Bionic Robots

- Begleitheft
- Activity booklet
- Manuel d'accompagnement
- Begeleidend boekje
- Cuaderno adjunto
- Folheto

fischertechnik®

GB+USA

1. Bionic – Nature as a Model

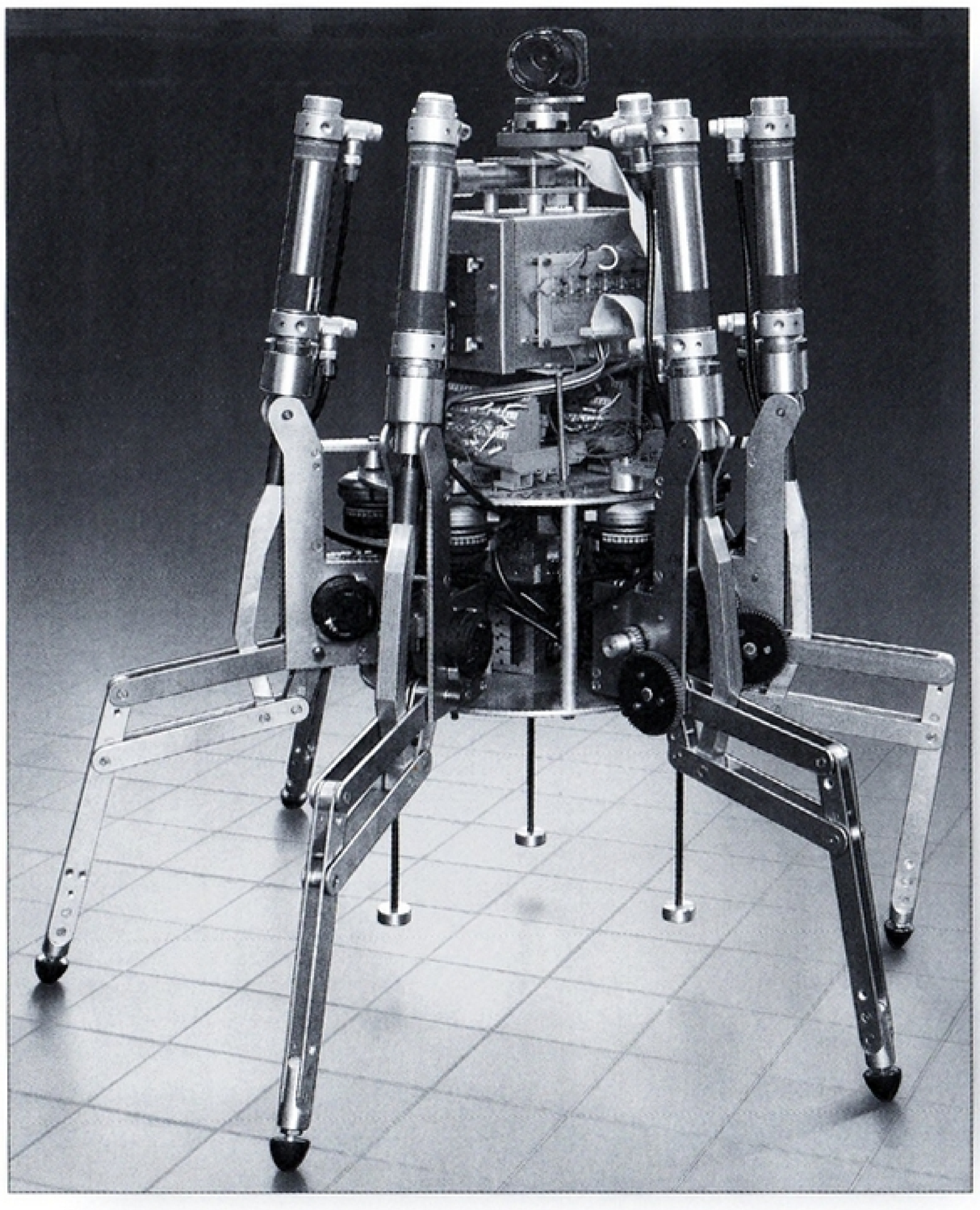
The term bionic is composed of the two terms biology and technique. This branch of science always tries to base its technical solutions on nature. Man has invented machines time and time again with the objective of moving farther, faster and more efficiently than possible naturally. This has been achieved in different ways corresponding to the respective requirements. Vehicles roll on wheels. In difficult terrains, which vehicles with wheels cannot handle, tracked vehicles or Caterpillars are used. Ships float on water or can operate underwater. In a few types of movement, nature serves as the model. For example, an airplane resembles a flying bird.

Over the past few years, scientists have been looking into another very widespread movement form: walking or running. Robots have been developed, which can move on legs. Such walking machines can be used in all places where wheel and tracked vehicles would hardly have a chance, for example, in very uneven or soft terrains, for climbing over obstacles, going up stairs, crossing ditches or for use in difficult to access and dangerous spots in nuclear power plants, mines or for rescue operations.

The first serious experiments in developing walking machines were conducted at a university in Tokyo in 1967. For the first time, work was oriented to the human way of walking instead of that of insects. The continual further development of these experiments results in the first two-legged walking machine in 1985. Today, these robots have more then 50 degrees of flexibility and numerous microprocessors. For example, they can read notes and play the organ with the help of a camera. You can even have a discussion with them.

An example of a six-legged walking robot is the electro-pneumatic walking robot Achilles, developed at the Royal Military Academy in Brussels. Equipped with a camera at the top and on its six legs, this robot reacts to raised or recessed obstacles (objects or holes).

Now fischertechnik has also devoted its attention to this exciting topic and has constructed robots, which are "wakened to life" with the Intelligent Interface and the LLWin software.



2. Requirements and Startup

You need the following articles in addition to the construction kit, so that you can build the models of the computing kit "Bionic Robots."

Intelligent Interface, Art. no. 30402
LLWin software (from Version 3.0), Art. no. 30407
Accu Set, Art. no. 34969

If you are not familiar with the LLWin software and the interface, you should first read through the LLWin software manual. It describes how you install the software and connect the interface. It is also very well suited for getting acquainted with how you can control fischertechnik models via a PC. Using a few components from the construction kit (motor and pushbuttons), you can first construct very simple model controls.

As soon as you are familiar with the software and the interface, you can then tackle the more challenging Bionic Robot models.

A CD-ROM is included in the construction kit, which contains LLWin example programs for the kit models. You need the LLWin software starting from Version 3.0 to open the programs. You can either leave the example programs on the CD and call them from LLWin with the command File - Open, or copy the complete BIONIC_ROBOTS folder from the CD in the project directory of LLWin onto your hard disk and open the examples from there.

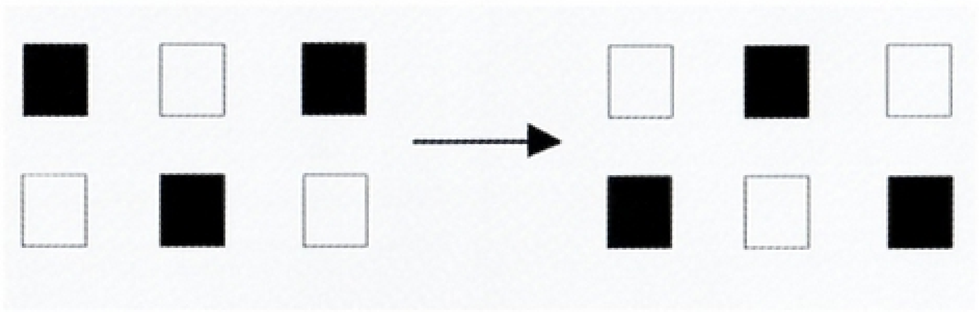
Before you construct the models, you need to assemble a few parts, e.g., cables and plugs. The construction instructions explain exactly what you have to do.

Now we're ready to start. You can now delve into the fascinating world of walking fischertechnik robots. As soon as you have constructed the first model and it starts to move in an almost eerie way, you will be thrilled by this technique, which has already been used in nature for moving for millions of years.

3. Walking on Six Legs

3.1 The Way Insects Walk

The way insects walk is excellently suited as a model for the drive of "mechanical six-legged" robots. In what is called a three-foot gait, three of the six legs are always raised at the same time from the ground: the front and back leg of one side together with the middle leg of the other side.

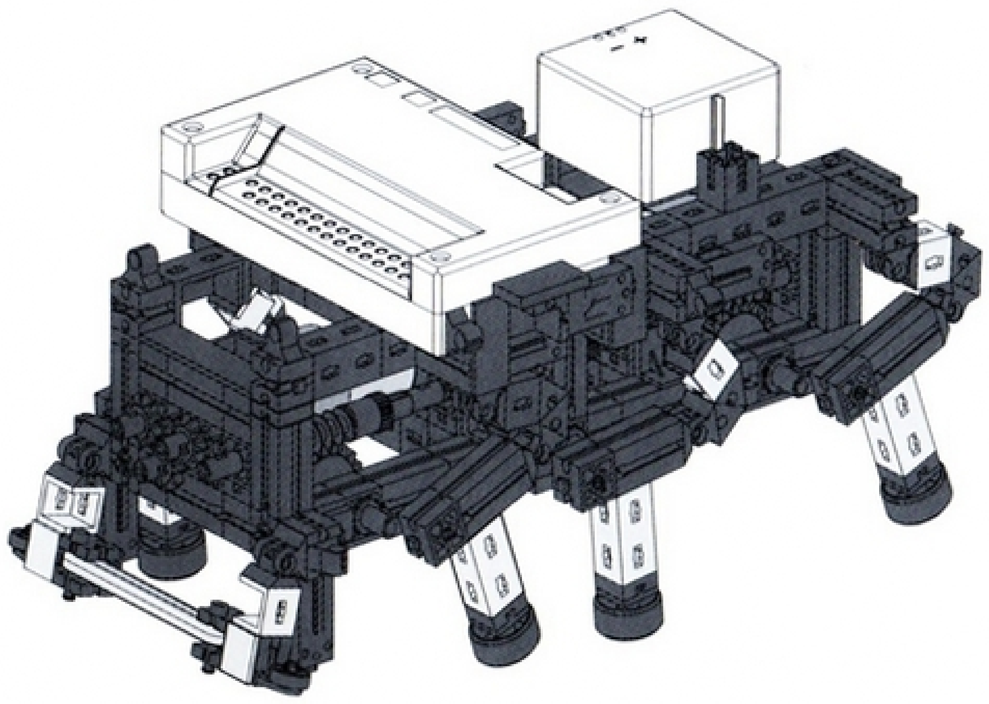


The legs, which remain on the ground (displayed in black), form a stable three-legged construct, so that the model always remains stable upright and does not fall over during walking.

