1455 Technic

LEGO

Programmable Systems

Teacher's materials

LEGO UK Ltd
Educational Division
2 Ruthin Road
Wrexham
LL13 7TQ

In order to maintain the highest level of support for users of Programmable Systems, it is extremely important that purchasers become registered. Support for the existing pack and news of future developments can then be guaranteed.

This is your customer reference number:

A 005402

Include this in all correspondence with LEGO UK Ltd, to ensure that your enquiry is dealt with as efficiently as possible.

Programmable Systems

Please register me as a user of Programmable Systems

Name ______________ Position ______________

Institution ______________

Address ______________

____

Postcode ______________ Tel ______________

Were the materials in satisfactory condition on arrival?

□ Yes □ No

Customer reference

A 005402

Detach this card and return it to the address overleaf.

No stamp is required if this is posted in the UK.

This portion may be discarded.

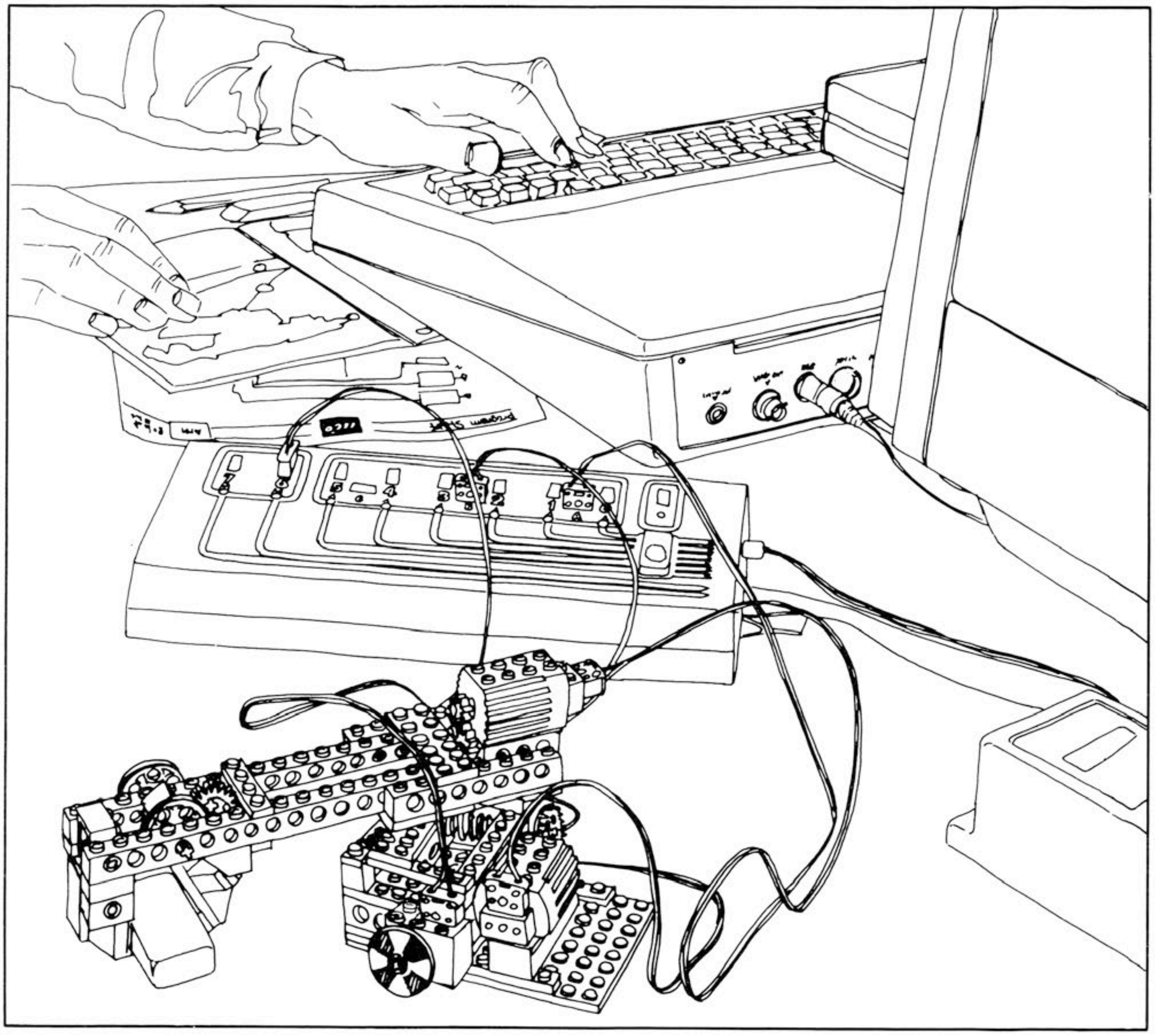
BUSINESS REPLY SERVICE
Licence No. WX. 23

1

LEGO UK LIMITED
RUTHIN ROAD
WREXHAM
CLWYD
LL13 7TQ

Teacher's materials LEGO Programmable Systems

Programmable Systems



Programmable Systems was developed as a result of a collaborative project between The LEGO Group, The Microelectronics Education Programme and British School Technology — Trent.

The development team

Alan Anov

Mike Bostock
Anthony Lucas
Mike Sharpe
Bob Windsor

Andrew Cooper
Alan Giles
Steve Rogers

Many people have made valuable contributions to the educational discussions which have shaped the course of this development and particular thanks are owed to the following:

Graham Bevis, MEP
Barry Evans, Morecombe High School, Lancs
Richard Fothergill, MEP
Roger Holmes, National Centre for School Technology
Mike Ive, HMI
Pauline Jay
Oliver Linton, Harrogate Grammar School, N Yorks
Dr John Martin, Salford University
Geoff Shillito, British School Technology — Trent
Noel Whalley, MEP

Trials teachers

Peter Caunt, Ashfield Comprehensive School, Notts
David Rogers, Ashfield Comprehensive School, Notts
Patricia Gould, Badminton School, Bristol
Ann Lloyd, Badminton School, Bristol
Alan Edwards, Blessed Edward Oldcorne School, Worcester
Bob Giles, Caterham High School, Ilford
Ray Elias, Caterham High School, Ilford
Martin Bassent, Chadwell Heath High School, Romford

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 01 0 Teacher's materials

Contents

Section 1: Introduction

The resource

The materials

Contents *(continued)*

Section 2: Classroom materials

The assignments

Section 3: Resources booklet

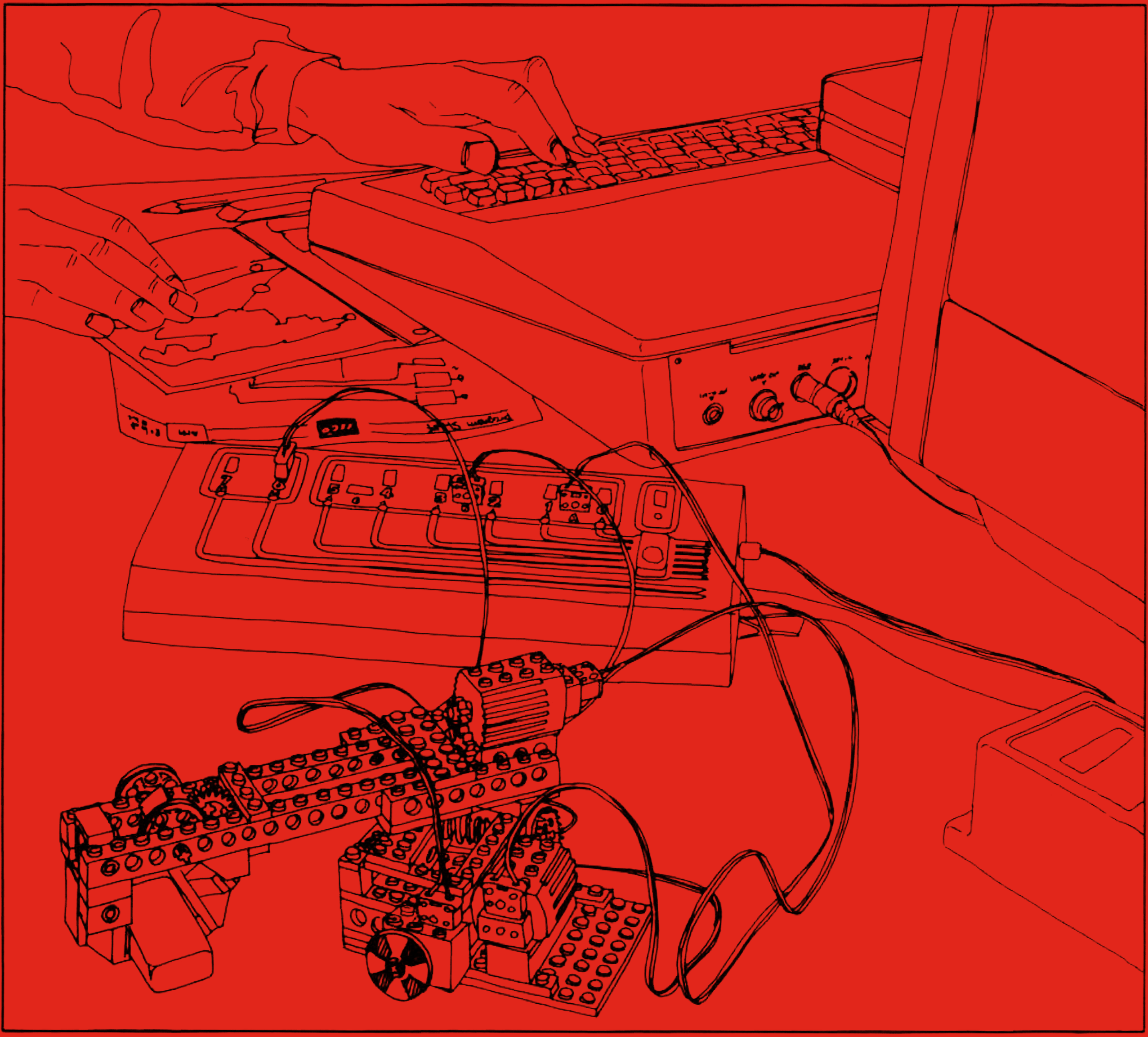
Section 4: Technical reference

Section 5: Duplication masters

Section 1

Introduction

Programmable Systems



A resource of hardware and software which develops understanding of Technology in a challenging way

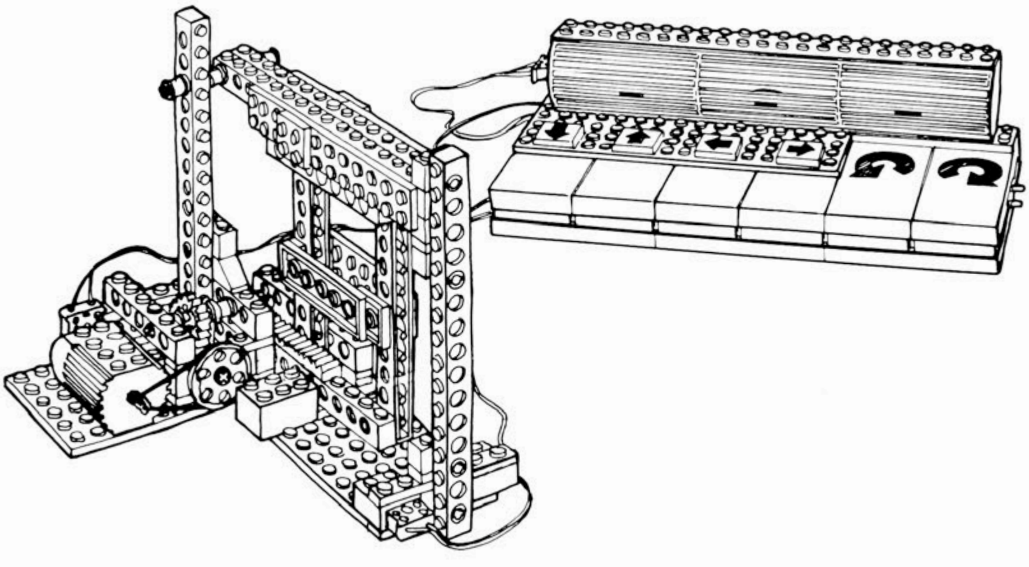
Overview

The aim of this material is to develop a student's competence in solving problems.

Students will be placed in learning situations which pose practical problems in an enjoyable way. This provides a progressive and effective learning strategy enabling students to practise technological skills valued in many walks of life.

Many students do not understand problem-solving nor do they appreciate microprocessor control yet this technology is all around us. These materials provide a framework within which to ***come to terms with*** important technological processes, to ***understand*** them and, most important, to ***apply*** them.

Students need confidence to approach problems and solve them successfully. They learn that by applying a combination of techniques, logic and imaginative ideas they can arrive at a sound and effective solution. Working within a team they learn to work co-operatively and to communicate their ideas. They learn to be critical of what they do through a process of self-assessment. They begin to understand some of the fundamental concepts of technological control.

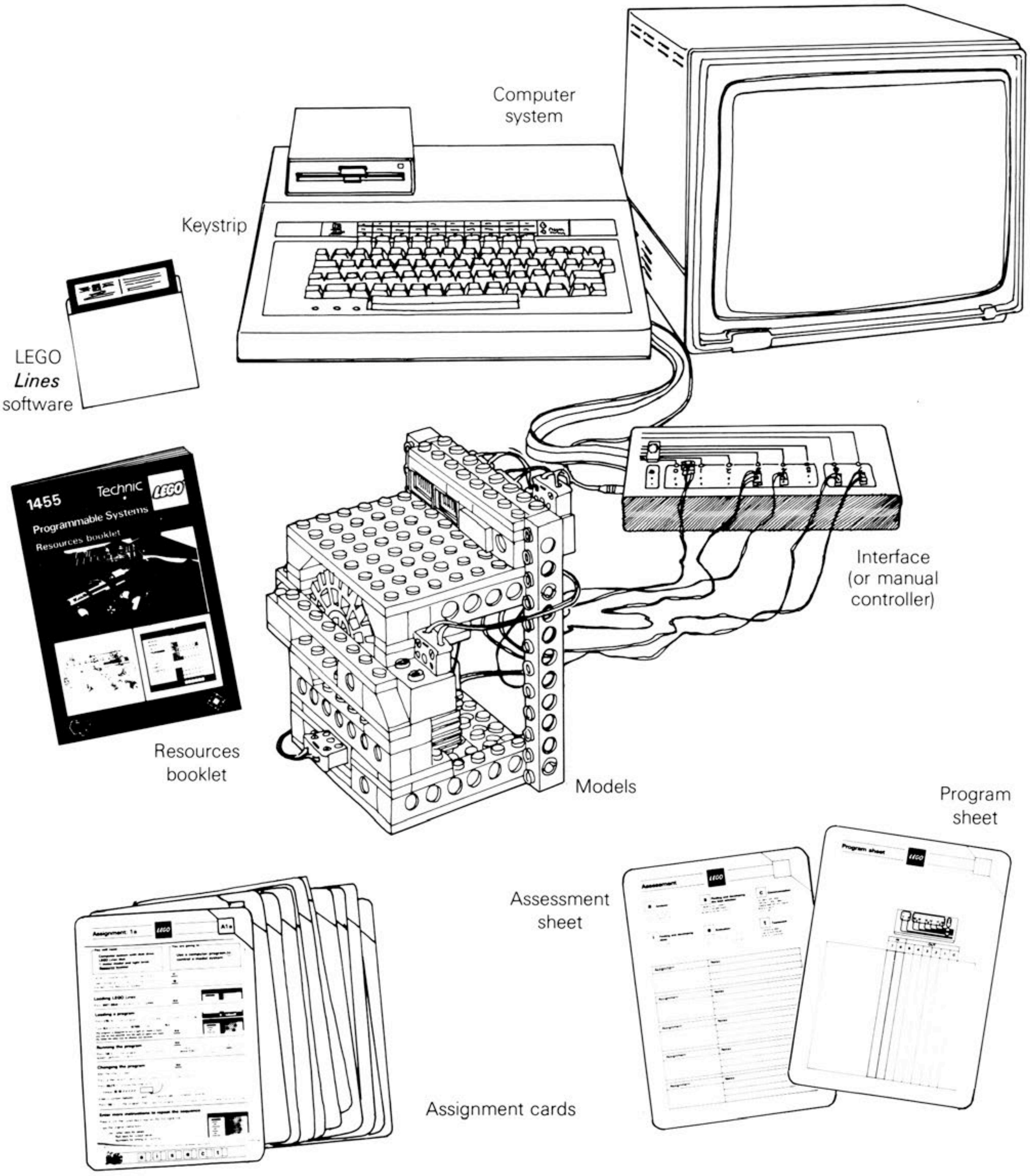


The materials are designed primarily for those in the 11–14 age group but are suitable for older students. They may be used in courses in Technology: Computer Studies, Control Technology, CDT, Science, Integrated Studies and General Studies.

Resources available

The material consist of a Resources booklet, LEGO ***Lines*** for designing control programs, LEGO models made from the 1090 set, and a series of assignments.

The assignments form a recommended course, but the materials as a whole form a substantial resource about technological control.



Introduction to the resource

This resource pack is suited to teachers who are new to the subject of control technology and those who, experienced in the field, are looking for effective materials to work with. Little pre-knowledge has been assumed by the authors, only some familiarity with the LEGO materials and the elements of the computer hardware. It is expected, however, that the teachers will familiarise themselves thoroughly with the materials before using them in the classroom. There is substantial support and introductory material to help teachers achieve this.

Main aims of the materials

- To provide a practical experience in the design of programmable systems using hardware and software in the form of rapidly-adaptable functional units.
- To develop an understanding of the characteristics of microelectronic systems, from binary operations to closed-loop control.
- To study the role of the microcomputer as an example of a programmable, decision-making, switchable device.
- To encourage systems thinking as it applies to the design cycle.
- To promote problem-solving activities in the context of recognisable examples of technological systems.

Controlling devices using a microelectronic system, computer or other, is an area of the curriculum which offers a great many educational opportunities. Students who are able to use this resource gain access to many fundamental technological concepts and processes.

Central to these experiences is the design process:

- Analysing the problem
- Researching alternative (but possible) solutions
- Refining ideas
- Selection of the most appropriate solution
- Building (hardware and software)
- Testing that solution
- Evaluating the success of that solution

and finally,

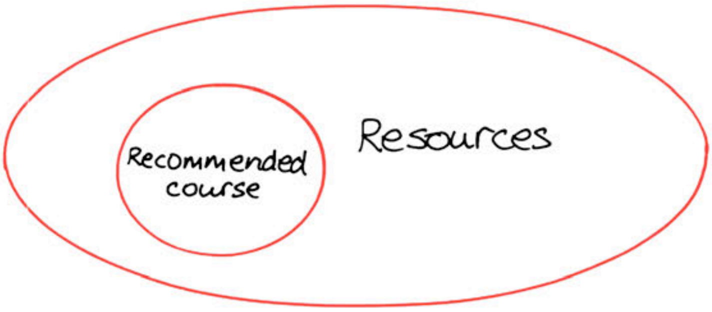
- Communicating the process of arriving at that solution to others.

Prominent also is ***systems thinking***, the capability of seeing the overall purpose, the holistic view, while paying attention to the detail, of understanding the interdependency and nature of the separate elements which interact to form a complete system.

Students involved in the design and control of such systems will play many roles. As they engage in the structural, mechanical and logical aspects they will be engineers, systems analysts and programmers. Each role will contribute its particular insight into some of the processes which characterise the world we live in.

Methodology

The materials as a whole should be regarded as a flexible resource. This has many implications for the way it should be used, from both the teacher's and the students' point of view. The authors have assumed that each teacher will know better than anybody else how to get the best results from the materials within his or her own context.



A recommended set of assignments is provided, together with a ***Resources booklet*** for the students. Once the effective use of this booklet is established, teachers should be released from the task of providing information on the techniques, concepts and practical detail of using the materials. In this way they can devote more attention to the management of the learning process, to the way in which learning takes place within the class, in the groups and for the individual student.

In the assignments, there is substantial cross-referencing to the ***Resources booklet***. This has been designed so that extracting information is as straightforward as possible; a ***Guide to contents*** and a ***Glossary of terms*** complement the actual resource pages on approaching problems, using the equipment and designing control programs. Once students can use this booklet with confidence, they can take far more responsibility for their own learning process. They will work at a pace most suited to their individual needs and should begin to utilise the materials in more constructive and inventive ways.

Such a process of student learning inevitably places a responsibility on teachers to develop their own strategy to exploit these materials. This will be concerned with how to manage the resource to the best advantage. Decisions can only be taken once the range and scope of the materials are understood. To support this understanding, an ***Introduction to the materials*** section has been included (pages 1.8 to 1.10).

As the understanding of the resource develops, teachers may wish to extend its scope, perhaps by adding articles or particular information sheets to the Resources booklet.

Further building materials are available in the form of LEGO Technic 1092 sets, or even the use of other materials with existing LEGO bricks. Ideas for this type of activity are included in the ***Teacher's guides*** to Technic 1 and Technic 2.

The LEGO ***Lines*** program is actually a suite of programs written in BBC BASIC. A series of procedures which enable you to emulate the way in which ***Lines*** controls the LEGO interface are included in the ***Technical reference*** section (pages 4.6 to 4.8) together with a demonstration BASIC program, SAMPLE, which puts them into action. This could form the basis of further work, particularly with the LEGO 1092 set.

Assignments

A number of student assignments are provided in the resource:

Assignment 0 — **A0 (a-c)**
Assignment 1 — **A1 (a-b)**
Assignment 2 — **A2 (a-f)**
Assignment 3 — **A3**
Assignment 4 — **A4 (a-b)**
Assignment 5 — **A5 (a-f)**
Assignment 6 — **A6**
Assignment 7 — **A7**
Assignment 8 — **A8**
Assignment 9 — **A9 (a-j)**

Assignment **A0** is intended as an introduction to the materials for the students and enough time should be allocated to ensure all students attempt the questions found there. All the assignments (**A1, A2, A7** and so on) are designed to be completed in a minimum of a double lesson. There is, however, a lot of material in each sheet and the recommended course could occupy a great deal longer than the time suggested.

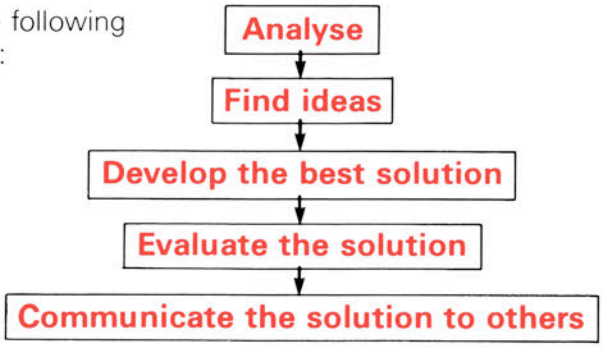
Although the assignments do appear in order, it is up to individual teachers to discover the most appropriate schedule and route through them, as dictated by their own situation.

Assignments have been combined to cater for a situation where there are five computer systems available but more than fifteen students, providing a workable mix of mechanical and computer control activities. These are grouped as follows:

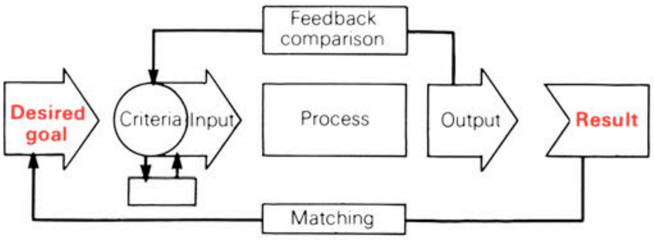
Assignment **A1a** with Assignment **A1b**
Assignment **A2 (a-f)** with Assignment **A3**
Assignment **A4a** with Assignment **A4b**
Assignment **A5 (a-f)** with Assignment **A6**

Assignments **A6**, **A7** and **A8** are central to the course in that, by the time these are encountered (if the assignments are being followed in order), the students will need the structured approach to problem-solving presented to them.

The authors have used the following model for problem-solving:

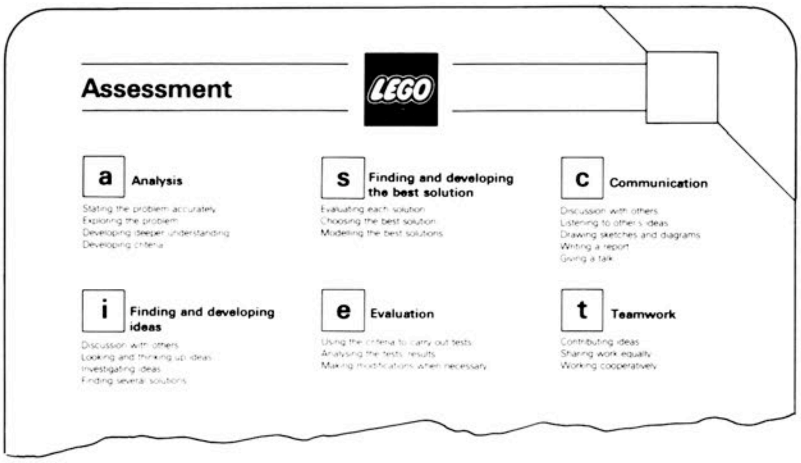


In reality, the technological process is far more complex than this and it is expected that the students will demand more of this simple model as their understanding grows.



Assessment

A process of continuous negotiated assessment is recommended. This is linked to the use of the boxed characters on the Assessment sheet, descriptors of the main elements of the problem-solving model outlined previously (page 1.5).



