the Y-axis controller to start the system.

The following elements must be handled by the system:

1) 2 Stepper motors.

2) 2 Inductive Sensors with NPN outputs (0V at the output when activated).

3) A drill which is activated by a voltage pulse of 0.1 second duration. The drill outputs 12V when the holes have been drilled and the drill unit has returned to its start position.

4) A Start button.

5) Internal synchronisation signals between the 2 Controllers. (See program description.)

The following page illustrates the electrical connection of the system.

The Controllers' User Outputs and Inputs are used in the following manner:

X-Controller.

Input 1

Used to receive "ready" signal from the Y-controller.

Input 2

Used to receive "ready" signal from the drill.

Input 3

Inductive Sensor (X-axis)

Output 1

Used to send "ready" signal to the Y-controller.

Output 2

Used to send start impulse to the drill.

Output 3

Not used.

Y-Controller.

Input 1

Used to receive "ready" signal from the X-controller.

Input 2

Used for the Start button.

Input 3

Inductive Sensor (Y-axis).

Output 1

Used to send "ready" signal to the X-controller.

Output 2

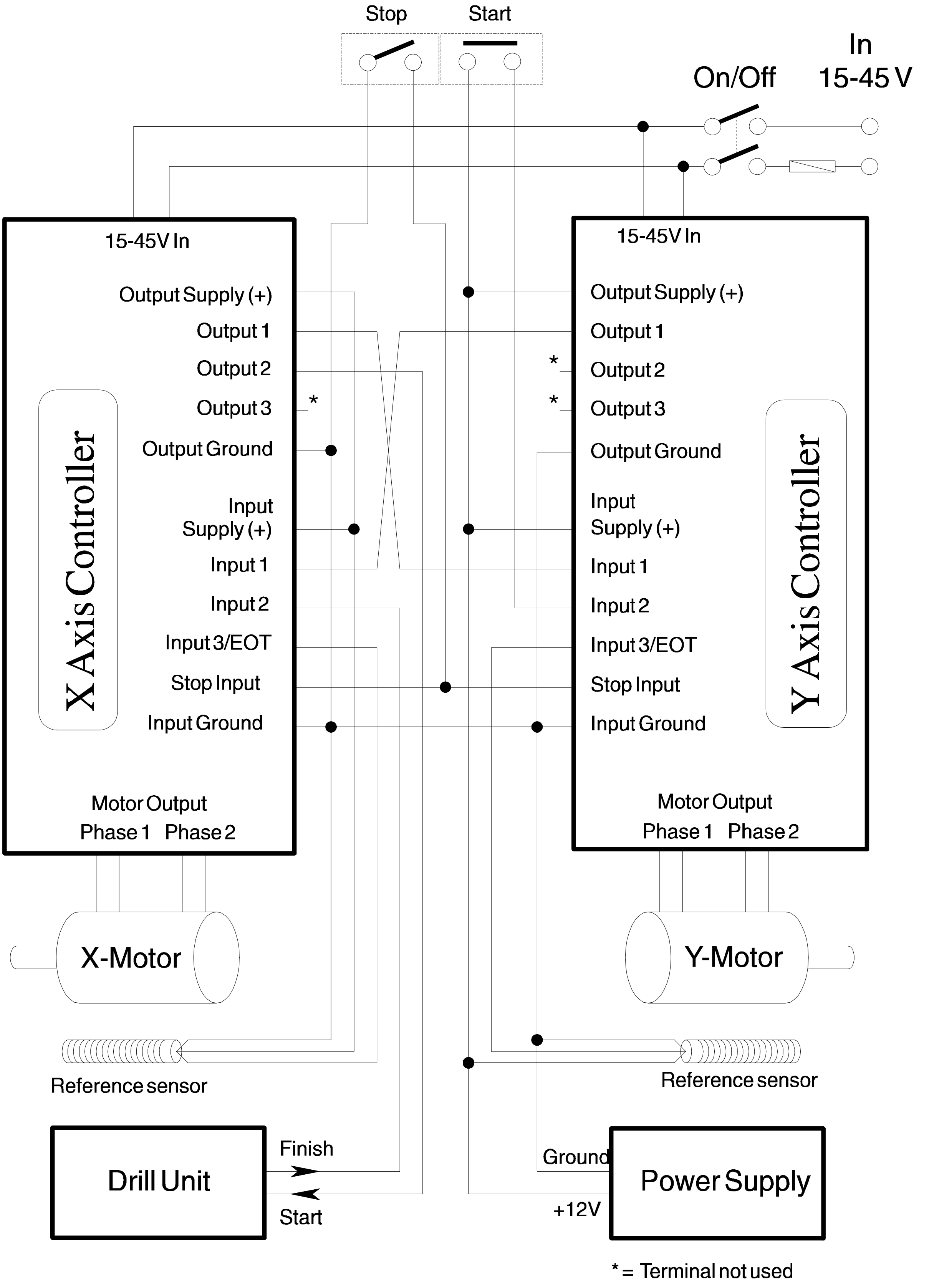
Not used.