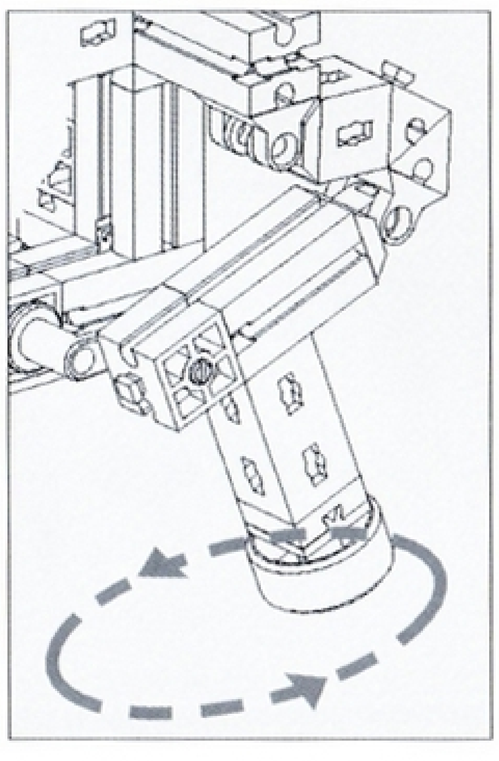
3.2 Model Mike

3.2.1 Assembly of the Model

Build the six-legged robot Mike now (see the assembly instructions on page 4). Load the power unit while you assemble it, so that you have sufficient energy to drive the model later.



The legs of the module are designed in such a way that a four-bar mechanism is produced. The construction type of the four-bar mechanism is called a crank and rocker mechanism. Driven by a crank, the movable elements of the gears sway back and forth. The distances between the individual joints and the position of the base (the lower end of the leg) are selected so that the base moves elliptically when the drive crank rotates. This creates movement, which is similar to a step when walking. The six cranks, which drive the legs, must be aligned precisely as shown in the assembly instructions. The three legs, which touch the ground at the same time, have the same crank setting. The cranks of the three legs, which are in the air at this time, are rotated by 180° for this. The correct setting of the cranks in relation to each other ensures that the model can walk in the correct step sequence, the three-foot gait.



The binding pieces and nuts, with which you attach the screws and toothed gears to the axles must be tight, so that they do not shift during the walking.

The right and left sides of the model are each driven by a motor (this is required for walking around curves). You must make certain for this that the middle leg on the one side is always in the same position as the front and back legs on the other side. This is synchronized by the software via the pushbuttons E1 and E2.

Use the interface diagnosis to test whether all pushbuttons and motors are connected correctly. Rotation direction of the motors: counter-clockwise rotation = forward.

3.2.2 The First Program

Now we are going to start to teach Mike something. The model should first walk straight. We will deal with walking around curves and reacting to obstacles later.

Task 1:

Program the model, so that it walks straight with a three-foot gait. Use the pushbuttons E1 and E2 for synchronizing the left and right legs. Make certain that the front and back legs on one side and the middle leg on the other side always have the same position. Also use pushbutton E8 as reset button.

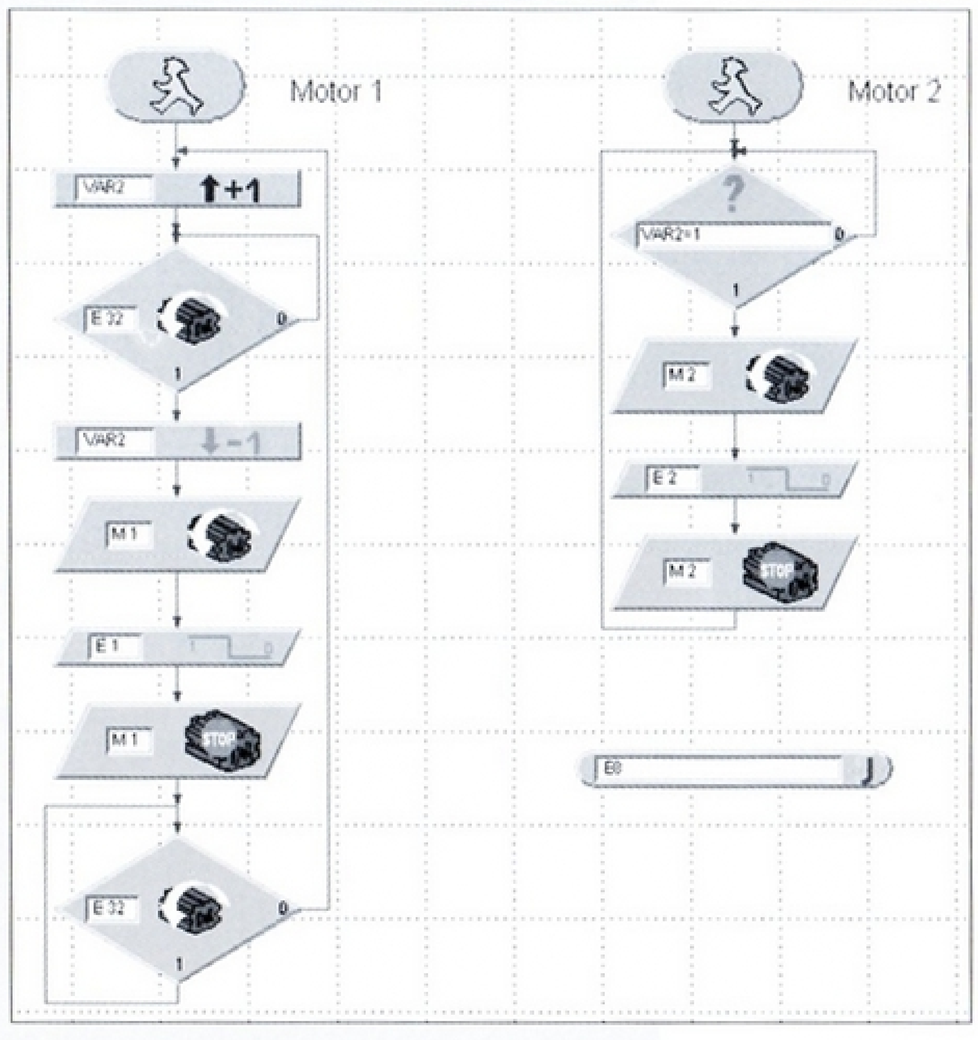
Tips:

Program a separate control process for each motor. Control the process for motor M2 using a variable VAR2.

If you do not need the interface during programming, interrupt the power supply between the power unit and the interface to save energy.

Solution:

The program for walking straight looks like this:



GB+USA

The variable VAR2 provides the impulse for starting motor M2. The motor M1 is started. As soon as pushbutton E1 is activated, M1 stops. As soon as E2 is activated, M2 stops. The first process waits until M2 has stopped (the status of motor M2 is queried via E32; also refer to the "Querying the Motor Status" in the LLWin manual).

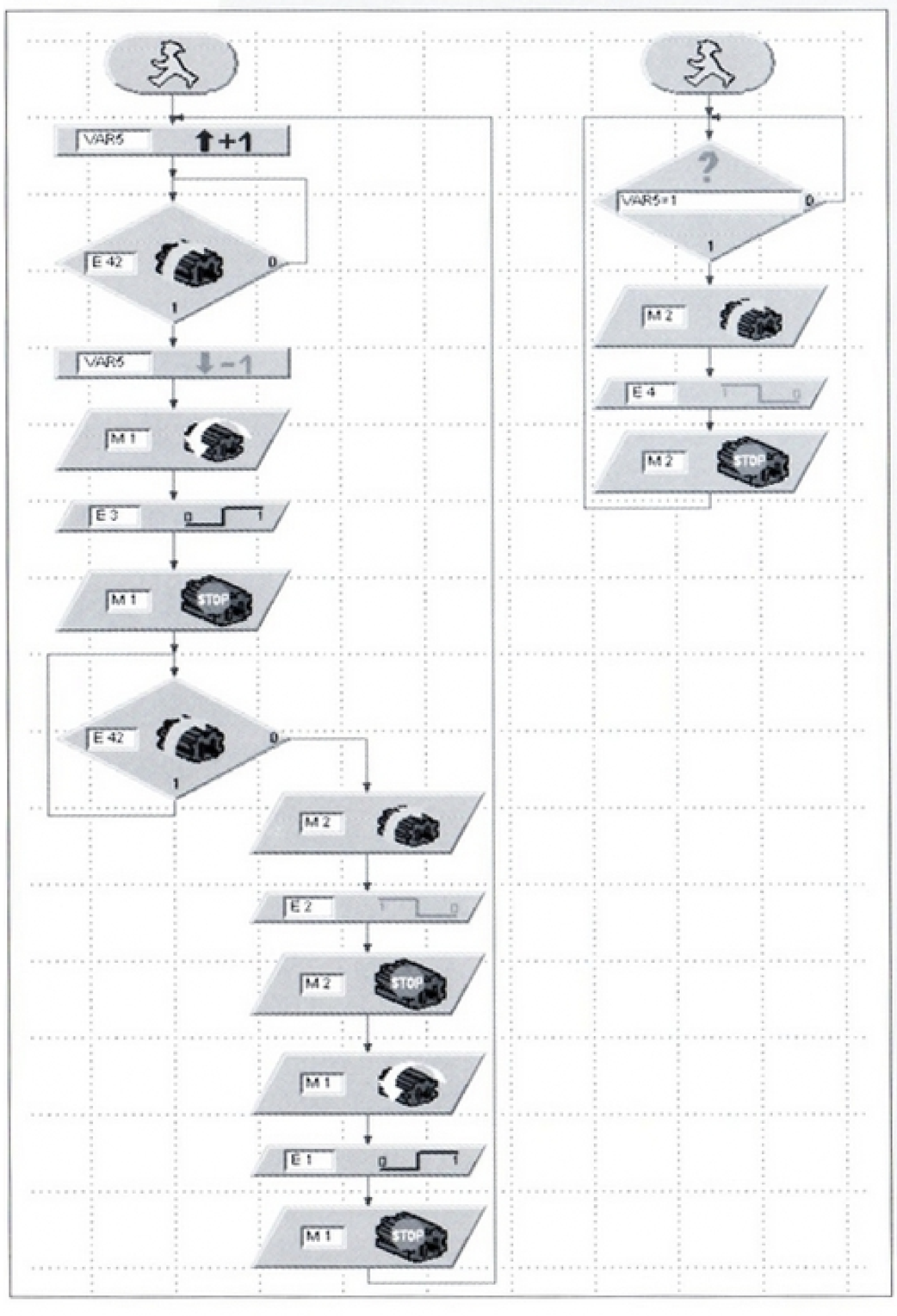
By the way, if you do not feel like creating this process yourself, you can find it as example project MIKE_STRAIGHT.MDL on the enclosed CD.
Start the project. If you have programmed everything correctly, the model comes to life and walks straight ahead. Congratulations! The first step has been taken.