Assessment

a **Analysis**

Stating the problem accurately
Exploring the problem
Developing deeper understanding
Developing criteria

s **Finding and developing the best solution**

Evaluating each solution
Choosing the best solution
Modelling the best solutions

c **Communication**

Discussion with others
Listening to other's ideas
Drawing sketches and diagrams
Writing a report
Giving a talk

i **Finding and developing ideas**

Discussion with others
Looking and thinking up ideas
Investigating ideas
Finding several solutions

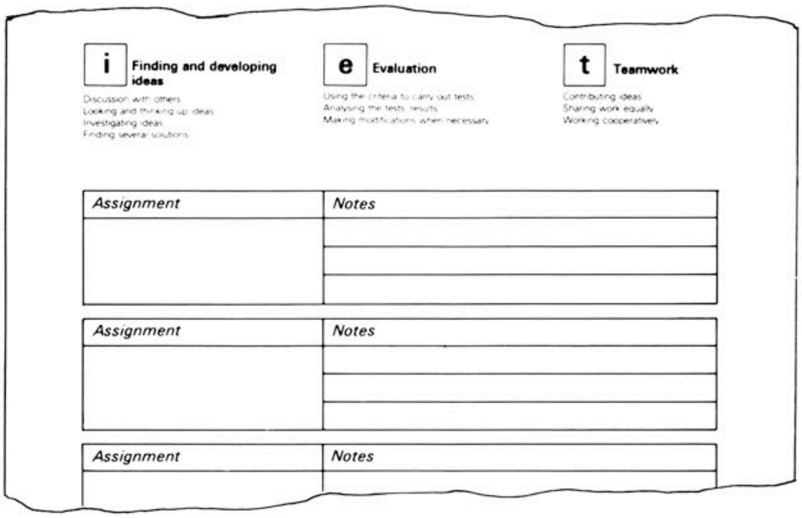
e **Evaluation**

Using the criteria to carry out tests
Analysing the tests results
Making modifications when necessary

t **Teamwork**

Contributing ideas
Sharing work equally
Working cooperatively

The boxed characters are repeated on the bottom of each assignment card (except **A0**). They are intended as a reminder to the students, providing a framework in which to assess their own progress. The teacher is also expected to assess group and individual performance and it is anticipated that, at some agreed point, students and teacher will get together to negotiate the overall assessment. This process has to be facilitated by the teacher and should occur in whichever way is most manageable in the particular context. The Assessment sheet provides space for students to write in a description of how they have performed on the assignment concerned.



i **Finding and developing ideas**

Discussion with others
Looking and thinking up ideas
Investigating ideas
Finding several solutions

e **Evaluation**

Using the criteria to carry out tests
Analysing the tests results
Making modifications when necessary

t **Teamwork**

Contributing ideas
Sharing work equally
Working cooperatively

Assignment	*Notes*

Assignment	*Notes*

Assignment	*Notes*

Students will soon appreciate that not all skills are required for all activities. The teacher will need to support this understanding and time should be made available during the course of each assignment to discuss whether or not a skill has been used. A formative mechanism of assessment is, therefore, being employed.

Management of materials — some questions and answers

— *What is the best physical location for the course (assuming a group of 30 students)?*

A shared computer area where there is desktop and demonstration space.

— *What is the best group size?*

Probably only two, but most likely three students will be the working group size, especially where the class size approaches thirty students.

— *How will I, a teacher, become sufficiently familiar with the materials?*

By working carefully and slowly through the ***Introduction to the materials*** section and the ***hands-on practice***.

— *What is the best way of organising the recommended course?*

Once you are familiar with the materials, you will use them as you think fit. The likelihood is that, first time round, you will follow the recommended course as presented in the Assignment cards. As you become more confident in their use, you will re-order and add new materials of your own.

— *What kind of computer system is required?*

Any BBC microcomputer (including the new Master Series) with a 5.25 inch 40- or 80-track disk drive. The disks (Master and Sample) supplied in the pack are 40-track only.

— *What is the best way of organising time?*

Each assignment is designed to take the equivalent double period but this, of course, is only a guide. You may find that your students take half the time, or twice as long. It is, however, recommended that you look to increase the time available for each assignment, to exploit the resource materials and to organise the assessment scheme.

— *How can the recommended assessment mechanism be employed?*

During the schools trials, most students responded very positively to this form of assessment. It has been shown that they are prepared to discuss what has been achieved in each assignment. It is then up to the teacher to mould this willingness to respond into a formative assessment process. A framework of descriptors is provided to work with, and the teacher may wish to incorporate more of these. Alternatively, a more traditional form of assessment may be followed, each assignment having clearly defined objectives which can be tested.

— *How should the students be introduced to the materials?*

It is recommended that all students work carefully and slowly through Assessment **A0 (a-c)** which has been specially written for this purpose.

— *Where do the materials lead?*

They generate a lot of enthusiasm on the part of the students. The resources may be used to explore other models or may be extended by other LEGO Technic sets, particularly 1092 or Technic 1 and 2.

The software may be extended by exploring the BASIC statements which generate it. (A full list of control procedures, emulating those in LEGO ***Lines***, appears in the ***Technical reference*** section, pages 4.6 to 4.8).

The method and content lead naturally to further courses in control technology and other cross-curricula activities.

Introduction to the materials

The success of these materials depends to a large extent on the individual teacher's familiarity and confidence with them.

This is an introductory section which teachers work through prior to presenting the materials in the classroom.

It takes you through:

- the hardware — the LEGO sets and the computer system
- the software — LEGO *Lines* and creating working masters
- the teacher's guide to the assignments
- the assignments
- the students' resources
- some hands-on practice.

Hardware

LEGO sets

A guide to the LEGO sets, entitled ***Directions for use***, can be found in the carton in which they arrive. This is accompanied by the series of construction guides which give an idea of the models which can be built.

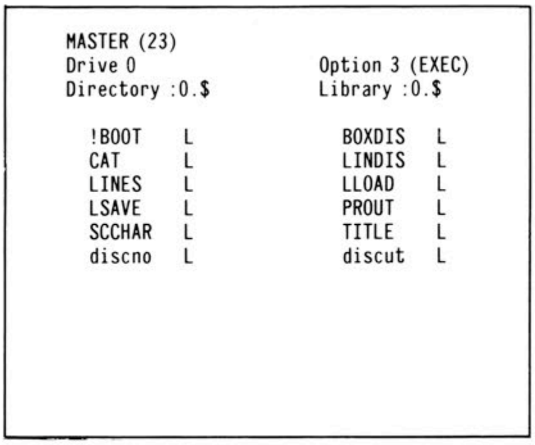
Computer system

LEGO ***Lines*** requires the computer you are using to be running standard BBC BASIC (see the ***Technical reference*** section, page 4.2). You should also ensure that you are familiar with the disk drive you will be using and all the other peripheral devices (monitor, printer and LEGO interface) you will use, particularly the way in which you connect them to the computer (refer to the Resources booklet, **R7**, if necessary). It is recommended that you check that all the devices and leads work as you expect them to.

Software

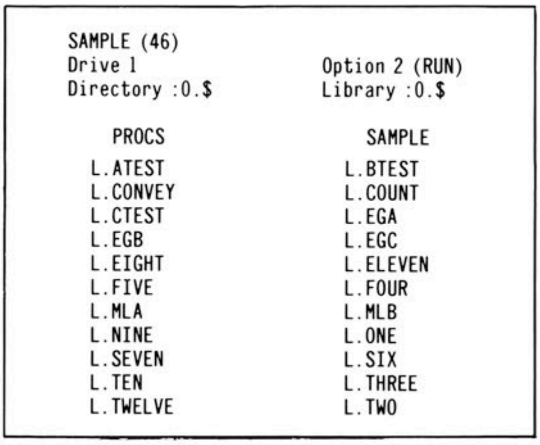
LEGO Lines

The software is supplied on two master disks.



```
MASTER (23)
Drive 0                  Option 3 (EXEC)
Directory :0.$           Library :0.$

  !BOOT    L               BOXDIS   L
  CAT      L               LINDIS   L
  LINES    L               LLOAD    L
  LSAVE    L               PROUT    L
  SCCHAR   L               TITLE    L
  discno   L               discut   L
```

Directory of master program disk



```
SAMPLE (46)
Drive 1                  Option 2 (RUN)
Directory :0.$           Library :0.$

    PROCS                    SAMPLE

  L.ATEST                  L.BTEST
  L.CONVEY                 L.COUNT
  L.CTEST                  L.EGA
  L.EGB                    L.EGC
  L.EIGHT                  L.ELEVEN
  L.FIVE                   L.FOUR
  L.MLA                    L.MLB
  L.NINE                   L.ONE
  L.SEVEN                  L.SIX
  L.TEN                    L.THREE
  L.TWELVE                 L.TWO
```

Directory of sample program disk

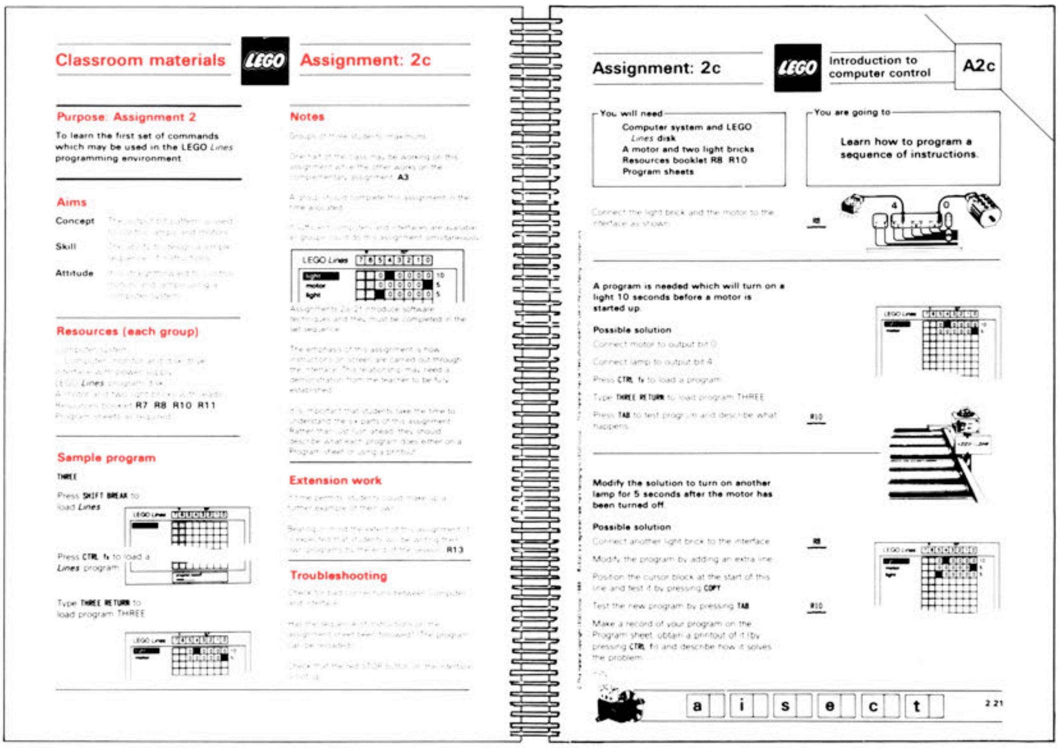
As soon as you receive this pack you should make a backup copy of both these disks (see *Technical reference* section, page 4.4). These backups are then your working masters. The originals should be stored in a safe place. LEGO *Lines* is actually a suite of programs, as you can see on the *Master program disk*, and the functions of the component programs are described in the *Technical reference* section (page 4.1). The programs on the *Sample programs disk* are described in more detail in the *LEGO Lines: sample programs* section (pages 1.30 to 1.40).

When groups are working on assignments involving LEGO *Lines*, they will require a working master disk, which will be a copy of your working master. In addition to the LEGO *Lines* software, they should have the appropriate sample program (indicated on the teacher's commentary page accompanying each assignment) loaded on the same disk. This will involve housekeeping to ensure that the working masters are kept up to date with the assignments currently being used. A detailed description of making backup copies, maintaining working masters and housekeeping procedures is given in the *Technical reference* section (pages 4.4 to 4.5). There should always be room on this working disk for the group to save its own programs.

Teacher's guide to the assignments

The assignments are to be found in the *Classroom materials* section which follows. The assignment cards are duplicated here, faced by a commentary designed for the teacher. Where an assignment is split into many parts, Assignment **A2 (a-f)** for example, this information is duplicated opposite each part, with minor differences relevant only to that part.

Each commentary states the purpose of that assignment, the concepts, skills and attitudes it promotes and lists of equipment and sample programs required. Notes and pointers on the assignment follow, together with ideas for extension work and a troubleshooting section. Often, the assignment itself requires the modification of one of the sample programs in order to solve a particular problem. Where this is the case, the modified program is given in the teacher's commentary for reference.



Classroom materials LEGO Assignment: 2c

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment

Aims

Concept The output bit pattern is used to control lamps and motors

Skill The ability to program a simple sequence of instructions

Attitude It is straightforward to control motors and lamps using a computer system

Resources (each group)

Computer system
Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Resources booklet R7 R8 R10 R11
Program sheets as required

Sample program

THREE

Press SHIFT BREAK to load *Lines*

Press CTRL f1 to load a *Lines* program

Type THREE RETURN to load program THREE

Notes

Groups of three students maximum

One half of the class may be working on this assignment while the other works on the complementary assignment A3

A group should complete this assignment in the time allocated

If sufficient computers and interfaces are available all groups could do this assignment simultaneously

Assignments 2a-2f introduce software techniques and they must be completed in the set sequence

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead they should describe what each program does either on a Program sheet or using a printout

Extension work

If time permits students could make up a further example of their own

Bearing in mind the extent of this assignment it is expected that students will be writing their own programs by the end of the session R13

Troubleshooting

Check for bad connections between computer and interface

Has the sequence of instructions on the assignment sheet been followed? The program can be reloaded

Check that the red STOP button on the interface [illegible]

Assignment: 2c LEGO Introduction to computer control A2c

You will need
Computer system and LEGO *Lines* disk
A motor and two light bricks
Resources booklet R8 R10
Program sheets

You are going to
Learn how to program a sequence of instructions.

Connect the light brick and the motor to the interface as shown R8

A program is needed which will turn on a light 10 seconds before a motor is started up.

Possible solution

Connect motor to output bit 0

Connect lamp to output bit 4

Press CTRL f1 to load a program

Type THREE RETURN to load program THREE

Press TAB to test program and describe what happens R10

Modify the solution to turn on another lamp for 5 seconds after the motor has been turned off.

Possible solution

Connect another light brick to the interface R8

Modify the program by adding an extra line

Position the cursor block at the start of this line and test it by pressing COPY

Test the new program by pressing TAB R10

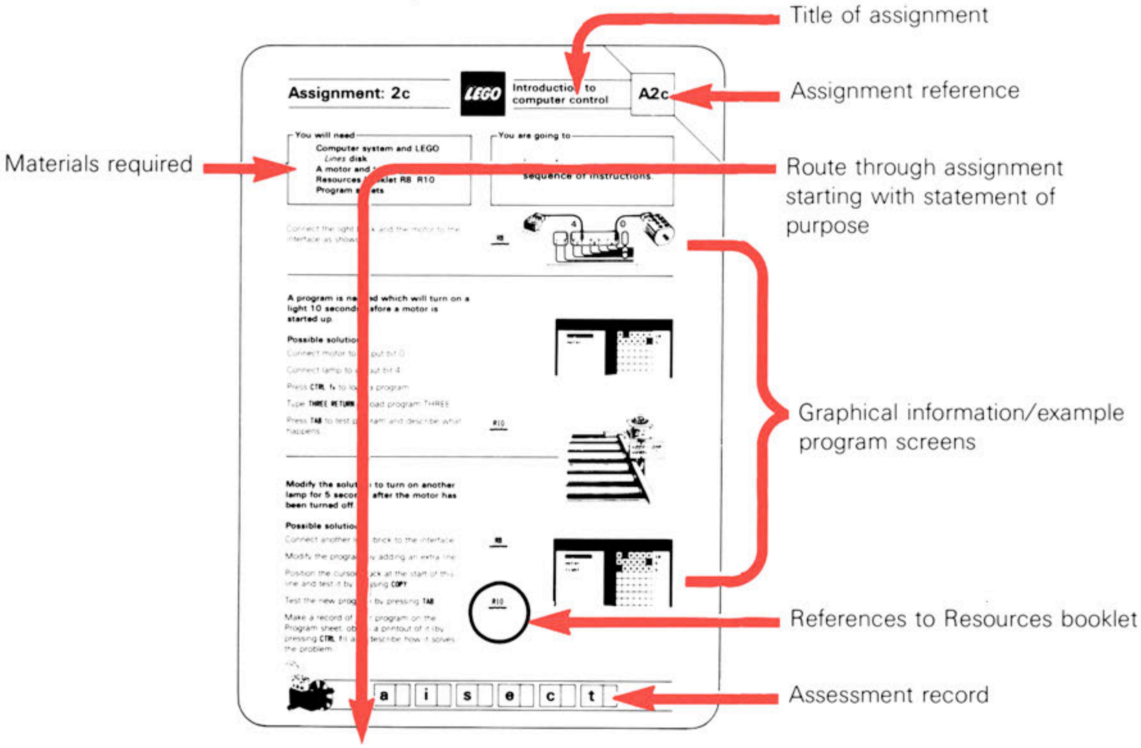
Make a record of your program on the Program sheet, obtain a printout of it (by pressing CTRL f1) and describe how it solves the problem

a i s e c t

2.21

The assignments

The layout of each card is intended to make the assignments as easy to follow as possible. Instructions for the student are clearly visible, together with the supporting information and reference to the ***Resources booklet***. The aim of the assignment (or part of an assignment) and the equipment needed to achieve it is clearly stated.


Title of assignment
Assignment reference
Materials required
Route through assignment starting with statement of purpose
Graphical information/example program screens
References to Resources booklet
Assessment record

Assignment: 2c | Introductio to computer control | **A2c**

You will need
- Computer system and LEGO *Lines* disk
- A motor and
- Resources klet R8 R10
- Program ets

You are going to

sequence of instructions.

Connect the light k and the motor to the interface as show R8

A program is ne d which will turn on a light 10 second efore a motor is started up.

Possible solutio

Connect motor to put bit 0

Connect lamp to ut bit 4

Press **CTRL f4** to lo a program

Type **THREE RETURN** oad program THREE

Press **TAB** to test p ram and describe what happens. R10

Modify the solut to turn on another lamp for 5 secor after the motor has been turned off.

Possible solutio

Connect another l brick to the interface. R8

Modify the progra y adding an extra line

Position the curso ack at the start of this line and test it by sing **COPY**

Test the new prog by pressing **TAB** R10

Make a record of r program on the Program sheet. ob a printout of it (by pressing **CTRL f1** a describe how it solves the problem.

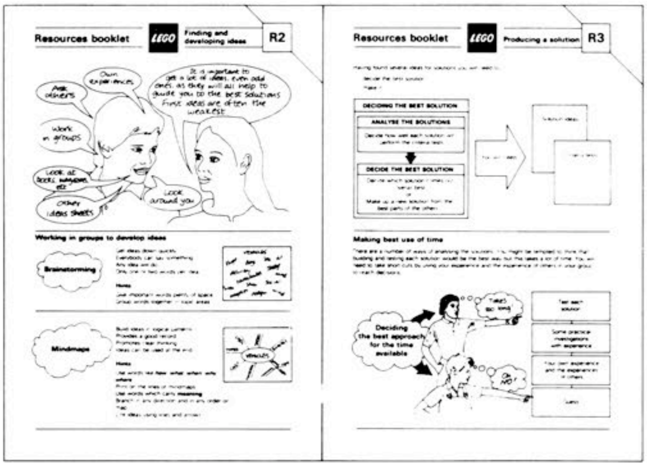
a | i | s | e | c | t

Students' resources booklet

This is an open resource which students can refer to for facts, ideas, techniques and background material. It is organised into three sections:

Organising yourself to solve problems
Using the equipment
Designing control programs using the LEGO materials

It is worth investigating this booklet thoroughly before using the materials in the classroom.


Resources booklet
Finding and developing ideas
R2
Ask others
Own experiences
Work in groups
Look at books, magazines etc
Other ideas sheets
It is important to get a lot of ideas, even odd ones, as they will all help to guide you to the best solutions. First ideas are often the weakest
Look around you
Working in groups to develop ideas
Brainstorming
Mindmaps
Resources booklet
Producing a solution
R3
DECIDING THE BEST SOLUTION
ANALYSE THE SOLUTIONS
DECIDE THE BEST SOLUTION
Making best use of time
Deciding the best approach for the time available
Takes too long
Oh no!

Hands-on practice

This section will familiarise the teacher with the scope of the materials available. It is recommended that the teacher spends time going through these activities in some detail.

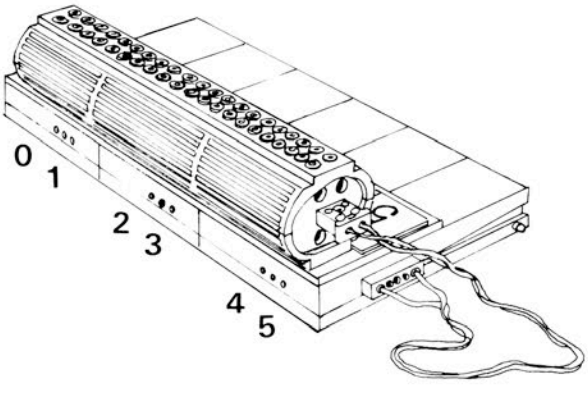
Using the 1090 B construction guide, build the Automatic door. Do not connect the leads to the interface as shown in the guide (the sample programs provided explore the full range of software commands available and they often utilise other connections).

You will first need to copy the sample programs ATEST, BTEST and CTEST from the Sample programs disk to your working disk using the procedures described in the Technical reference section, page 4.5.

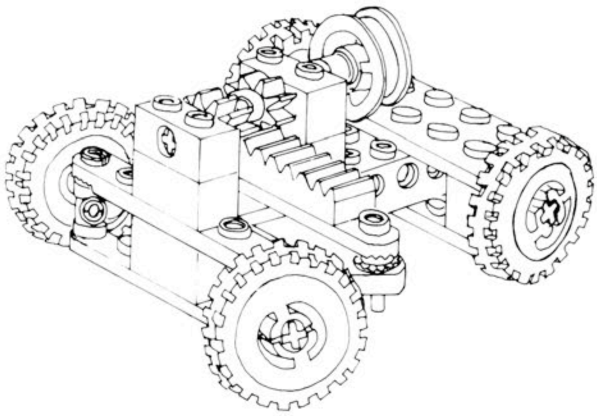
Identify the gearbox in this model, using Resources booklet **R18**.

Build a separate gearbox as suggested by Assignment **A3**.

Using Resources booklet **R6**, work your way through Assignment **A1b** to familiarise yourself with the LEGO manual controller 1039. For this purpose, use the Automatic door you have just built instead of the two-motor model suggested there. Refer to the teacher's commentary as you work your way through.

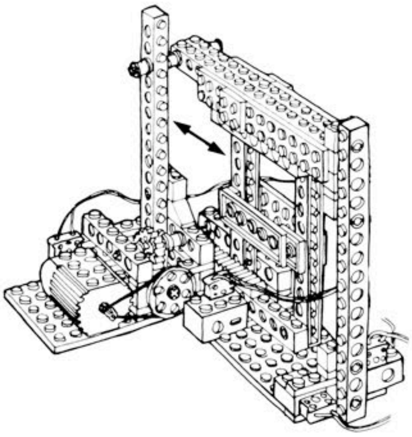


Make a vehicle (as described in Assignment **A6**) and control it using the manual controller. Explore drive and steering mechanisms using Resources booklet **R15** to **R18**.



The materials – hands-on practice

Set up your computer system (keyboard, monitor and disk drive) as usual and insert the LEGO ***Lines*** keystrip under the plastic strip above the red function keys. Connect the interface to the user port on the underside of the computer (this is clearly labelled). Ensure that the interface power supply is connected and switch the system on (**R7**).

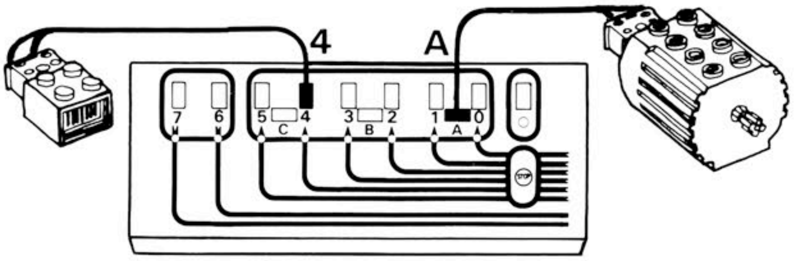


As you switch on, all the indicator lights on the interface panel should come on if the stop button is up. If they fail to do so, you should check connections to the power supply.

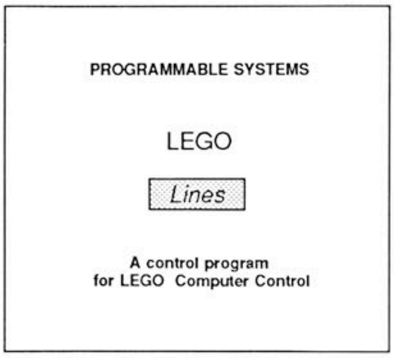
If the button is pressed down, all the lights will go out, with exception of the continuous power output — marked ***4v***.

Use the Automatic door model, starting with the door closed.

Connect the motor and one of the lights to the interface. The light should be connected to the socket marked 4 (output bit 4) and the motor to that marked A (output bits A). Check that the red STOP button is up (**R8**).



Insert the disk into the drive (drive 0 if you have more than one slot). Hold down **SHIFT**, press and release **BREAK** and then release **SHIFT**. This will autoboot (automatically load and run) LEGO ***Lines***. An introductory screen will appear followed by the program screen. (Press the **SPACEBAR** to speed up the appearance of the program screen).

