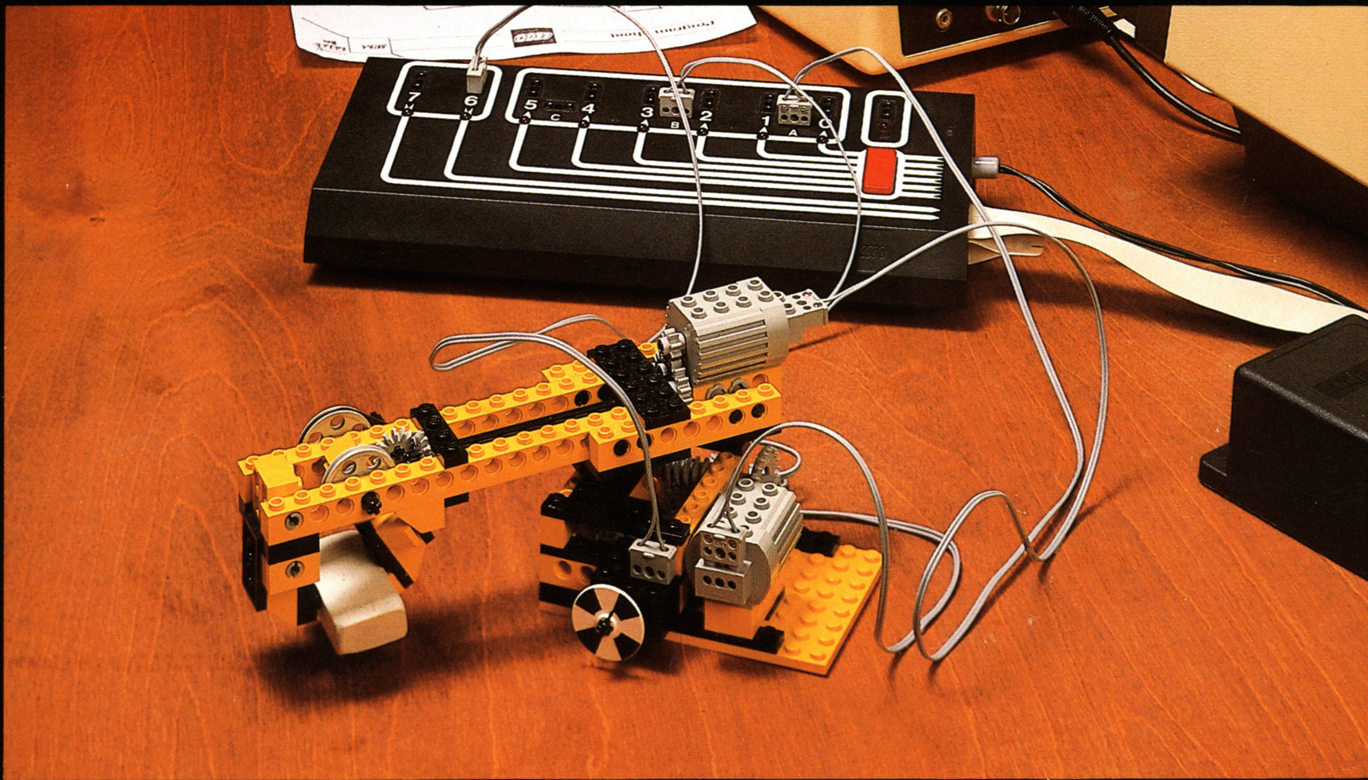
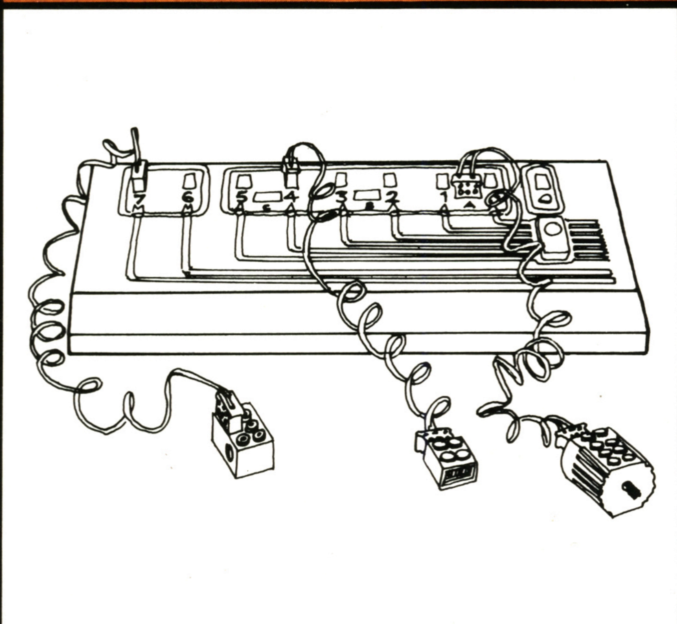
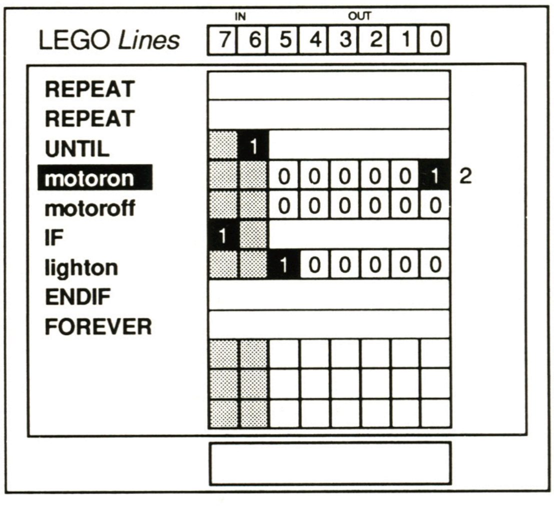
1455 Technic

Programmable Systems

Resources booklet



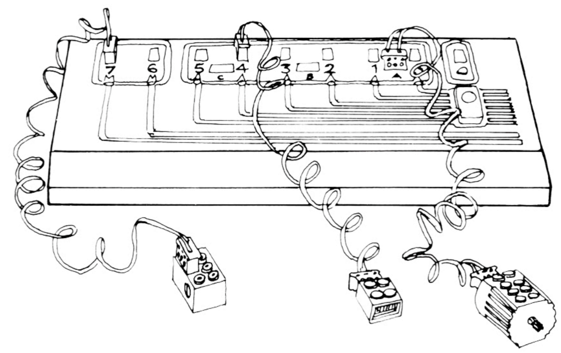


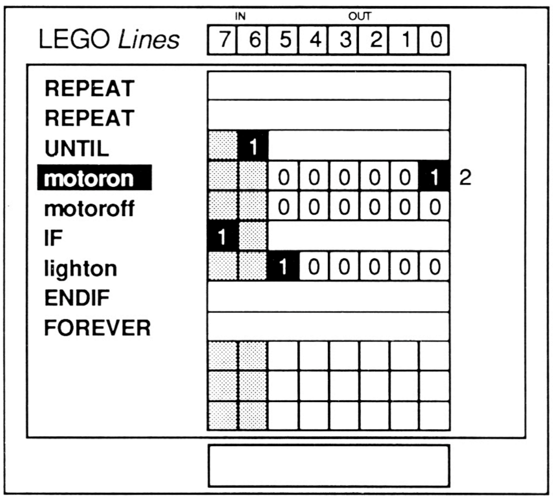


Resources booklet

This booklet is designed to give you useful information and to help you as you work your way through the Programmable Systems material.

It also provides you with a great deal of information about Technology and you should be able to use it as a resource at any time.





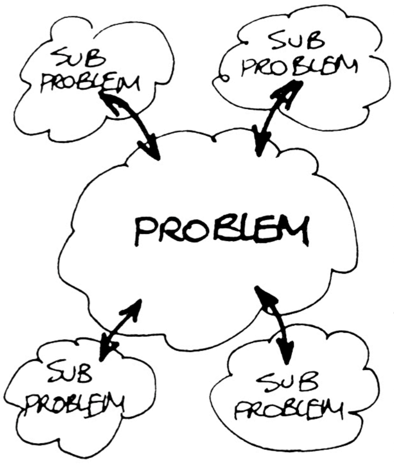
Designed and produced by:

Tecmedia Limited
5 Granby Street
Loughborough LE11 3DU
Tel 0509 230248

ISBN 1 869953 00 2 Programmable Systems (Pack)
ISBN 1 869953 02 9 Resources booklet

Guide to contents

Designing control programs using the LEGO materials



State the problem accurately

- A clear statement (giving all the essential detail) is required, before you begin.

Explore the problem

- Draw up a list of things you need to find out about.
- Decide how you are going to find out about the things on your list.

Develop a deeper understanding

- Carry out investigations to find the answers to the (sub) problems you can identify.

Develop criteria

- Discuss or make notes about what the solution should do.
- Develop tests against which you will judge your solution.

Criteria

A list of statements describing what is required of the solution.

How do I make a list of criteria?

Here are some ways

Ask yourself

See what others have done

Ask others

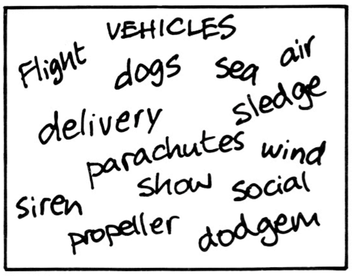
Working in groups to develop ideas

Brainstorming

Get ideas down quickly
Everybody can say something
Any idea will do
Only one or two words per idea

Hints

Give important words plenty of space
Group words together in topic areas

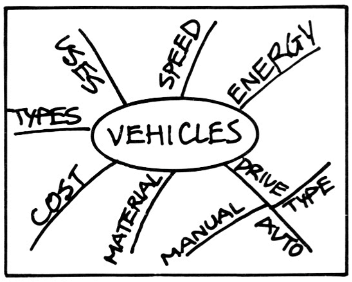


Mindmaps

Build ideas in logical patterns
Provides a good record
Promotes clear thinking
Ideas can be used at the end

Hints

Use words like ***how***, ***what***, ***when***, ***why***, ***where***
Print on the lines of mindmaps
Use words which carry **meaning**
Branch in any direction and in any order on map
Link ideas using lines and arrows



Having found several ideas for solutions you will need to:

— decide the best solution
— make it.

DECIDING THE BEST SOLUTION

ANALYSE THE SOLUTIONS

Decide how well each solution will perform the criteria tests

DECIDE THE BEST SOLUTION

Decide which solution comes out overall best
or
Make up a new solution from the best parts of the others.

You will need

Solution ideas

Criteria tests

Making best use of time

There are a number of ways of analysing the solutions. You might be tempted to think that building and testing each solution would be the best way but this takes a lot of time. You will need to take short cuts by using your experience and the experience of others in your group to reach decisions.

Test each solution

Some practical investigations with experience

Your own experience and the experiences of others

Guess

What does evaluation mean?

Evaluate means to decide the value of something or how good it is.

