stepper motor control

part 3: construction and software

Having built the PCB and secured it to the front panel, you should be ready to take the stepper motor control into use. This is best done in step-by-step fashion (pun intended), and we will guide you from adjusting the step-down voltage regulator right up to operating the PC terminal program. In the unfortunate case of the project not working as expected, help is available in the section 'Faultfinding'.

The step-down voltage converter is taken into use before fitting the integrated circuits and before securing the SMC board to 80C166 board. When an input supply voltage of about 10 V is applied to the converter, it should supply an output voltage of between 4.6 V and 5.4 V. Preset P5 is used to accurately set a level of 5.0-5.1 V. The converter's output voltage should remain stable when the input voltage is raised to 40 V. Two green LEDs, D11 and D12, light to indicate the presence of the supply voltage.

Once the step-down converter is know to function properly, the PCB is plugged on to 80C166 board fitted with the SMC firmware. Circuits IC13 and IC14 (74xx123) have to be present for indicating the clock signals and, later, for the stand-by mode. Additionally, a PC may be connected up to the serial interface. This PC should run a communication program set to ASCII transfer, 9600 baud, 8 databits, 1 stopbit, and no parity (9600,8,n,1).

After you switch on the supply voltage, the 80C166 board should produce its version and initialisation texts on the PC display. When the final *OK* appears, the stepper motor control software is ready for use.

Next, you have to verify that the SMC control software responds to pushbutton action, and supplies all clock and direction signals. On alternately pressing the *Left* and *Right* pushbuttons, the motor direction LEDs should respond accordingly by lighting or going out. The Tick (clock pulse) LED should light for the duration of the motor clock pulse.

Once this works as it should, the supply voltage may be removed and the GALs and power driver ICs installed in their sockets. Next, the stepper motors are connected up, one at a time, and each to its own power driver. Install the jumpers for the stepping order, current and standby mode, and then perform the analogue current adjustment with the potentiometers. Having done this the stepper motors should turn in both directions when the relevant control pushbuttons are pressed. If the motors respond properly to pushbutton control it should also be possible to control them using the PC.

If stepper motors are used with active zero-search, the motor direction has to be checked and modified if necessary. To change the direction, simply swap two wires of a phase winding.

Next, you may enable and test the sensor inputs by fitting the optocouplers. Whenever a voltage is applied to a sensor input (IN5-IN10, IC17 and IC18), the PC display should produce the associated report *s1* through *s6*. The same effect may also be achieved by short-circuiting the optocoupler outputs.

The state of 'Zero' (i.e., motor-home) inputs 1 through 4 (IC16) may be read in the same way as the sensor inputs. However, before this can be done, a zero (home) search command should be issued to the relevant motor. The PC display will then indicate the associated reports *n1* through *n4*.

FAULTFINDING

In all cases, check your soldering work, as this is the common cause of malfunctions. The same goes basically for the components mounted on the board. Further error sources include:

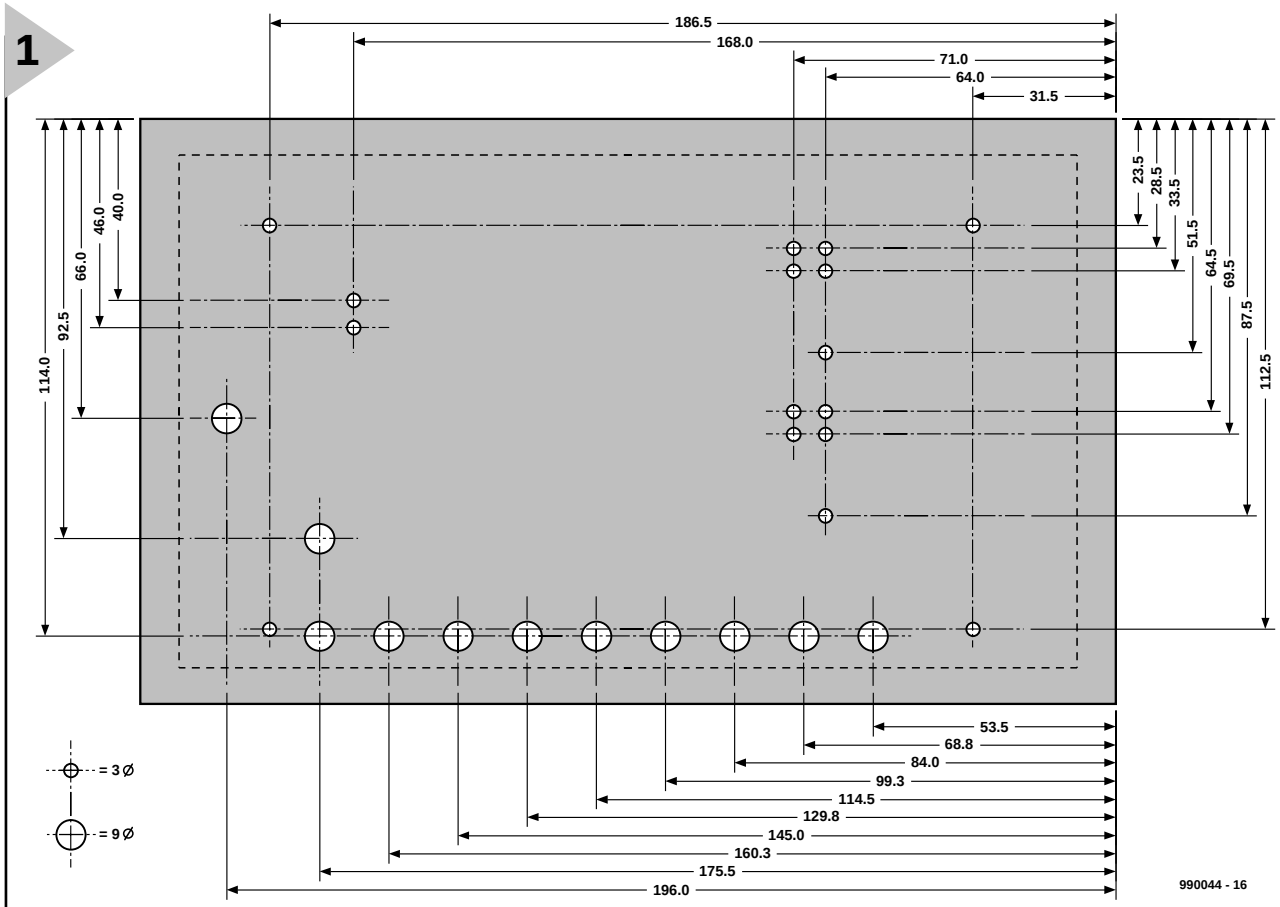
Motor does not operate at all

- Step-down voltage regulator does not supply 5 V.
- 80C166 controller board not connected.
- No SMC program in EPROMs, or H and L EPROM interchanged.
- Clock signal not available; lost between SMC board and 80C166 board (boxheader pin bent). Power driver ICs (IC1-IC8) missing.
- GAL (IC9-IC11) missing, not programmed or incorrectly programmed.

High-pitch sound heard, jerky movement of motor spindle, or no movement at all.

- Clock frequency or lower frequency too high.
- Too low current selected on jumpers JP9-JP20.
- With analogue current control, potentiometers P1-P4 set to wrong value, or jumpers JP9-JP20 not set