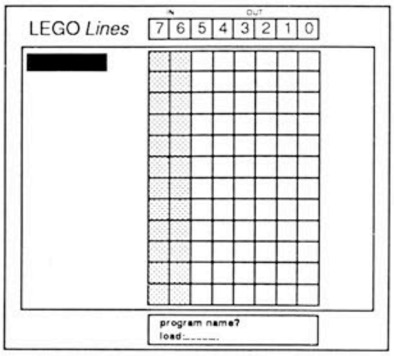

PROGRAMMABLE SYSTEMS

LEGO

Lines

A control program
for LEGO Computer Control

Press f_7 to turn output bit 0 on. Check what happens on the screen. You should find that a **1** has appeared in column 0. Press f_7 again to turn that bit off. Now press f_7 again and then press the **COPY** key. The door should open or it should attempt to close. If neither of these happen then something is not connected correctly.


LEGO *Lines*

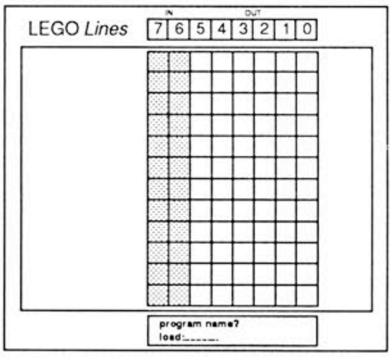
7	6	5	4	3	2	1	0

program name?
load:______

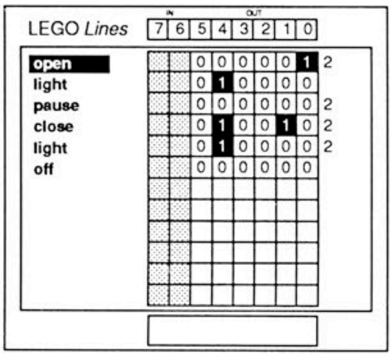
If the model starts with the door closed, and if your instruction attempts to close the door you should reverse the action of the motor by reversing the way the motor lead plugs into either the interface or into the motor itself (**R8**).

Press f_7 to set the output bit 0 to off and press f_6 to set bit 1 to on. Check that this now reverses the direction of the door.

Close the door by an appropriate method and then ensure that bit 0 will open the door when it receives an on message from the computer. By holding down **CTRL** and pressing **f8** load the program called ATEST. You will notice that the indicator block on the bottom of the screen asks you to input the name of the program. Type **ATEST** and press **RETURN**. Wait a few moments while the program loads and then press **TAB** to run it. The program is designed to open the door and light a lamp. The door should pause before it closes and then the program continues and switches off the light. When you run the program again watch the screen and see how, as the program is executed line by line, the highlighted block (the cursor block) appears at the start of the line currently being performed. The lights on the interface panel indicate how the bits are turned on and off by the software.



```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0

             program name?
             load:______
```



```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
open             0 0 0 0 0 1 2
light            0 1 0 0 0 0
pause            0 0 0 0 0 0 2
close            0 1 0 0 1 0 2
light            0 1 0 0 0 0 2
off              0 0 0 0 0 0
```

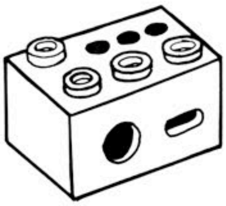
When the program has finished, the computer will beep and the cursor block will return to the start of the program.

To demonstrate the editing facilities press ↓ twice. The cursor block should move down two lines.

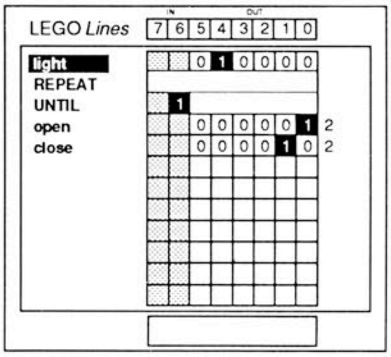
You will see from the key strip that pressing **f9** will delete that line completely from the program. Do this. Observe what happens on the screen and then run that new program again.

Make a record of one of the lines labelled **light**, then delete that line and try to run the program again.

Now find an opto-sensor brick and connect it to the input bit 6 on the interface. What happens to the indicator light on the interface when you slowly cover the opto-sensor with your finger? It will respond to changes in light intensity (**R9**).

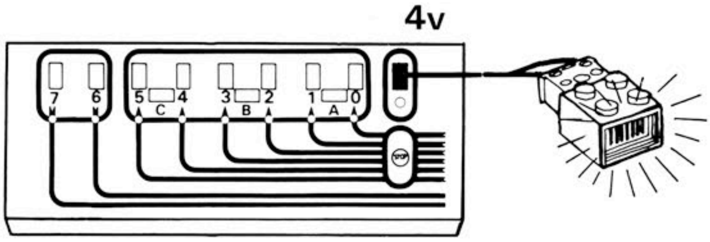


Hold down **CTRL** and press **f8** and load the program BTEST (remember to press **RETURN**). Before you run this program make sure that the indicator light on the interface panel is off (input bit 6 set to 0 — off) by placing your finger on the sensor (or not as the case may be). If you have done this and you press **TAB** to run the program you will observe that it will wait until you make bit 6 on (by covering or uncovering the opto-sensor) before it opens the door and closes it again.



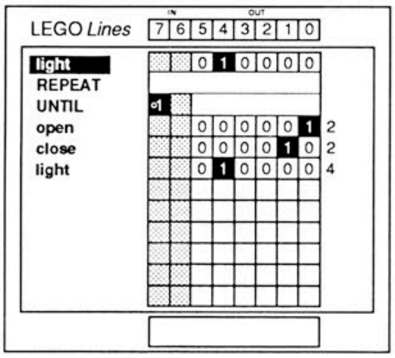
```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
light            0 1 0 0 0 0
REPEAT
UNTIL          1
open             0 0 0 0 0 1 2
close            0 0 0 0 1 0 2
```

When you use the opto-sensor on a model it is usual to use a light brick with it. This is because the state of the brick *floats*, it could be on or off initially. The light brick will force it to one particular state to begin with and you can work from that certainty. Normally the 4v power supply is used to provide a light if this is required. Alternatively, a light may be instructed to come on within the program (**R7, R9**).



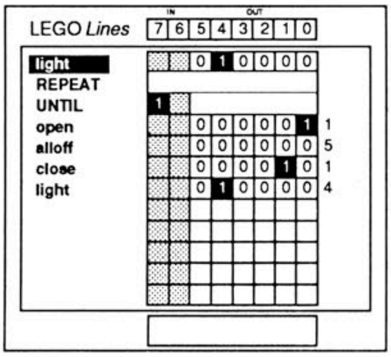
To add a new line at the end of the program arrange for the cursor block to move down to an appropriate position. Type in the letters **l i g h t** and this will appear on the screen in the left-hand side. This is a label. If you then type **3** on the number keys you will enter a value on the screen in the right-hand side indicating that the light should stay on for 3 seconds. The output bit for the light that you have connected to the interface is bit 4. This particular bit should be currently set to 0 — off, and to turn it on you need to press **f₃**. Notice that this corresponds to bit 4 on the keyboard strip. Test this additional line out by positioning the cursor block on that line and pressing **COPY**. This key executes the line where the cursor block is (as displayed in the indicator box at the bottom of the screen). This provides a very useful facility, especially when setting up a model as it is necessary to test the output lines to see in what direction they cause the motor to turn.

To alter the state on which the sensor operates, position the cursor block on the **UNTIL** line. This line will keep sending the program back to the **REPEAT** line until it discovers the state defined by the input bits, in this case, until bit 6 is set to 1 — on. Press **f₁** — the value of bit 6 will alter to 0. Notice that the value of bit 7 is currently shaded, meaning that it doesn't matter what value it is — it takes any value. If you press **f₀**, however, you will alter that and set bit 7 to 1 — on; if you press **f₀** again, you will reverse the sense of that bit to 0 — off. To achieve a certain value on one bit is straightforward enough but to make an input bit take any value you will need to hold down the **SHIFT** key and press either **f₀** or **f₁** for the bit required. Test this out and then alter the program so that bit 6 takes any value and bit 7 takes a 1 value — on. Before you run the program you will have to connect the opto-sensor to bit 7 (**R9, R10**).



LEGO *Lines*	IN		OUT						
	7	6	5	4	3	2	1	0	
light	▨	▨	0	1	0	0	0	0	
REPEAT									
UNTIL	1	▨							
open	▨	▨	0	0	0	0	0	1	2
close	▨	▨	0	0	0	0	1	0	2
light	▨	▨	0	1	0	0	0	0	4

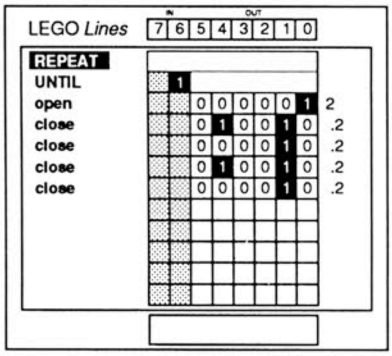
You can alter the timing of a line by positioning the cursor block over that line and using the number keys. Use the → key to move the cursor block into the right-hand side of the screen and then use the **DELETE** key to erase the current value. Then insert the new value required. Try it by altering the time interval on the open and close lines to **1.0** seconds each. Test out your alterations by first using the **COPY** key and then running the program.



```
LEGO Lines      IN          OUT
            7 6 5 4 3 2 1 0
light           0 1 0 0 0 0
REPEAT
UNTIL       1
open            0 0 0 0 0 1   1
alloff          0 0 0 0 0 0   5
close           0 0 0 0 1 0   1
light           0 1 0 0 0 0   4
```

To insert a new line before the **close** line, position the cursor block over this line and press **f8**. Then insert a new label, **alloff** and an output state of all off for five seconds. You are given a blank line which, even though you cannot see it, has all output bits set off. To confirm this, press **f7** (output bit 0) and you will see the zero entered in the other bits. Press **f7** again to turn bit 0 off, so all are now off.

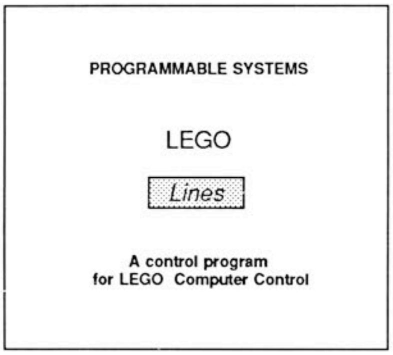
Now run this new program.



```
LEGO Lines      IN          OUT
            7 6 5 4 3 2 1 0
REPEAT
UNTIL         1
open            0 0 0 0 0 1   2
close           0 1 0 0 1 0   .2
close           0 0 0 0 1 0   .2
close           0 1 0 0 1 0   .2
close           0 0 0 0 1 0   .2
```

Can you make the light flash on and off while the door is closing? (a solution is shown in the program called CTEST which uses input bit 6 for the opto sensor).

LEGO *Lines* User Guide



PROGRAMMABLE SYSTEMS

LEGO

Lines

A control program
for LEGO Computer Control

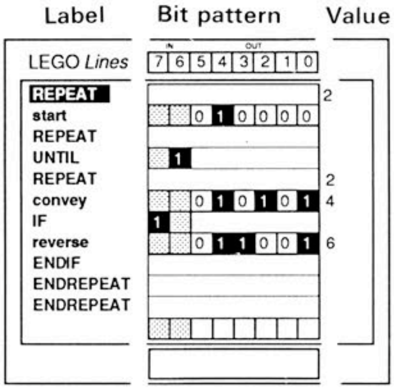
Lines is designed to be an introduction to building a control program. It is suitable for students of all abilities from the age of eleven upwards. The software is designed to run on a BBC Model B, BBC Model B+ and the BBC Master Series computers. The programs are disk-based and can be used with either 80- or 40-track disk drives. (*See footnote*.)

Lines is a controller which treats a series of instructions as a control program which it can use to control a LEGO model through the LEGO interface.

Control is effected by supplying or denying power to the connections on the interface (switching motors or lamps on and off) and by testing the signals coming into the interface (from the opto-sensor bricks) to see if certain conditions are met.

Each instruction occupies a single line on screen. The instruction describes a state on the interface or a condition which it expects to find. The instruction line has the following structure:

description + **bit pattern** + **value**



Label	IN 7	6	5	OUT 4	3	2	1	0	Value
LEGO *Lines*	7	6	5	4	3	2	1	0	
REPEAT									2
start	▨	▨	0	1	0	0	0	0	
REPEAT									
UNTIL	▨	1							
REPEAT									2
convey	▨	▨	0	1	0	1	0	1	4
IF	1	▨							
reverse	▨	▨	0	1	1	0	0	1	6
ENDIF									
ENDREPEAT									
ENDREPEAT									
	▨	▨							

The description may either be a *label* (a reminder of what the line does) or a keyword (a *Lines* reserved word).

The bit pattern defines the output state or the condition being tested. Bits 0–5 are the ***output*** bits. If there is a 1 in the bit it is said to be on and, when that line is executed, power will be sent to that output connection. A 0 means that no power will be sent. The two bits 7 and 6 are ***input*** bits. No power is sent to these bits, instead they are read to see if they are on or off, that is depending on the state of the opto-sensor brick. This information can be used to condition what action is performed next.

The value controls either how long an output activity is performed for, or how many times the activity will be executed before moving on to the next set of instructions.

Lines is designed to accept instructions with the minimum of keystrokes. These will be written to the screen automatically and in any order. Any individual line (or group of lines) may be tested at any stage independently of the rest of the program. In this way, the logical contribution of a single line or a number of lines may be observed.

Note: Disks are supplied in 40-track format but should be copied appropriately onto 80- or 40-track working disks.

Creating a *Lines* program

Note: in explaining the facilities and features of LEGO Lines the following keystroke convention is used:

key 1	means press the named key
f8	means press **f8** to insert a line
key 1 key 2	means hold down the first key, press and release the second key and then release the first.
CTRL f8	means hold down the **CTRL** key and press **f8** to load a ***Lines*** program.

Loading *Lines*

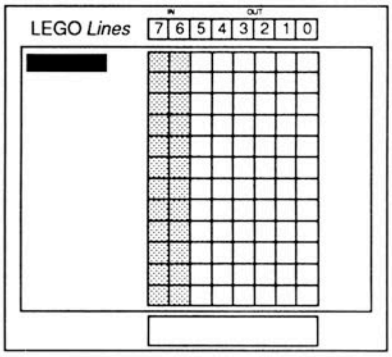
Lines has an ***autoboot*** facility which enables it to be loaded and run automatically. Put the master disk in drive 0 and press:

SHIFT BREAK

The first screen to appear is the title screen. There is a short pause before the main ***Lines*** program is loaded and the second screen, the ***Lines*** editing screen, appears. (The pause can be cancelled by pressing any key.)

The screen has three areas:

- The left hand side of the screen is used to display labels.
- The middle section of the screen shows the bit pattern (defining the way in which the computer and program are communicating with the interface).
- The right hand side is used to show numerical and time values.



The Lines *editing screen*

The blue box situated at the top left hand corner of the screen is the cursor block. It locates where the next item will be entered and is used for editing purposes. The cursor can be moved up and down and from side to side by using the arrow keys on the keyboard.

Entering information in any of the three screen areas is achieved by the use of three different types of key on the keyboard:

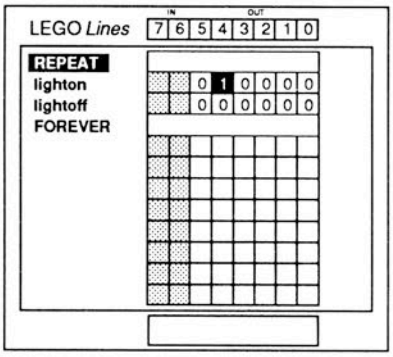
- The letter keys of the QWERTY keyboard are used to enter labels.
- The number keys, 0–9, are used to enter numerical or time values.
- The red function keys are used to enter the appropriate bit patterns.

Setting a bit pattern

The middle section contains eight bits in a layout resembling that on the interface. The positions are marked, from left to right, 7, 6, 5, 4, 3, 2, 1, 0. The numbers 7 and 6 represent the input bit patterns and the others represent the output bit patterns (**R8, R9**).

Testing or running a line will mean that *Lines* will activate the interface to send power to all the output bits which contain a 1 — that are on — in that line. Input bit patterns have green indicator lights, output bit patterns have red indicator lights.

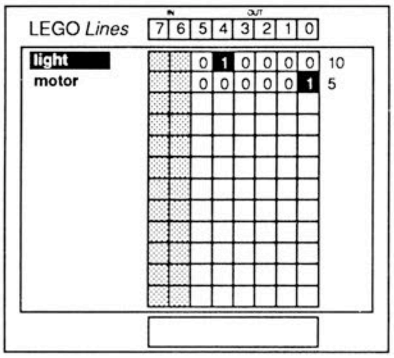
The red function keys set the bits to either off or on. Pressing one of the function keys 'toggles' the state of that bit, on off on and so on. Which function key activates which bit is shown by the keystrip which fits behind the clear plastic strip above the function keys.



LEGO *Lines*	IN 7	6	OUT 5	4	3	2	1	0
REPEAT								
lighton	▨	▨	0	1	0	0	0	0
lightoff	▨	▨	0	0	0	0	0	0
FOREVER								

Initially, all the output bits are set off and the input bits have no state, described as **any value**, meaning that they are not to be tested at all in that line.

Bit patterns set the LEGO interface and will only change when a new bit pattern is output. The bits will remain on until they are explicitly altered, either by another line or by the end of the program.



LEGO *Lines*	IN 7	6	OUT 5	4	3	2	1	0	
light	▨	▨	0	1	0	0	0	0	10
motor	▨	▨	0	0	0	0	0	1	5

An output bit pattern can be entered on its own, that is to say that it does not require a label to be present. Input bit patterns can only be set, however, when the line starts with a keyword.

Keywords

A label is a descriptive word which is entered to identify the line by its function. In the same area, however, *Lines* will accept a number of reserved words, ***keywords***, which have special meaning in terms of function. They are used to allow loop structures (an activity or group of activities being performed repeatedly) and condition structures (activities being performed as a result of a decision to proceed) to be incorporated in the program design.

There are several keywords in LEGO *Lines*. These are:

```
REPEAT
REPEAT n
UNTIL
FOREVER
ENDREPEAT

IF
ENDIF

COUNT n
```

These keywords are entered in the same way as the labels. The letters of the words are entered in lower case until the full keyword is typed, then the word is reprinted in upper case automatically.

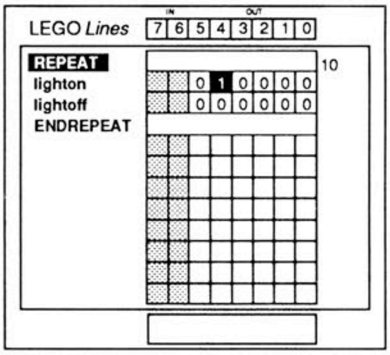
Loop structures

REPEAT FOREVER

When a set of lines is to be repeated indefinitely then the keyword **FOREVER** can be used.The lines ended by **FOREVER** will be repeated continuously or until the **ESCAPE** key is pressed.

REPEAT *n* ENDREPEAT

This allows the same lines to be repeated *n* times. **ENDREPEAT** is used to end the **REPEAT** *n* structure.



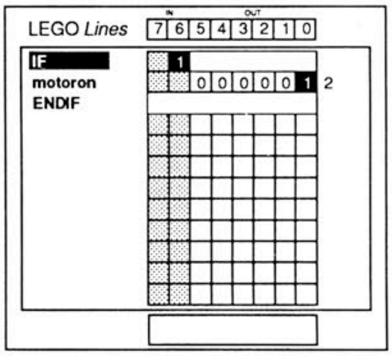
```
LEGO Lines    IN      OUT
              7 6 5 4 3 2 1 0
REPEAT                          10
lighton           0 1 0 0 0 0
lightoff          0 0 0 0 0 0
ENDREPEAT
```

Condition structures

IF ENDIF

Lines can be skipped over or operated upon according to whether a certain bit pattern (the condition) is recognised. The **IF** structure is ended by **ENDIF**. The input bit pattern to be tested is set using the function keys **f₀** and **f₁**. If the bit pattern programmed into the input bit locations is the same as the bit pattern being received through the interface, the lines within the **IF ENDIF** structure are acted upon.

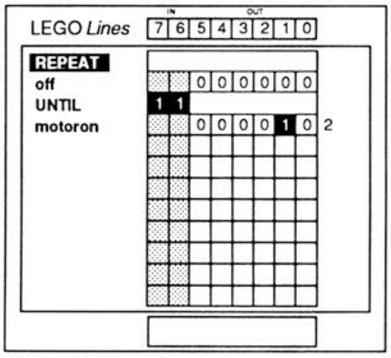
If the bit pattern being received through the interface is not the same, the next line to be executed will be the line following the **ENDIF**. There are two input bits and both may be used to set the input pattern to be tested. If only one input bit pattern is required then the other bit pattern may be cancelled by pressing **SHIFT** and either **f₁** or **f₀**, to reset **any value**.



```
LEGO Lines    IN      OUT
              7 6 5 4 3 2 1 0
IF              1
motoron           0 0 0 0 0 1   2
ENDIF
```

REPEAT UNTIL

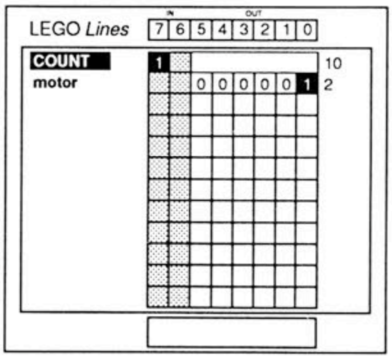
REPEAT loops can end with the **UNTIL** condition. This operates in exactly the same way as the **IF ENDIF** structure except that the input pattern is tested at the end of the lines included in the loop, rather than at the beginning. The lines will, therefore, be repeated until a certain input bit pattern is recognised and then the line after the **UNTIL** will be executed.



```
LEGO Lines    IN      OUT
              7 6 5 4 3 2 1 0
REPEAT
off               0 0 0 0 0 0
UNTIL         1 1
motoron           0 0 0 0 1 0   2
```

COUNT *n*

This keyword is used in conjunction with the LEGO black and white segmented disk. **COUNT** a number (*n*) of times can be used to count changes on the disk from black to white or from white to black. The **COUNT** function will only count on one of the input bit channels, either bit 7 or bit 6, so the input bit NOT being counted MUST be set to **any value** using the **SHIFT** key and either **f₀** or **f₁** as appropriate.



```
LEGO Lines    IN      OUT
              7 6 5 4 3 2 1 0
COUNT         1                 10
motor             0 0 0 0 0 1   2
```

Running a *Lines* program

Pressing **TAB** causes the program to be executed. The cursor block will move to the first line of the program and will then step through the lines as each is performed.

The **ESCAPE** key is used as a stop button if program execution is to be abandoned. Because the performance of a line may take a little time, the **ESCAPE** key needs to be held down until the program has definitely halted.

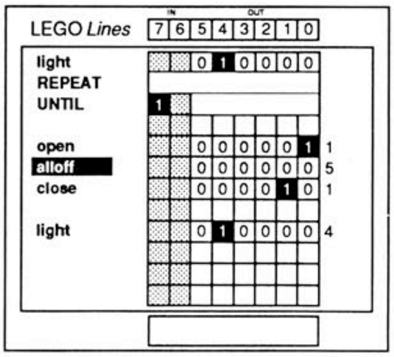
Pressing **SPACE** will halt program execution for as long as it is held down. Releasing **SPACE** continues the program.

Testing a bit pattern

Using the arrow keys to position the cursor on the desired output bit pattern and pressing the **COPY** key transmits that pattern to the LEGO interface and will continue to do so while the key is held down. This feature is useful in checking that that the outputs are connected as required before actually running a program.

Running part of a program

In addition to running a single line, ***Lines*** allows the testing of a group of lines. This can be done by positioning the cursor at the beginning of the part of the program to be tested and inserting a new line (by pressing **f8**). Move the cursor to beyond the last line of the part to be tested and insert another line. As long as the cursor block is within these two blank lines, pressing **TAB** will result in only that part of the program being run. The blank lines will need to be deleted, of course, to enable the whole program to be run.



```
                     IN          OUT
LEGO Lines     7 6 5 4 3 2 1 0

light              0 1 0 0 0 0
REPEAT
UNTIL          1


open               0 0 0 0 0 1  1
alloff             0 0 0 0 0 0  5
close              0 0 0 0 1 0  1

light              0 1 0 0 0 0  4
```

Editing a *Lines* program

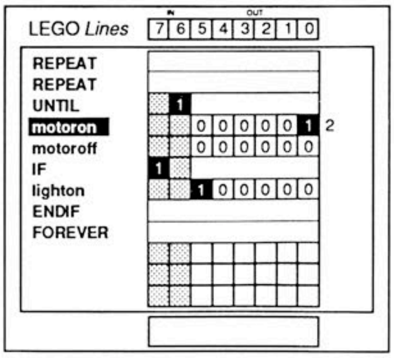
Alterations are made by moving the cursor (using the arrrow keys ↓ ↑ ← →) to the part of the program which requires changing. If a bit pattern needs alteration then the appropriate change can be made simply by switching the required bits off and on in the usual way. Changing a label or a value involves positioning the cursor over it, pressing **DELETE**, and entering the new label or value.

Pressing **SHIFT ↓** takes the cursor to the last line of the program.

Pressing **SHIFT ↑** takes the cursor to the first line of the program.

Editing keywords

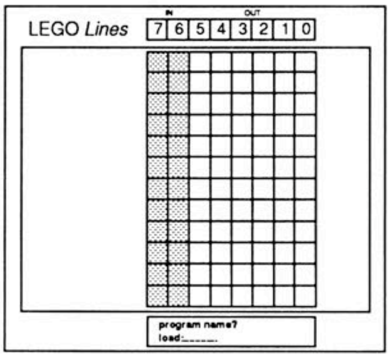
Keywords can only be edited by using **f9** (to delete the line) and **f8** (to insert a new line).



```
                     IN          OUT
LEGO Lines     7 6 5 4 3 2 1 0

REPEAT
REPEAT
UNTIL            1
motoron            0 0 0 0 0 1  2
motoroff           0 0 0 0 0 0
IF             1
lighton            1 0 0 0 0 0
ENDIF
FOREVER
```

Lines utilities

Lines includes a number of utility programs which enable the student to take full advantage of this programming environment. These are also designed to require the minimum number of keystrokes and, in achieving this, make full use of the function keys.