3.2.3 Turning Left

Of course, we are not satisfied that Mike can only walk straight ahead. As the next step, we want him to turn around.

Task 2:
Program Mike, so that he turns to the left (counter-clockwise).

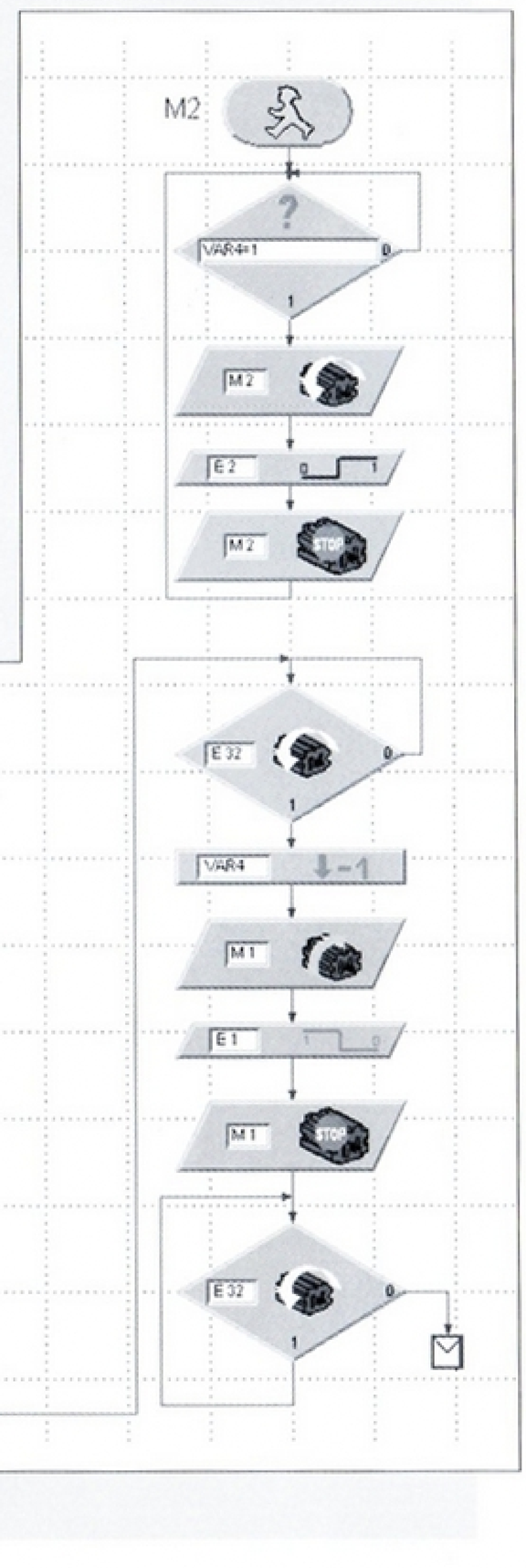
Tips:
The model turns to the left if M1 turns to the left and M2 to the right. Of course, you can also drive the model unsynchronized.
It also turns then, but then there are spots where the model can fall over forward. This can be avoided with the following process:



3.2.4 Left, Right, Forward, Backward

Task 3:
Now program each of the functions STRAIGHT, BACK, LEFT and RIGHT as subprograms, so that you can use them flexibly in various projects later.

Tips:
The procedure for copying an existing process into a subprogram is described in the LLWin manual.
Use a different variable (VAR2-VAR5) in each subprogram to start the process for motor M2.
To ensure that the model does not fall over during the turn to the right, it must first move one step forward with the right side and the one step forward with the left side. Then both sides can take a step together, etc.
The subprogram for this is structured as follows:



fischertechnik®

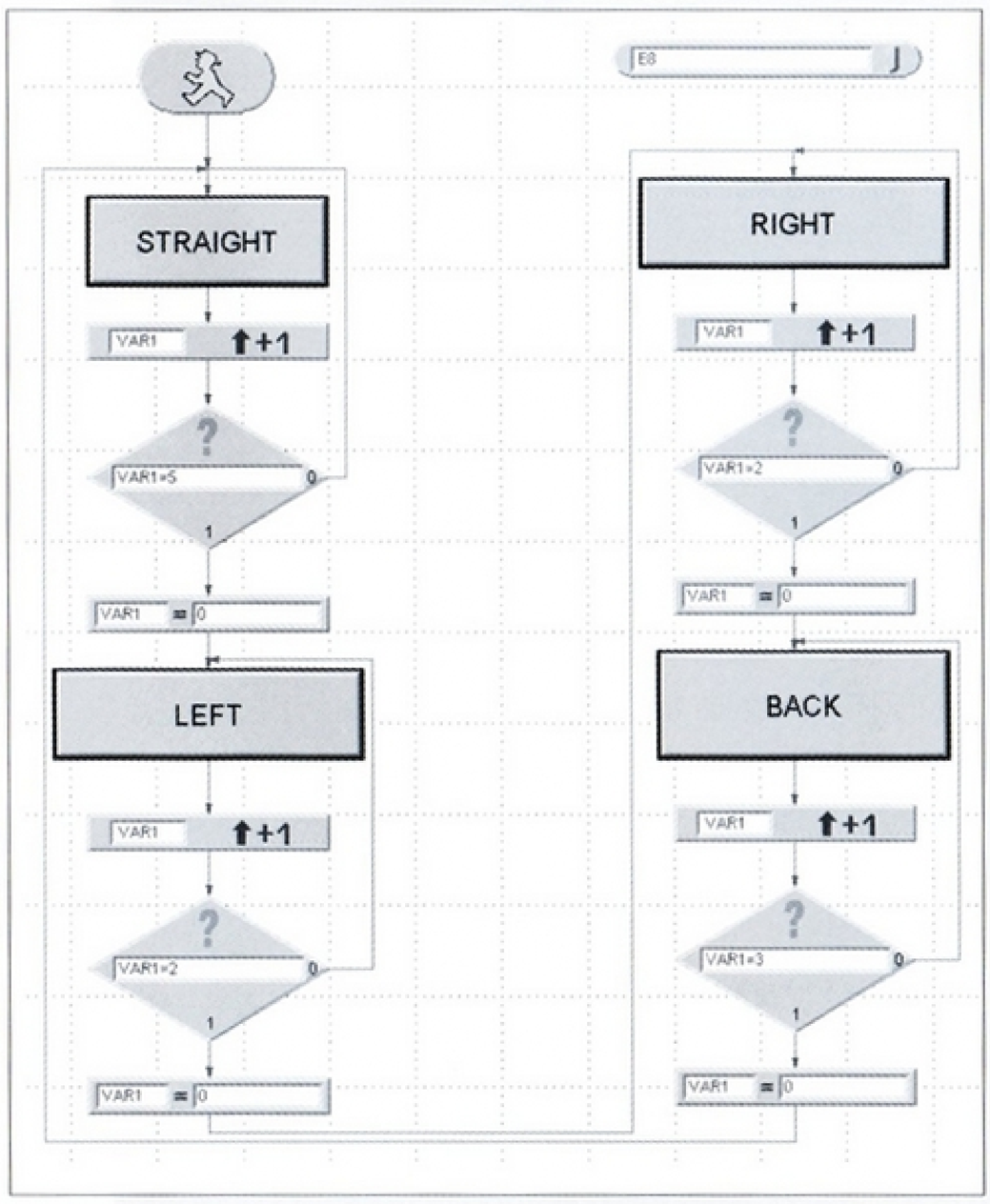
We have not printed the other subprograms here. If you have difficulty programming a process, you can find the finished subprograms in the MIKE_TEMPLATE.MDL file on the CD. The main program of this project is blank. You can find a list with existing subprograms, which you can insert in the main program, in the function block under the "Subprograms" heading.

But don't look right away there to see how it works. First try to figure out the solution yourself. If you do not succeed, you can always look up the solution. To try out all subprograms, we now want to get Mike to dance.

Task 4:

Program Mike, so that he takes five steps forward, turns two steps to the left, then two steps to the right and then three steps back before starting all over again. Use the variable Var1 as count variable for the number of steps. Use E8 as reset pushbutton.

Solution:



This project is called MIKE_DANCE.MDL.