When you evaluate the final solution you should

- Carry out the criteria tests
- Analyse the test results

When you are doing the tests

Decide on

- The success of each test
- Reasons for weaknesses
- Possible modifications

Note

There could be a number of reasons for weaknesses:

— Poor design
— Limitations of kit
— Lack of time
— Poor criteria

When you are analysing the results

Think about

- A general description of how well your solution solves the problem
- Possible improvements
- How easily the model solution could be adapted for real use

Purpose

Give useful information to others
Convince others your ideas are best
Record ideas and information

Your report should be

Easily understood **Quickly understood** **Interesting**

WRITTEN REPORT

Contents

Title *Author(s)*

The problem — short statement (a few lines)

The solution — short statement (a few lines)

Criteria used

Solutions considered (with evaluation of each)

Reasons for choosing your solution

Development (problems encountered and how they were overcome)

Results of evaluating the final solution

SPOKEN REPORT

Preparation

Good large diagrams
(only essential detail)

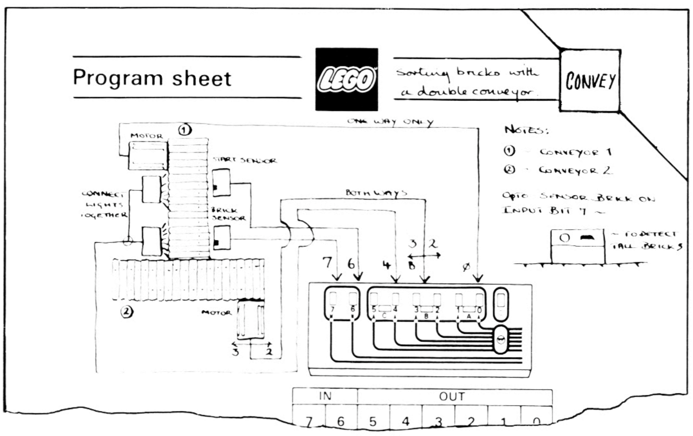
Content of talk

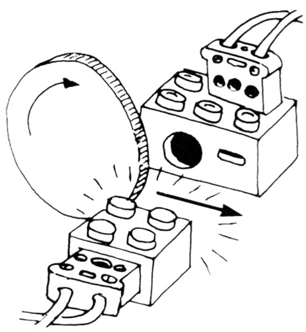
The problem
The criteria used
Solutions considered
The solution chosen
The reasons for choice
Development problems

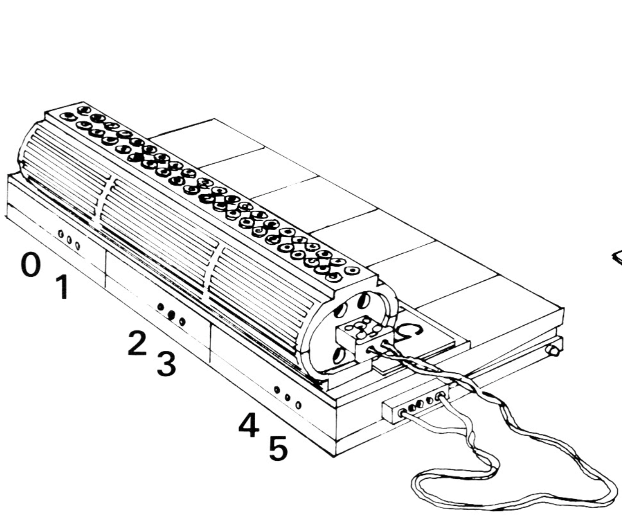
Hints

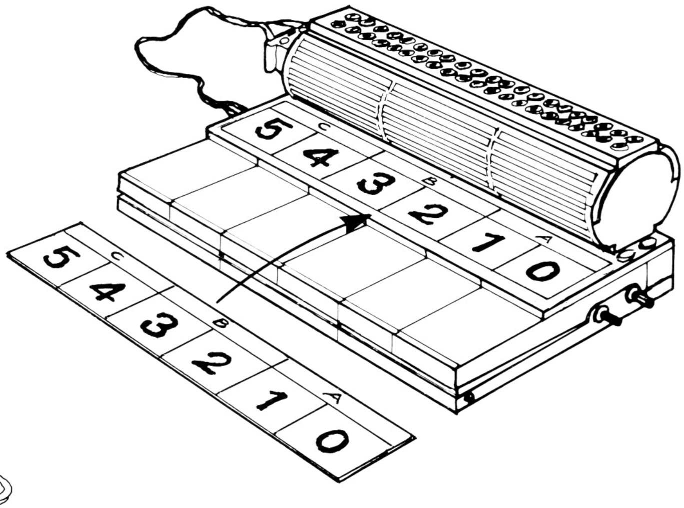
Keep the task simple
Talk to the whole audience
Do not speak too fast
Speak loud enough to be heard

Remember: sketches and diagrams are worth many words.