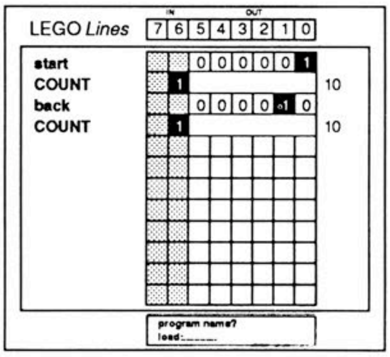
```
                IN          OUT
LEGO Lines   7 6 5 4 3 2 1 0

program name?
load:______
```

Loading and Saving *Lines* programs on disk

A program can be loaded into *Lines* by pressing **CTRL f8**. The following prompt will appear at the bottom of the screen:

```
Program name?
LOAD:........
```

Lines expects the filename of a *Lines* program, stored on the disk in the current drive.



```
                IN          OUT
LEGO Lines   7 6 5 4 3 2 1 0
start              0 0 0 0 0 1
COUNT          1                 10
back               0 0 0 0 1 0
COUNT          1                 10

program name?
load:______
```

A *Lines* program can be saved by pressing **CTRL f9**. The following prompt will appear:

```
Program name?
SAVE:........
```

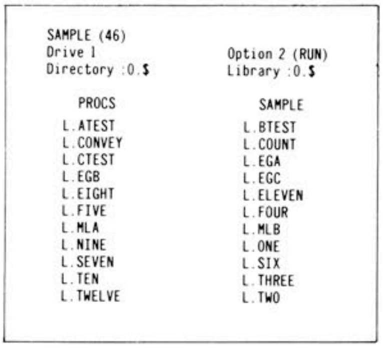
The name entered will identify that program on the disk.

Note: See *Error messages during load and save operations* for a description of possible errors which can occur.

Display a disk directory

During load and save operations, it is often useful to be able to read what is on the disk, to confirm that a program has saved correctly or to list the programs available to load. Pressing **CTRL f3** will display a disk directory.

The files prefixed `L.` are files which have been created by *Lines*, that is, control programs.



```
SAMPLE (46)
Drive 1                  Option 2 (RUN)
Directory :0.$           Library :0.$

   PROCS                    SAMPLE
  L.ATEST                  L.BTEST
  L.CONVEY                 L.COUNT
  L.CTEST                  L.EGA
  L.EGB                    L.EGC
  L.EIGHT                  L.ELEVEN
  L.FIVE                   L.FOUR
  L.MLA                    L.MLB
  L.NINE                   L.ONE
  L.SEVEN                  L.SIX
  L.TEN                    L.THREE
  L.TWELVE                 L.TWO
```

This is a directory of the Sample programs disk

Inserting a line

This is effected by pressing **f8**. A blank line (with all output bits set off and input bits as **`any value`**) will be inserted before the line on which the cursor is positioned.

Deleting a line

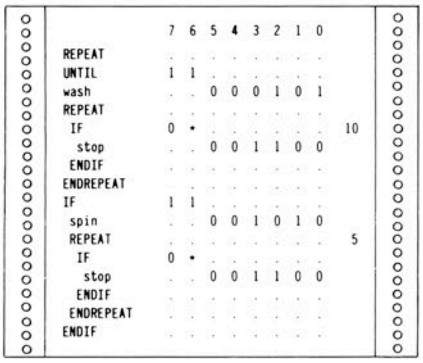
This is effected by positioning the cursor on the line to be deleted and pressing **f9**.

Wipe

An entire program can be deleted by pressing **CTRL f6**. The current program is removed and the blank *Lines* editing screen reappears. A program which has been wiped cannot be retrieved (unless, of course, it has previously been saved to disk).

Printing a *Lines* program

Pressing **CTRL f₇** enables a printout of the program you are currently working on to be sent to a printer. Refer to Technical reference section, page 4.2. Nested structures will appear indented.



```
             7 6 5 4 3 2 1 0
REPEAT       . . . . . . . .
UNTIL        1 1 . . . . . .
wash         . . 0 0 0 1 0 1
REPEAT       . . . . . . . .
 IF          0 • . . . . . .    10
  stop       . . 0 0 1 1 0 0
 ENDIF       . . . . . . . .
ENDREPEAT    . . . . . . . .
IF           1 1 . . . . . .
 spin        . . 0 0 1 0 1 0
 REPEAT      . . . . . . . .    5
  IF         0 • . . . . . .
   stop      . . 0 0 1 1 0 0
  ENDIF      . . . . . . . .
 ENDREPEAT   . . . . . . . .
ENDIF        . . . . . . . .
```

Boxed display option

This feature is used to highlight the structured nature of programs written in the *Lines* environment. Pressing **CTRL f₄** displays a structured flow diagram of the current program, as an aid to efficient programming. Pressing **ESCAPE** or **CTRL f₄** again returns the user to the editing screen.

Using ESCAPE

The **ESCAPE** key can be used during the save, load, printout, and boxed display operations as a means to return immediately to the *Lines* editing screen.

Ending a *Lines* session

Lines is ended by pressing **CTRL f₅**. The screen will be cleared and the user is returned to BASIC.

The disk utility program (DISCUT)

If you have a double or dual disk drive, you may want to save or load programs using a drive other than drive 0. To do this, you run the program DISCUT (see Technical Reference, page 4.2).

Note: Running this program with a single disk drive is unnecessary and may introduce errors.

Error messages

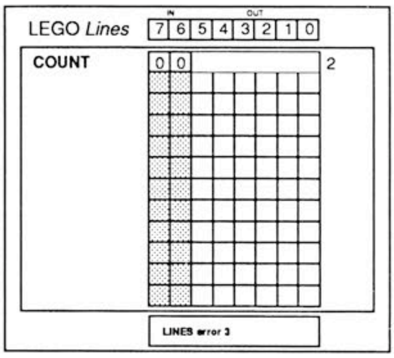
Run errors

After pressing the **TAB** key to run a program, *Lines* does various checks to ensure that the structures within the program are correct. If there are any errors then a message will be displayed in the form:

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

These numbers refer to the following errors:

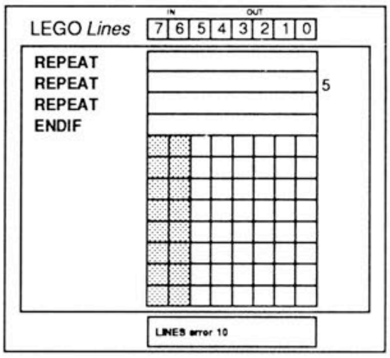
1 The numbers of **REPEAT/REPEAT** *n* levels exceeds eight.
2 The number of **IF** levels exceeds eight.
3 Input condition not properly set in **COUNT**.
4 Number to **COUNT** is not set.
5 **REPEAT** without **UNTIL/FOREVER**.
6 **REPEAT** *n* without **ENDREPEAT**.
7 **IF** without **ENDIF**.
8 **UNTIL/FOREVER** without **REPEAT**.



LINES error 3

COUNT *uses only one of the two input lines, the other should be set to* any value.

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.



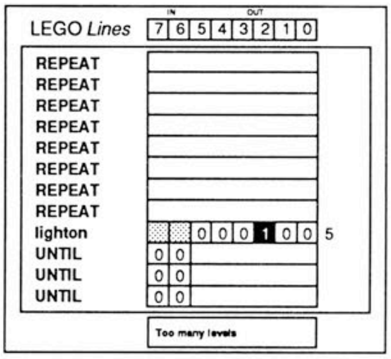
```
LEGO Lines      IN          OUT
               7 6 5 4 3 2 1 0
REPEAT
REPEAT                            5
REPEAT
ENDIF

LINES error 10
```

Error message in boxed display operation

There is one error message which may be encountered in the boxed display operation:

`Too many levels`

Only six levels of nested **REPEAT** and **IF** structures can be accommodated on screen and if a display of a more complex program is attempted then this message appears.



```
LEGO Lines      IN          OUT
               7 6 5 4 3 2 1 0
REPEAT
REPEAT
REPEAT
REPEAT
REPEAT
REPEAT
REPEAT
REPEAT
lighton            0 0 0 1 0 0   5
UNTIL          0 0
UNTIL          0 0
UNTIL          0 0

Too many levels
```

Error messages during load and save operations

`File exists`
`Replace? ...`

This message is displayed when a filename entered, in order to save the file, is the same as one already on the disk. *Lines* expects either a **YES** (to overwrite the named file with the current version) or **NO** (to re-enter a different file name).

`File does not`
`exist`

This message appears when the filename typed in response to the load prompt does not match any file named on the disk.

`Disk error`

If this message appears, there are a number of possible causes:

— The write-protect sticker has not been removed prior to a save operation.

— The disk (or the 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.

LEGO *Lines*: program design

This section is designed to assist the teacher and, consequently, the student in the design and testing of programs using the LEGO ***Lines*** software environment.

Suggestions for the organisation of pupil-based work are also included, reflecting the overall philosophy of the package and the fact that pupils should become capable of solving a range of technological problems using the materials provided.

Three aspects of program design are discussed:

- designing algorithms and testing them using the software
- identifying key software techniques from worked examples
- suggestions about the many tasks pupils could perform using this package.

Algorithms

An algorithm is a set sequence of steps which describes a solution to a particular problem.

There are a number of ways to go about designing a computer program but here is a method which the authors recommend:

1. Analyse the problem by producing a logical statement or a sentence of what the solution(s) to that problem may be. A complex problem will have to be broken down into a number of smaller sub-problems; each of which should also be described in sentence form.

2. Identify the logical structures which are present in these sentences. The use of a ***structured flow diagram*** is usually of great assistance, especially as this can be compared with the boxed display, available as a software utility.

3. Identify the program instructions which correspond to the structures previously identified.

4. Link the sub-problems together to form a complete solution to the problem.

5. Test the solution(s) and make appropriate changes to the program if necessary.

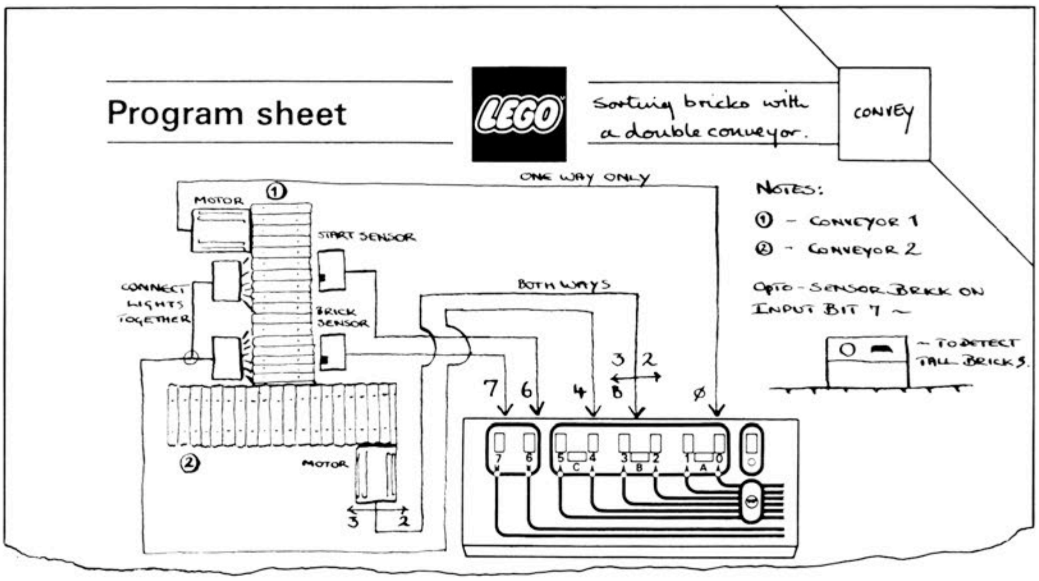
6. Choose the most effective solution.

The development of a computer program is an example of the larger technological problem-solving process at work and it should be subjected to the same rigorous testing as all technological activities.

As you progress through these steps, it will become necessary to convert statements like 'switch motor on' to lines of instruction, involving the creation of bit patterns. The following has proved a useful guide for this process:

1 Define the use of the interface by drawing a schematic diagram of the model being used and identify which bits correspond to which input or output connections. (The program sheet has been designed to facilitate this.) An example is shown below.

Example: Double conveyor used to sort large and small bricks



2 For each of the sentences or blocks in the structured flow diagram identify:

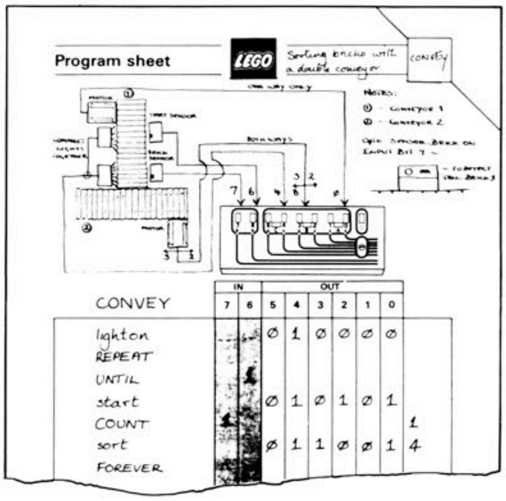
- the starting values of the bits
- the changes in those bit patterns once they are subjected to the processes of the algorithm
- the end values of the bit (that is, the start values of the next sentence or block).

3 Mentally trace the algorithm through line by line focussing on the bit pattern as it changes.

4 Carefully observe that the model behaves as you expect it to.

Sorting conveyors

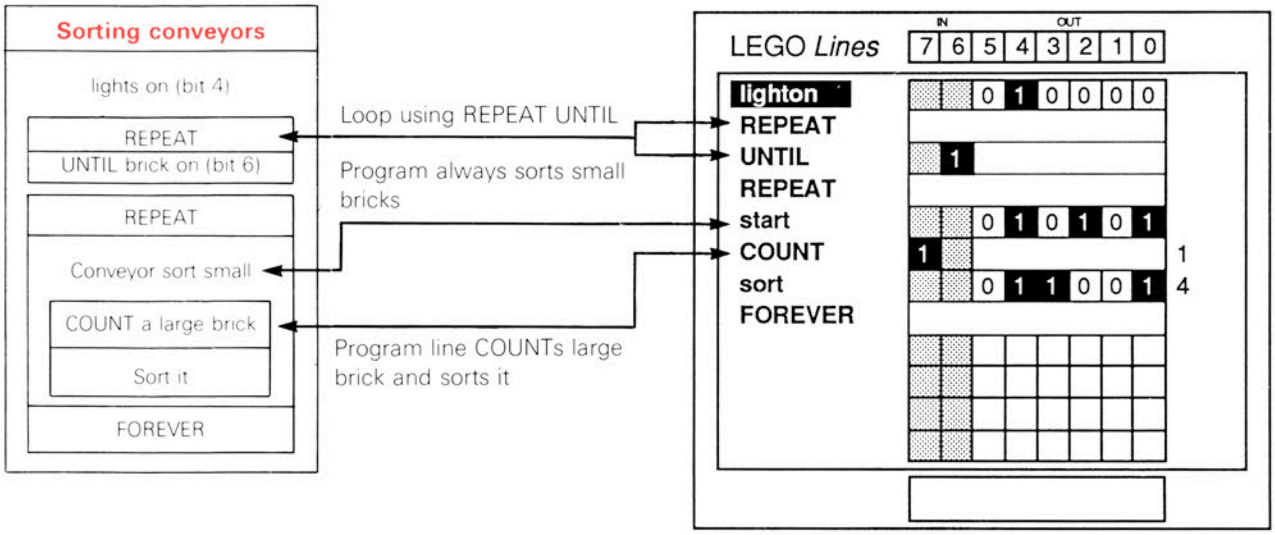
```
lights on (bit 4)
REPEAT
UNTIL brick on (bit 6)
REPEAT
    Conveyor sort small
        COUNT a large brick
        Sort it
FOREVER
```


Program sheet
Sorting bricks with a double conveyor
CONVEY

CONVEY	IN 7	IN 6	OUT 5	OUT 4	OUT 3	OUT 2	OUT 1	OUT 0	
lighton			Ø	1	Ø	Ø	Ø	Ø	
REPEAT									
UNTIL		1							
start			Ø	1	Ø	1	Ø	1	
COUNT	1								1
sort			Ø	1	1	Ø	Ø	1	4
FOREVER									

Key software techniques

Twelve software functions are introduced in the students' assignments and these are detailed in the ***sample programs*** section (page 1.30). Together with a **COUNT**ing function they form the essential introductory building blocks for programming.



The software environment allows the creation of programs of 30 lines maximum and containing up to 8 nested loop structures.

It is a very powerful aid to programming in itself, but it does have a limit. If two input sensors are used it is complex to construct a condition (**UNTIL** or **IF**) which will operate on one input **OR** the other at the same time.

Care must be taken when using **REPEAT UNTIL** loops where the **UNTIL** condition is waiting for an event to take place and there are other actions in the loop. Occasionally an input trigger or signal will appear to be missed. Good program design and appropriate use of light bricks (to force the required signal from the opto-sensor brick) will overcome this.

For example:

Start motor when bit 6 is on

```
LIGHT
REPEAT
UNTIL (bit 6 on)
(where•the light shines
into the opto-sensor)
```

rather than

```
REPEAT
UNTIL (bit 6 on)
(open to ambiguity)
```

Switch two motors and lamp on

```
MOTORS (motor 1 on) (motor 2 on)
LAMP (lamp on)
REPEAT
UNTIL (bit 6 on)
```

rather than

```
REPEAT
MOTORS
LAMP
UNTIL (bit 6 on)
```

Structured flow diagram

The technique of employing structured flow diagrams is an important aid to program design. By using such a device students are forced to adopt a 'top-down' approach to solving a problem using the programming environment.

Example

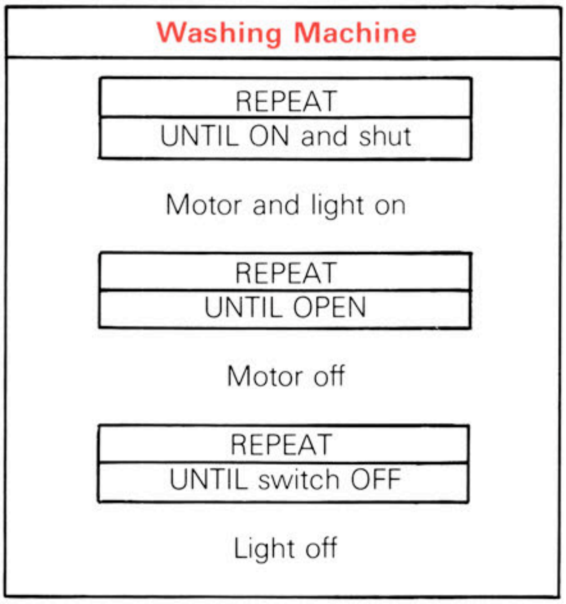
Turn the washing machine and the wash cycle indicator light on when the door is closed and the switch is on. Turn the motor off when the door is opened, and the light off when the switch is off.

Identify structures:

Actions	Turn motor and light on Turn motor off Turn light off
Test	When door is closed and switch is on When door is opened When switch is off
Loops	Both off until door and switch on Motor on until door opened Light on until switch off

sequence ↓

Produce structured flow diagram



Washing Machine

REPEAT
UNTIL ON and shut

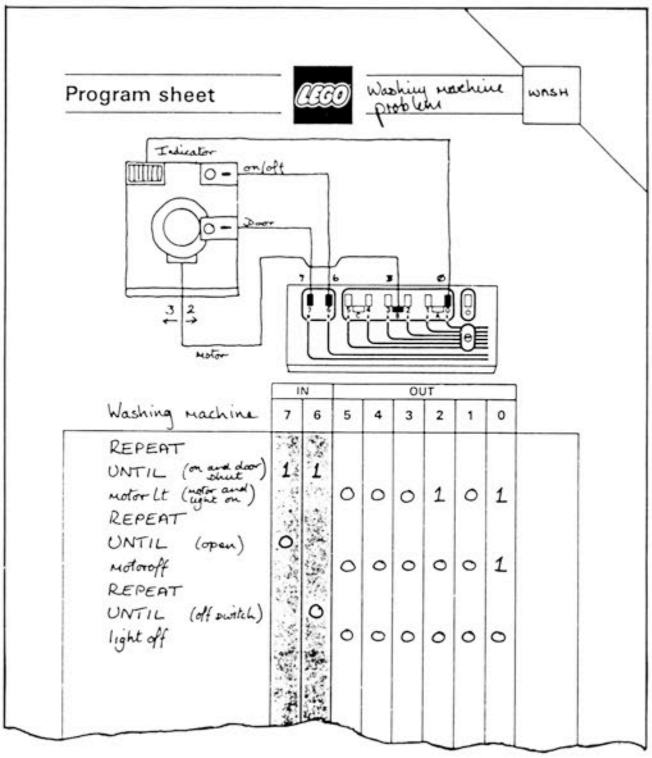
Motor and light on

REPEAT
UNTIL OPEN

Motor off

REPEAT
UNTIL switch OFF

Light off



Program sheet

Washing machine problem

WASH

Indicator
on/off
Door
3
2
Motor
7
6
3
0

	IN		OUT					
Washing machine	7	6	5	4	3	2	1	0
REPEAT								
UNTIL (on and door shut)	1	1						
motor Lt (motor and light on)			0	0	0	1	0	1
REPEAT								
UNTIL (open)	0							
motoroff			0	0	0	0	0	1
REPEAT								
UNTIL (off switch)		0						
light off			0	0	0	0	0	0

All of the models will support many different programs some of which will perform a particular task better than others. A 'good' solution is one which does the job it is required to do and is elegant in design, in the sense that it is efficient yet uncluttered and easy to follow.

The following are samples of problems which can be solved using the washing machine model (1090 C) and the programming environment.

1 Turn the washing machine motor on and then off after a certain time has elapsed.

2 Turn the washing machine motor on for a certain time and indicate that it is on by a warning light. Then turn the light and motor off.

3 Turn the motor on for a certain time only if an on/off switch is turned on.

4 Turn the motor and indicator light on for a certain time only if an on/off switch is turned on.

5 Turn the motor on for a certain time only if an on/off switch is turned on and the door is closed.

6 Turn the light on when the on/off switch is on and then turn the motor on when the door is closed. The motor should not turn on unless the on/off switch is on.

7 Turn the washing machine on when the door is closed and the switch is on. Execute a wash cycle (drum turns one way for a short while, then the other) and a spin cycle (drum spins quickly, in one direction only) indicating when each is taking place.

8 Turn the washing machine on when the door is closed and the switch is on. Execute a wash cycle and a spin cycle indicating when each is taking place.

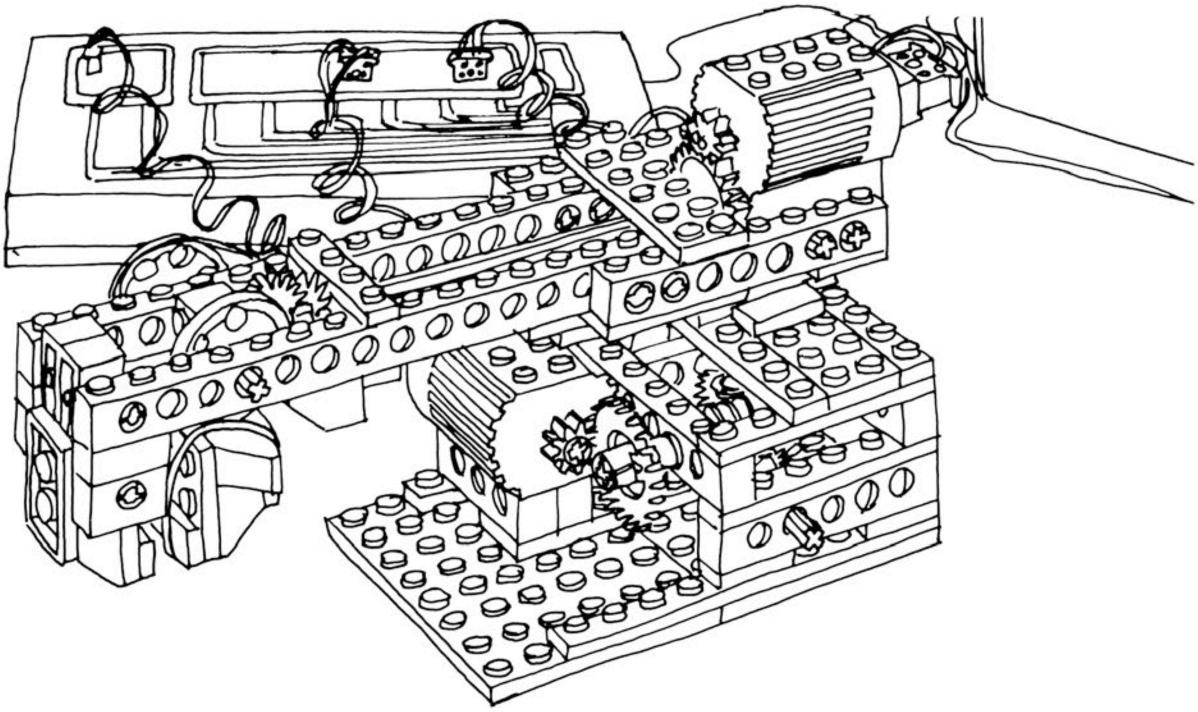
9 Incorporate an emergency switch into the programs.

Adapting or combining models

Pre-built models can provide a useful starting point but students will naturally suggest improvements to the design of particular models and ways of improving performance. It is possible, for example, to add a gearing mechanism to the washing machine so that the wash and spin cycles operate at different speeds. The Robot Arm model (1090 E) could easily be adapted to operate in an up/down plane, and so on.