Design by K.-.H. Domnick



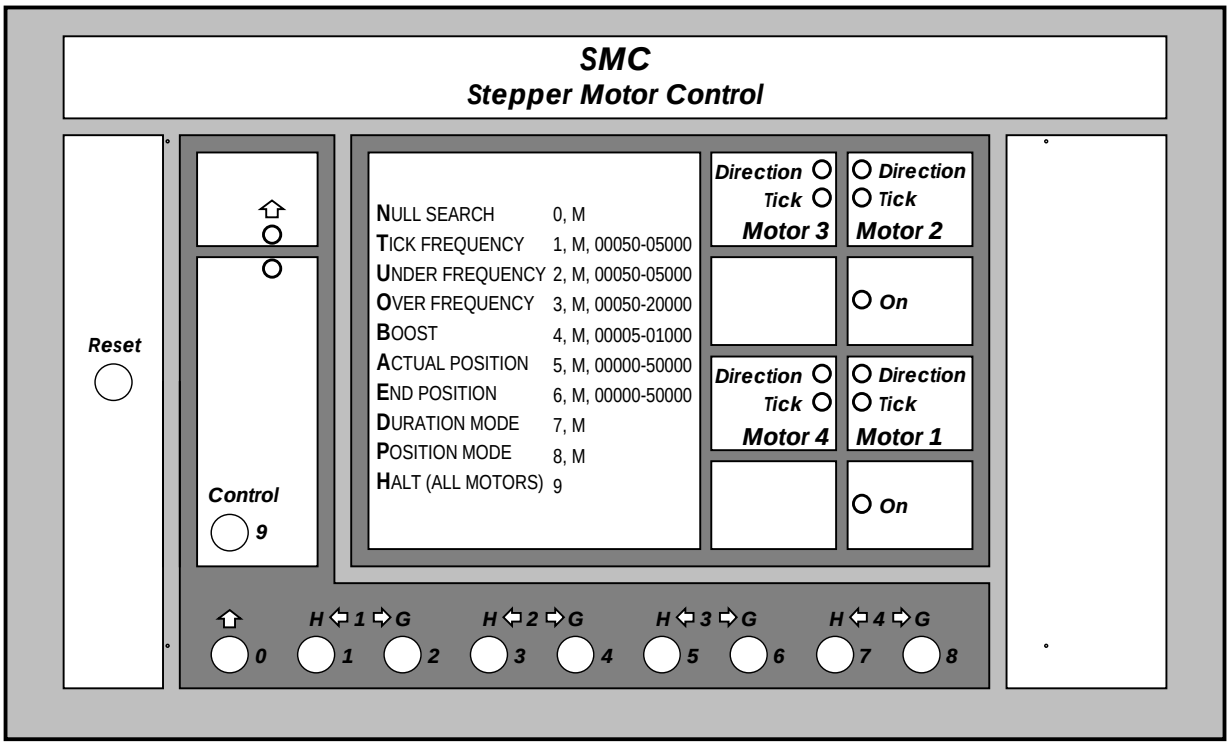


Figure 1. Front panel drilling details and suggested lettering.

for full current.

- Too low supply voltage for stepper motor(s).

Motor runs briefly, then stops:

- Missing motor direction signal; missing or broken link with 80C166 board (boxheader pin bent).

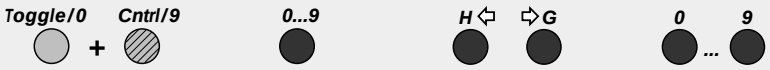
Motor fails to find spindle home position:

- Wrong direction on stepper motor.
- Wrong polarity on home switch or sensor (normally-closed instead of normally-open contact).
- Missing optocoupler IC16.

Unable to communicate with PC:

- Wrong COM port selected on PC.
- Serial wires RxD and TxD interchanged.
- Jumpers for CTS and RTS missing, or wrong configuration.
- Missing SIO component or associ-

Table 1. Programming Mode



Function	Pushbutton	Motor	Value = # sequence	Serial
Tick Frequency	1	M 1 - 4	00.050 - 05.000	T 50 - 5.000
Under Frequency	2	M 1 - 4	00.050 - 05.000	U 50 - 5.000
Over Frequency	3	M 1 - 4	00.050 - 20.000	O 50 - 20.000
Boost	4	M 1 - 4	00.005 - 01.000	B 5 - 1.000
Actual Position	5	M 1 - 4	00.000 - 50.000	A -2.147.483,648 - +2.147.483.648
End Position	6	M 1 - 4	00.000 - 50.000	E -2.147.483.648 - +2.147.483.648
Duration Mode	7	M 1 - 4	-	D 1 - 4
Position-Mode	8	M 1 - 4	-	P 1 - 4
Halt (all motors)	9	-	-	H 9
Null search	0	M 1 - 4	-	N 1 – 4

Table 1. The pushbuttons on the SMC front panel have different functions in Programming Mode. Values to be entered into the SMC should include leading zeroes to ensure uniform length.

ated tantalum capacitors.

Nothing happens…
Further problems may be caused by construction errors on the SMC board or a missing, incorrectly specified or even defective part. Check your work thoroughly, or ask a 'second opinion'.