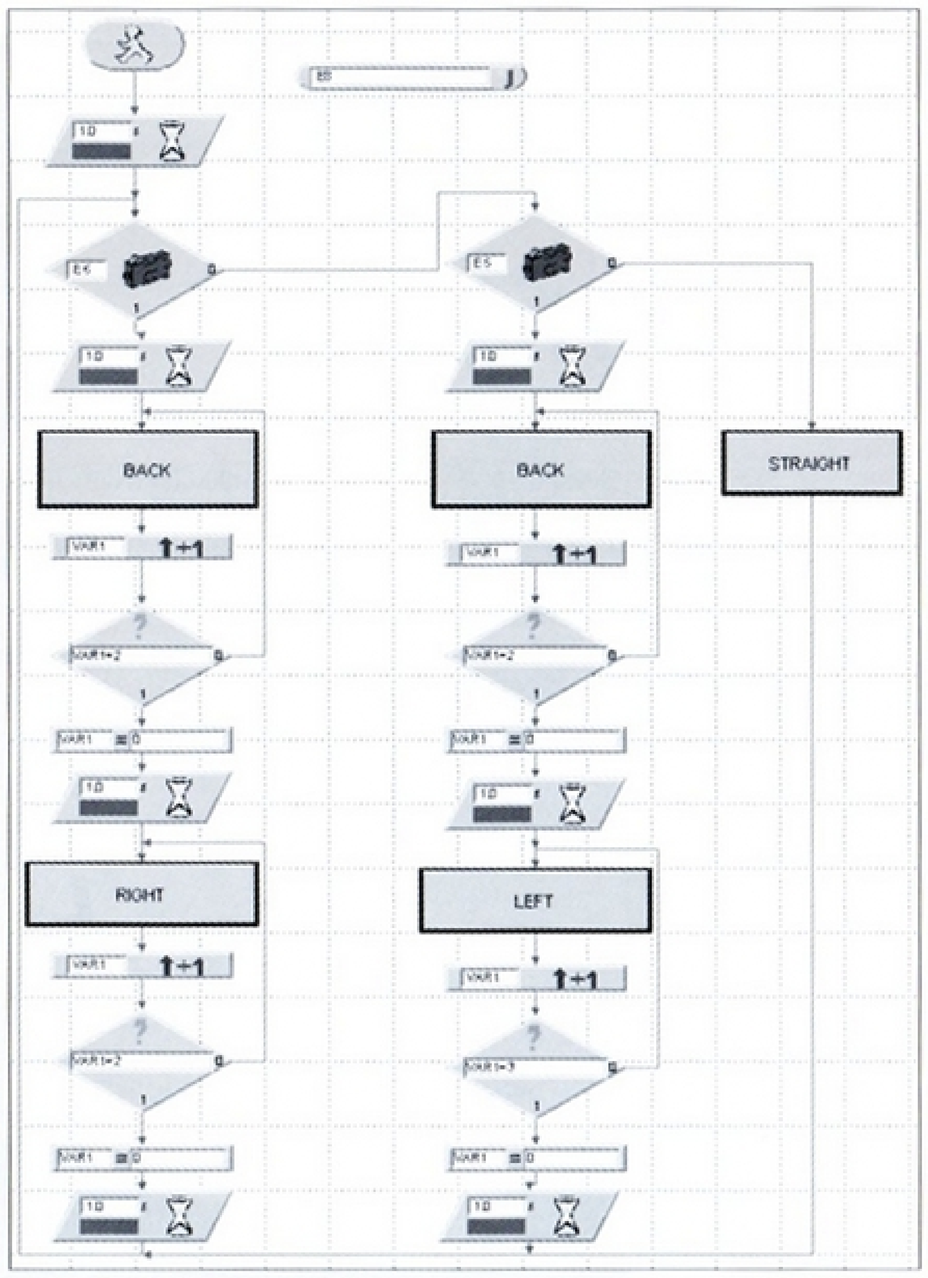
GB+USA

3.2.5 Detecting Obstacles

As a last step, we want to get Mike to detect obstacles with his movable fender ("sensor" would be a better term) and avoid them.

Task 5a:

Program Mike to that when he detects an obstacle at his left sensor (pushbutton E6), he first takes four steps back and then two steps to the right. If there is an obstacle at his right sensor (pushbutton E5), he should take four steps back and then three steps to the left.



Solution:

Mike first walks straight ahead. After each step, the pushbuttons E5 and E6 are queried. If E6 is pressed, the program branches to the left process (first back, then right). If E5 is pressed, the program branches to the middle process (first back, then left).

Because the pushbuttons E5 and E6 are only queried after each full step, it takes a relatively long time until Mike reacts to an obstacle.

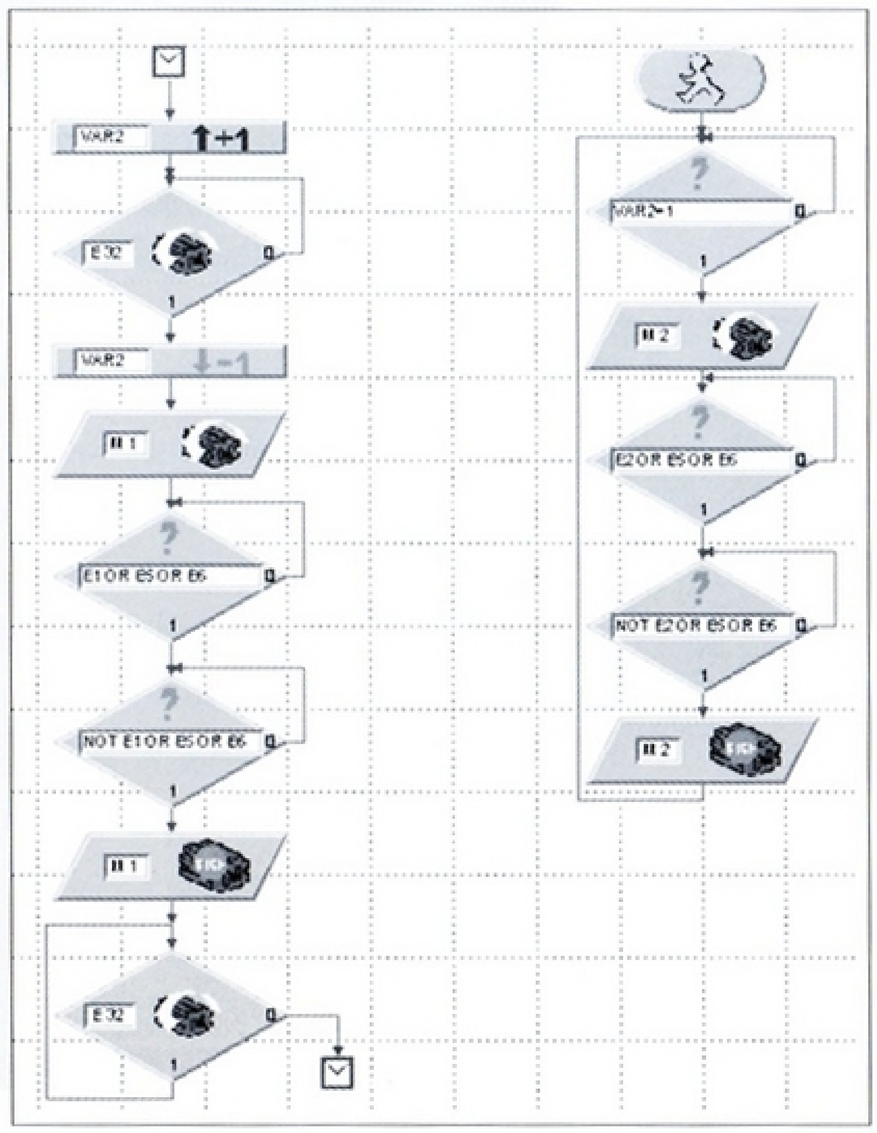
Task 5b:

Optimize the STRAIGHT subprogram, so that Mike can react more quickly to an obstacle.

Tip:

Do not use the EDGE function block but instead the COMPARE function block for querying the push buttons E1 and E2. Also query whether E5 or E6 is pressed.

Solution:



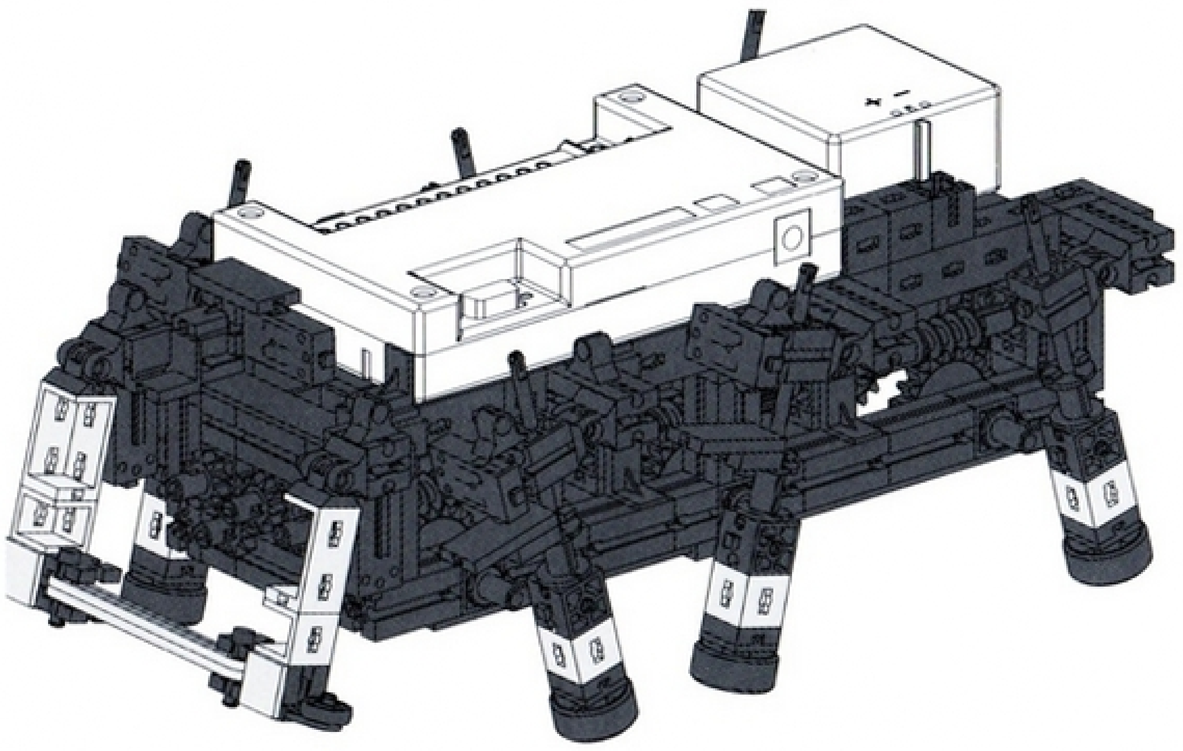
GB+USA

Now Mike should function perfectly. This program is also on the CD under MIKE_OBSTACLE.MDL. You can also integrate the improved subprogram in the MIKE_TEMPLATE.MDL project. If E5 and E6 are not queried in another program, that does not matter at all. We have stored the improved template under MIKE_TEMPLATE_OBSTACLE.MDL.

Now that we have dealt with the first six-legged robot in detail, let's turn to the second model, which also has six legs. We call it "Jack".