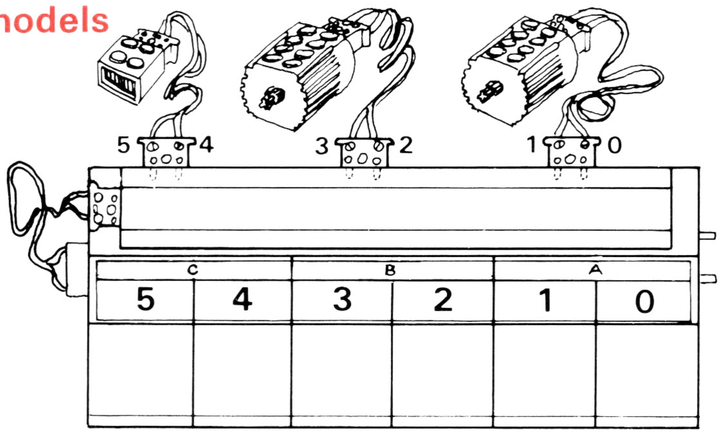




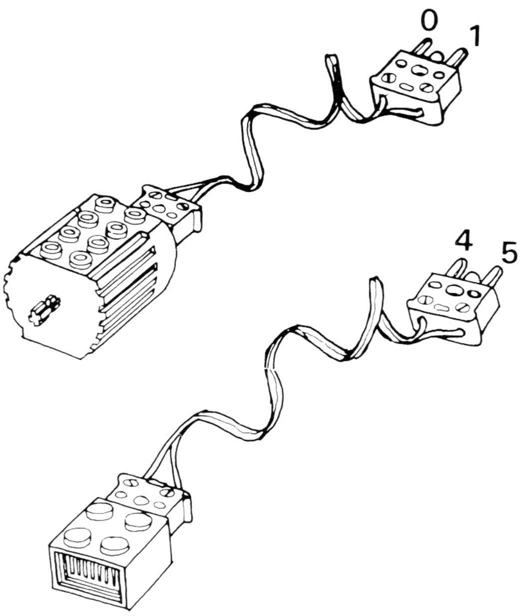


Put keyboard strip over keys

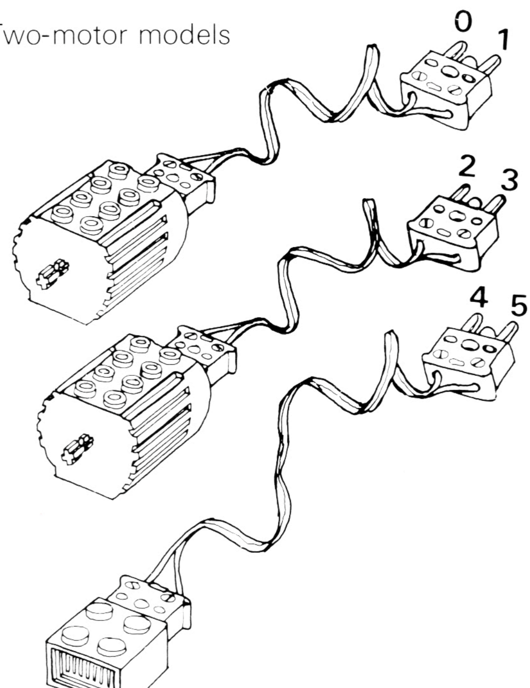
Connecting models



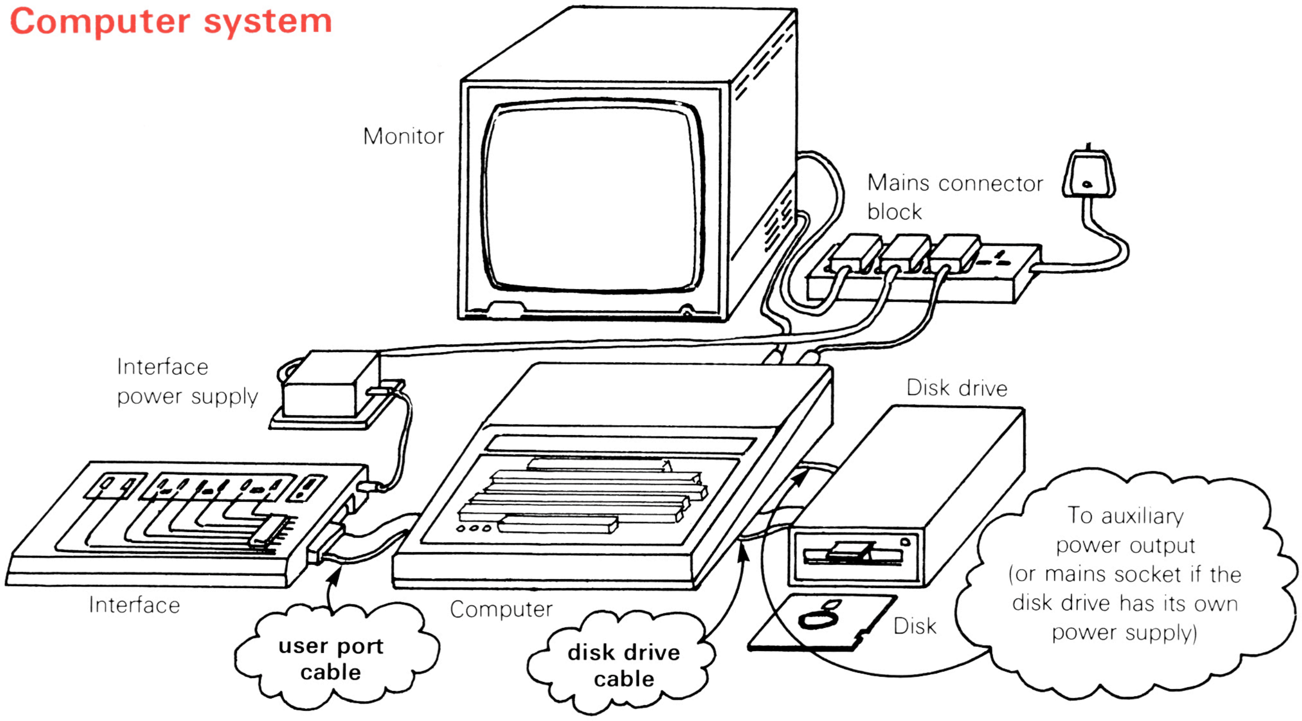
One-motor models



Two-motor models

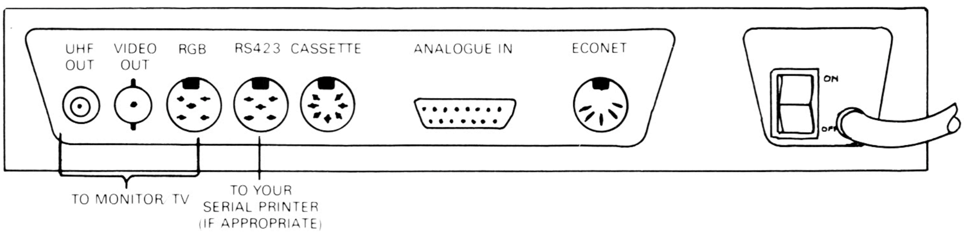


Computer system

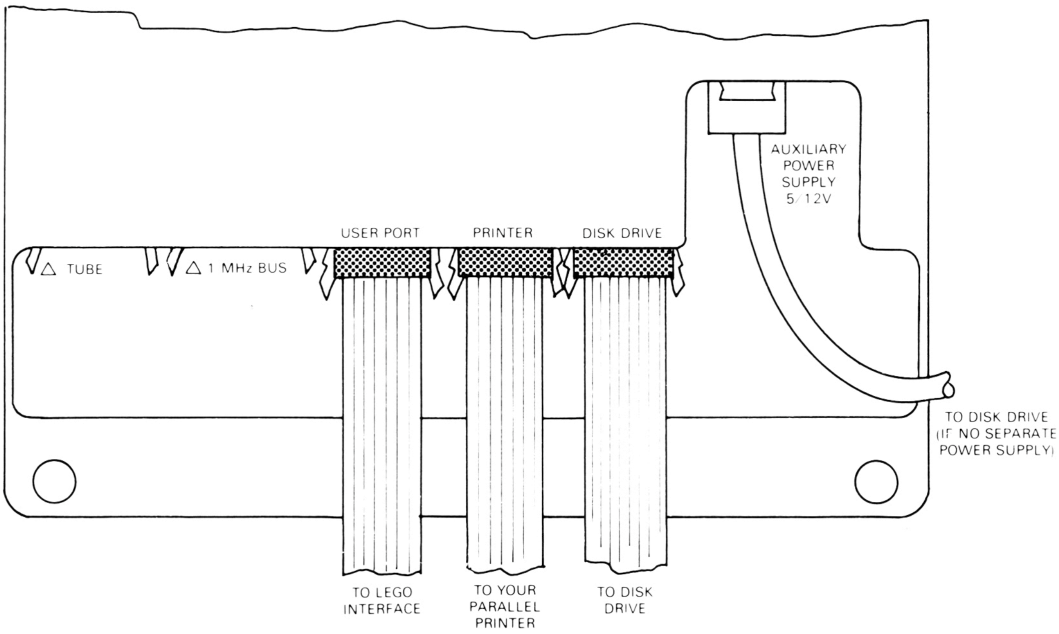


BBC B connections

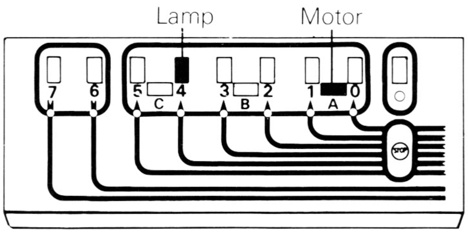
Rear



Underside



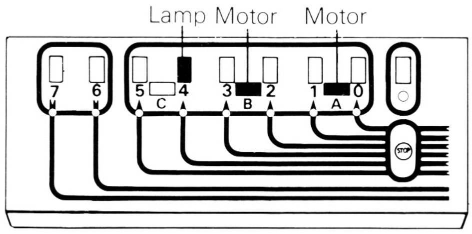
Interface connections for One motor models



4 v power supply

STOP — Press to switch off all outputs

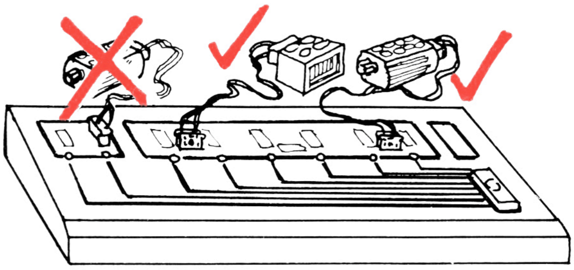
Interface connections for Two motor models

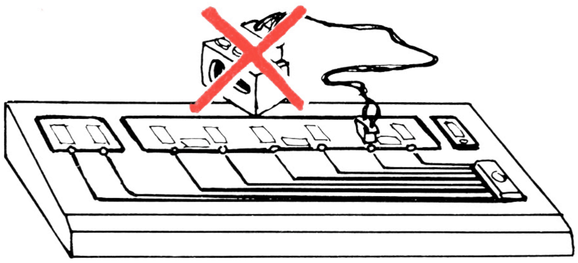


4 v power supply

STOP — Press to switch off all outputs

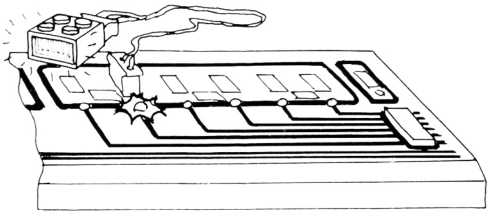
Output connections



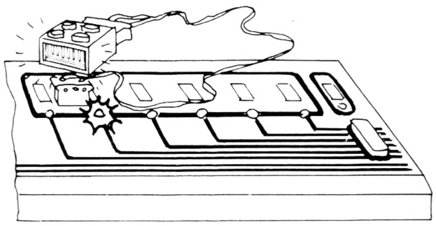


Use of output connections

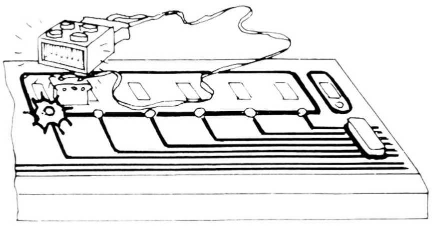
Light on



Light on

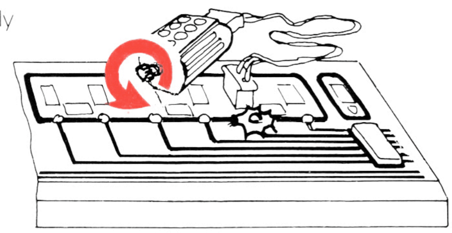


Light on

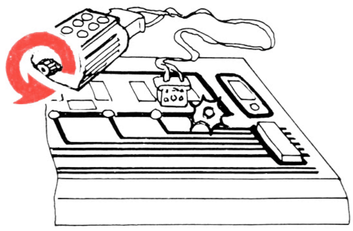


Motor on

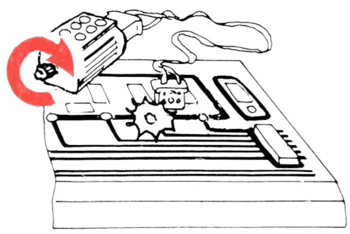
One way only



Motor on

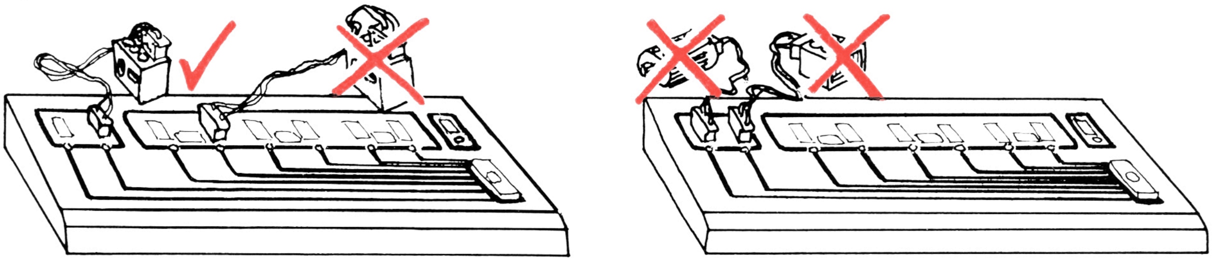


One way



Other way

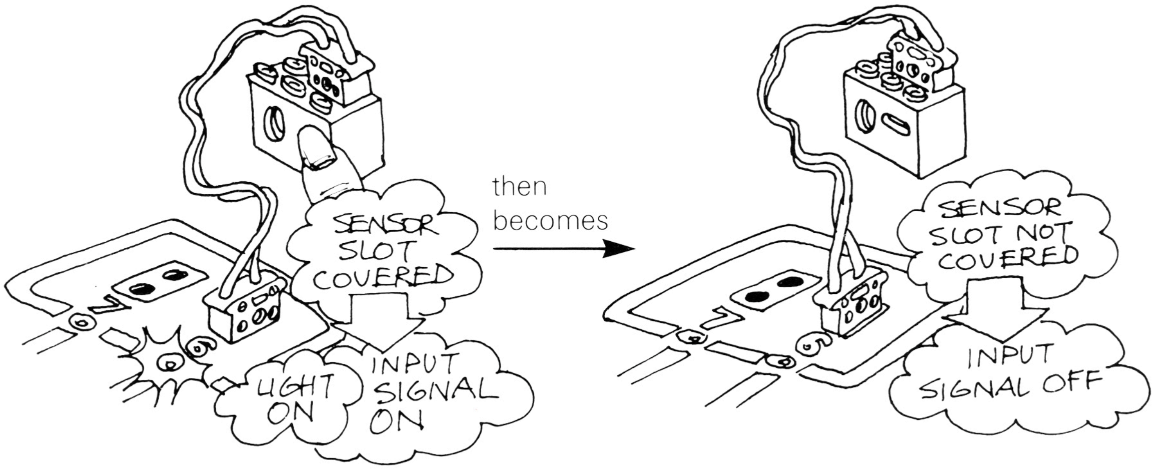
Input connections



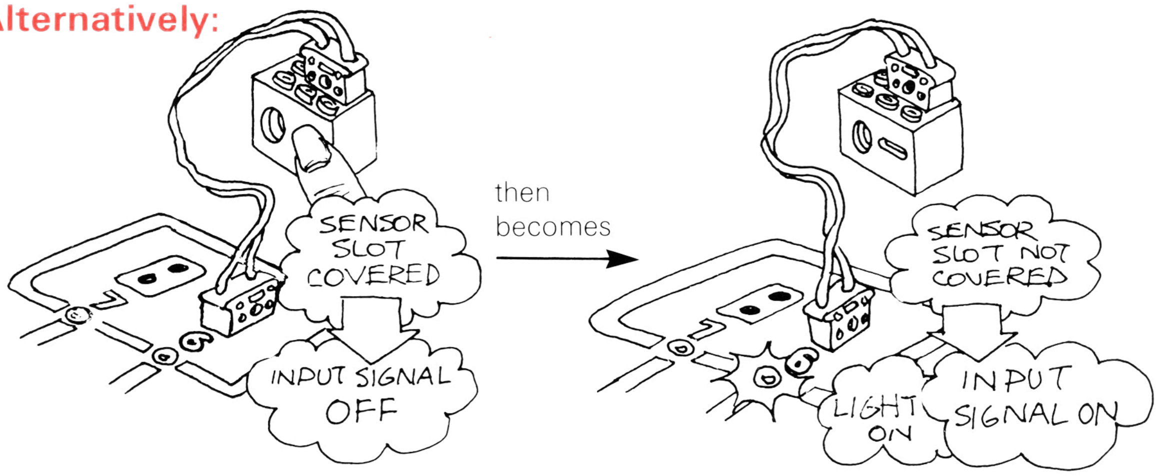
Opto-sensor brick

Electronic sensors take information from the outside world and produce electronic signals to represent that information.

The opto-sensor itself nestles in a hole in the brick. It will react to both external light and to reflected light. Care must be taken in that it will sometimes be on initially, other times off initially. You can force the sensor to be either on or off by shining a light into it from the light brick.



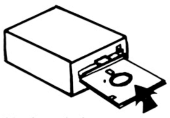
Alternatively:



Connect the opto-sensor brick to bit 6 on the interface. Is the indicator light on or off? When you place your finger over the sensor it will respond to ***changes*** of light conditions.

Change the state of bit 6 a number of times.