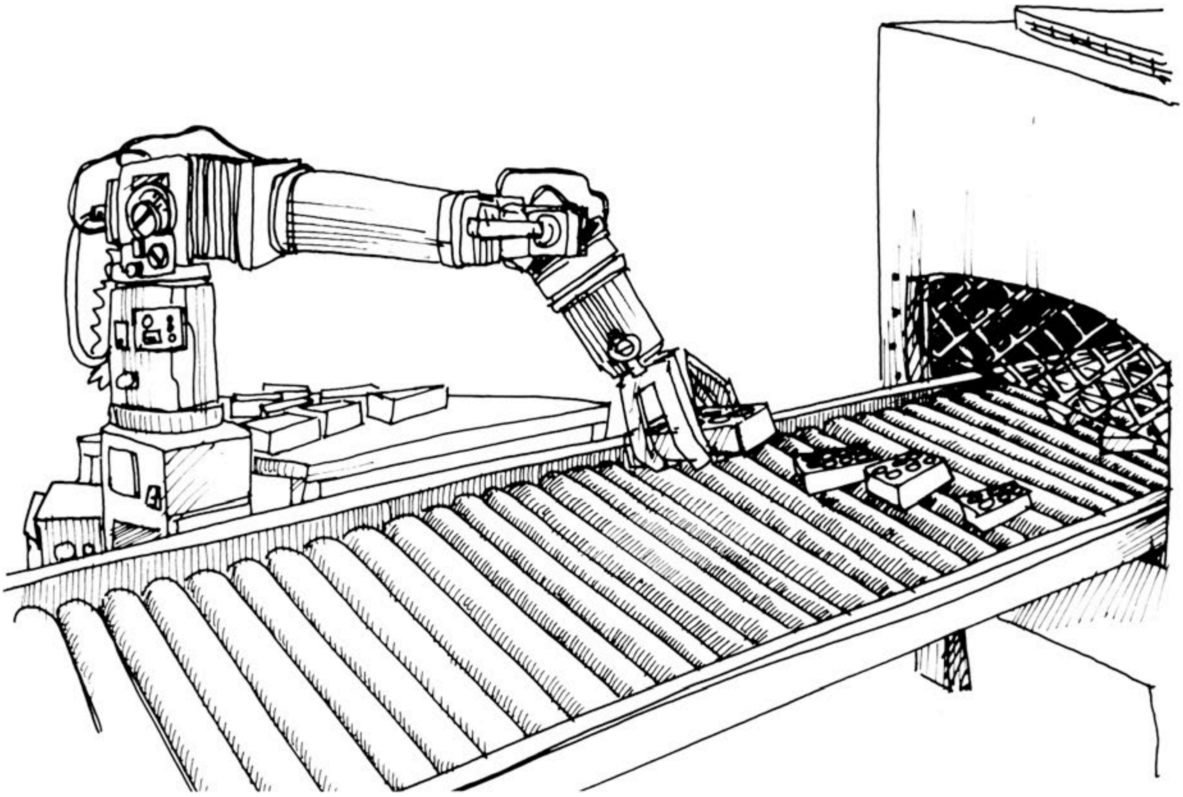
These extensions are to be encouraged as they promote the understanding that both software and hardware solutions combine to form the best solution to a particular problem and students need to understand this clearly. Selection of the most appropriate solution is a very important part of the technological process.

It is expected that as students gain in confidence they will wish to move away from the assignments into problems which they define for themselves. Indeed motivation for such activities will be high and results are likely to be good.

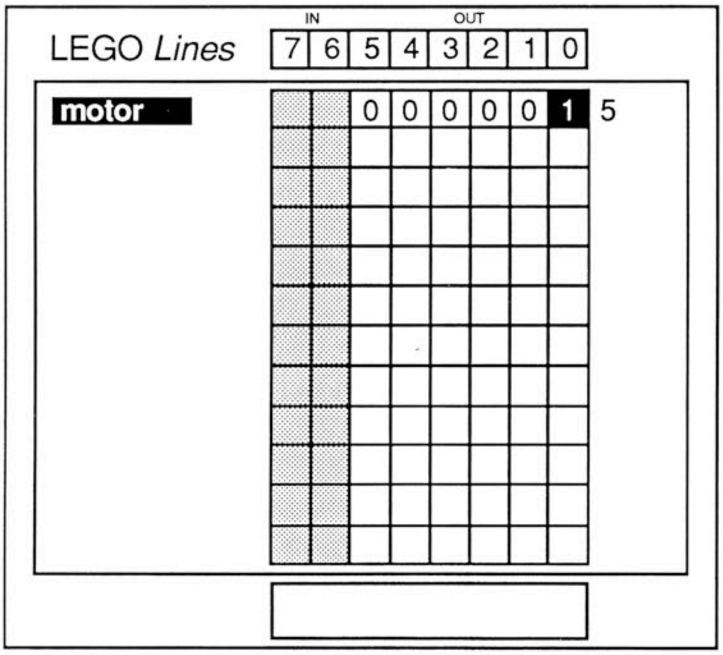


Open-ended problem-solving should be encouraged by discussions of particular environmental, manufacturing, or social needs. It will be important to consider the ways in which computer control has something more to offer than simple mechanical solutions.

It will also be important to refer the students to the problem-solving process they will have already experienced through the recommended course. This will now be a valuable resource.



ONE



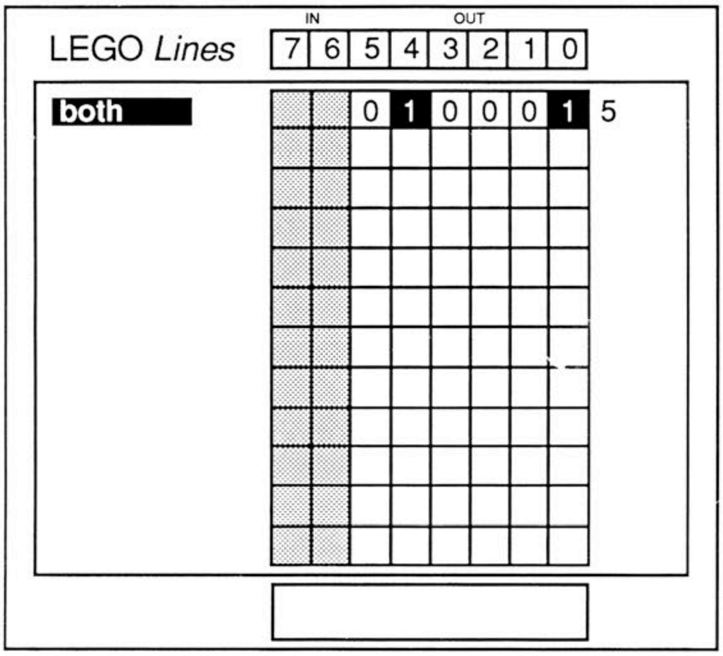
LEGO *Lines*

	IN		OUT						
	7	6	5	4	3	2	1	0	
motor	▨	▨	0	0	0	0	0	1	5
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							

ONE
motor

This program demonstrates control of a single motor in one direction for 5 seconds.

TWO



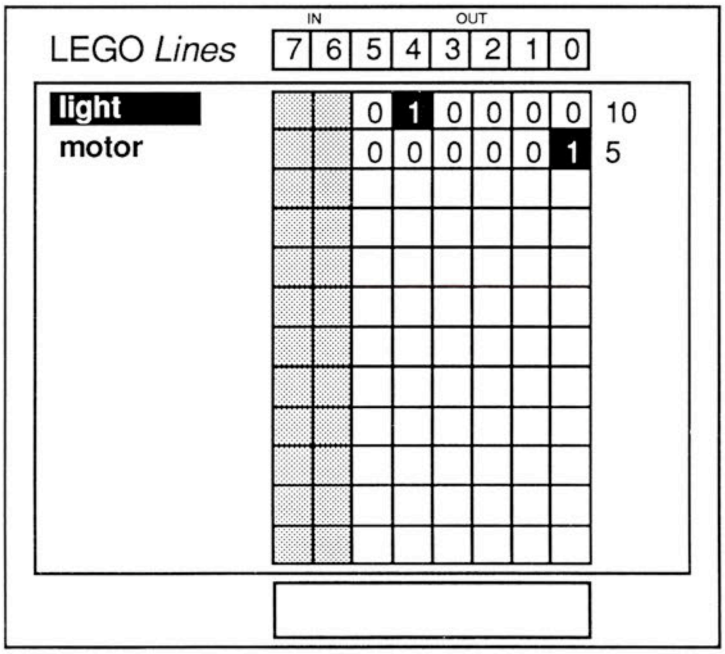
LEGO *Lines*

	IN		OUT						
	7	6	5	4	3	2	1	0	
both	▨	▨	0	1	0	0	0	1	5
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							

TWO
both

This program turns on a motor and a lamp for 5 seconds.

THREE



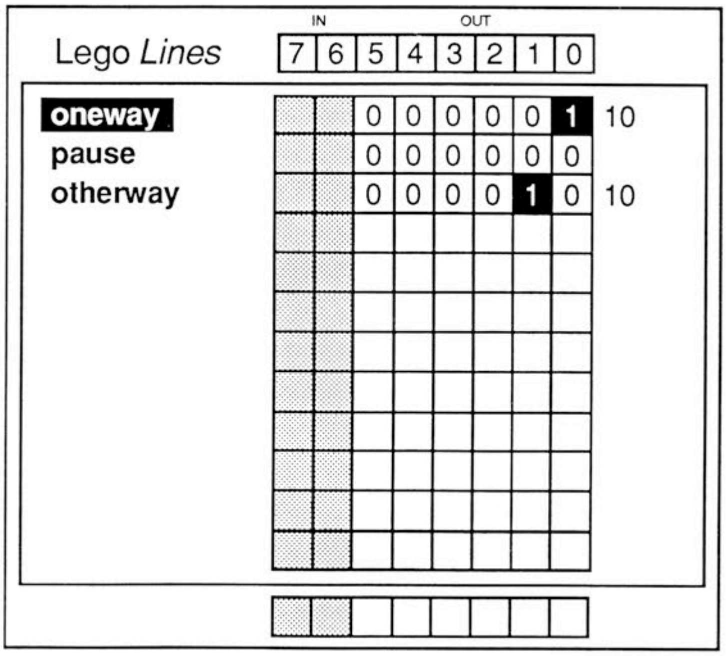
	IN		OUT						
LEGO *Lines*	7	6	5	4	3	2	1	0	
light	▨	▨	0	1	0	0	0	0	10
motor	▨	▨	0	0	0	0	0	1	5
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							

THREE

light
motor

This program is made up of a sequence of two instructions.

FOUR



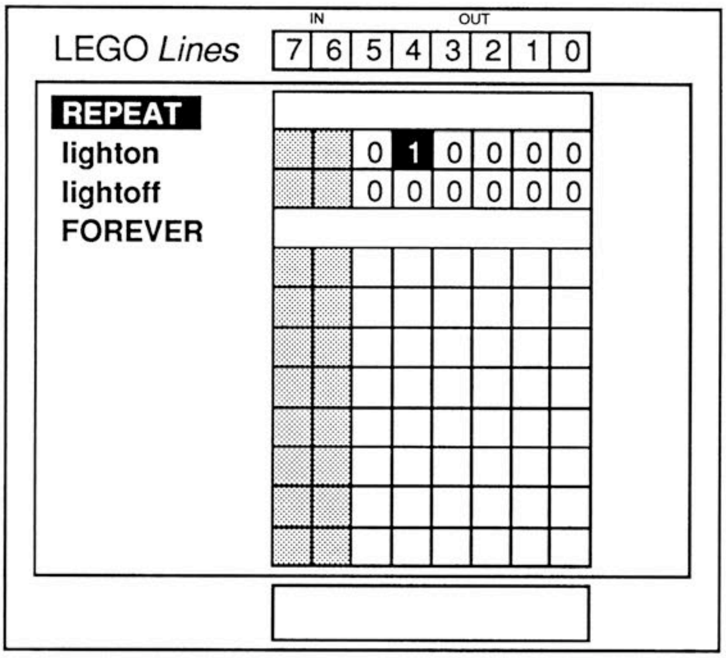
	IN		OUT						
Lego *Lines*	7	6	5	4	3	2	1	0	
oneway	▨	▨	0	0	0	0	0	1	10
pause	▨	▨	0	0	0	0	0	0	
otherway	▨	▨	0	0	0	0	1	0	10
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							

FOUR

oneway
pause
otherway

This program contains a sequence of three instructions which demonstrates rotation of a motor in clockwise and anticlockwise directions.

FIVE



LEGO *Lines*

	IN	IN	OUT	OUT	OUT	OUT	OUT	OUT
	7	6	5	4	3	2	1	0
REPEAT								
lighton	▨	▨	0	1	0	0	0	0
lightoff	▨	▨	0	0	0	0	0	0
FOREVER								

FIVE

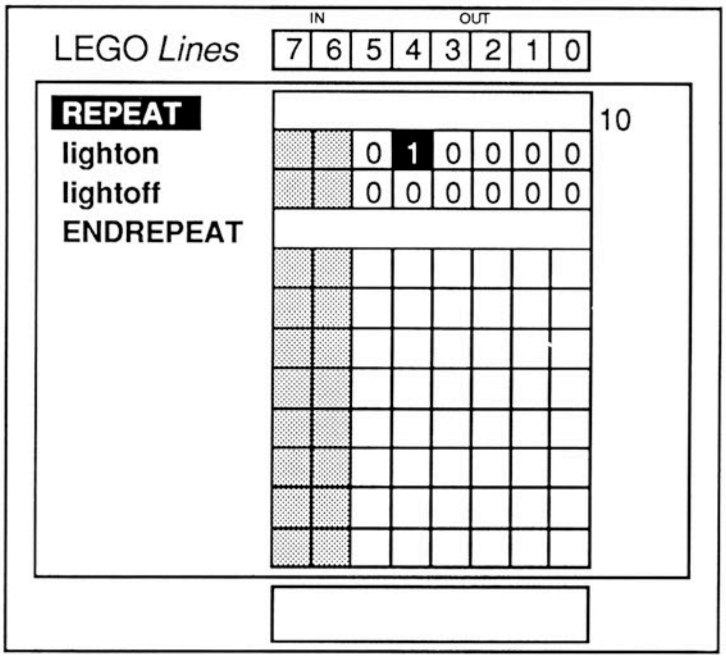
REPEAT

lighton
lightoff

FOREVER

This program turns the lamp on and off forever (until **ESCAPE** is pressed).

SIX



LEGO *Lines*

	IN	IN	OUT	OUT	OUT	OUT	OUT	OUT
	7	6	5	4	3	2	1	0
REPEAT	10							
lighton	▨	▨	0	1	0	0	0	0
lightoff	▨	▨	0	0	0	0	0	0
ENDREPEAT								

SIX

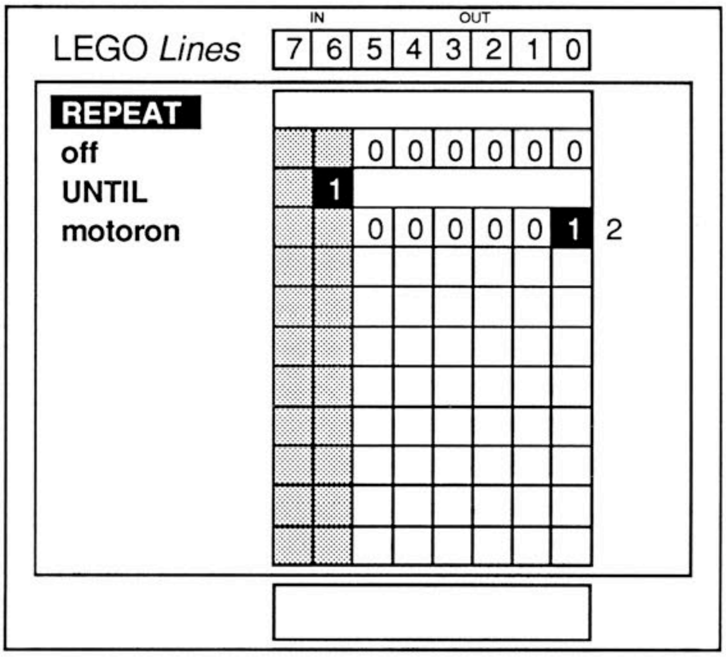
REPEAT

lighton
lightoff

ENDREPEAT

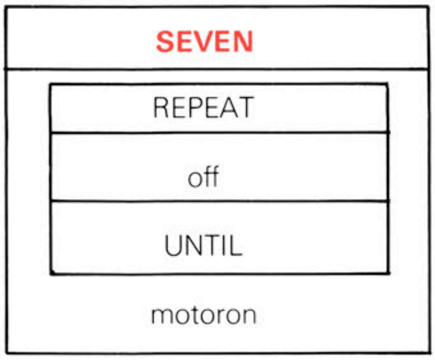
This program demonstrates a sequence of instructions being repeated a set number of times.

SEVEN



LEGO *Lines*

	IN		OUT						
	7	6	5	4	3	2	1	0	
REPEAT									
off	▨	▨	0	0	0	0	0	0	
UNTIL	▨	1							
motoron	▨	▨	0	0	0	0	0	1	2
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							



SEVEN

REPEAT

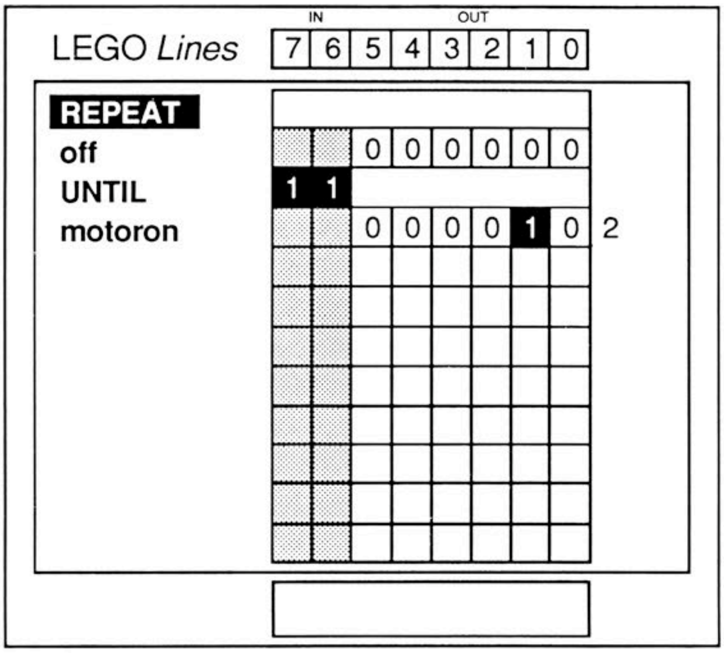
off

UNTIL

motoron

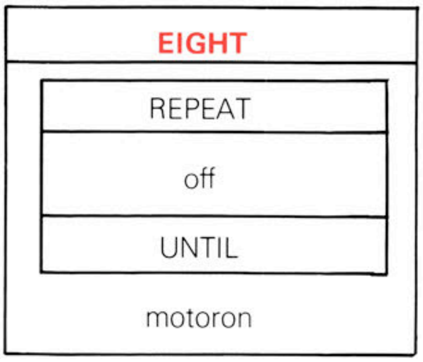
This program illustrates the use of **REPEAT UNTIL** using one sensor as an input.

EIGHT



LEGO *Lines*

	IN		OUT						
	7	6	5	4	3	2	1	0	
REPEAT									
off	▨	▨	0	0	0	0	0	0	
UNTIL	1	1							
motoron	▨	▨	0	0	0	0	1	0	2
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							



EIGHT

REPEAT

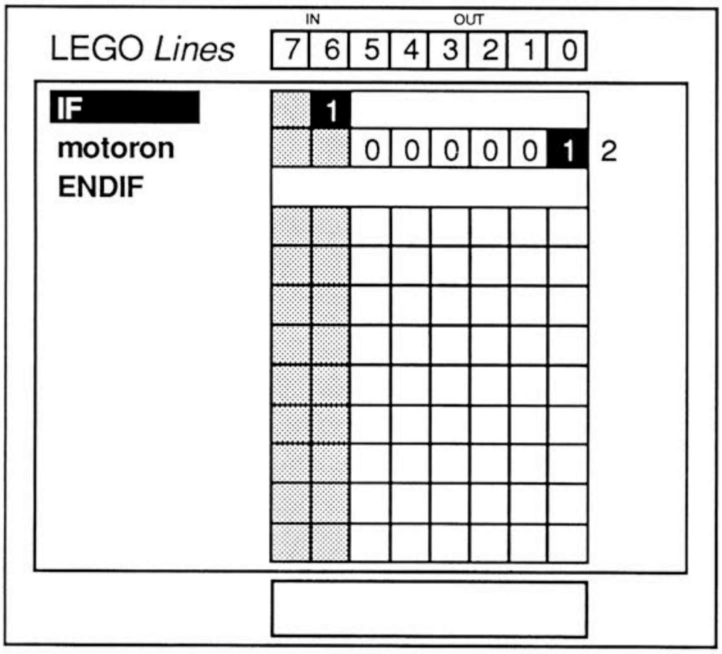
off

UNTIL

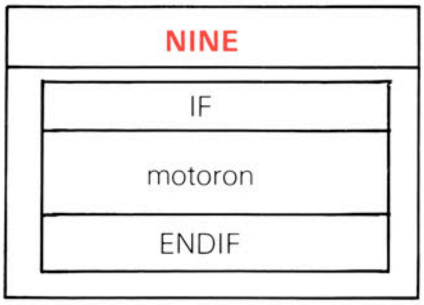
motoron

This program illustrates the use of two sensors and the **REPEAT UNTIL** structure.

NINE


LEGO *Lines*

	IN 7	6	5	4	OUT 3	2	1	0	
IF		1							
motoron			0	0	0	0	0	1	2
ENDIF									


NINE

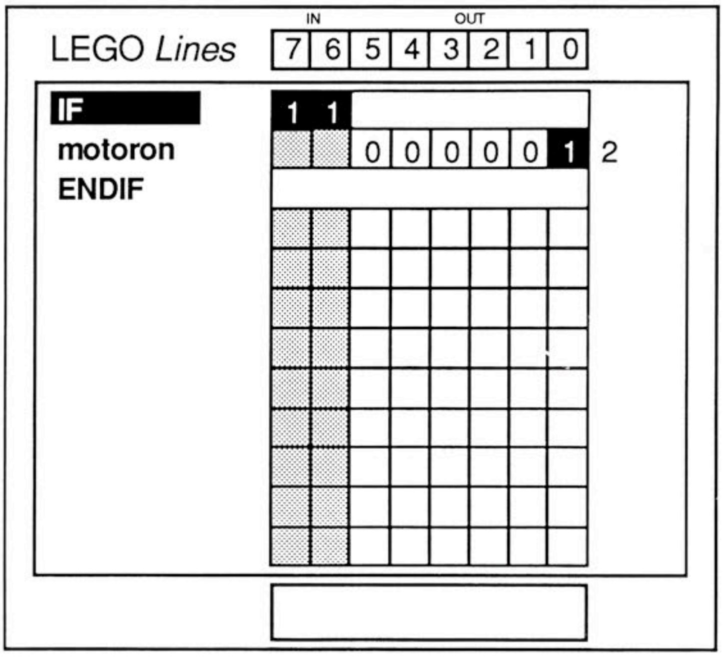
IF

motoron

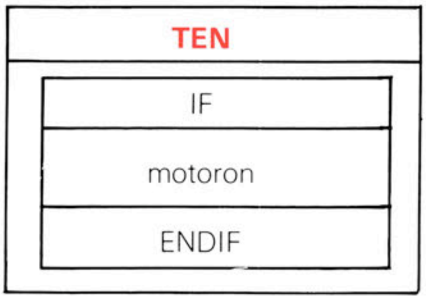
ENDIF

This program illustrates the use of `IF ENDIF`.

TEN


LEGO *Lines*

	IN 7	6	5	4	OUT 3	2	1	0	
IF	1	1							
motoron			0	0	0	0	0	1	2
ENDIF									


TEN

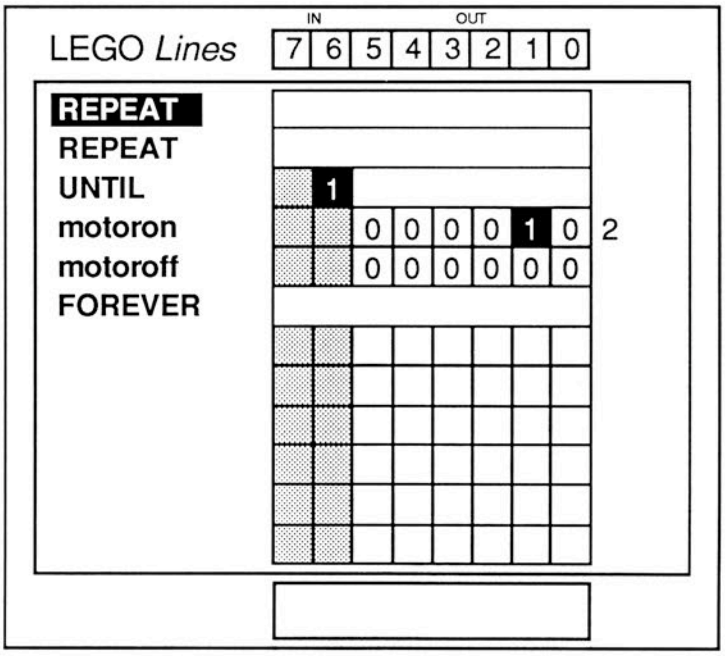
IF

motoron

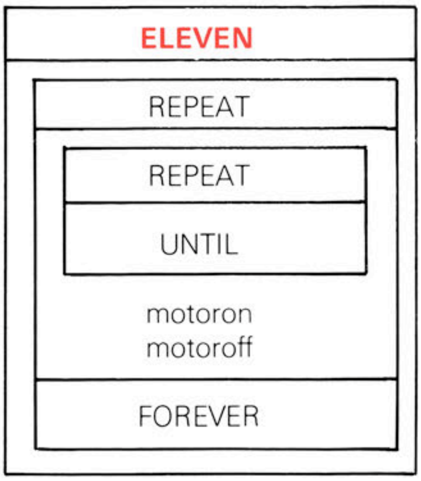
ENDIF

This program turns a motor on if input bit 7 and input bit 6 are both on.