OPERATION

The ten pushbuttons on the SMC front panel allow direct and simple control of the stepper motors, as well as parameter entry. Complex control sequences are not possible using the pushbuttons — inevitably a personal computer is then called for.

Normal operation
SMC makes a distinction between Normal Operation and Programming Mode.

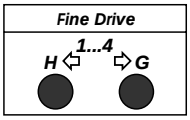


Figure 2. Normal operation without Toggle pushbutton pressed.

When pushbuttons 1…4 **Left** or 1…4 **Right** are pressed, the motor will turn in the selected direction at the selected clock frequency, until the relevant pushbutton is released.

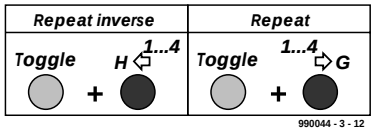


Figure 3. Normal operation with Toggle pushbutton pressed.

When the pushbutton 1…4 **Left** is pressed, the motor will turn in the opposite direction by the previous number of steps (equals command '*X — Repeat Inverse*). When the pushbutton 1…4 **Right** is pressed, the motor turns in the previously selected direction by the previous number of steps (equals command *Repeat*). The effect of Normal Operation with the Toggle switch pressed or not be interchanged with the aid of DIP switch number 2.

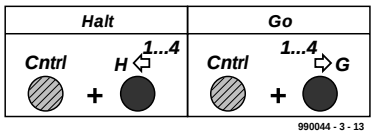


Figure 4. Normal operation with Cntrl pushbutton pressed.

Pushbutton 1…4 **Left** allows the stepper motor to be halted (equals command *Halt*). Pushbutton 1…4 **Right** starts the motor (equals command *Go*). Note however that a motor will only start if the values representing the current and end positions are different.

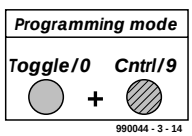


Figure 5. Enable/disable Programming Mode.

Programming Mode
To change from Normal Operation to Programming Mode, keep the Toggle pushbutton down while also pressing the Cntrl (Control) pushbutton. The yellow LED will start to flash.

To disable or terminate Programming Mode, repeat the above sequence. The yellow LED will go out.

Programming Mode is automatically ended after a complete or a wrong entry. With number entry, the Toggle pushbutton represents the value 0, and the Cntrl pushbutton the value 9. In Programming Mode, the pushbuttons have different functions.

When entering values into the SMC, leading zeroes should be included to make sure the values consist of the same number of digits.

SOFTWARE — INTERNALLY

All clock and direction signals are generated in software on the 80C166 controller board. Each clock is recorded as a 32-bit number and may be called up or modified as *Actual* (current) *Position*. After a successful 'home' search of the motor spindle, the relevant value is made 0. When the command *Go* is issued the controller computes a ramp (gradient) table from the parameters *Under Frequency* (lower frequency), *Over Frequency* (upper frequency) and *Boost* (acceleration rate).

Using these values, clock pulses are produced until the high-speed phase is reached. If the *Duration* mode (continuous operation) is enabled, all subsequent clock pulses take on the value assigned to *Over Frequency*. When the *Halt* command is issued, the values that make up the deceleration ramp (i.e., a look-up table) are processed until the motor stops.

If the spindle positioning mode is enabled, the processor also looks at the table entries representing the difference between *Actual Position* and *End Position*, as well as the number of steps