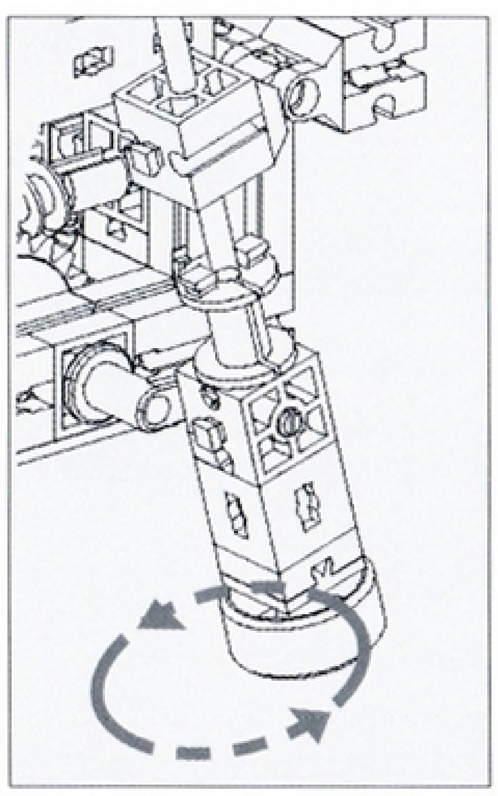
3.3 Model Jack

Jack is also one of the six-legged fischertechnik models. However, the design of his legs differs considerably from those of Mike's.
Now build the model as described in the assembly instructions starting on page 12. By the way, the assembly steps 1-13 are the same for Mike and Jack. Consequently, you need not disassemble Mike completely before you start to build Jack.



3.3.1 The Design

The leg design of Jack also involves a four-bar mechanism. The construction type used here is called an inverted slider crank. The connecting shaft is positioned in a movable longitudinal guide, which swings back and forth when the crank rotates. The curve, which the base of the leg moves along, does not have such an ellipse shape as the model Mike does, but is more of a circle.
As a result, Jack's body goes up and down more during walking than that of Mike. The steps are shorter. But Jack can overcome small obstacles, which Mick cannot. This gear design also reminds us more of a leg than is the case with that of Mike's. When Jack walks, it looks like he is walking on stilts.
He also moves in the three-foot gait of insects.
It is also important for this model to align the cranks precisely as in the assembly instructions, and to tighten the binding pieces and nuts well.



3.3.2 The Programming

It is logical that you can use programs for Jack with which Mike functioned too. Try it!

Task 1:
Operate Jack using the MIKE_OBSTACLE.MDL program. What can you observe?

Observation:
The model walks forward and backward without any problems. But it falls over forward when it turns to the left or right.

Task 2:
How can you explain this?

Solution:
The two models have different leg designs. The pushbuttons E1-E4 are also activated at a different position on the leg. Consequently, the way in which Mike turns need in no way function for Jack – bad luck.

Of course, we don't like this at all and want to find a solution as quickly as possible.

Task 3:
Try to program Jack, so that he detects obstacles as Mike does, but does not fall over forward when he turns.

Tips:
Save the MIKE_OBSTACLE.MDL project under the name JACK_OBSTACLE.MDL and make the necessary changes there.

When Jack turns, both motors can also run simultaneously. The alternating switching on and off is eliminated. The legs must be in the correct starting position at the beginning of turning, i.e., after walking backward.

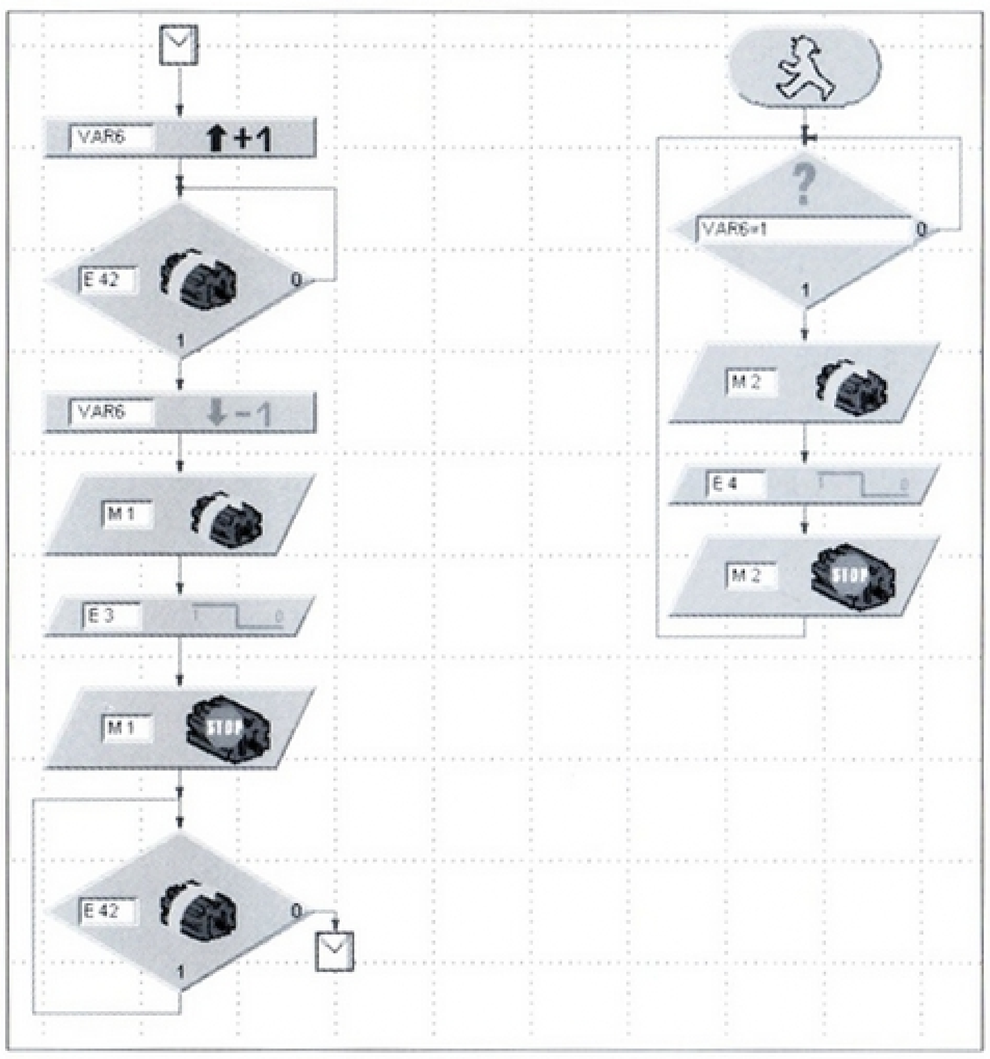
Left turn (counter-clockwise):
If the model should turn to the left, the crank of the front left leg must point backward at the beginning of the turn, and that of the front right leg must point forward. This is the case when the pushbutton E2 on the left side and the pushbutton E1 on the right side are pressed during walking backward and then released again.

Right turn (clockwise):
If the model should turn to the right, the crank of the front left leg must point forward at the beginning of the turn, and that of the front right leg must point backward. This is the case when the pushbutton E4 on the left side and the pushbutton E3 on the right side are pressed during walking backward and then released again.

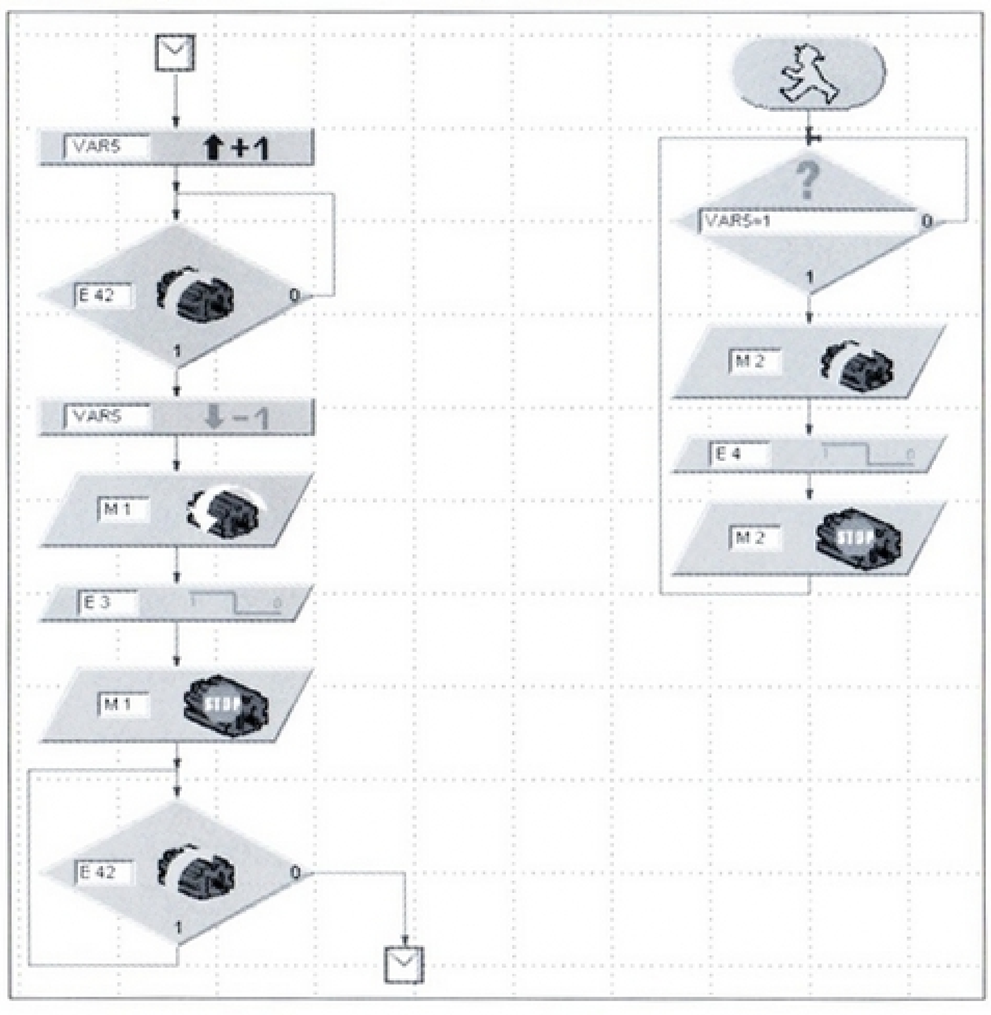
Rather complicated, isn't it? But don't worry; we'll have it solved soon.

Two different subprograms must be used for the backward movement.
If the model should turn left, you synchronize the steps during walking back with the pushbuttons E1 and E2. This corresponds to the BACK subprogram from the MIKE_OBSTACLE.MDL project.
If the model should turn right, you synchronize the steps during walking back with the pushbuttons E3 and E4.