Loading LEGO *Lines*



1 Connect computer system
2 Insert LEGO *Lines* disk into disk drive
3 Autoboot program SHIFT BREAK.

Keyboard convention

Key 1	Press key
Key 1 Key 2	means hold 1st key down, press and release 2nd key, then release 1st key

Writing a program

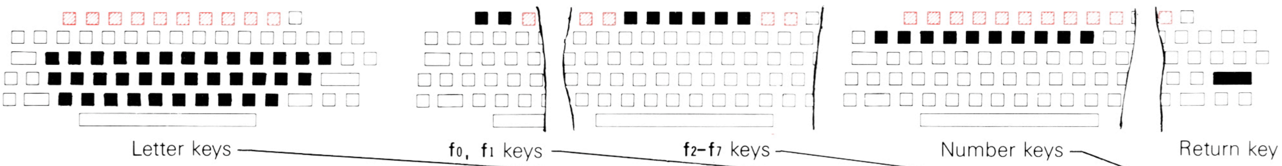
Write label or keyword

Set input bits

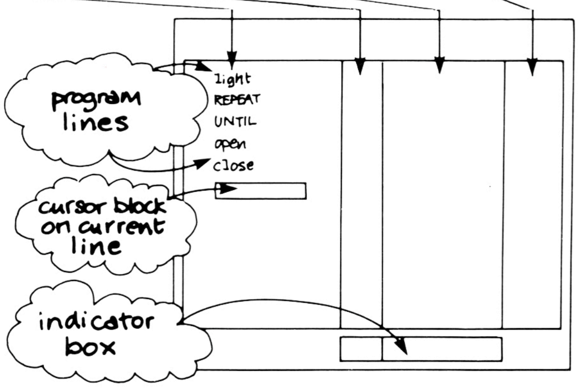
Set output bits

Set time or count

Move to next line



Changing a program line



Move cursor block

Up	↑	Down	↓
Left	←	Right	→
To beginning of program	SHIFT ↑	To end of program	SHIFT ↓

Deleting

Character	DEL
Line	f_9
Program (WIPE)	CTRL f_6

Running a program

Program	TAB
Test of current line (displayed in indicator box)	COPY

Loading and saving a *Lines* program

Load	CTRL f_8
Save	CTRL f_9

Inserting

New line	f_8

Stopping a program

Halt/start	SPACE / SPACE
Stop (TAB to start again)	ESCAPE
End and return to BASIC	CTRL f_5

Other utilities

Boxed display CTRL f_4
(CTRL f_4 or ESCAPE to return to main display)

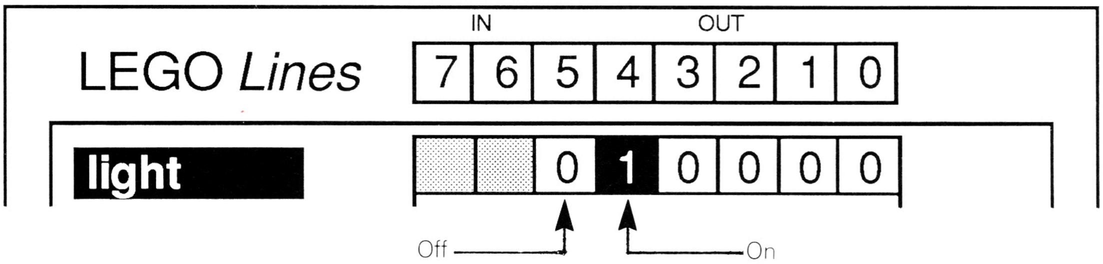
Print	CTRL f_7
Disk directory	CTRL f_3

The LEGO models can be controlled by the LEGO *Lines* program running on the microcomputer.

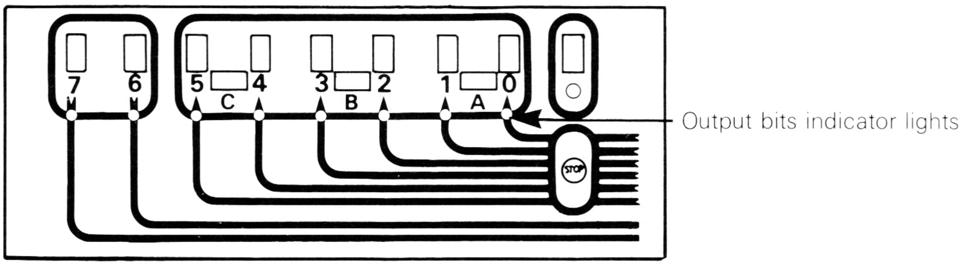
Physical control over the models is achieved by using a pattern of signals, made up of 8 bits, which link the model and the computer together via the interface.

Each of the 8 bits may be set on or off and one pattern of the bits controls the model in a different way to another pattern.

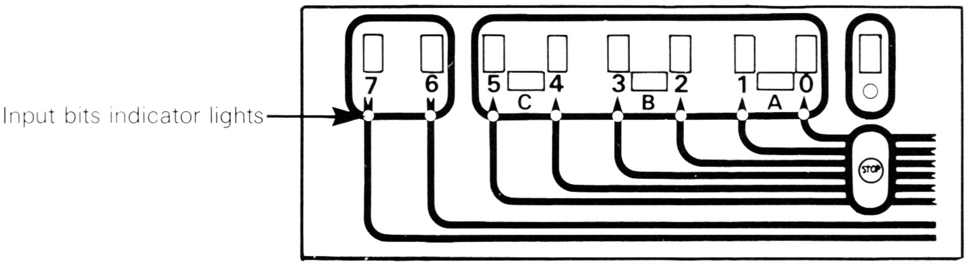
On the screen, a bit which is set on is represented by a figure one in that bit's box whereas a bit which is set off is represented by a zero.



The LEGO *Lines* program is designed to use six of the eight bits as **output** bits. These control the motor and lamp devices from the LEGO set. These are numbered 0-5 on the interface and the indicator lights will show whether they are set on or off at any time.



The other two bits are **input** bits. These are designed to receive messages from the LEGO opto-sensors. These are the bits numbered 6 and 7, and the indicator lights will, again, show their on or off state.

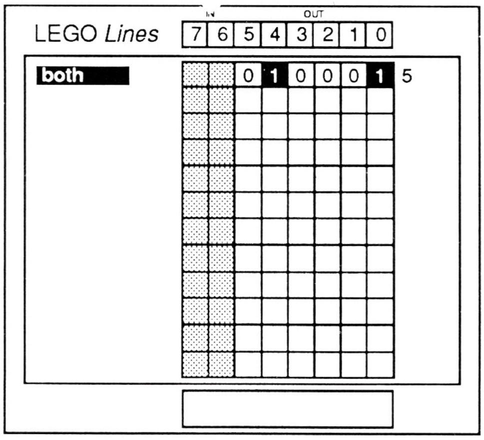


LEGO *Lines* makes use of these bits to physically control the LEGO devices.

Logical control over the models is achieved by using the commands and statements available in LEGO *Lines* called *reserved words*.

Using the commands available and combining them in certain ways, the user can build up a sequence of lines which make up a control program.

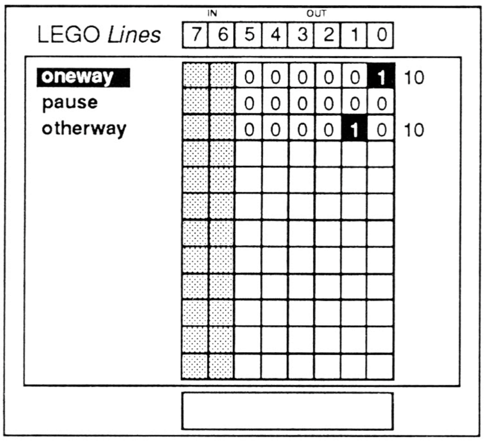
The smallest program which can be created is a single line which will instruct LEGO *Lines* to control the devices attached to the computer system in a particular way. Each line can be given a label to indicate what is happening to the model when that line is being executed.



Program "TWO"

This program lights a lamp and turns on a motor.

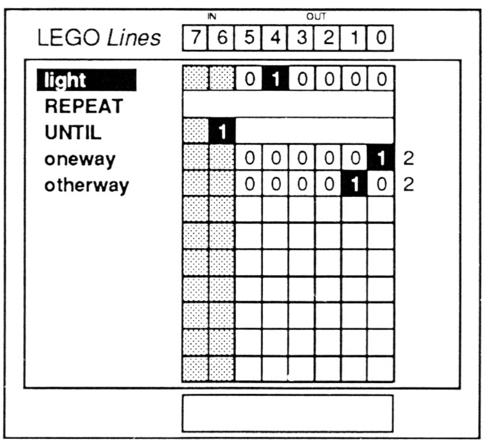
A program usually contains a sequence of lines and these will be executed in the order they are set out from top to bottom.



Program "FOUR"

This program makes a motor revolve one way and then the other way.

You will notice that these two programs do not involve any lines which have input bits in them.



Program "MLB"

The input bits allow LEGO *Lines* to make decisions and, when required, to give commands to control a model.

This program lights a lamp and then waits until input bit 6 is turned on before it makes the motor rotate one way and then the other.

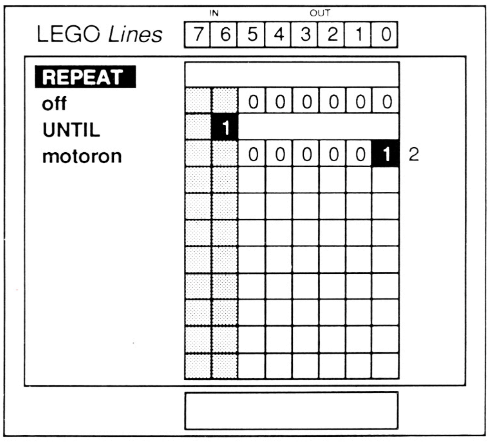
The LEGO *Lines* program has a number of reserved words which must not be used as labels because they are words associated with its important decision-making capability. They will always appear in capitals on the screen to distinguish them from labels, which always appear in lower case.

These are: **REPEAT** **UNTIL** **ENDREPEAT** **FOREVER** **IF** **ENDIF** **COUNT**

Using these reserved words provides structure to a control program.

The following structures are available:

REPEAT UNTIL



Program "SEVEN"

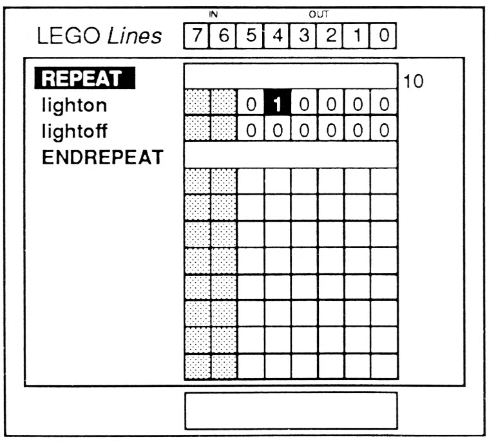
Lines will either be repeated or ignored **UNTIL** a certain input pattern is recognised by the program.

The input pattern on the **UNTIL** line is set using f_0, f_1 and, to set **any value**, **SHIFT** f_0 or **SHIFT** f_1.

This program waits until input bit 6 is set on before it switches on the motor.

Notice that input bit 7 is set to **any value**, meaning that the **UNTIL** line will ignore the state of this particular bit and only look at input bit 6.

REPEAT ENDREPEAT

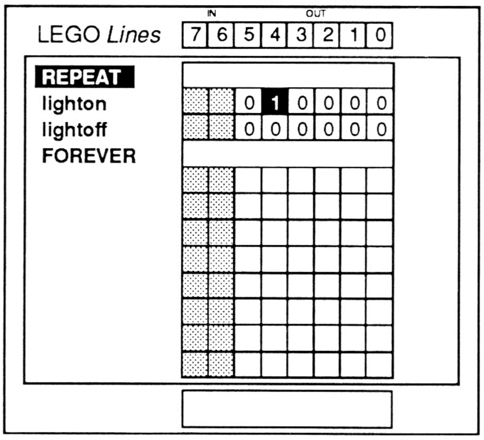


Program "SIX"

Lines between these two reserved words will be repeated a number of times. **ENDREPEAT** marks the end of the sequence of lines to be repeated. The number in the right-hand column of the screen on the **REPEAT** line (entered using the number keys) is how many times the **REPEAT** sequence takes place.

This program turns a lamp on and off ten times.

REPEAT FOREVER

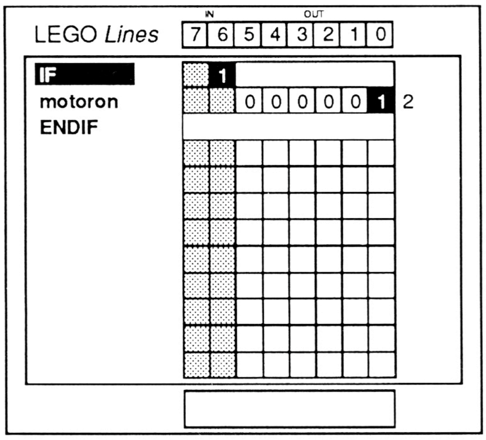


Program "FIVE"

Lines between the **REPEAT** line and the **FOREVER** line will be repeated until the **ESCAPE** key is used to stop the control program.

This program turns a lamp on and off forever, or until the **ESCAPE** key is used to stop it.