Output 3

Not used.

5.6 Application Example for Controller Type SMC25

See Chapter 2 for the locations of individual connectors.



5.6 Application Example for Controller Type SMC25

Program for Control of the Drilling Application.
Before programming is started, the values of all parameters should be defined.

Holes are to be drilled in the aluminium block at the following coordinates:

Hole 1 - 4013,7387
Hole 2 - 5164,1949

The coordinates are specified in steps relative to the reference point of the inductive sensors.

The drill unit is activated by a voltage impulse of 0.1s duration. Once the holes have been drilled, the drill unit moves to its start position and outputs a constant voltage of 12V to indicate that the drilling operation is complete.
The sequence of instructions to be programmed for the application can be described as follows:

1. If the start button is activated, continue to point 2.
2. Go to position 4013,7387
3. Drill hole
4. Go to position 5164,1949
5. Drill hole
6. End, Go to point 1

If this procedure is followed, the system will wait until the start button is activated, drill 2 holes and wait until the start button is activated again.

Before the instructions for an application are programmed, it is recommended that a flow chart is made to give a better overview of the programming task. The remaining work is then largely determining and using the appropriate commands.

A flow chart for the drilling application is given on the following page.

5.6 Application Example for Controller Type SMC25

Program Flow Chart for Control of X-Y Table.

X-Controller.	**Y-Controller.**
Start:	Start:
1) Continue when "ready" signal is received from Y-controller.	**1)** If the Start button is activated, continue to point 2.
2) -	**2)** Send "ready" signal to the X-controller.
3) Move to position 4013.	**3)** Move to position 7387.

The X-axis motor has to move the shortest distance. The X-controller must therefore wait until the Y-controller is finished.

4) Continue when "ready" signal is received from Y-controller.	**4)** Send "ready" signal to the X-controller.
5) Send a start signal to the drill unit.	**5)** Continue when "ready" signal is received from the X-controller.
6) Continue when the drill unit is finished.	-
7) Send "ready" signal to the Y-controller.	-
8) Move to position 5164.	**8)** Move to position 1949.

The Y-axis motor has to move the shortest distance. The X-controller must both move to the specified position and start the drilling operation. The Y-controller must therefore wait until the X-controller is finished.

9) Send a start signal to the drill unit.	**9)** Continue when "ready" signal is received from the X-controller.
10) Continue when the drill unit is finished.	-
11) Send "ready" signal to the Y-controller.	**11)** -
12) Go to point 1.	**12)** Go to point 1

Note:
A dash in the above indicates that the respective Controller is waiting for the other Controller to complete an operation and send a "ready" signal.

5.6 Application Example for Controller Type SMC25

Program for Control of X-Y Table.

The basis for the actual program has thus been laid in the above overview and flow chart. The actual Controller instructions can then be programmed as shown below. Note that the numbers to the left of the columns below refer to the respective step in the above flowchart. Note that not all steps in the flowchart can be directly translated to a single program command.

For example, each time the X-controller sends a "ready" signal to the Y-controller, or vice-versa, 3 program commands are required.

Each time a ready signal is transmitted between the two Controllers, this is accomplished by sending a voltage impulse. The impulse duration is set to 1 second, since the receiver must be able to register the signal. The impulse is sent by the sender activating its output. A delay of 1 second is then made, after which the sender deactivates its output. This operation involves the use of 3 program commands: *A1 - D10 - C1*.

X-Controller.			**Y-Controller.**		
1)	H-	(Reset)	**1)**	H-	(Reset)
2)	W1	(Wait for Input 1)	**2)**	W2	(Wait for Input 2)
-			**3)**	A1	(Activate Output 1)
-			-	D10	(Wait 1 second)
-			-	C1	(Deactivate Output 1)
4)	G+4013	(Move to position +4013)	**4)**	G+7387	(Move to position +7387)
5)	A1	(Activate Output 1)	**5)**	A1	(Activate Output 1)
-			-	D10	(Wait 1 second)
-			-	C1	(Deactivate Output 1)
6)	A2	(Activate Output 2)	**6)**	W1	(Wait for Input 1)
-	D1	(Wait 0.1 seconds)	-		
-	C2	(Deactivate Output 2)	-		
7)	W2	(Wait for Input 2)	-		
8)	A1	(Activate Output 1)	-		
-	D10	(Wait 1 second)	-		
-	C1	(Deactivate Output 1)	-		
9)	G+5164	(Move to position +5164)	-		
10)	A2	(Activate Output 2)	**9)**	G+1949	(Move to position +1949)
-	D1	(Wait 0.1 seconds)	**10)**	W1	(Wait for Input 1)
-	C2	(Deactivate Output 2)	-		
11)	W2	(Wait for Input 2)	-		
12)	A1	(Activate Output 1)	-		
-	D10	(Wait 1 second)	-		
-	C1	(Deactivate Output 1)	-		
13)	J1	(Jump to point 2)	**13)**	J1	(Jump to point 2)

5.7 Index