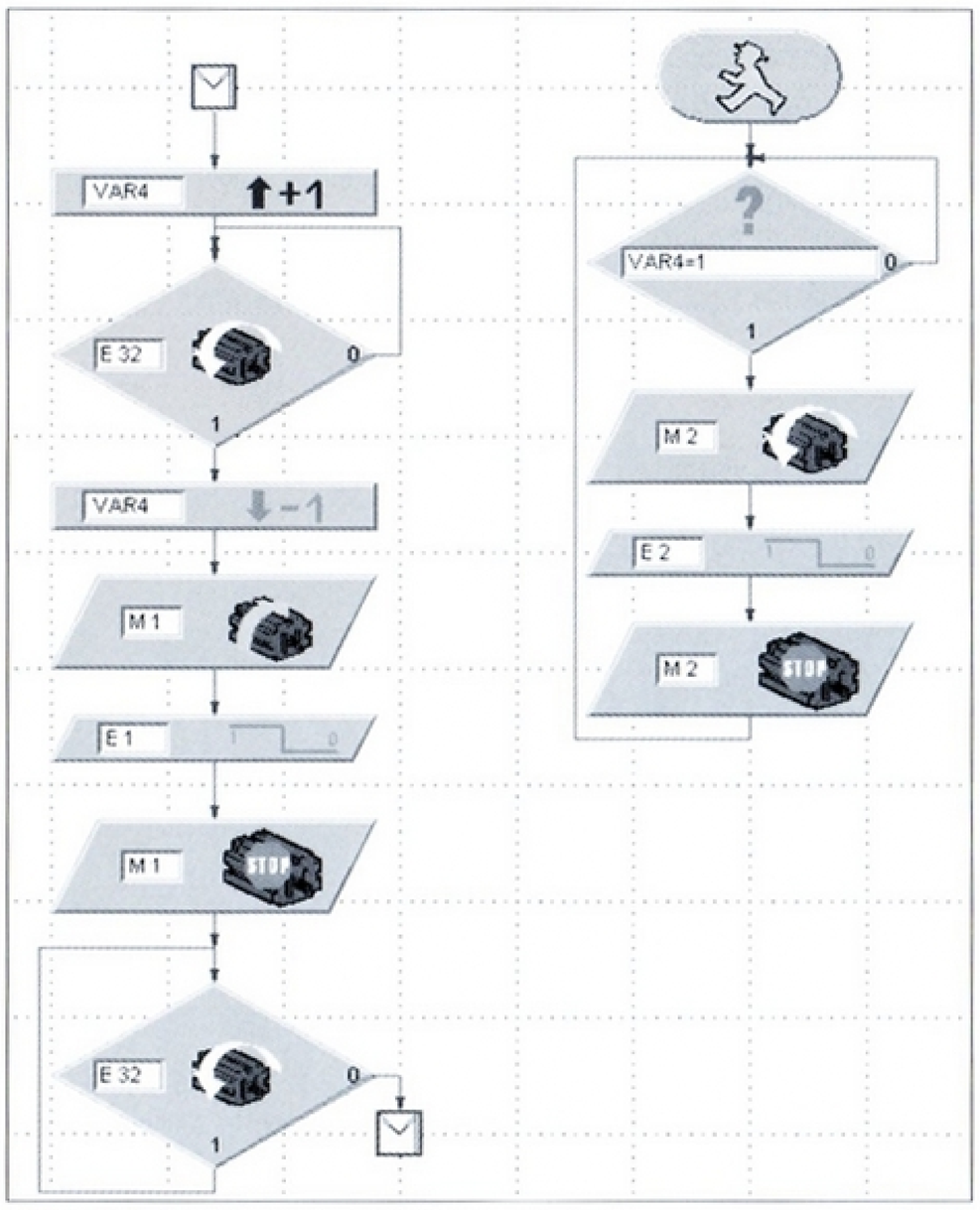
Rename the subprogram BACK, and rename it to BACK_L using the command SUBPROGRAM - RENAME. Then copy it using SUBPROGRAM - COPY into a second subprogram BACK_R. Change the pushbuttons for the synchronization there to E3 and E4. Don't forget to use a new variable VAR6 for BACK_R for the synchronization; otherwise, you will have perfect chaos. BACK_R then looks like this:



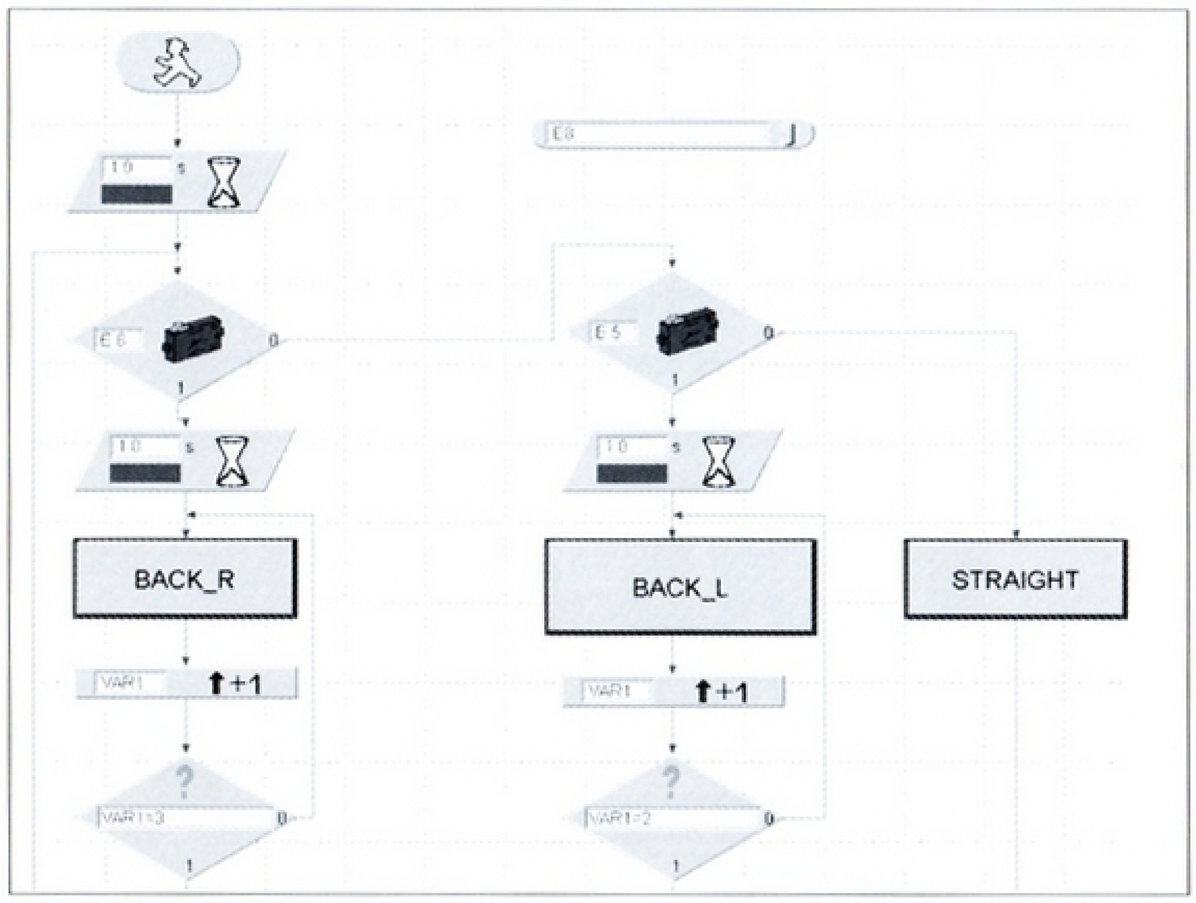
Now you need to change the subprograms for the turning itself, so that the two motors always run simultaneously. The LEFT subprogram is composed of the following blocks:



You can see that a few blocks can be eliminated compared to the subprogram in the MIKE_OBSTACLE.MDL project.
The subprogram for the turn to the right looks similar, only with different motor rotation directions. The pushbuttons E1 and E2 are also used for synchronizing the two motors.



First replace the subprogram BACK_L with BACK_R for turning to the right in the branch in the main program.



The rest of the main program remains unchanged.

You did it! If you didn't make any mistakes, Jack should now walk without falling over when he turns. If something does not function and you have no idea why, don't worry about it because it really was a hard nut to crack. At any rate, you can simply call the finished JACK_OBSTACLE.MDL project from the CD and run the model with it.

If you figured out how to solve the problem yourself, you have reason to be proud, because now you are a pro among programmers.

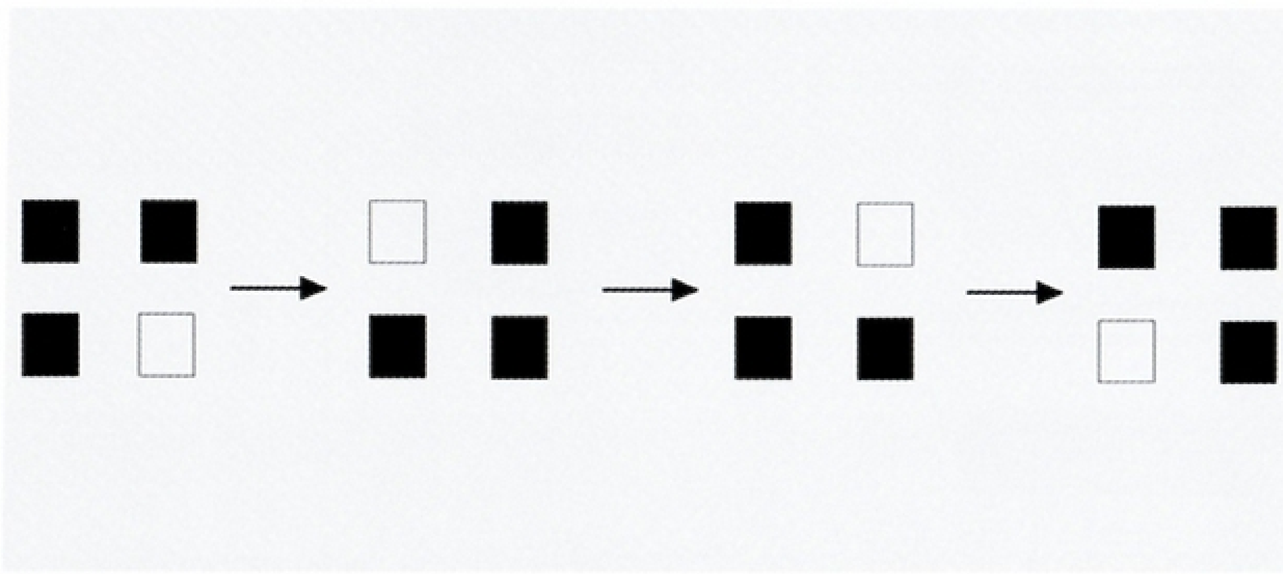
GB+USA

4. Walking on Four Legs

4.1 The Way Mammals Walk

To design a walking robot with four legs, let's use nature as a model again and take a look at how mammals move.