IF ENDIF



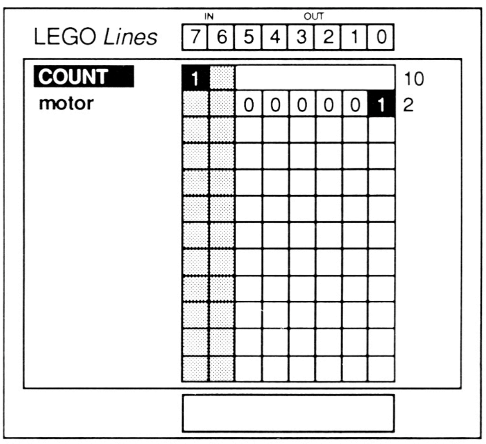
Program "NINE"

Lines will be executed or skipped over according to a certain pattern on the input bits.

The input pattern on the **IF** line is set using f_0, f_1 and, to set **any value**, **SHIFT** f_0 or **SHIFT** f_1. **ENDIF** is used to mark the end of the group of lines in question.

This program turns on a motor if input bit 6 is set on.

COUNT



Program "COUNT"

This reserved word is used to count input signals on a single input bit before it goes on to execute the lines which follow. The number of input signals it is counting is entered on the right hand side of the **COUNT** line.

COUNT uses only one of the two input lines (7 or 6) and requires the other input line to be set to **any value** using **SHIFT** f_0 or **SHIFT** f_1 as appropriate.

This program turns on a motor if input bit 7 is set on 10 times.

Assignments 2 and 5 introduce the use of these reserved words.

Errors

Run errors

After pressing the **TAB** key to run a program, *Lines* performs various checks to ensure that the structures within the program are correct.

If an error is found, *Lines* will not run the program, displaying instead a message giving the type of error found, in the form:

Lines error *n* (where *n* is a number)

These numbers refer to the following errors:

1 The number of **REPEAT/REPEAT** *n* levels exceeds eight.
2 The number of **IF** levels exceeds eight.
3 Input condition not properly set in **COUNT**.
4 Number to **COUNT** is not set.
5 **REPEAT** without **UNTIL/FOREVER**.
6 **REPEAT** *n* without **ENDREPEAT**.
7 **IF** without **ENDIF**.
8 **UNTIL/FOREVER** without **REPEAT**.
9 **ENDREPEAT** without **REPEAT** *n*.
10 **ENDIF** without **IF**.
11 **ENDREPEAT** ending a **REPEAT UNTIL/FOREVER** structure.
12 **UNTIL/FOREVER** ending a **REPEAT** *n* **ENDREPEAT** structure.
13 Incomplete **IF ENDIF** structure within a **REPEAT** *n* **ENDREPEAT** structure.
14 Incomplete **IF ENDIF** structure within a **REPEAT UNTIL/FOREVER** structure.
15 Incomplete **REPEAT** *n* **ENDREPEAT** structure within an **IF ENDIF** structure.

Boxed display error

Too many levels Only six levels of nested **REPEAT** and **IF** structures can be displayed on screen. This message is shown if you attempt to display a more complex program on screen.

Loading and saving errors

File does not exist This message appears when you are attempting to load a file from the disk but the name you have typed in is not found as a file name on the disk. Pressing **CTRL f3** will display a list of the correct names on the disk, and you can ensure that, next time, you enter the correct name.

File exists Replace? . . . This message is displayed when you are attempting to save a file (*Lines* program) and there is already a file with that name on the disk. *Lines* expects a **YES** to overwrite the disk file with the one you are working on, or a **NO** to abandon your attempt to save the file with that name.

Disk error If this message is displayed, there are a number of possible causes:

— The write-protect sticker has not been removed before trying to save a program.
— The disk (or disk catalogue) is full.
— The disk has been corrupted.
— The disk has been incorrectly formatted.
— The program **discut** has been used on a single drive system and the default drive has been set to 1.

'Feedback is important in order to know where you are.'

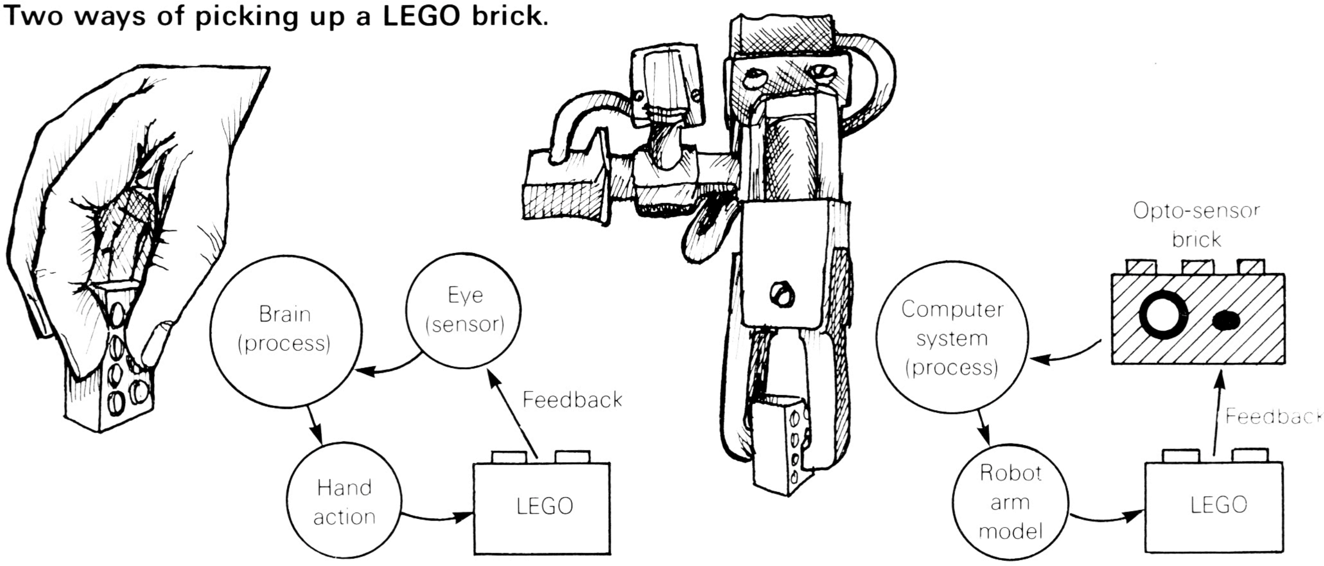
First write down the sentence about feedback.

Now try to do the same thing again, only this time do it *with your eyes shut*.

You were able to write the first sentence better because your eyes sensed where your hand was and fed this information to the brain.

This is called ***feedback***.

Two ways of picking up a LEGO brick.



The brain acts on the information it receives from the eye.

The computer system acts on the information it receives from the opto-sensor.

A ***dedicated*** computer system will constantly monitor the feedback from the job it is doing. Frequently, however, computer systems are required to do more than one job at the same time — this is like constantly opening and closing your eyes while you do something. Computer systems usually operate at faster speeds compared to devices they are controlling, and so they are capable of processing instructions to control more than one task.

Developing a program

A program is a sequence of instructions which solves a particular problem. The best test of a program is whether it works.

Other important criteria are whether it is efficient in terms of statements and speed, and whether it can be easily understood and followed.

Problem

a	**Analysis**	Express the problem as a sentence or sentences in English. The clearer the statement of the problem the neater the solution will be. Determine also what the solution is required to do.
i	**Ideas**	Identify **Actions** (and give them a meaningful label) **Tests** — when actions should or should not take place **Loops** — when a sequence of tests and/or actions are repeated.
s	**Developing solutions**	Use a structured flow diagram (Resources booklet **R14**) to produce solutions.
	Best solution	Choose the best solution and translate the diagram into a program which you can test on the computer.
e	**Evaluate**	Judge whether the program achieves what you required it to do.
c	**Communicate the solution**	State what the problem was and describe your solution.

Solution

In order to construct a structured flow diagram it is necessary to take a sentence which describes a solution to a problem and identify:

The ***sequence*** of events
The ***actions*** which make up these events
The ***tests*** which decide whether or not those actions take place
The ***loops*** in which actions or sequences of actions take place.

Look both ways and cross the road slowly if there are no vehicles coming in either direction

Sequence	Look both ways — if no vehicles — cross the road slowly
Actions	Look both ways — cross the road slowly
Tests	If there are no vehicles coming in either direction
Loops	—

In a structured flow diagram the idea is to place actions into specially shaped boxes

IF a vehicle is coming
actions

REPEAT
actions
UNTIL road crossed

For the example of crossing the road:

Crossing road
Look both ways
IF no vehicle
Cross road slowly

Crossing road
IF no vehicles
Look both ways
Cross road slowly

Loops

REPEAT

actions

UNTIL ?

There are a number of ways to organise the condition (?) which finishes the REPEAT/UNTIL loop.

REPEAT UNTIL a certain input message is received.

REPEAT UNTIL a counter reaches a certain value, in other words REPEAT a number of times

REPEAT FOREVER

In the example:

Crossing road

REPEAT

Look both ways

Cross road slowly

UNTIL other side

But watch out — a vehicle may be approaching!