ELEVEN



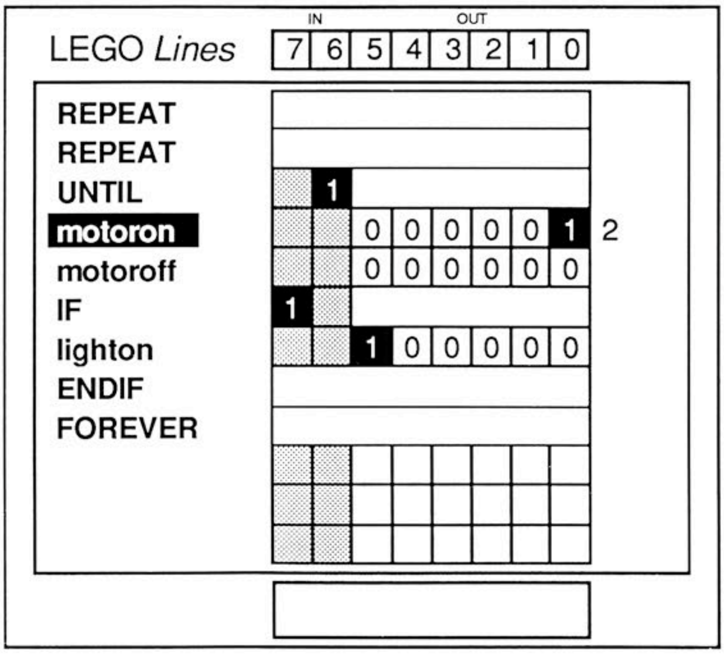
```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
REPEAT
REPEAT
UNTIL        ▨ 1
motoron      ▨ ▨ 0 0 0 0 1 0  2
motoroff     ▨ ▨ 0 0 0 0 0 0
FOREVER
```



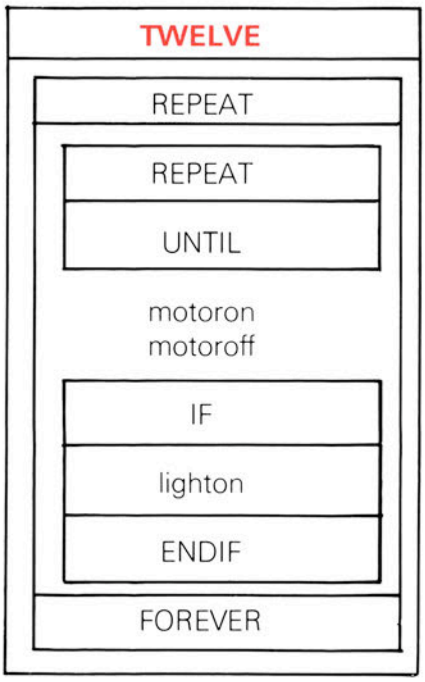
```
ELEVEN
REPEAT
    REPEAT
    UNTIL
    motoron
    motoroff
FOREVER
```

This program continually looks for an input and, when it has been found, executes a sequence of instructions.

TWELVE



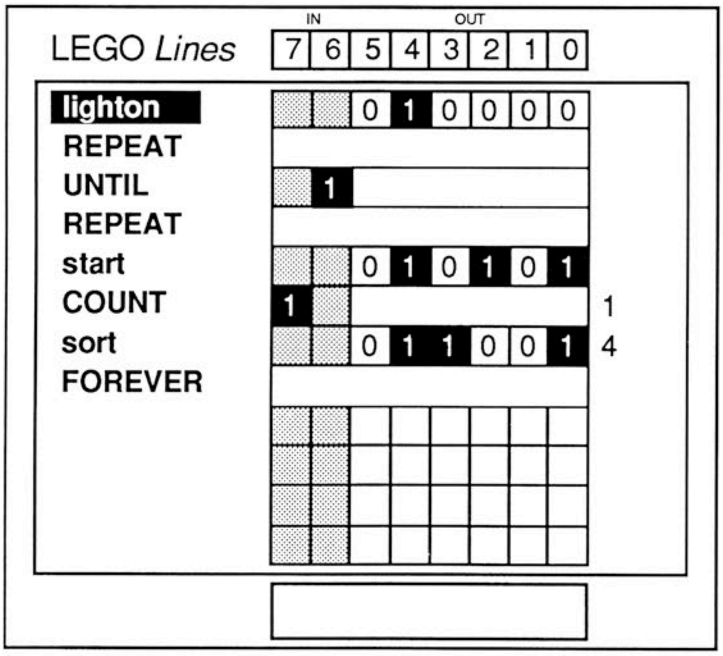
```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
REPEAT
REPEAT
UNTIL        ▨ 1
motoron      ▨ ▨ 0 0 0 0 0 1  2
motoroff     ▨ ▨ 0 0 0 0 0 0
IF           1 ▨
lighton      ▨ ▨ 1 0 0 0 0 0
ENDIF
FOREVER
```



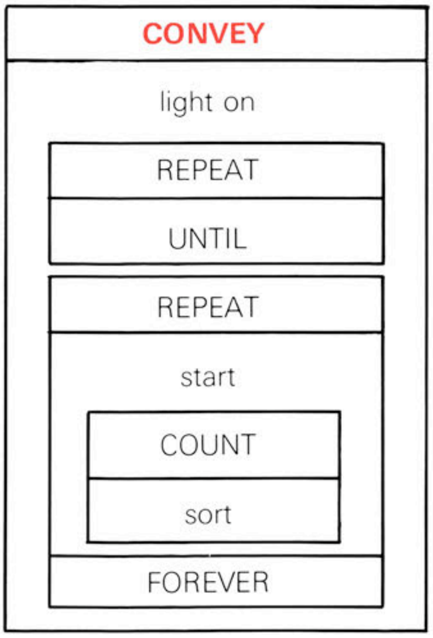
```
TWELVE
REPEAT
    REPEAT
    UNTIL
    motoron
    motoroff
    IF
    lighton
    ENDIF
FOREVER
```

This program continually executes a sequence of instructions which includes an `IF ENDIF` structure.

CONVEY



LEGO *Lines*	IN		OUT						
	7	6	5	4	3	2	1	0	
lighton	▨	▨	0	1	0	0	0	0	
REPEAT									
UNTIL	▨	1							
REPEAT									
start	▨	▨	0	1	0	1	0	1	
COUNT	1	▨							1
sort	▨	▨	0	1	1	0	0	1	4
FOREVER									
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							



CONVEY

light on

REPEAT

UNTIL

REPEAT

start

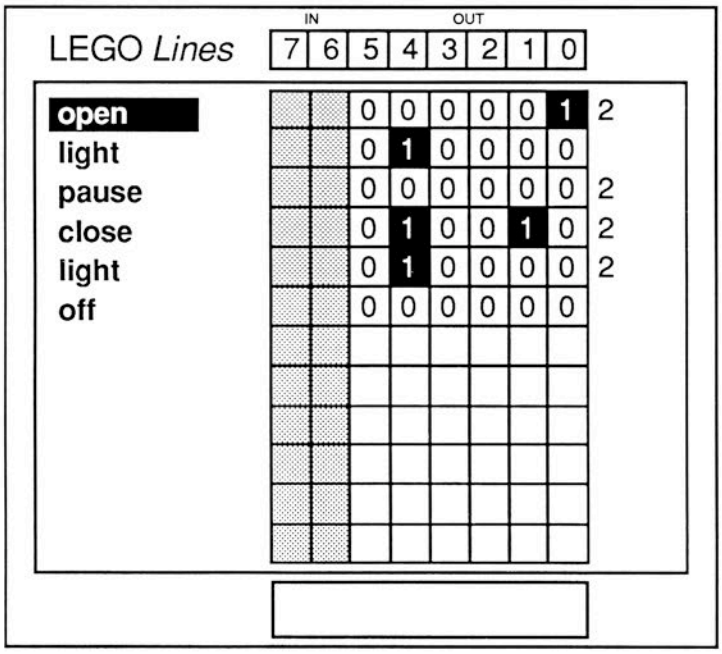
COUNT

sort

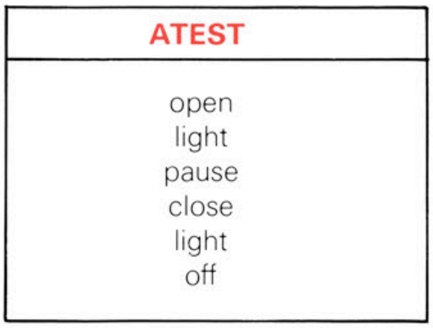
FOREVER

This program is designed to sort large bricks from small bricks on the double conveyor model.

ATEST



LEGO *Lines*	IN		OUT						
	7	6	5	4	3	2	1	0	
open	▨	▨	0	0	0	0	0	1	2
light	▨	▨	0	1	0	0	0	0	
pause	▨	▨	0	0	0	0	0	0	2
close	▨	▨	0	1	0	0	1	0	2
light	▨	▨	0	1	0	0	0	0	2
off	▨	▨	0	0	0	0	0	0	
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							

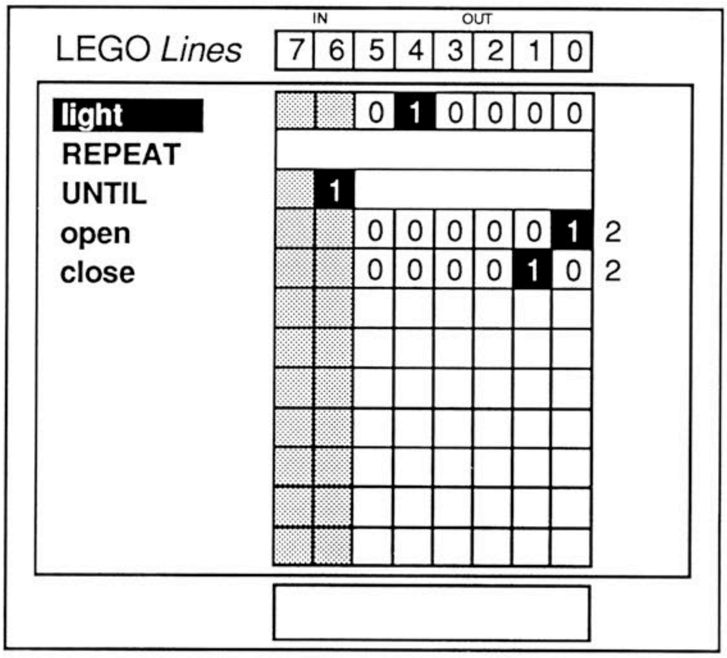


ATEST

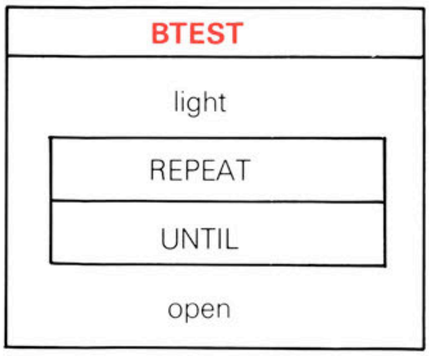
open
light
pause
close
light
off

This program is designed to open the door and light a lamp. The door should pause before it closes and then the light is switched off after 2 seconds.

BTEST



LEGO *Lines*	IN		OUT						
	7	6	5	4	3	2	1	0	
light	▨	▨	0	1	0	0	0	0	
REPEAT									
UNTIL	▨	1							
open	▨	▨	0	0	0	0	0	1	2
close	▨	▨	0	0	0	0	1	0	2



BTEST

light

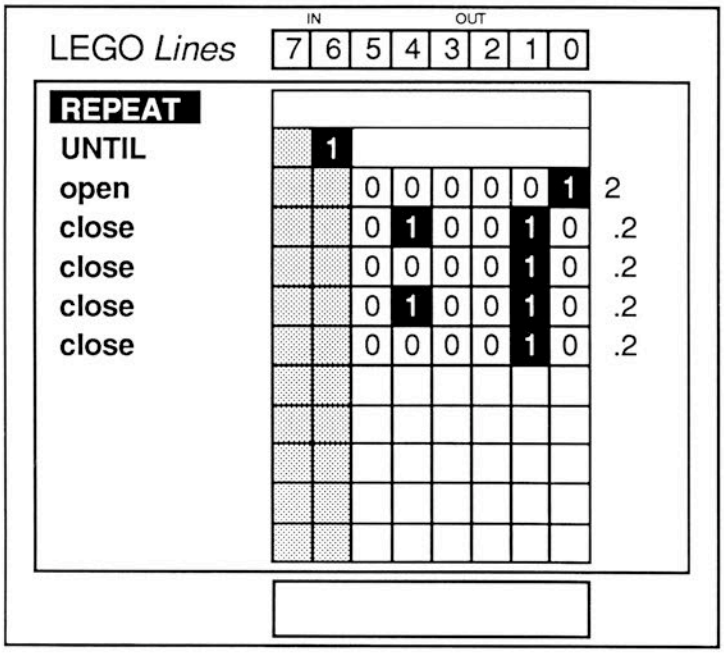
REPEAT

UNTIL

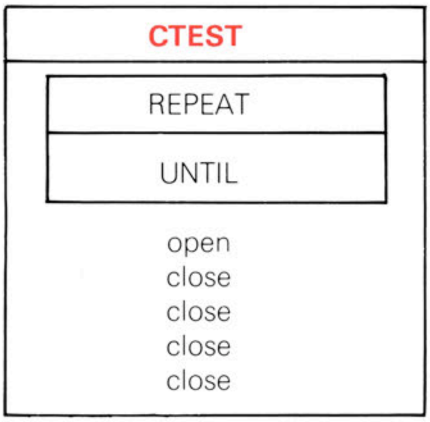
open

This program waits until input bit 6 is on before it opens and closes the automatic door.

CTEST



LEGO *Lines*	IN		OUT						
	7	6	5	4	3	2	1	0	
REPEAT									
UNTIL	▨	1							
open	▨	▨	0	0	0	0	0	1	2
close	▨	▨	0	1	0	0	1	0	.2
close	▨	▨	0	0	0	0	1	0	.2
close	▨	▨	0	1	0	0	1	0	.2
close	▨	▨	0	0	0	0	1	0	.2



CTEST

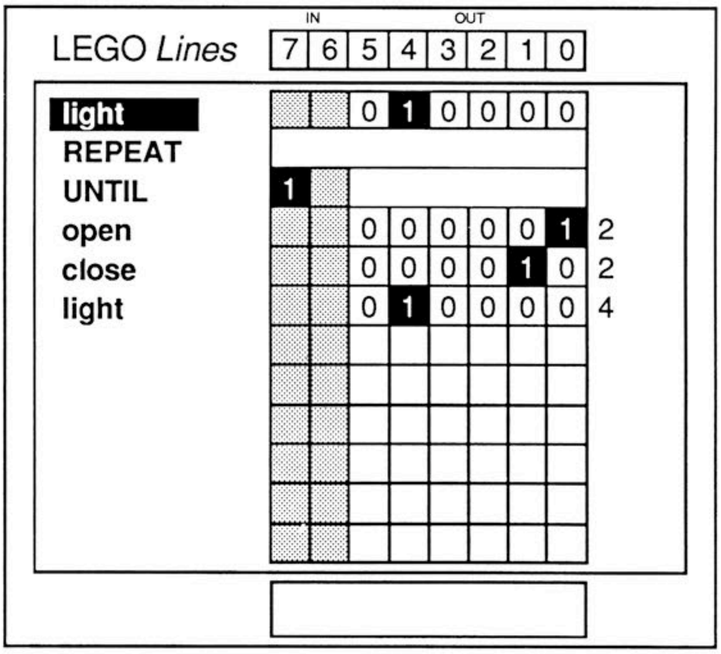
REPEAT

UNTIL

open
close
close
close
close

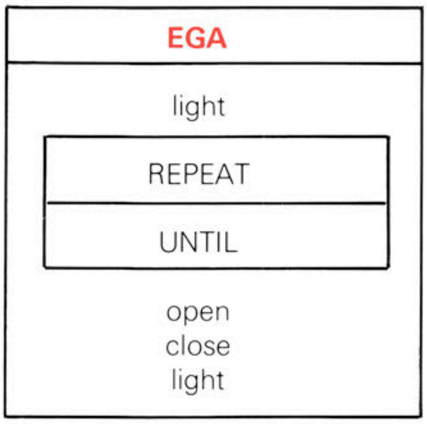
This program is designed to wait until an appropriate signal opens the door and then the light will flash on and off as the door closes.

EGA



LEGO *Lines*

	IN		OUT						
	7	6	5	4	3	2	1	0	
light	▨	▨	0	1	0	0	0	0	
REPEAT									
UNTIL	1	▨							
open	▨	▨	0	0	0	0	0	1	2
close	▨	▨	0	0	0	0	1	0	2
light	▨	▨	0	1	0	0	0	0	4



EGA

light

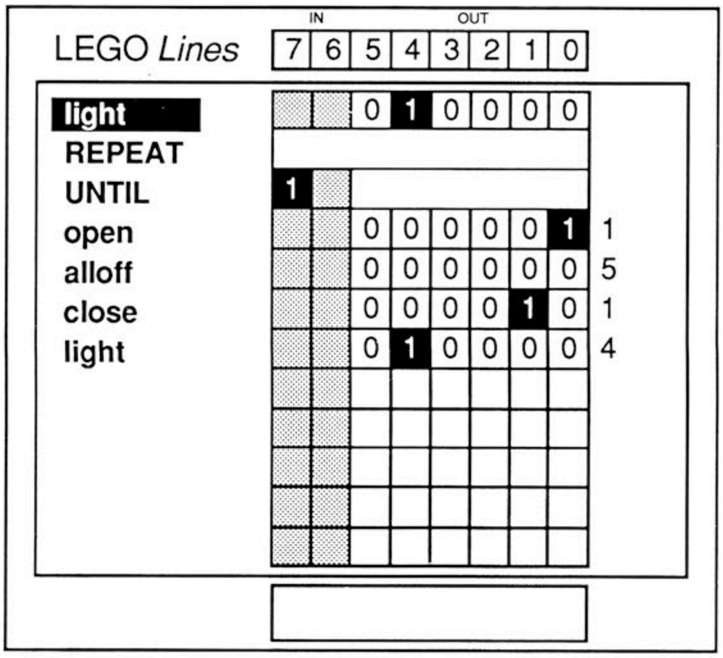
REPEAT

UNTIL

open
close
light

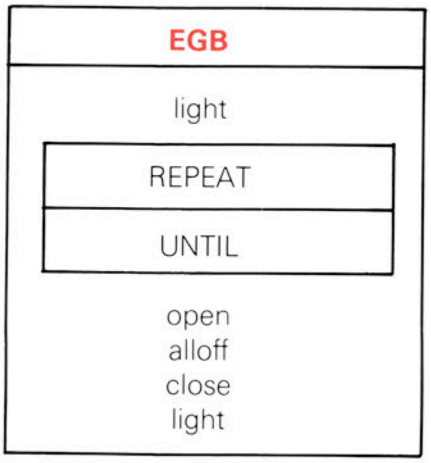
This program lights a lamp and waits until input bit 7 is made to switch the door to open and close. The light will stay on for 4 seconds when the door is closed.

EGB



LEGO *Lines*

	IN		OUT						
	7	6	5	4	3	2	1	0	
light	▨	▨	0	1	0	0	0	0	
REPEAT									
UNTIL	1	▨							
open	▨	▨	0	0	0	0	0	1	1
alloff	▨	▨	0	0	0	0	0	0	5
close	▨	▨	0	0	0	0	1	0	1
light	▨	▨	0	1	0	0	0	0	4



EGB

light

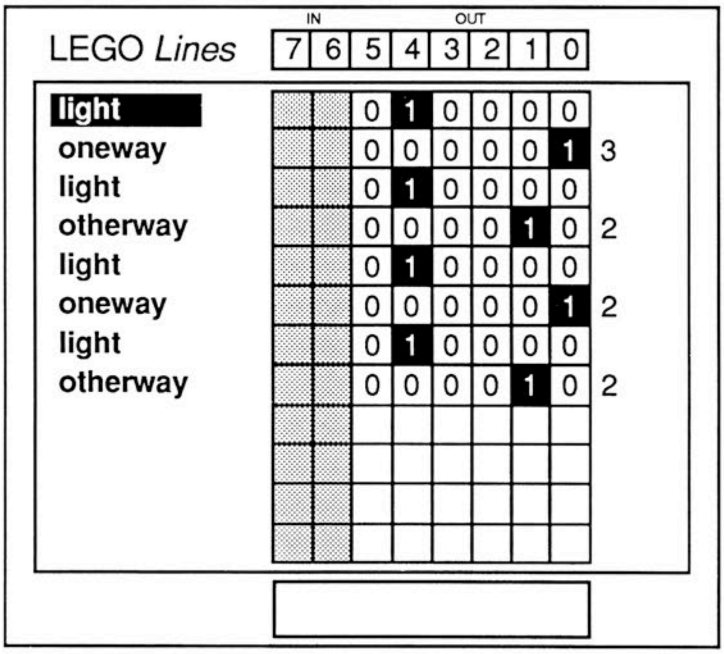
REPEAT

UNTIL

open
alloff
close
light

This program performs in the same way as the last program, except that it includes a new line, labelled *alloff*.

EGC



LEGO *Lines*

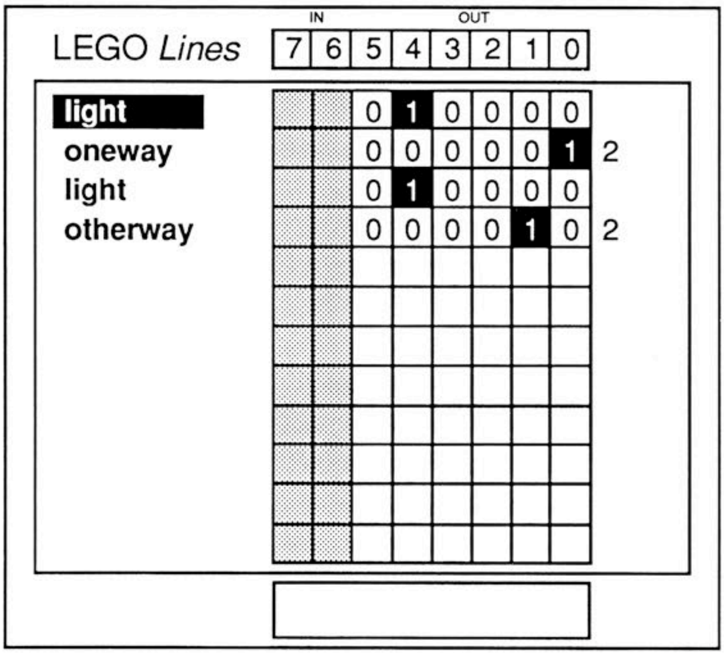
	IN		OUT						
	7	6	5	4	3	2	1	0	
light			0	1	0	0	0	0	
oneway			0	0	0	0	0	1	3
light			0	1	0	0	0	0	
otherway			0	0	0	0	1	0	2
light			0	1	0	0	0	0	
oneway			0	0	0	0	0	1	2
light			0	1	0	0	0	0	
otherway			0	0	0	0	1	0	2

EGC

light
oneway
light
otherway
light
oneway
light
otherway

This program demonstrates the use of output bits A to control a motor and output bit 4 to control a light. It is an extension of **MLA**.

MLA



LEGO *Lines*

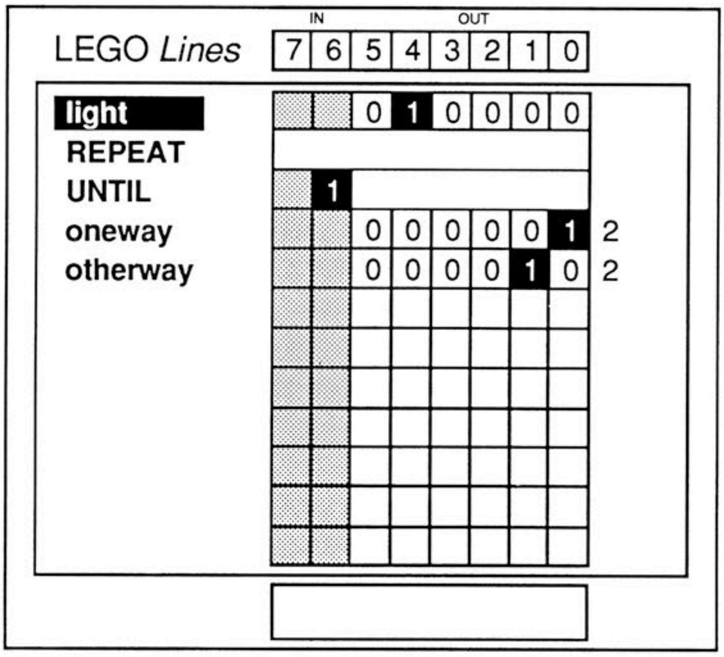
	IN		OUT						
	7	6	5	4	3	2	1	0	
light			0	1	0	0	0	0	
oneway			0	0	0	0	0	1	2
light			0	1	0	0	0	0	
otherway			0	0	0	0	1	0	2

MLA

light
oneway
light
otherway

This program demonstrates the use of output bits A (0 and 1) and 4 to control a motor and a lamp.