Table 2. Commands and reports

?	Help	-	Request help
A	* Actual Position	±2.147.483.648	Set/request current position
B	* Boost	5 – 1.000	Set/request motor acceleration (clocks per ms)
C	Copy Data	-	Copy data from memory into EEPROM
D	Duration mode	0/1 – 4/9	Switch on continuous mode (Position mode off)
E	* End position	±2.147.483.648	Set/request end position
E	Report	1 – 4	Report "End Position reached"
F	Fine drive	0/1 – 4/9	Turn slowly using clock frequency in selected direction
G	Go	0/1 – 4/9	Start motor / turn to end position
H	Halt	0/1 – 4/9	Halt motor
I	Info	-	Request info (status with semicolon preceding)
J	With zeropoint search	0/1 – 4/9	Turn to home sensor and then clear current position to 0
K	Without zeropoint search	0/1 – 4/9	No home sensor, clear current position to 0 immediately
L	Left rotate	0/1 – 4/9	Turn left using clock frequency (slow)
M	Motor actual *	1 - 4	Set/request motor no. for subsequent commands
N	Null search	0 / 1 - 4 / 9	Turn to spindle home (zeropoint) using clock frequency
n	Report	1 - 4	Report "Zeropoint reached"
O	* Over Frequency	50 - 20.000 Hz	Set/request upper frequency (fast)
P	Position Mode	0 / 1 - 4 / 9	Position mode on (Duration mode off)
q	Report	0 - 9	Acknowledge (q0) and error reports (q1-q9)
R	Right rotate	0 / 1 - 4 / 9	Turn right using clock frequency (slow)
S	Status	- / 0 / 1 - 4 / 9	Request status: `M1,T500,U1000,O10000,B500,....`
s	Report	1 - 8	Report "Sensor reached"
T	* Tick frequency	50 - 5.000 Hz	Set/request clock frequency (slow)
U	* Under Frequency	50 - 5.000 Hz	Set/request lower frequency (Start/Stop)
V	Version.	-	Request program version number
v	Report	1 - 4	Report "Leaving zeropoint"
W	Repeat	0 / 1 - 4 / 9	Repeat previous no. of steps in previous direction
X	Repeat inverse	0 / 1 - 4 / 9	Repeat previous no. of steps in opposite direction
Z	* Int. position	±2.147.483.648	Set/request intermediate position
z	Report	1 - 4	Report "Int. position reached"
	Possible values:	0	= current motor (set beforehand using M1 - M4)
		1 – 4	= indicated motor
		9	= all motors

for the high-speed phase. If more steps are required for the two ramps than for the difference between *Actual Position* and *End Position*, the motor will be unable to reach the high-speed phase.

Because all required clocks and steps have been computed after the start, all parameters are open to modifications (with the exception of the continuous or positioning modes).

If you want to stop a running motor before the spindle reaches the end position, you have to use the *Halt* command. It is possible to issue a *Null Search* command immediately after *Go*. On reaching the end position the motor will then automatically turn to the home position.

Another interesting command is *Int. Position* (intermediate position). When its associated value equals that of *Actual Position*, the program transmits a report (*z1-z4*) via the serial interface. This parameter may be modified at any time and as often as you want. Reports are also transmitted on reaching the end (target) position (*e1-e4*) or zero (null) position (*n1-n4*), on leaving the zero (null) position (*v1-v4*) or on detecting an active sensor input (*s1-s8*).

The command *Null Search* (turn to spindle home position) only causes the motor to actually turn to the home position if a 'home' sensor is available and the parameter *With zeropoint search* has been enabled. With *No home sensor*, only the current position is reset to zero, and a report (*n1-n4*) is transmitted.

PC COMMUNICATION

The SMC and the PC communicate through as serial link employing the following parameters:
9600 bits/second, 8 databits, 1 stopbit, no parity (9600,8,n,1). Using a terminal emulation program or general-purpose communication software for the PC (Telix, ProComm, HyperTerminal) you should be able to control and interrogate the Stepper Motor Control in plain ASCII. Complex sequences to be performed by the stepper motor will require a program, however, that supplies the relevant commands and evaluates the various reports returned by the SMC.

SMC wants to receive all commands either as a character or as a complete word without numbers or special characters. The system is not case-sensitive:

```
M1 / m2 / M 3 /m 4 / Motor 2
```

A line may contain more than one command provided a comma or colon (:) is used as a delimiter:

Table 3. Error reports

Q0	OK
Q1	Wrong command
Q2	Wrong motor number
Q3	Wrong value
Q6	RAM checksum error
Q7	EEPROM checksum error
Q8	EEPROM write error

```
m1, t500 / M1, T500 /
M1:T500:M2:T800 / Motor 1,
Tickfrequency 500
```

After a semicolon (;) comment may be inserted up to the end of the line:

```
M1,T500 ; Clock frequency
500 Hz for Motor 2 <ret>
```

Status reports are transmitted in uppercase characters, while position reports appear in lowercase. Information and comment is always preceded by a semicolon (;). Each line ends with a Carriage Return and a Line Feed (CR-LF sequence).

All commands and reports are listed in **Table 2**.

(990044-2)