In structured flow diagrams tests and loops may contain each other

Crossing road

IF no vehicles

REPEAT

Look both ways

Cross road slowly

UNTIL other side

Crossing road

REPEAT

Look both ways

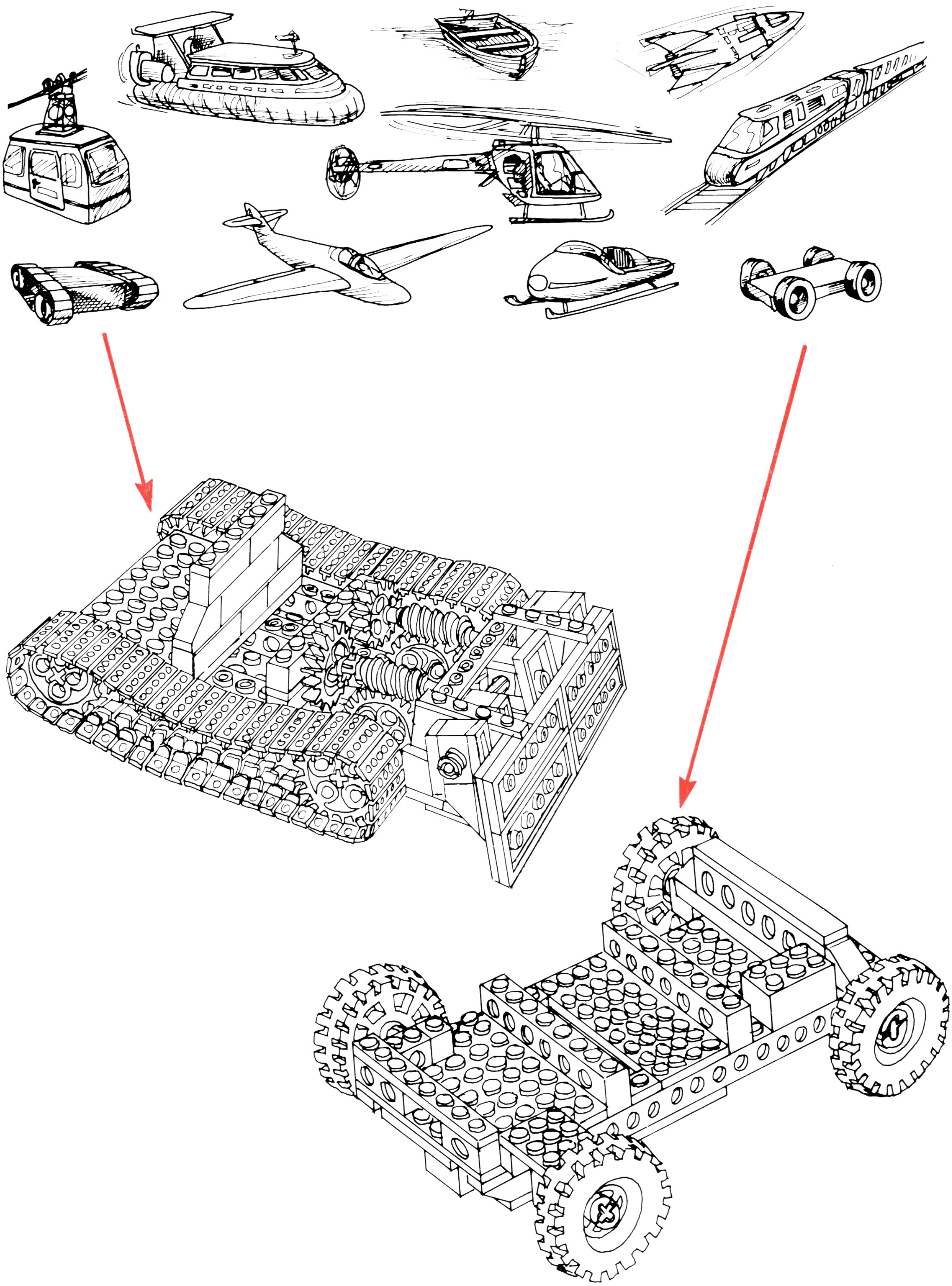
Cross road slowly

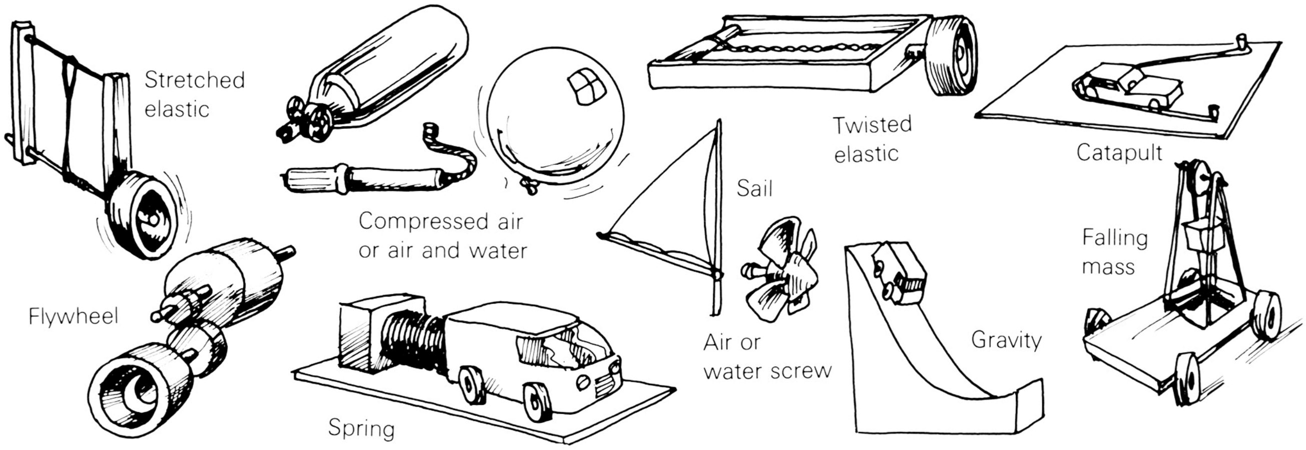
IF vehicle approaches

Go to nearest kerb

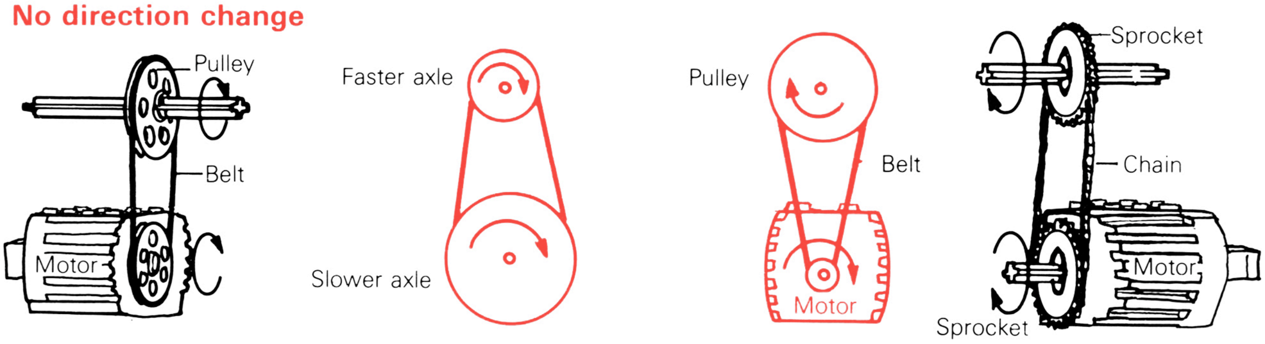
UNTIL other side

Loops or tests which are within other loops or tests are described as *nested*.