MLB



LEGO *Lines*

	IN 7	6	OUT 5	4	3	2	1	0	
light	▨	▨	0	1	0	0	0	0	
REPEAT									
UNTIL	▨	1							
oneway	▨	▨	0	0	0	0	0	1	2
otherway	▨	▨	0	0	0	0	1	0	2

MLB

light

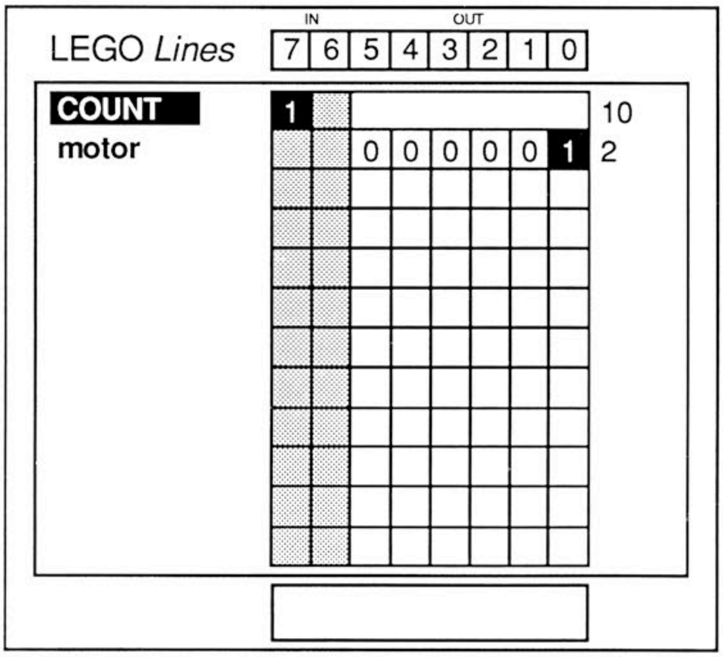
REPEAT

UNTIL

oneway
otherway

This program is designed to wait until a signal is received on bit 6 before the motor is switched on. The motor will rotate clockwise and anticlockwise for 2 seconds.

COUNT



LEGO *Lines*

	IN 7	6	OUT 5	4	3	2	1	0	
COUNT	1	▨							10
motor	▨	▨	0	0	0	0	0	1	2

COUNT

COUNT

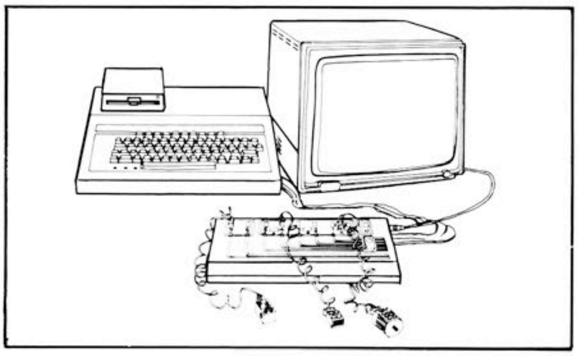
motor

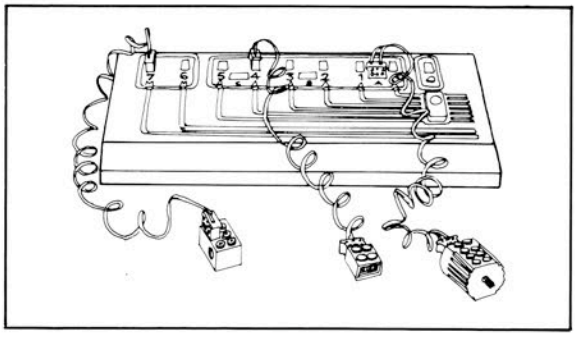
This program is designed to wait until the sensor is 'switched' 10 times before turning a motor on.

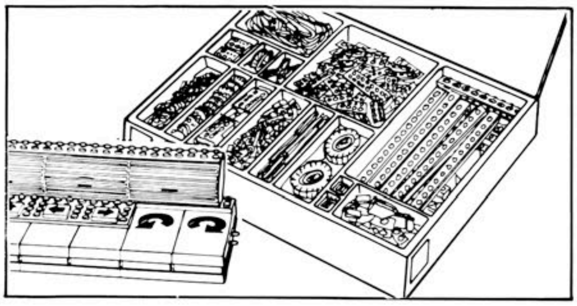
Section 2
Classroom materials

Programmable Systems equipment

Content







Recommended minimum quantities for class of 30 pupils working in groups of three:

10	LEGO 1090 Sets
10	Manual controllers 1039
5	LEGO Interfaces 1093
5	Computer systems
5	Working disks

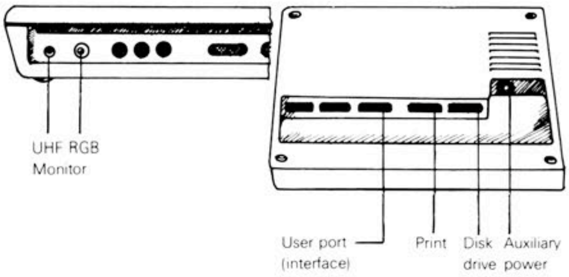
Hints on use

Familiarity with materials

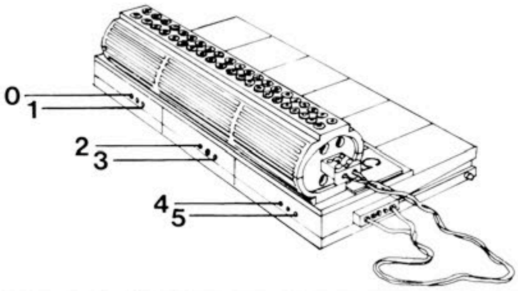
An introduction to the use of the equipment has been provided in the Teacher's materials.

Connecting the system

The computer system is connected as shown in the Resources booklet **R7**.



The manual controller



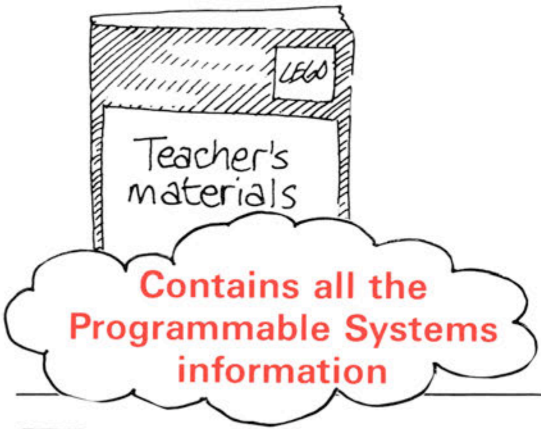
Storage and handling

Electronic equipment

Connecting leads

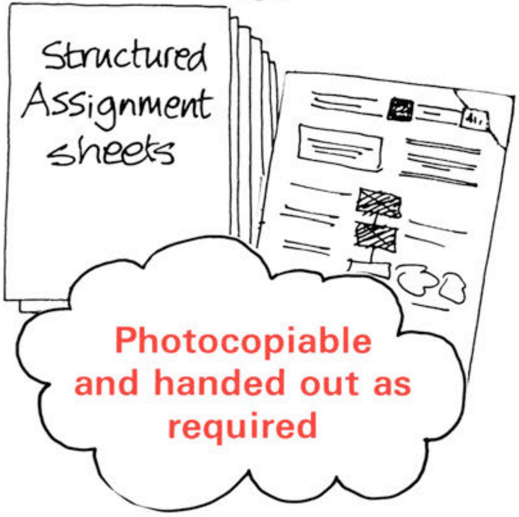
Software disks

Programmable Systems resources



Contains all the Programmable Systems information

Contents



Photocopiable and handed out as required

A0 Introduction to materials (a–c)
A1 Introduction to Programmable systems (a–b)
A2 Introduction to computer control (a–f)
A3 Introduction to manual control
A4 Sensing and feedback (a–b)
A5 Computer control (a–f)
A6 Manual control
A7 Solving a problem
A8 Designing a structured program
A9 Designing programmed systems (a–j)

Included in LEGO sets

1090 A Ferris wheel
B Automatic door
C Washing machine
D Double conveyor
E Robot arm
1039 Manual controller

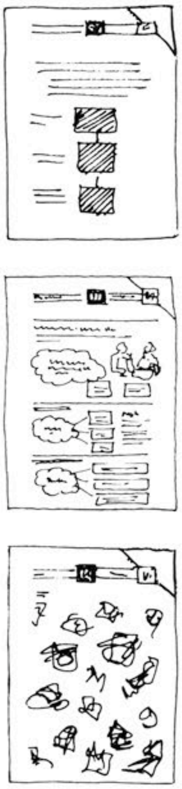
Recommended minimum quantities for class of 30 pupils working in groups of three:

Assignment sheets	As required	Manual controller Keystrips	1 per controller
Construction guides	1 per model	Program sheets	As required
Resources booklets	1 per group	Assessment sheets	As required
Keyboard strips	1 per computer		

Classroom materials

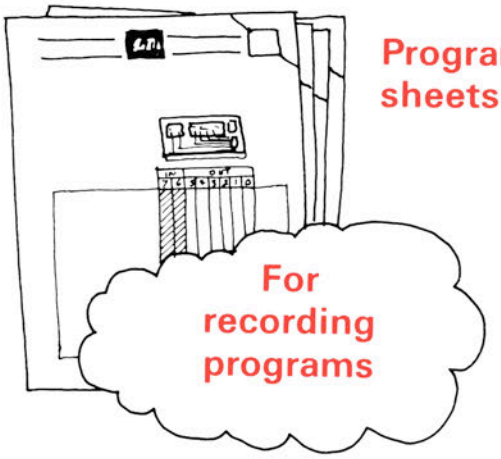
Guide to resources

How to use equipment and develop solutions to problems



Assessment sheets

For recording progress

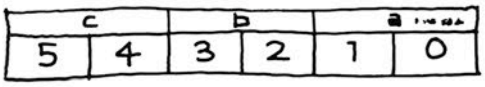


Program sheets

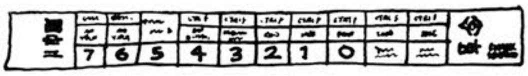
For recording programs

Keyboard strips

Manual controller

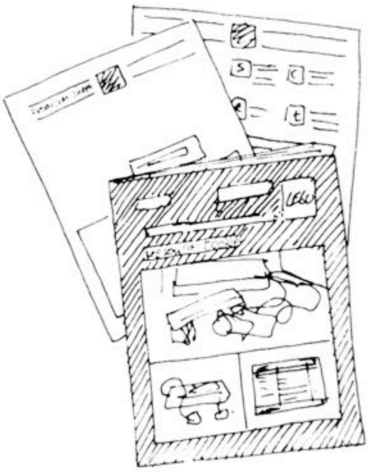


Computer



Guide to assignments

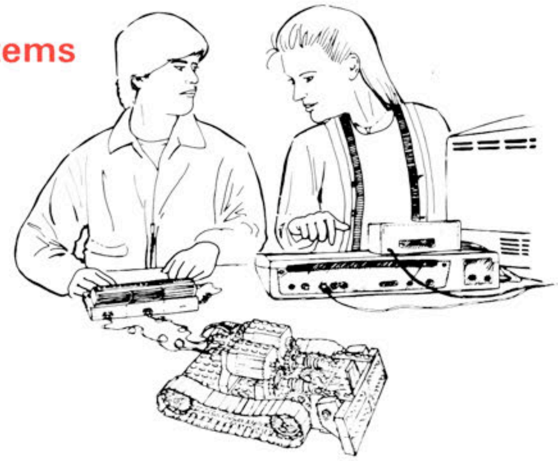
0 Introduction to the materials (page 2.8)



Gaining familiarity and confidence in the use of the materials.

Cards:	A0 (a–c)
Models:	1090 D (recommended)
Software:	CONVEY
Notes:	Teacher-led demonstration of hardware and software; students explore resources on their own.

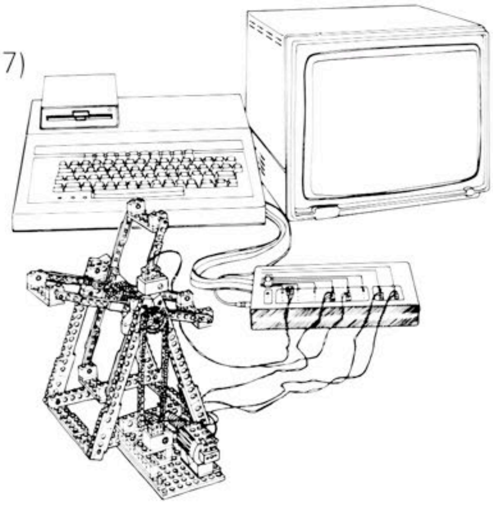
1 Introduction to Programmable Systems (page 2.13)



Controlling models using a computer and a manual controller.

Cards:	A1 (a–b)
Models:	1090 A, 1090 B, 1090 D, 1090 E
Software:	MLA (A1a only)
Notes:	Models need to be built before lesson.

2 Introduction to computer control (page 2.17)



Learning how to control using software.

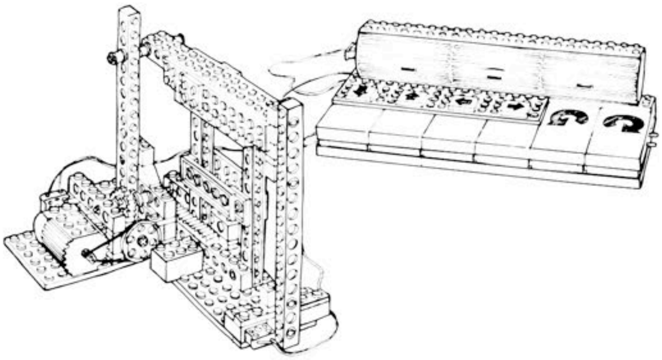
Cards:	A2 (a–f)
Software:	ONE to SIX
Notes:	Motors and lights are extracted from models as required. An opto-sensor brick is also required.

3 Introduction to manual control (page 2.29)

Learning how to control using a mechanical system.

Card: A3

Notes: Gearbox units can be assembled from any of the construction guides (1090 B for example) but an example is supplied.



4 Sensing and feedback (page 2.31)

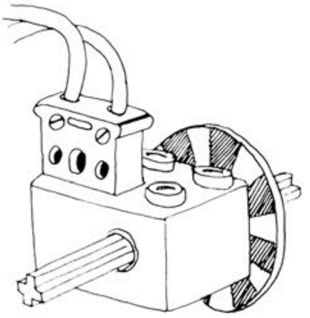
Learning how to use sensors.

Cards: A4 (a–b)

Models: 1090 A, 1090 B, 1090 D, 1090 E

Software: MLB, COUNT

Notes: Motors and lights are extracted from models as required. An opto-sensor brick is also required.



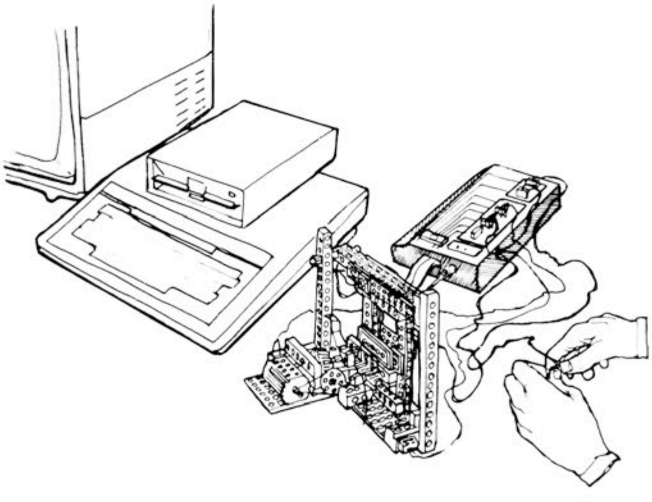
5 Computer control (page 2.35)

Learning how to use sensors for computer control.

Cards: A5 (a–f)

Software: SEVEN to TWELVE

Notes: Motors, light bricks and opto-sensors are extracted from models and used to test sub-systems.



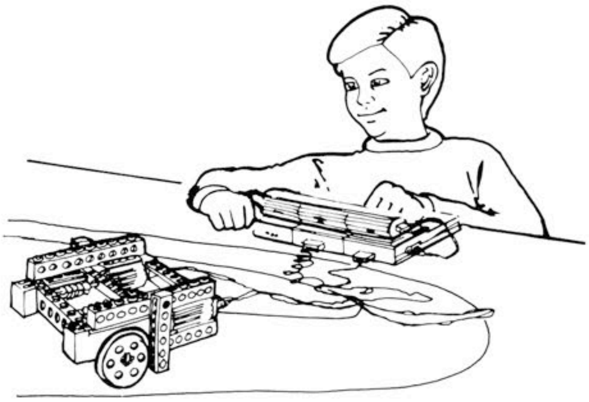
6 Manual control (page 2.47)

Learning about feedback through human control.

Card: A6

Model: A vehicle is built from the 1090 kit.

Notes: Models should be dismantled before and after this assignment.

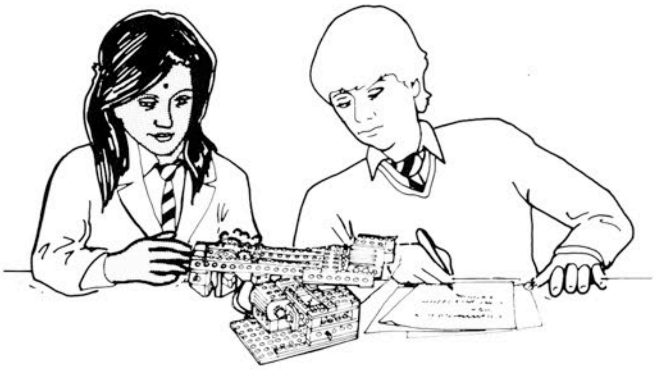


7 Solving a problem (page 2.49)

Learning a structured approach to solving problems.

Card: A7

Notes: Teacher may wish to have a demonstration buggy already built, showing a solution to the problems of Assignment 6.



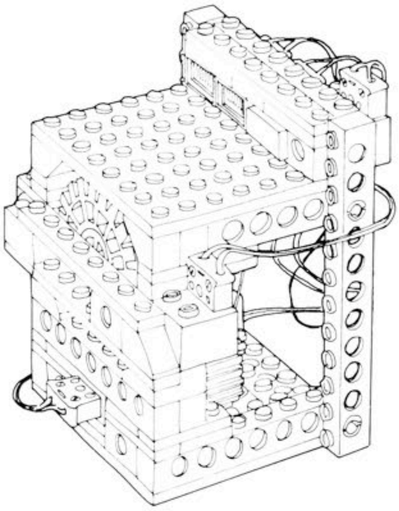
8 Designing a structured program (page 2.51)

Learning how to use the problem-solving process for computer control problems.

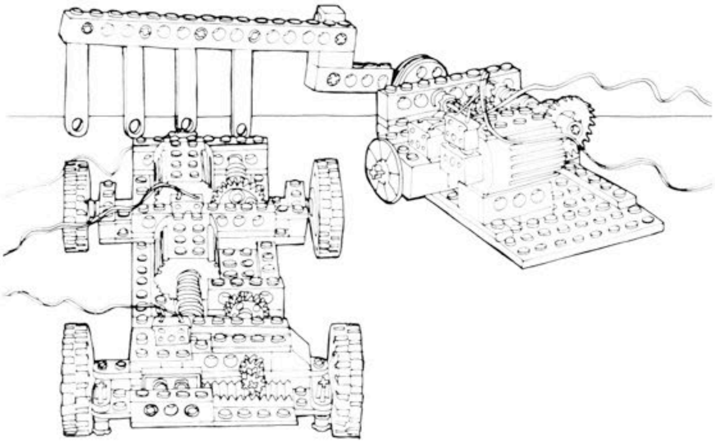
Card: A8

Model: 1090 C

Notes: 1090 C is constructed either before or during this assignment. There is no sample demonstration program for the problems posed.



9 Designing programmed systems (page 2.53)



Applying the process to a computer control problem.

Cards: A9 (a–j)

Notes: A series of open-ended problems to be solved using all available resources.

Assignments 9a-9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f-9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

The problems:

A9a A sliding door allowing customers in and out but otherwise remaining shut to conserve heat and energy.

A9b A safe and economic Ferris wheel ride.

A9c A device to pick LEGO bricks from a conveyor and deposit them on a platform nearby.

A9d A mobile space vehicle, capable of avoiding obstacles in its path.

A9e A conveyor system which can sort large boxes from small ones.

A9f A safe and efficient automatic barrier for a car park.

A9g A set of curtains which open and close automatically.

A9h An automatically controlled roundabout.

A9i A device to stamp a pattern on to a plasticine block.

A9j A theatrical head which comes alive as members of the cast pass it.

Purpose: Assignment 0

To introduce the students to the resources available.

Aims

Skill To introduce the use of:
— Resources booklet
— Program sheet
— Assignment sheet
— Assessment sheet
— Hardware and software

Attitude To develop confidence in the use of these materials.

Resources (each group)

Resources booklet
Program sheet for each student
Assignment sheet for each student
One-motor model (1090 B)
Two-motor model (1090 E)
Leads
Manual controller (without batteries)
Interface

Resources (teacher)

Computer system
LEGO *Lines* program disk
Sample programs disk

Notes

Teachers should ensure that the students are familiar with the materials before continuing with the other assignments. This will be achieved by this teacher-led assignment and a demonstration of the LEGO *Lines* software.

A suitable demonstration could be based on the use of the double conveyor (1090 D) which introduces many of the key concepts such as systems, feedback, sensing, processing and programming. Discussion could be encouraged which explores why such a machine might be useful and how the various bits (sub-systems) go to make up the whole machine. At this stage, showing the computer system in operation is more important than appreciating how it works.

The Program sheet and Assessment sheet corners have been left blank for the student to write on some reminder (a program name, perhaps) of the activity being pursued.

The answers to the questions are readily found and can themselves be the subject of debate and agreement. This applies particularly to those questions whose answer is a definition.

Assignment: 0

Introduction to the materials

A0a

You will need

Resources booklet
Program sheet
Assessment sheet

You are going to

Learn how to use the materials

Using the materials you are going to learn about three different ways to control devices:

— human control
— mechanical control
— electronic control

To do this, you will be given Assignment cards, like this one. This assignment, Assignment 0, comes in three parts, A0a, A0b and A0c.

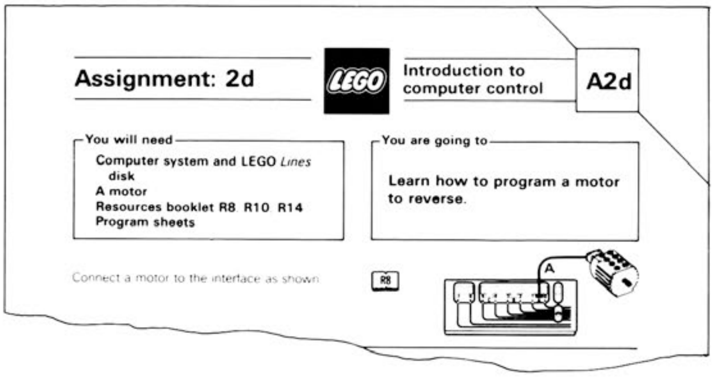
To help you in all the assignments, there are the Resources booklet, the Program sheet and the Assessment sheet. Make sure you have these in front of you for Assignment 0.

Assignment cards

The Assignment cards, like this one, are designed to be as easy to use as possible. On the top right-hand corner of each one you will find a reference number, A0a on this one (that is Assignment 0, part a). Underneath this you will find a statement of what you are going to do in the assignment.

The box on the top left-hand side of the card outlines what you will need in order to do the assignment. You should make sure you have these before you start.

The rest of the card tells you what to do. Drawings and sample programs take you through the activity. Where extra help is available in the Resources booklet, you are shown where to find it by the figure in the middle column like this: R8



Assignment: 2d

Introduction to computer control

A2d

You will need

Computer system and LEGO *Lines* disk
A motor
Resources booklet R8 R10 R14
Program sheets

You are going to

Learn how to program a motor to reverse.

Connect a motor to the interface as shown

R8

Program sheet

Use resources page **R11** to discover what you will be writing in each of the columns.

Assignment: 0

Introduction to the materials

A0b

You will need

Resources booklet
Program sheet

You are going to

Explore the Resources booklet

Resources booklet

You will find that this booklet contains a great deal of information. The *Guide to contents* helps you to find what you want quickly and a *Glossary of terms* explains some of the words you will meet.

Use the Resources booklet to answer these questions.

1 What is the purpose of this booklet?

2 What is *Technology*?

3 What resources would you use if you wanted help in designing control programs, using the LEGO materials?

4 What is the title of **R6**, and what kind of information do you find there?

5 If you were stuck because you didn't know what the term *Cursor block* means, where would you look for help? What is a Cursor block?

6 What is LEGO *Lines*?

7 What are *SFD*'s?

8 What does *Evaluation* mean?

9 If you had been told to produce a report, which resource page would you use?

10 Look at resource page **R10** and answer these questions:

What is a keyboard convention?

If you saw Press CTRL f8 in the text, what would you do?

Which keys are used to write labels?

Which keys are used to set output bits?

Which keys move the cursor block to the next line?

Which key would you use to start a program?

Which key stops a program temporarily?

Which key enables you to insert a new line?

Which key tests a single line of the program?

11 How would you connect a model with one light brick and one motor to the manual controller? How would you connect it to the interface (use the top of your Program sheet to answer). Do the same for a model with two motors and a light brick.