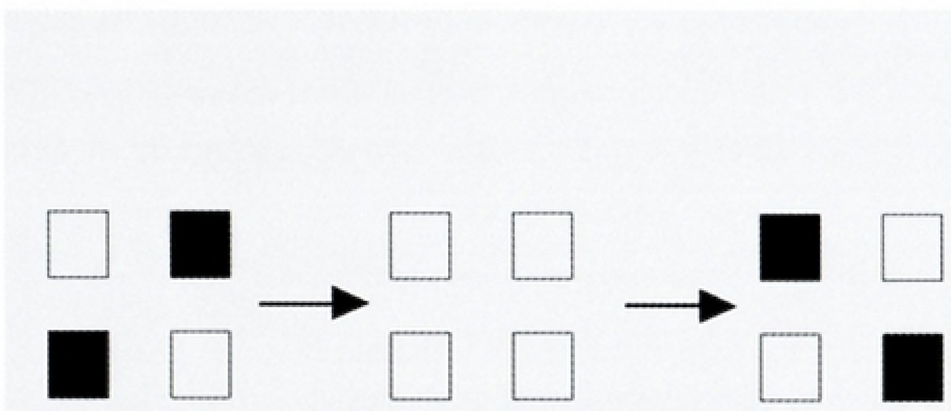
The slowest and most reliable gait is a step. One leg looks for a new place to position itself, while the body of the animal is support on three legs. The animals move by taking steps on opposite sides in the following sequence: right foreleg, left hind leg, left foreleg, right hind leg forward. The black boxes represent the legs, which are on the ground, and the white boxes represent the lifted leg.



If we want to use this gait for a walking robot, we need to consider the following:

Imagine that you saw off one leg from a table with four legs. What happens? Right, the table falls over. Consequently, the three legs no longer form a stable tripod, as was the case with the six-legged models. This complicates the design of a four-legged robot.

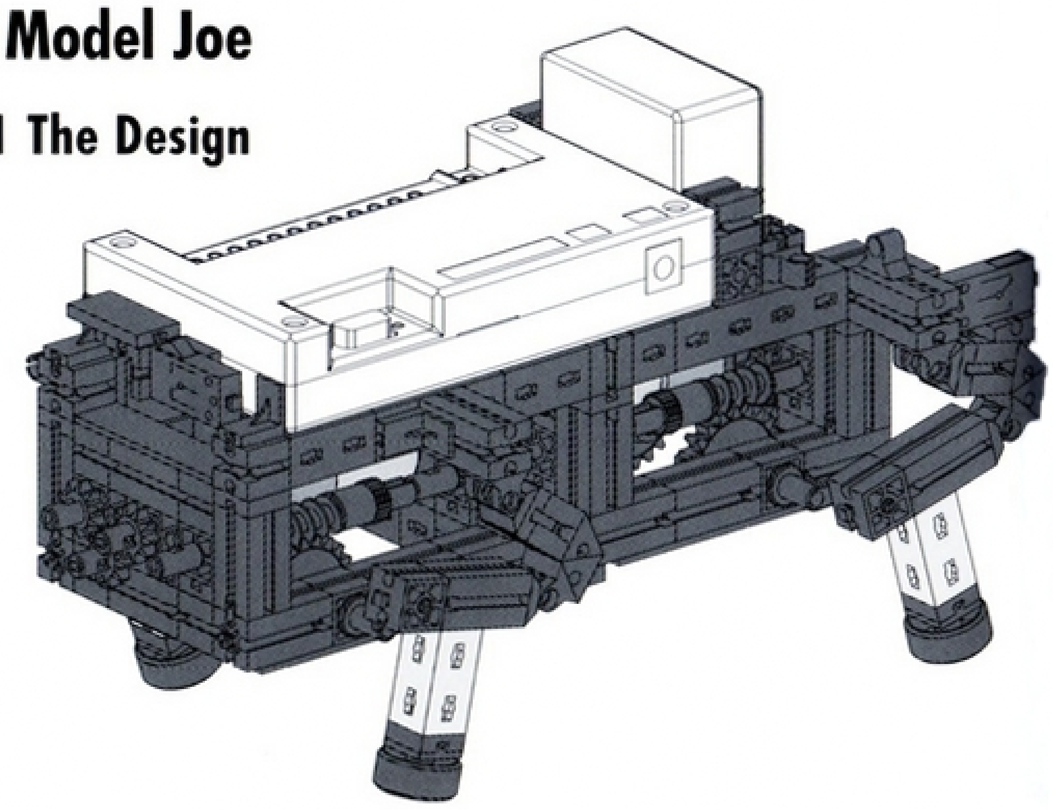
The faster mammals move, the more instable their gait becomes. Let's take a brief look at the gait trotting. The legs are lifted synchronously in a diagonal direction during trotting. But before they touch the ground, the two other legs are already lifted. This means that there is no contact at all to the ground at times.



You can certainly imagine that a gait in which there is no contract to the ground at times is not necessarily suited to a fischertechnik model, in which the interface and power unit should be contained.
Consequently, let's try it using the gait "step."

4.2 Model Joe

4.2.1 The Design



Now build the model as described in the assembly instructions starting on page 20.

The design of the legs is the same as for Mike. But the position of the cranks, which drive the legs, must be completely different for Joe. The cranks are offset from one another by 90°. You must align it exactly as described in the assembly instructions. The left and right sides are again synchronized via the two pushbuttons E1 and E2. Then we get the required step sequence.
To make sure that the model does not fall over as soon as a leg is lifted, the center of gravity of the model must be set so that the model tips at the correct moment and is supported by the leg taking the load just then.

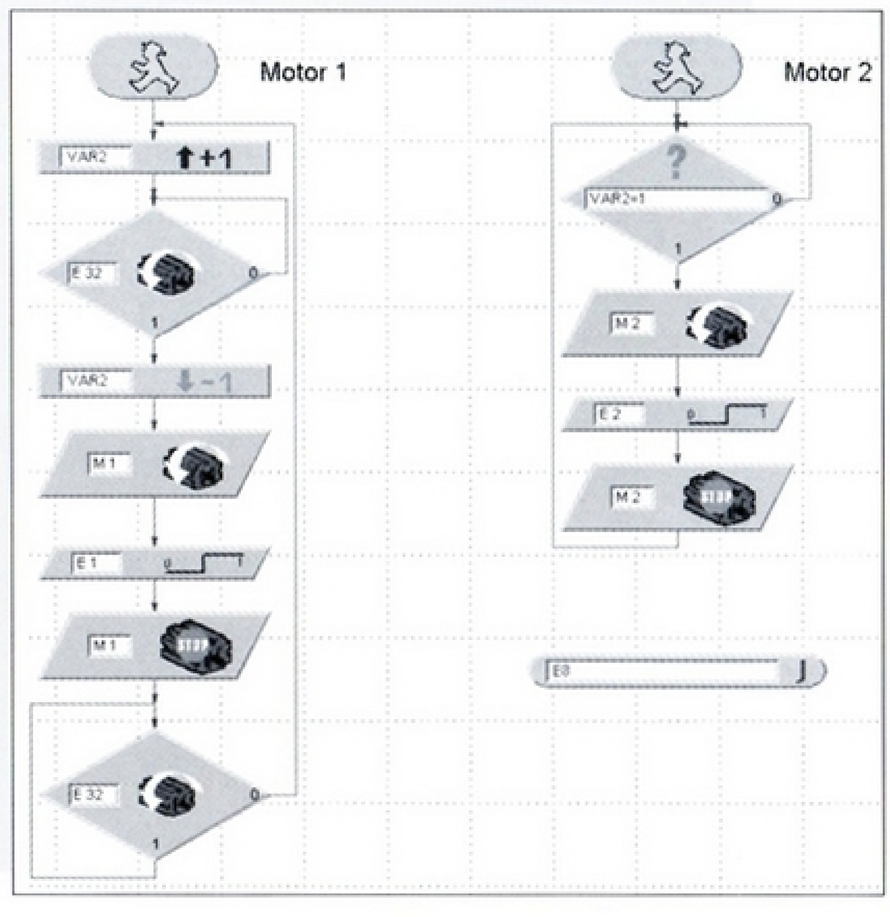
4.2.2 The Programming

We will only program walking straight ahead for this model.

Task 1:
Program Joe, so that he moves forward in the gait step.

Tips:
Use a separate process for each motor, and synchronize the two sides using the pushbuttons E1 and E2.
Use E8 as reset pushbutton.

Solution:



fischertechnik®

We use the 0-1 edge of the pushbuttons for synchronizing in this program. In the moment when the pushbuttons are pressed, the cranks of all four legs have the correct position with respect to each other. We'll call the project JOE.MDL.

You can see that Joe moves a lot more slowly and ponderously than Mike and Jack. Due to the necessary shifting of weight, the body wavers rather strongly and the walking style is far from being as elegant as that of the six-legged robots.

If you feel like it, try to get this model to walk around curves. Simply try to see if you can do it. Good luck!

5. Walking on Two Legs

5.1 Two-Legged Walkers

Walking on two legs did not originate in the species of mammals, but instead was first practiced by a few reptiles. Monitor lizards, iguanas, agamidae and run-lizards only use their hind legs when running away. This lets them achieve big steps and consequently become very fast. They need strong hind legs for this, a long balancing tail and flat terrain.

Birds are also two-legged creatures. The ostrich is the fastest among the birds that run. It can achieve continuous speeds of up to 60 km/h.

The most perfect two-legged creature is man. The completely upright gait requires the stretching of the hip joint. This is achieved by the large buttocks muscle. The legs can also be "locked in place" in the knee joints and consequently fixed in an energy-saving position.