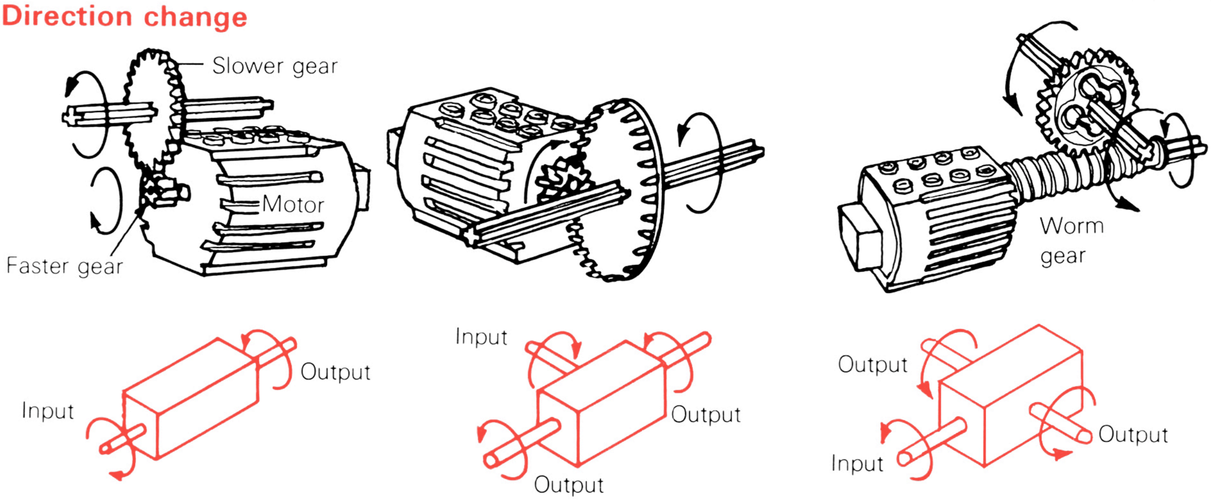




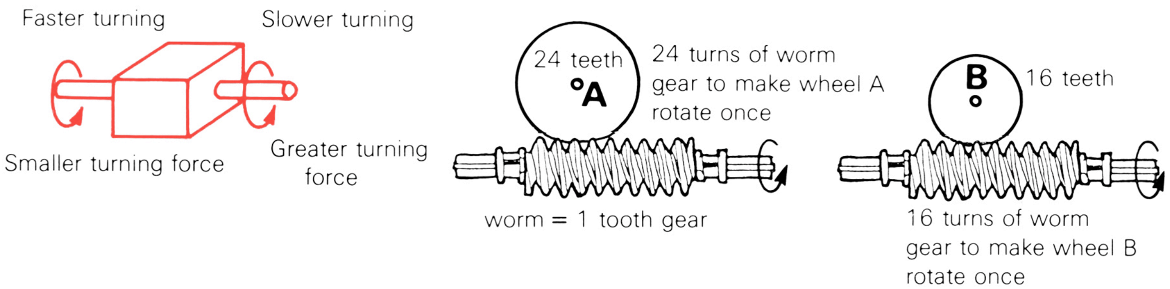
No direction change



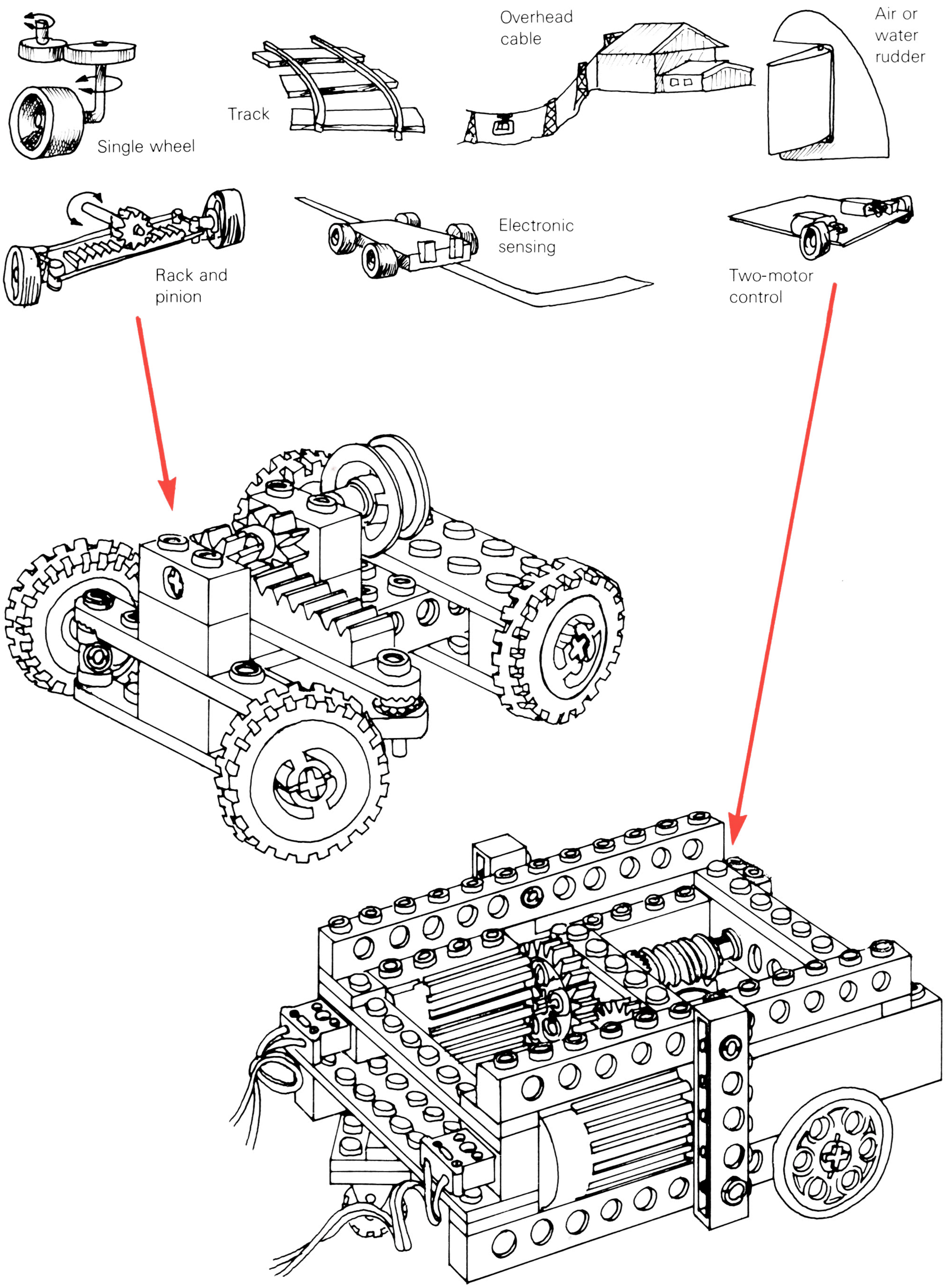
Direction change



Change of speed

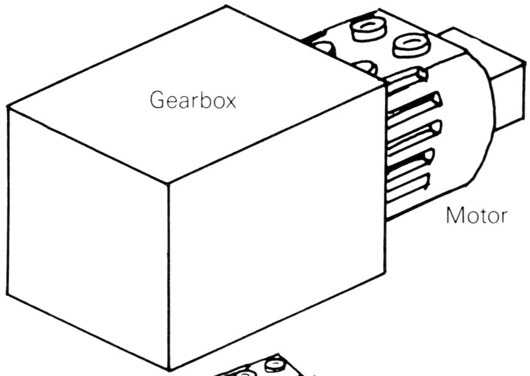


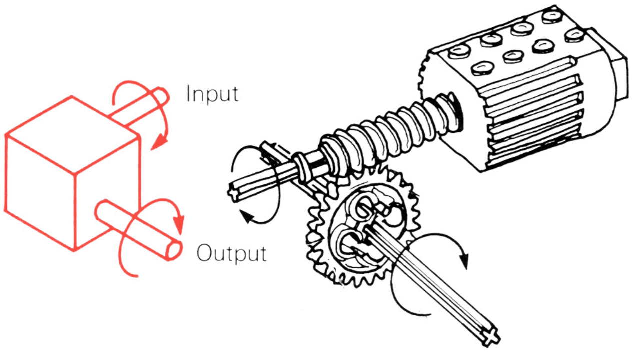
Vehicles — ideas on steering mechanisms

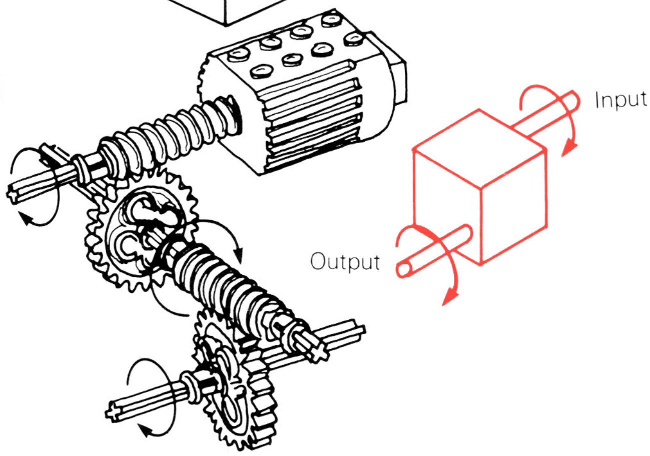


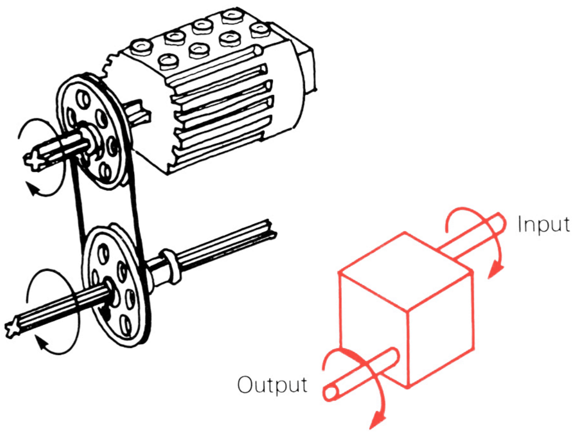
Gearboxes can be used to:

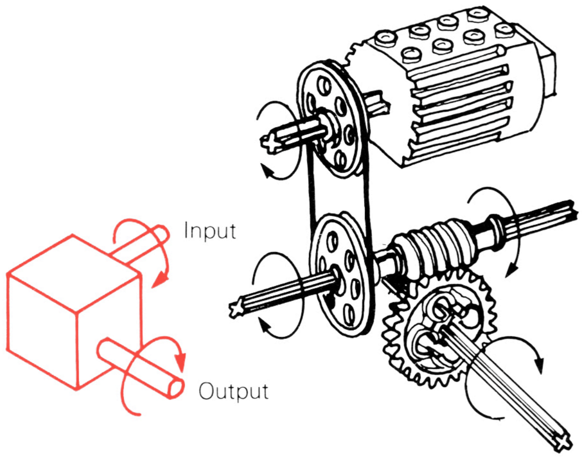
— change speed
— change direction
— change turning force.



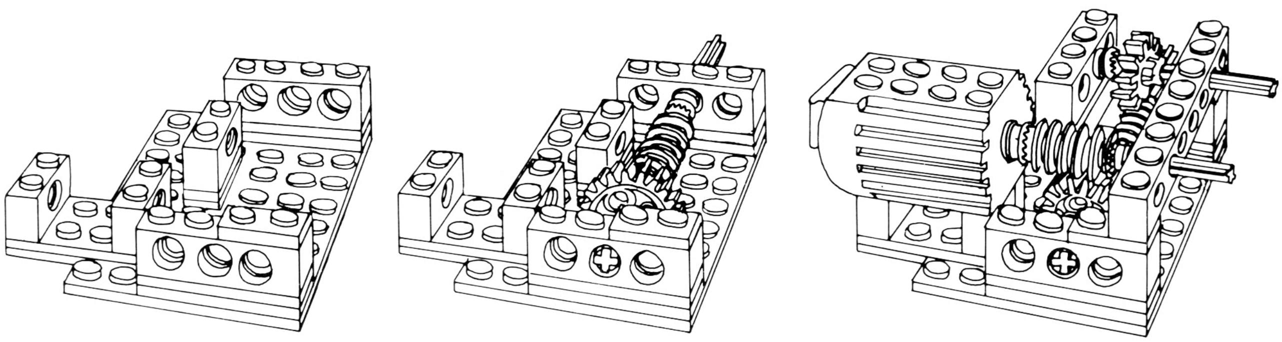


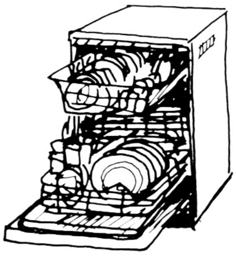


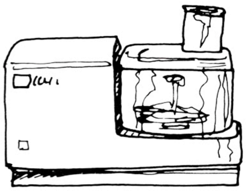


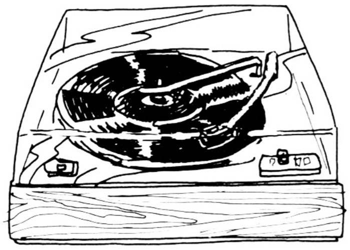


Gearbox for use in Assignment 3

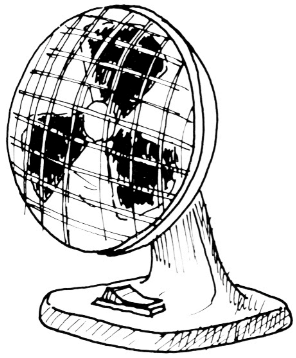
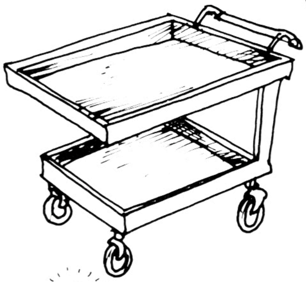
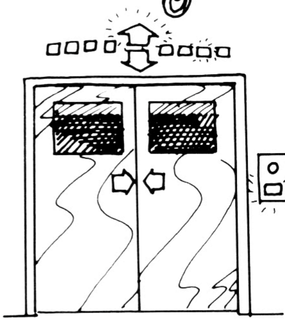
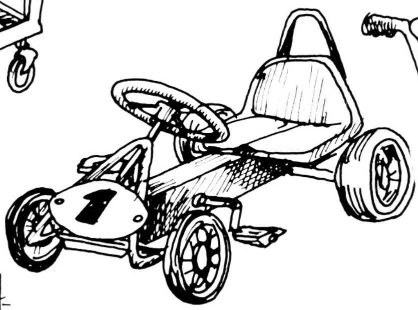
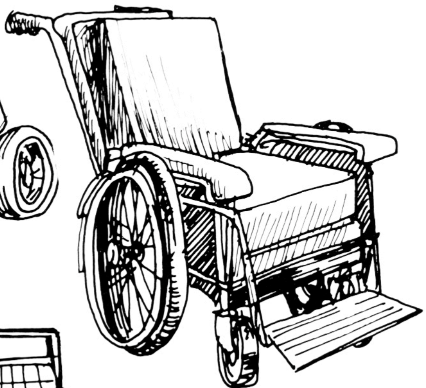
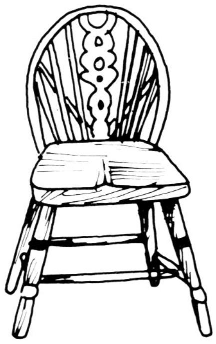


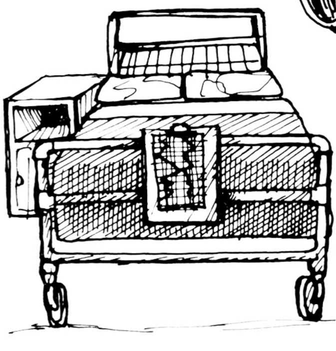
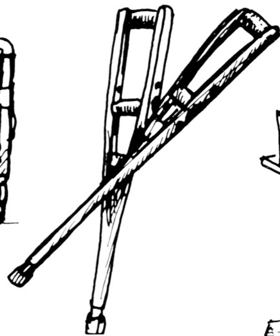


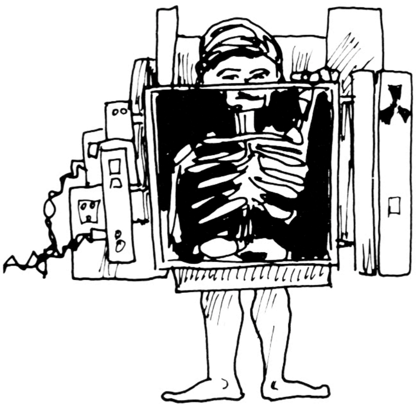




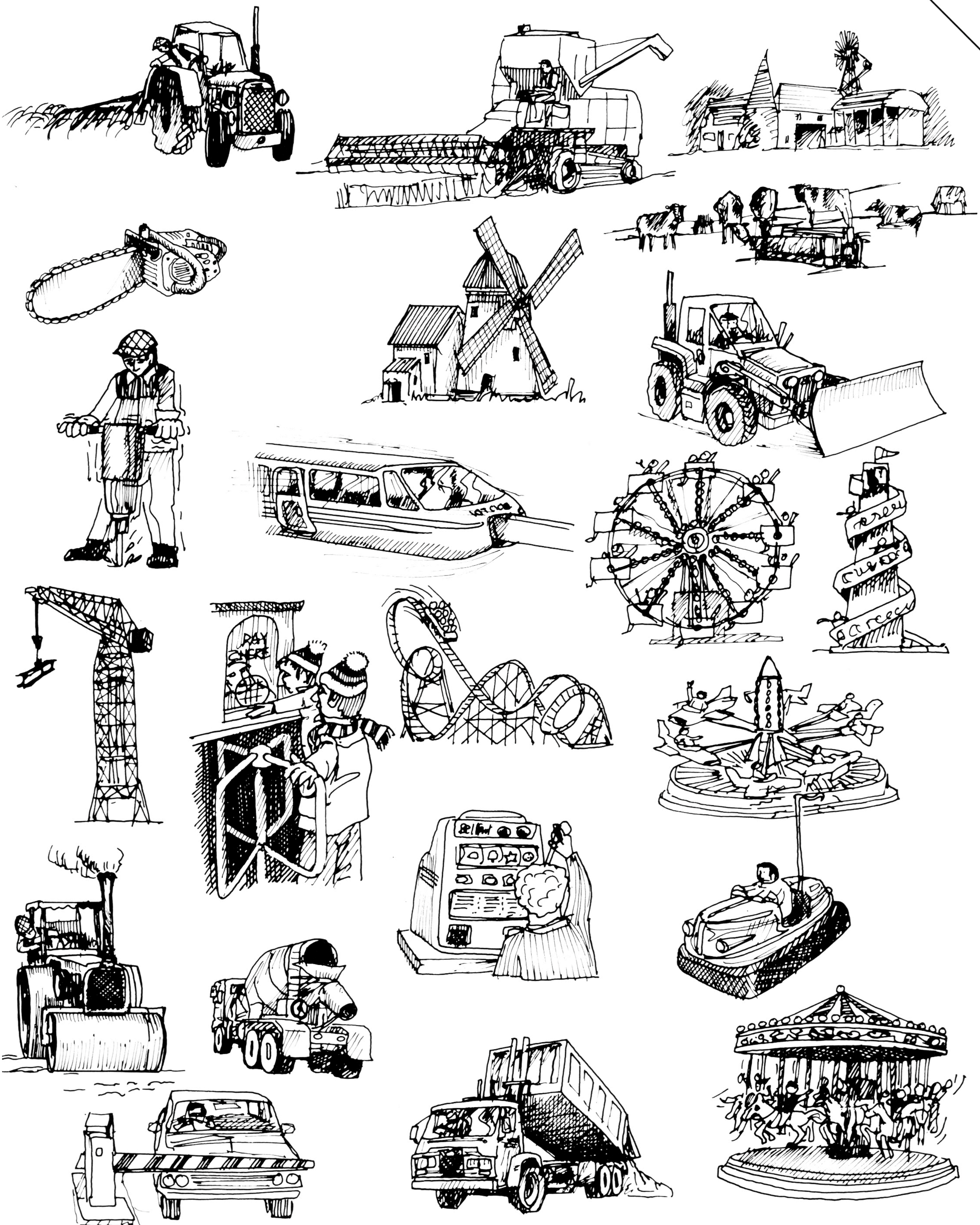


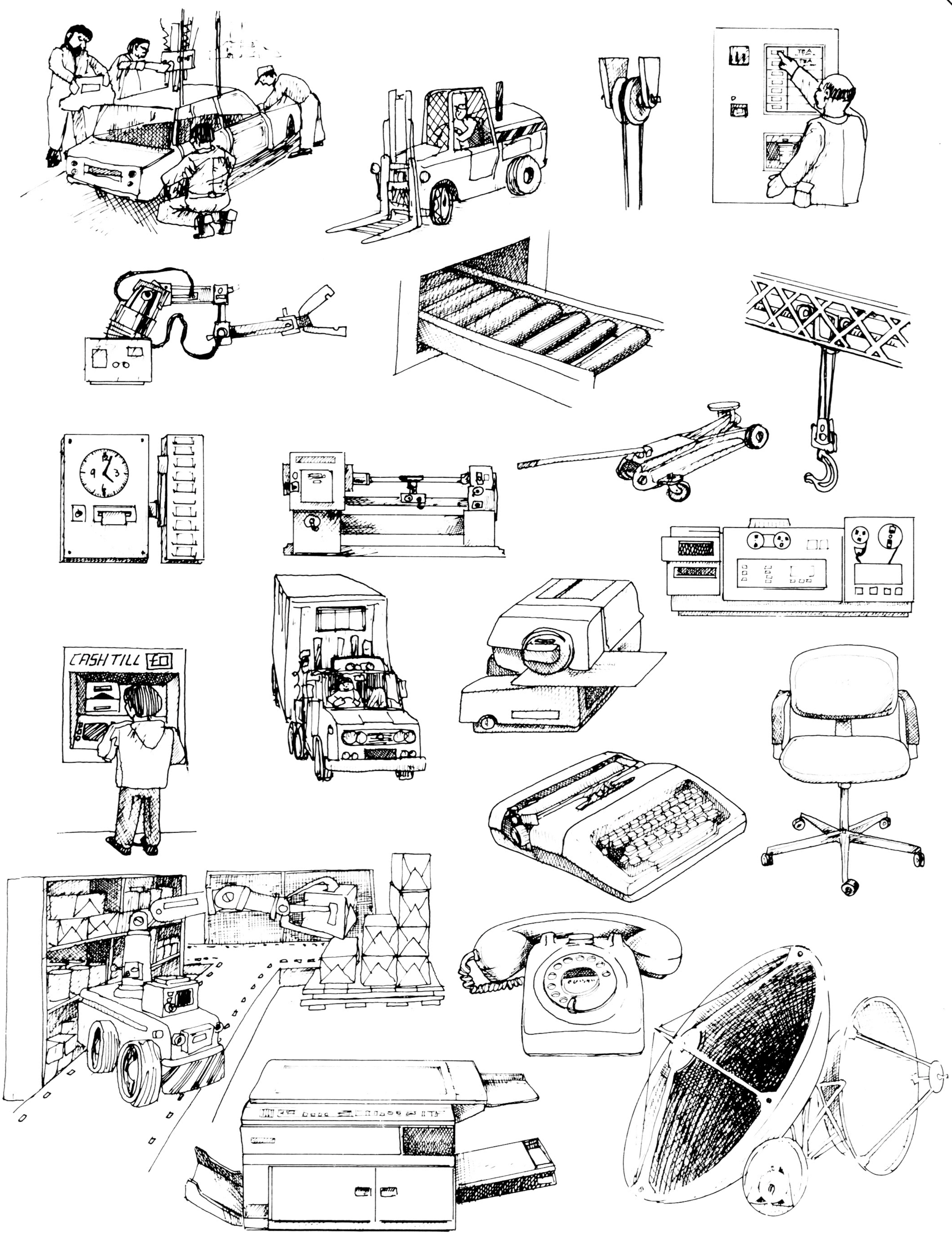










Welcome to SAVA STORES


CASH TILL

Glossary of terms

Algorithm A set or sequence of instructions which describes a solution to a particular problem. **R13 R14**

Any value This term is used in connection with the input bits and describes a state when it doesn't matter whether the input signal from the opto-sensor is ***on*** or ***off***. This only applies when the keywords `IF`, `UNTIL` or `COUNT` are being used to test an input signal on a line. If both input bits are being tested, both cannot be ***any value***: this is an error. If pulses on one input time are being counted the other input line must be set to any value. **R11**

Assignment This is the basic unit of coursework presented in this resource pack. There are ten assignments, **A0 (a–c)** to **A9 (a–j)**. Most have more than one part.

Autoboot A term which is used to describe the automatic loading and running of a program from a disk. To autoboot LEGO *Lines* you simply insert the disk into the drive and press `SHIFT BREAK`. **R10**

Bit Short for *binary digit*. Computers act on signals held in the memory. These signals are combinations of on and off electronic pulses (binary 1 and 0). Because the interface is controlled by a pattern of bits, the connections on the interface itself are referred to as the *input bits* (6 and 7) and the *output bits* (0–5). **R8 R9**

Closed-loop control In a closed loop, the instructions will be repeated until a new condition is found, at which point the program continues. This condition is tested for either at the start or the finish of the loop, and always involves testing an input signal. **R12**

Commands Software commands enable you to control what the computer does. The autoboot command is an example, as is pressing `CTRL f8` to load a program. **R11**

Computer system All the different pieces of computer equipment (monitor, printer, disk drive, cables and leads as well as the computer itself) go to make up the computer system. **R7**

Condition In LEGO *Lines*, this term describes the pattern of input signals being tested. The pattern you set on the screen is the condition and this is matched against what is happening on the interface. If they are the same, the condition is met and the `IF` or `UNTIL` command is then executed. **R11**

Criteria A list of statements of what the solution to a problem must achieve if it is to be called sucessful. A successful vanilla ice-cream should be white, creamy, sweet and, of course, taste of vanilla. These are all ***criteria***. **R1**

Cursor block The cursor block is the oblong which appears on screen to indicate where the next character will appear when it is typed. It is also used to show which line is being performed when a LEGO *Lines* program is being run. **R10**

Drive mechanism This is the means by which the energy from the motor is transferred to the axles, pulleys or gears of a mechanical device. **R16**

Feedback The process by which a controlling system, such as a computer, obtains necessary information about the device it is controlling. **R12**

Gearbox An arrangement of gears which is linked to a motor in order to change the speed, direction or turning force of the output shaft. **R18**

Hard copy Anything which ends up on paper (printout, or just program sheets) which can therefore be saved for future reference, unlike information appearing on screen which is lost. **R10**

Hardware The physical objects making up the system. Printers, monitors, disk drives and so on, are all items of ***hardware***. **R7**

Human senses The five senses (sight, hearing, touch, taste and smell) which provide information which is fed back to the controlling system — the brain — which then acts on this information. **R12**

Indicator lights The red and green lights on the interface. The green (input bits 6 and 7) lights indicate the state of the opto-sensor while the red (output) lights show power being sent to the output bits 0–5. **R7–R9**

Inertia This term refers to the difficulty experienced in making objects change their state; used here in the sense of getting things to move. When you switch a LEGO motor on, there is a very slight delay in moving the other parts attached to it. This is due to *inertia*, which must be overcome. **R12**

Input bit pattern Connections through which the computer system receives information from the opto-sensors. **R9**

Interface Used to describe the place where two things meet. The term is normally used to refer to a piece of equipment, such as the LEGO interface where signals from the computer meet signals from the outside world. **R7–R9**

Label In LEGO *Lines*, a label is a word which is entered in the left hand part of the line to describe the action which is taking place (it always appears in lower case). **R11**

LEGO *Lines* A set of specially designed programs to control devices attached to a computer. **R11–R12**

Loop An instruction or a group of instructions which is repeated a set number of times (an open loop) or until a particular condition is met (a closed loop). **R13–R14**

Manual control Where devices are controlled by human beings pressing switches, they are said to be ***manually controlled***. The LEGO manual controller enables you to control models easily. **R6**

Nested loops/tests Loops or tests which are within other loops or tests are described as being *nested*. **R14**

Open loop control Here, the controlling device does not seek any feedback to decide whether to continue the sequence of instructions. **R12**

Opto-sensor brick The LEGO opto-sensor brick contains a sensor which reacts to changes in the level of light which falls upon it. When a change is detected, an electronic signal is sent to the interface. **R9 R12**

Output devices In the LEGO materials, these are the light bricks and motors which may be connected to the manual controller or the output bits of the interface. **R8**

Output bit pattern Connections through which the computer system directs power, to control devices attached to it. **R8**

Printout A copy of any LEGO *Lines* program may be obtained on a printer (assuming you have one connected to the computer!) by pressing `CTRL f7`. **R7 R10**

Problem-solving process A process, made up of a series of stages, which enables you to approach, and then come to a solution for any type of problem. Here, we are concerned with technological problems. **R1–R5**

Program A sequence of instructions which, when executed in the proper order, enables a computer to solve a problem. Using LEGO *Lines*, programs may be written to control devices. These programs can be developed and recorded on the program sheets designed for this purpose. **R11 R13**

Reserved words These are the keywords which appear in capital letters when you enter them in the left hand part of the program line. Unlike labels, which merely describe what is happening, these keywords make things happen and are very important in writing efficient programs. **R11**

Simulation When you want to test something, it can be very expensive to build it first and then test it — what happens if it is wrong? (Imagine testing a new aeroplane!) Instead you build a model and test this. Only when you are certain that the real thing is worth building do you go ahead. This is called simulation.

Software This is a set of instructions which makes a computer work. LEGO *Lines* (the program, not the disk it comes on) is an example of *software*. **R11**

Structured flow diagram (*sfd*) A diagram which illustrates the way in which the instructions making up the program solve the problem. **R14**

System Any combination of parts which go together to accomplish a goal. The drinks dispenser in Assignment 5 is a system made up of sub-systems, so is the computer itself. **R7**

Value The right hand part of the program line is where you enter values. These are of two kinds. Time values (in seconds and tenths of seconds) control how long the instruction on that line will be performed for before the program moves on. Number values (whole numbers only) control how many times a loop is repeated or how many input signals are to be counted by the `COUNT` keyword. **R10 R11**

LEGO UK Limited
EDUCATIONAL DIVISION
RUTHIN ROAD WREXHAM CLWYD LL13 7TQ

Designed and produced by Tecmedia Limited Granby Street Loughborough LE11 3DU