Movement on two legs is the most difficult of all gaits, because it requires a pronounced sense of balance in addition to the described anatomic prerequisites. People think of walking on two legs as a matter of course and simple. But if we consider that when one leg is lifted, the whole body is resting on only one leg and must be balanced, we realize that keeping your balance during this movement type is structured in a very complicated fashion. Even a newborn person is not able to walk immediately on two legs. He first crawls "on all fours" before he stands up and learns to walk.

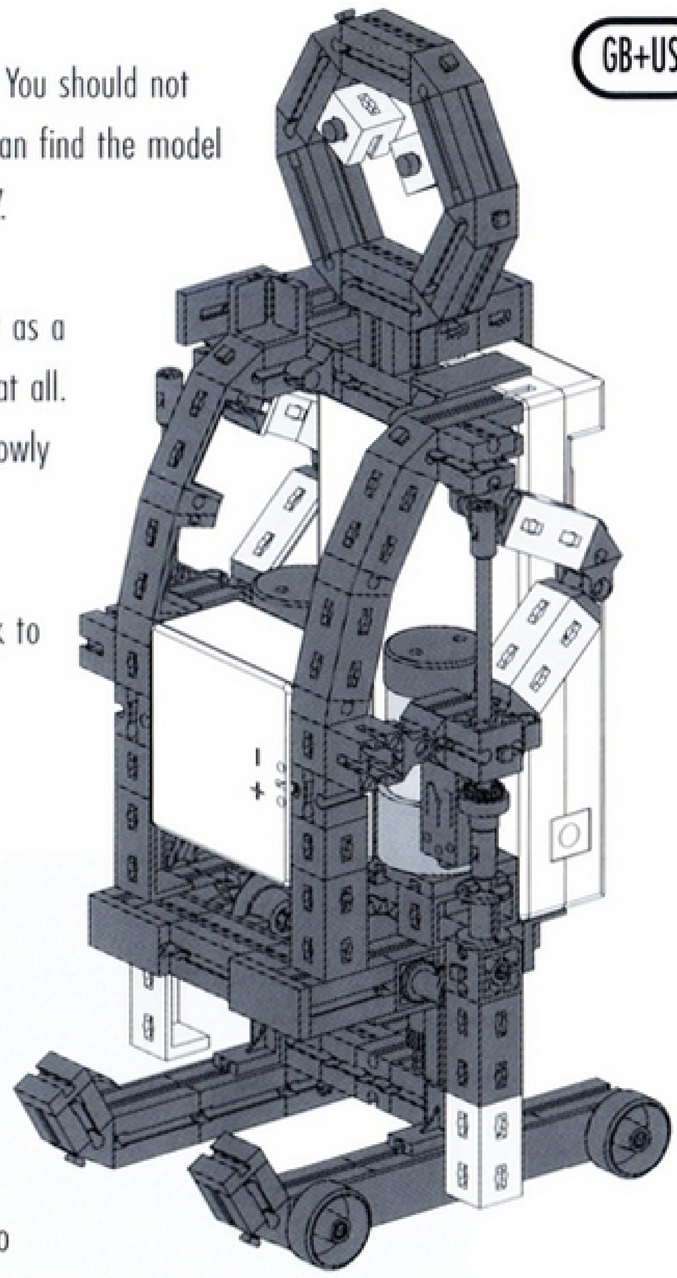
At Waseda University in Tokyo, two-legged robots have already been developed, which move using numerous joints, various sensors, cameras and high-performance microprocessors and maintain their balance by shifting their weight.

But that would be too much work and complicated for our Bionic Robots construction kit. We have seen that we are already reaching the limits of what we can do with walking on four legs with a fischertechnik model.

5.2 Model Jim

But to make sure that we do not only consider this topic in theory, we have designed a two-legged skier here at the end. We call him Jim. Although he does not have a lot to do with two-legged walkers, he is very nice and tries his best to move forward in some way. You should not miss the chance to have this fun. You can find the model in the assembly instructions on page 27.

GB+USA



You can simply use the JOE.MDL project as a program. You need not even change is at all. Jim also functions using it and prods slowly forward.

But we want to give you one more task to perform.

Task 1:
Program Jim, so that he walks approx. 50 cm forward, the turns to the right 180°, walks the same path back (forwards), then turns 180° to the left, walks the same path again, etc. Use the terminal parameter EA for the number of steps straight ahead, EB for the number of steps left and EC for the steps right. Use E8 as reset pushbutton again.

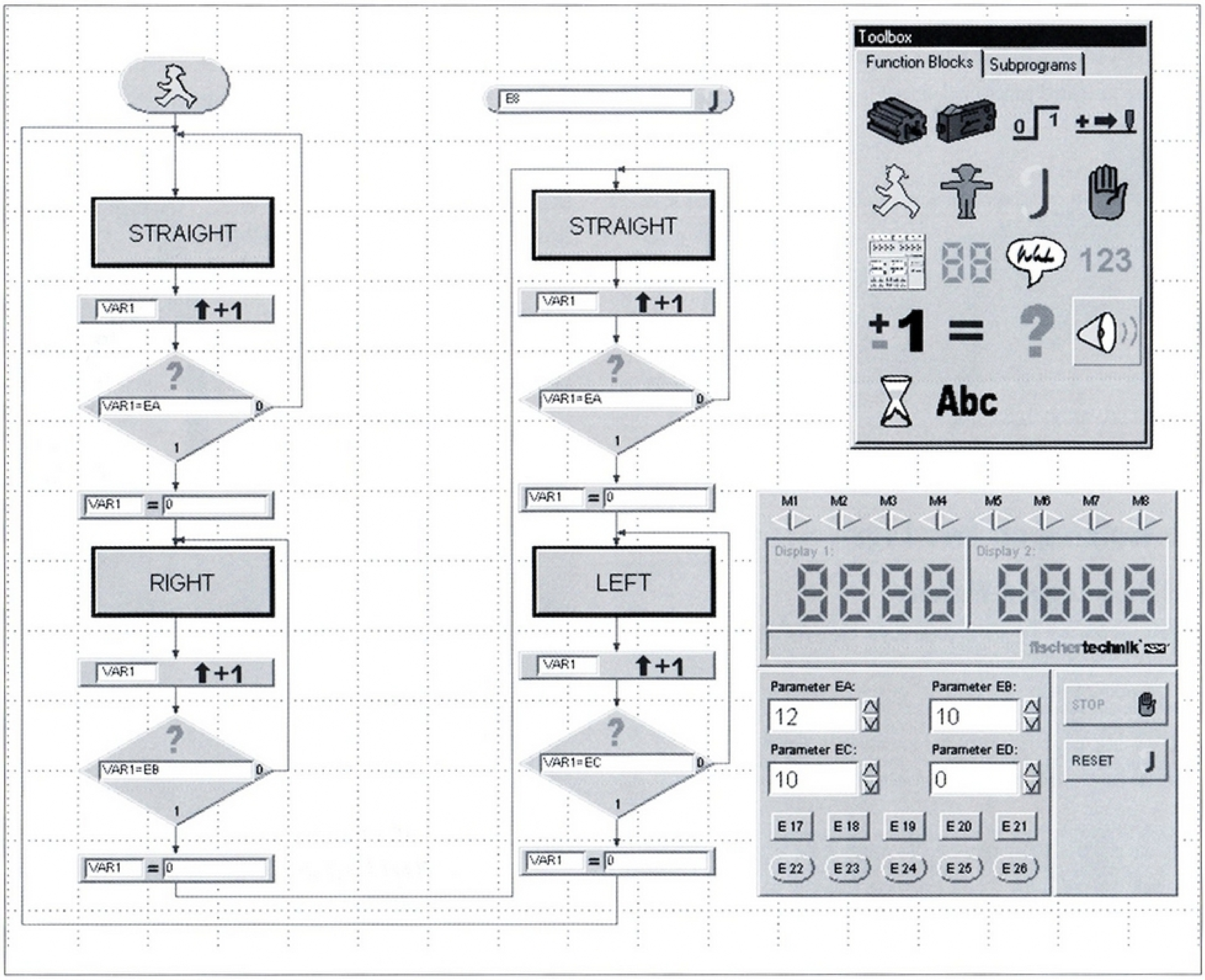
Tips:
Save the JOE.MDL project under JIM.MDL. Make a Straight ahead subprogram out of the main program (select and cut out function blocks, create a new subprogram via EDIT - SUBPROGRAM, insert the function blocks, and supplement SUBIN and SUBOUT; also refer to the LLWin manual).

Create the required subprograms LEFT and RIGHT from this subprogram using the command SUBPROGRAM - COPY. Change the motor rotation direction in it and the query of the motor directions correspondingly. Use a different control variable for motor 2 in each subprogram.

Then program the main program similar to MIKE_DANCE.MDL, except that you use the settable terminal parameters EA-EC for the number of steps. You have to try out how many steps Jim needs to turn 180° or to move forward a half meter.

Solution:
The main program is shown below. If required, you can see the subprograms directly on the screen. We also call the project JIM.MDL.

GB+USA



We added something directly after the BACK subprogram in the project, even though it was not absolutely necessary here. But you certainly want Jim to take other paths too. Maybe he must also move backward sometimes.

6. Summary

You have most certainly seen on your journey through the world of Bionic Robots from fischertechnik that it was not always easy to get the four guys walking. It is simply more difficult to move on legs than to roll on wheels. The programming of synchronization between the left and right sides for turning right or left especially requires a bit of concentration. But for all those who have fun building the models anyway, we copied all programs ready-to-use onto the CD, so that everyone can build and operate the models.

If you are among the pros for programming, you certainly have a lot more ideas about which tasks you can program for Mike, Jack, Joe or Jim, either using additional sensors, so that they do not fall off a table or can find their way around in a labyrinth.

Using additional components, you can also equip them with head, snouts or trunks, or tails. There are not limits to your imagination. Use it!

Bionic Robots

- Begleitheft
- Activity booklet
- Manuel d'accompagnement
- Begeleidend boekje
- Cuaderno adjunto
- Folheto

fischerwerke
Artur Fischer GmbH & Co. KG
Weinhalde 14–18
D-72178 Waldachtal

Telefon: 0 74 43/12-43 69
Fax: 0 74 43/12-45 91
email: fischertechnik@fischerwerke.de
http://www.fischertechnik.de

fischertechnik®

62 965 • ZE • 11/01 • Printed in Germany • Technische Änderungen vorbehalten • Subject to technical modifications