12 What is an output bit 4? An input bit 6? Output bits B?

Assignment: 0

Introduction to the materials

A0c

You will need

Assessment sheet
Resources booklet

You are going to

Learn how to use the Assessment sheet

Assessment sheet

This is to help you measure the success of what you are doing and of what you have learnt. At the top there are headings for you to use to describe how you have got on, next to boxes containing letters (these letters also appear on the bottom of every Assignment sheet). Help is available on these headings on the following Resources booklet pages:

R1 R2 R3 R4 R5

1 What does **i** stand for?

2 Where would you look to find out more about what **a i s e c** and **t** stand for?

3 What are *Criteria*?

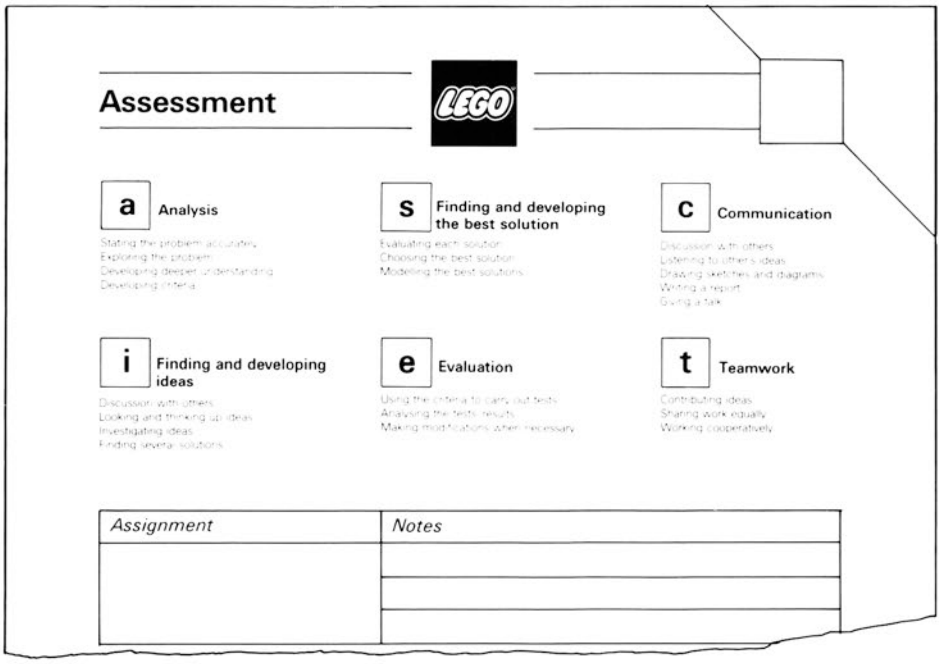
4 If you were asked how well you thought your group worked as a team, what questions would you ask yourself?

5 **c** has five elements in it. Which do you think will be the most difficult for you to achieve?

6 Do you think you will use all the items on the Assessment sheet in every single assignment you do?

7 Are there any terms you can't find out about on the Assessment sheet?

8 Fill in an Assessment sheet for Assignment 0, describing what you think you have learnt from answering these questions.



Assessment

a Analysis

Stating the problem accurately
Exploring the problem
Developing deeper understanding
Developing criteria

s Finding and developing the best solution

Evaluating each solution
Choosing the best solution
Modelling the best solutions

c Communication

Discussion with others
Listening to other's ideas
Drawing sketches and diagrams
Writing a report
Giving a talk

i Finding and developing ideas

Discussion with others
Looking and thinking up ideas
Investigating ideas
Finding several solutions

e Evaluation

Using the criteria to carry out tests
Analysing the tests results
Making modifications when necessary

t Teamwork

Contributing ideas
Sharing work equally
Working cooperatively

Assignment	*Notes*

Assignment: 1a

Purpose: Assignment 1

To control models using a computer and a manual controller.

Aims

Concept	Electrical devices can be controlled by computers.
Concept	A program consists of a series of instructions.
Skill	Manipulation of the manual controller to achieve a desired result.
Skill	Manipulation of a computer program to achieve a desired result.
Attitude	It is easy to alter the content of a control program in a software environment.

Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
One-motor model with leads (1090 A or 1090 B)
Resources booklet **R7**, **R8**, **R11**, **R12**

Sample programs

```
MLA
EGC
```

Notes

Groups of three students (maximum).

This part of the assignment will occupy at least one half of a double lesson.

One half of the class may be working on this while the other works on assignment, **A1b**.

Each group should complete both parts of this assignment in the time allocated.

Each student in the group should use the computer keyboard and gain confidence in the ability to control a computer system.

If desired, the class could be taken through the activity by the teacher using a demonstration of a model working, OHPs and the LEGO *Lines* documentation.

This would reinforce the work covered in Assignment 0 and establish the use of the Resources booklet.

Extension work

Further instructions could increase the complexity of the programs. These should be discussed before being put into effect.

In particular, students should be encouraged to record and describe their solution.

A program listing is available as output to a printer, by pressing **CTRL f7**, and this could form the basis of a description.

Troubleshooting

Keyboard problems. All students must establish control of the keyboard in this lesson.

All students should understand and use the following keystroke convention:

Press **Key 1**	Press key
Press **Key 1 Key 2**	Hold down first key, press other key, release first key

Program may have been altered by previous users or disk may be damaged.

Connections may be faulty or incorrectly set.

Model may not be a single motor model.

Disk drive or printer may not be standard.

Assignment: 1a

Introduction to programmable systems

A1a

You will need

Computer system with disk drive
LEGO *Lines* disk
1 motor model and light brick
Resource booklet

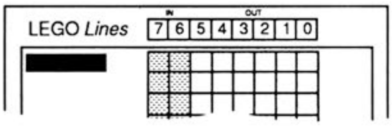
You are going to

Use a computer program to control a model system.

Connect computer system and insert disk. R7

Collect model and connect it to the interface as directed in the Resources booklet. R8

Loading LEGO *Lines*

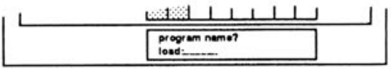


```
LEGO Lines      IN        OUT
              7 6 5 4 3 2 1 0
```

Press **SHIFT BREAK** to autoboot LEGO *Lines* R10

Loading a program

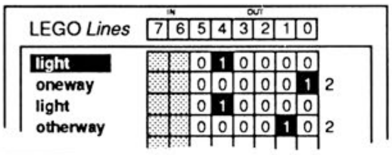
Press **CTRL f8** to load program — this message should appear →



```
program name?
load:______
```

Type **MLA** and then press **RETURN** to load program **MLA**.

This program is designed to turn a light on, rotate a motor one way for two seconds, turn the light on again and rotate the motor the other way for another two seconds.



```
LEGO Lines      IN        OUT
              7 6 5 4 3 2 1 0
light             0 1 0 0 0 0
oneway            0 0 0 0 0 1  2
light             0 1 0 0 0 0
otherway          0 0 0 0 1 0  2
```

Running the program

Press **TAB** to run the program — check that the system performs correctly. If not → check connections and then Try again

Changing the program

R10

Alter the time of rotation

Press ↓ then → and check that the cursor block has moved down and across

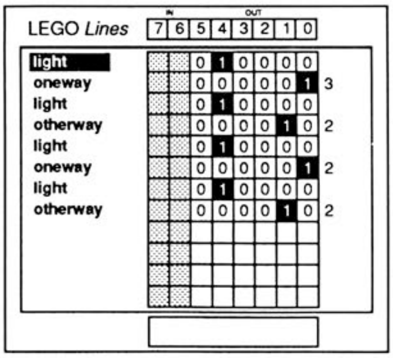
Press **DELETE** to remove the time on this line:

Oneway ■ ■ Ø Ø Ø Ø Ø 1 2.0

Enter a number between 1.0 and 3.5. Check to see it is entered correctly.

Press **TAB** to run the program. Then copy the program onto a Program sheet and describe what it does.

Enter more instructions to repeat the sequence



```
LEGO Lines      IN        OUT
              7 6 5 4 3 2 1 0
light             0 1 0 0 0 0
oneway            0 0 0 0 0 1  3
light             0 1 0 0 0 0
otherway          0 0 0 0 1 0  2
light             0 1 0 0 0 0
oneway            0 0 0 0 0 1  2
light             0 1 0 0 0 0
otherway          0 0 0 0 1 0  2
```

Press ↓ until the cursor block reaches the first blank line.

Copy the original instructions

Use: Letter keys for labels
Red keys for output values
Numbers for timing or counting
RETURN to go to the next line. R10

Press **TAB** to run program and then produce a printout of this 'new' program and describe what it does.

Purpose: Assignment 1

To control models using a computer and a manual controller.

Aims

Concept	Electrical devices can be controlled by computers.
Concept	A program consists of a series of instructions.
Skill	Manipulation of the manual controller to achieve a desired result.
Skill	Manipulation of a computer program to achieve a desired result.
Attitude	It is easy to alter the content of a control program in a software environment.

Resources (each group)

Manual controller (1039) with keystrip
Two-motor model with leads (1090 D or 1090 E)
Program sheets as required.
Batteries — spares

Notes

Groups of three (maximum).

This part of the assignment will occupy at least one half of a double lesson.

One half of the class may be working on this while the other works on assignment, **A1a**.

Each group should complete both parts of this assignment in the time allocated.

Each student in the group should use the manual controller and test a sequence of instructions they have written, preferably onto the Program sheet.

A description of each program in everyday English could accompany each set of instructions. Check that time values are being entered in the right hand column of the Program sheet.

Extension work

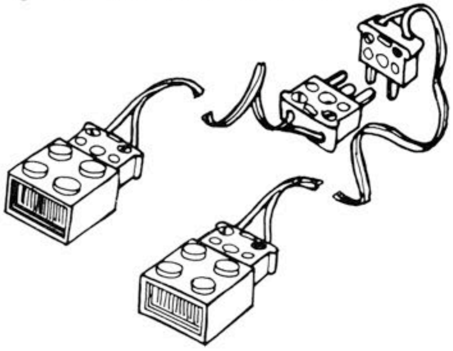
Repeat the assignment using a different two-motor model.

Troubleshooting

Two-motor model may not be connected correctly. Check **R8**.

The keyboard strip might not have been used to number keys correctly.

In this example, the lights should be connected by plugging one lead into the other before inserting into the manual controller.



Batteries may be dead — are extra ones available?

Assignment: 1b

Introduction to programmable systems

A1b

You will need

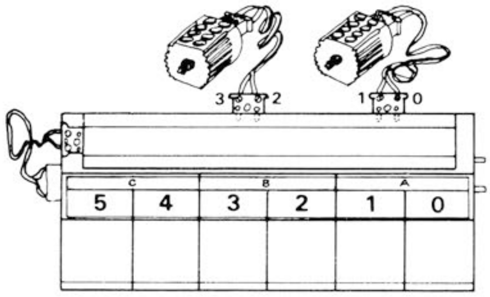
Manual controller and keystrip
Two-motor model (1090 D or 1090 E)
Resources booklet R6, R8
Program sheets

You are going to

Control a model using the manual controller.

Connect a two-motor model to the manual controller as shown in the Resources booklet.

R8



Press each key on the controller reasonably hard in turn.

Note what happens to the model.

A summary of your actions is given below:

The 1 means that key 1 is pressed or *on*

	5	4	3	2	1	Ø
Key 0	Ø	Ø	Ø	Ø	Ø	1
Key 1	Ø	Ø	Ø	Ø	1	Ø
Key 2						
Key 3						
Key 4						
Key 5						

Draw this table and complete it.

Try pressing keys 1 and 3 at the same time:

What happens to the model?

Why?

	5	4	3	2	1	Ø
Key 1 and Key 3	Ø	Ø	1	Ø	1	Ø

Check the direction the motors turn when you press keys 2 and Ø at the same time. If this is not the same, can you change the leads round so it is?

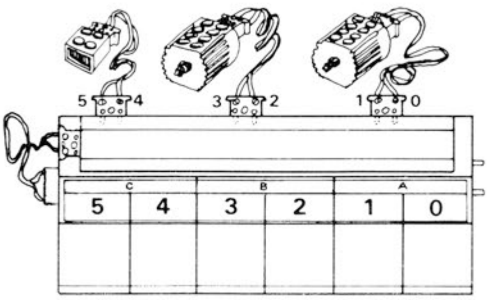
R8

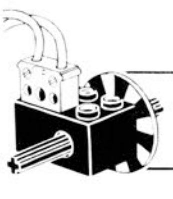
Here is a sequence of instructions.

	5	4	3	2	1	Ø	time
light on	Ø	1	Ø	Ø	Ø	Ø	5.0
one way	Ø	Ø	Ø	1	Ø	1	10.0
light on	Ø	1	Ø	Ø	Ø	Ø	5.0
other way	Ø	Ø	1	Ø	1	Ø	10.0

Instead of a key to press, we now have a *label* (to describe what is happening) and a time value to tell you how long to do it.

Now write some programs (on the program sheet) and try them out. Describe what you intend your models to do, and whether they do it.





Assignment: 2a

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept The output bit pattern is used to control lamps and motors.

Skill The ability to design a simple sequence of instructions.

Attitude It is straightforward to control motors and lamps using a computer system.

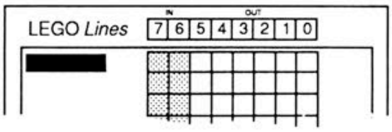
Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Resources booklet **R7**, **R8**, **R10**, **R11**
Program sheets as required.

Sample program

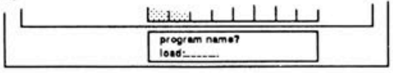
ONE

Press **SHIFT BREAK** to load *Lines*.



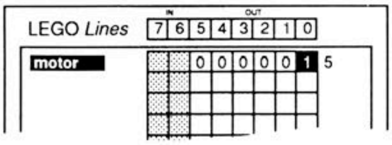
```
LEGO Lines   IN        OUT
             7 6 5 4 3 2 1 0
```

Press **CTRL f8** to load a *Lines* program.



```
program name?
load:______
```

Type **ONE** press **RETURN** to load program ONE.



```
LEGO Lines   IN        OUT
             7 6 5 4 3 2 1 0
motor            0 0 0 0 0 1  5
```

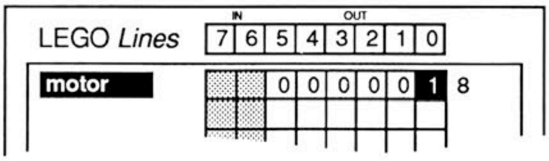
Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available, all groups could do this assignment simultaneously.



```
LEGO Lines   IN        OUT
             7 6 5 4 3 2 1 0
motor            0 0 0 0 0 1  8
```

Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

Assignment: 2a

Introduction to computer control

A2a

You will need

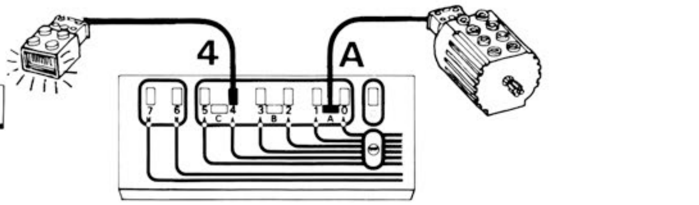
Computer system and LEGO *Lines* disk
A motor and a light brick
Resources booklet

You are going to

Learn how to control a motor and light brick.

Take a motor and a light brick from your model and connect them to the interface as shown.

R8



A washing machine manufacturer needs a control program which will spin clothes at the end of the wash cycle for 5 seconds.

The LEGO motor represents the washing machine motor.

R8

Possible solution

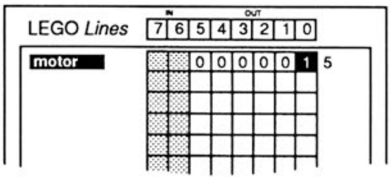
Press **SHIFT BREAK** to load LEGO *Lines*.

Press **CTRL f8** to load program.

Type **ONE** press **RETURN** to load ONE.

Press **TAB** to test program and describe what this instruction does.

R10



LEGO *Lines*

	IN 7	6	OUT 5	4	3	2	1	0	
motor			0	0	0	0	0	1	5

An electric fan (*represented by the LEGO motor*) is required to come on for 8 seconds when a button is pressed.

Possible solution

Change time of last solution

→ to move cursor block to 'time'

Press **DELETE**

Press **8** to enter new time.

Press **TAB** to test program and describe what it does — look especially at the interface.

R10

An outside light is required to come on for 8 seconds when a button is pressed.

Possible solution

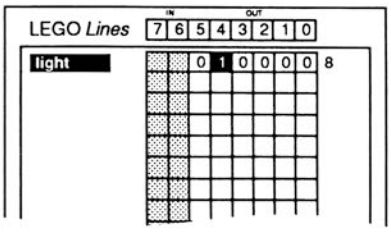
Press **f7** to set bit 0 off.

Press **f3** to set bit 4 on. (Look for a '1' in column 4).

DELETE label and enter another word to describe what is taking place.

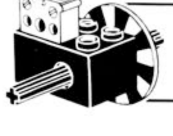
Press **TAB** to test program and describe what it does.

R10



LEGO *Lines*

	IN 7	6	OUT 5	4	3	2	1	0	
light			0	1	0	0	0	0	8



Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept The output bit pattern is used to control lamps and motors.

Skill The ability to design a simple sequence of instructions.

Attitude It is straightforward to control motors and lamps using a computer system.

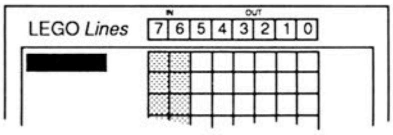
Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Resources booklet **R7**, **R8**, **R10**, **R11**
Program sheets as required.

Sample program

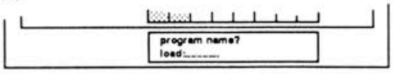
`TWO`

Press `SHIFT BREAK` to load *Lines*.



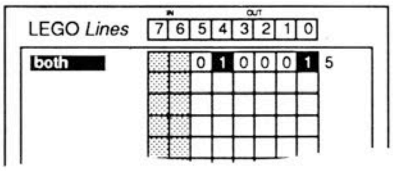
LEGO *Lines* IN OUT 7 6 5 4 3 2 1 0

Press `CTRL f8` to load a *Lines* program.



program name?
load:

Type `TWO` press `RETURN` to load program TWO.



LEGO *Lines* IN OUT 7 6 5 4 3 2 1 0

both 0 1 0 0 0 1 5

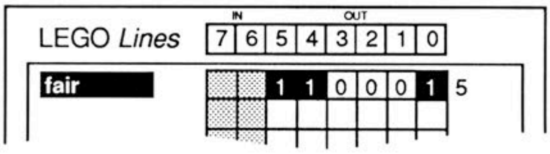
Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available, all groups could do this assignment simultaneously.



LEGO *Lines* IN OUT 7 6 5 4 3 2 1 0

fair 1 1 0 0 0 1 5

Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

Assignment: 2b

Introduction to computer control

A2b

You will need

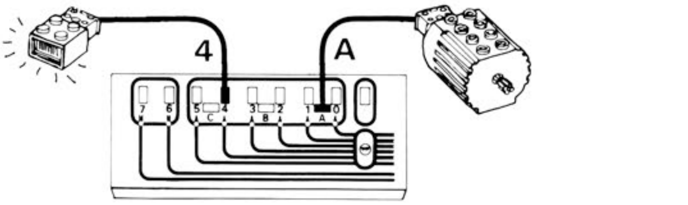
Computer system and LEGO *Lines* disk
A motor and two light bricks
Resources booklet R8, R10
Program sheets

You are going to

Learn how to control two or more output devices.

Use a light brick and a motor and connect them to the interface as shown.

R8



A program is required which operates a fairground ride (*represented by the LEGO motor*) and a light which shows that the ride is in progress for 5 seconds.

Possible solution

Connect motor to output bits A and lamp to output bits 4 on the interface.

R8

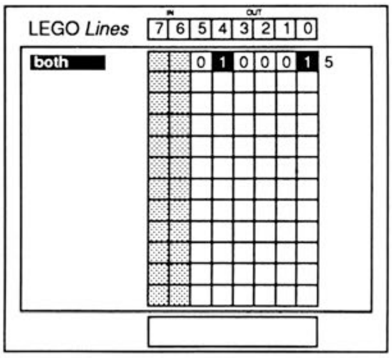
Press **CTRL f8** to load program.

Type **TWO** press **RETURN** to load TWO.

Press **TAB** to test program.

R10

Describe what happens on screen, on the interface and to the motor and light.



LEGO *Lines*	7	6	5	4	3	2	1	0	
both	▨	▨	0	1	0	0	0	1	5

Improve the fairground ride to include two lights.

Possible solution

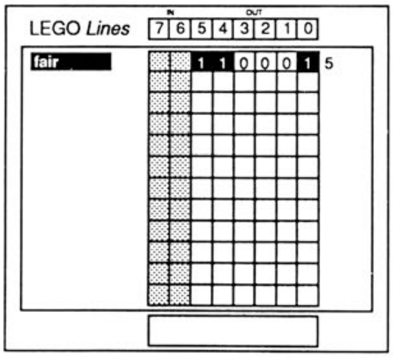
Connect another light brick to the interface.

Change the line so that both the lights and the motor are on for 5 seconds.

Press **TAB** to test program.

R10

Make a record of your program on the Program sheet, obtain a printout of it (by pressing **CTRL f7**) and describe how it solves the problem.



LEGO *Lines*	7	6	5	4	3	2	1	0	
fair	▨	▨	1	1	0	0	0	1	5

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept The output bit pattern is used to control lamps and motors.

Skill The ability to design a simple sequence of instructions.

Attitude It is straightforward to control motors and lamps using a computer system.

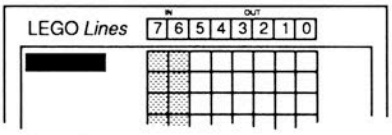
Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Resources booklet **R7**, **R8**, **R10**, **R11**
Program sheets as required.

Sample program

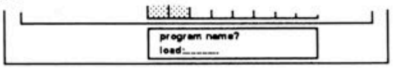
THREE

Press **SHIFT BREAK** to load *Lines*.



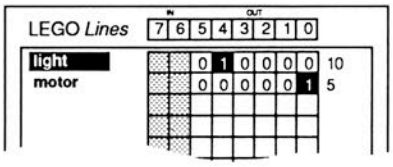
```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
```

Press **CTRL f8** to load a *Lines* program.



```
program name?
load:______
```

Type **THREE** press **RETURN** to load program THREE.



```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
light            0 1 0 0 0 0  10
motor            0 0 0 0 0 1  5
```

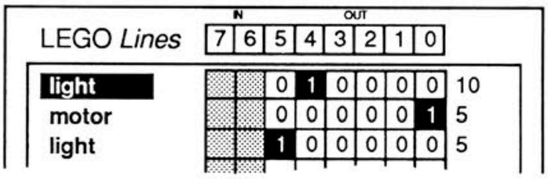
Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available, all groups could do this assignment simultaneously.



```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
light            0 1 0 0 0 0  10
motor            0 0 0 0 0 1  5
light            1 0 0 0 0 0  5
```

Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

Assignment: 2c

LEGO

Introduction to computer control

A2c

You will need

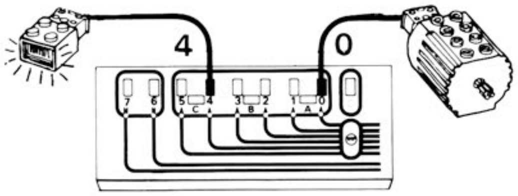
Computer system and LEGO *Lines* disk
A motor and two light bricks
Resources booklet R8, R10
Program sheets

You are going to

Learn how to program a sequence of instructions.

Connect the light brick and the motor to the interface as shown.

R8



A program is needed which will turn on a light 10 seconds before a motor is started up.

Possible solution

Connect motor to output bit 0.

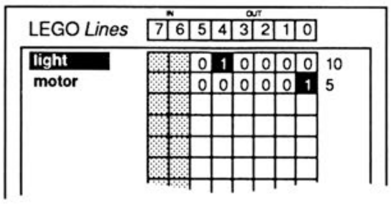
Connect lamp to output bit 4.

Press **CTRL f8** to load a program.

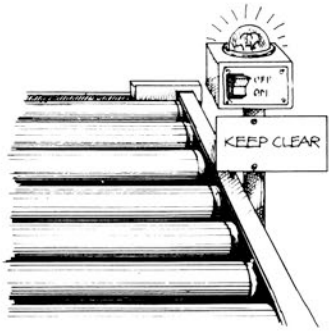
Type **THREE** press **RETURN** to load program THREE.

R10

Press **TAB** to test program and describe what happens.


LEGO *Lines*

	7	6	5	4	3	2	1	0	
	IN		OUT						
light			0	1	0	0	0	0	10
motor			0	0	0	0	0	1	5



Modify the solution to turn on another lamp for 5 seconds after the motor has been turned off.

Possible solution

R8

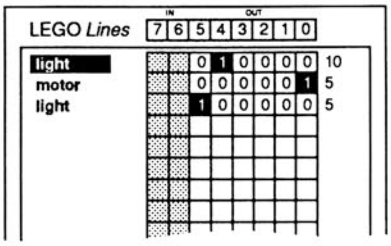
Connect another light brick to the interface.

Modify the program by adding an extra line.

Position the cursor block at the start of this line and test it by pressing **COPY**.

R10

Test the new program by pressing **TAB**.


LEGO *Lines*

	7	6	5	4	3	2	1	0	
	IN		OUT						
light			0	1	0	0	0	0	10
motor			0	0	0	0	0	1	5
light			1	0	0	0	0	0	5

Make a record of your program on the Program sheet, obtain a printout of it (by pressing **CTRL f7**) and describe how it solves the problem.

Assignment: 2d

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept The output bit pattern is used to control lamps and motors.

Skill The ability to design a simple sequence of instructions.

Attitude It is straightforward to control motors and lamps using a computer system.

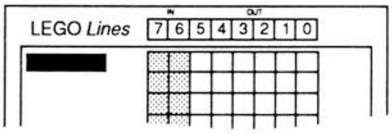
Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor with leads
Resources booklet **R7**, **R8**, **R10**, **R11**, **R14**
Program sheets as required.

Sample program

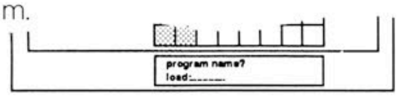
FOUR

Press **SHIFT BREAK** to load *Lines*.



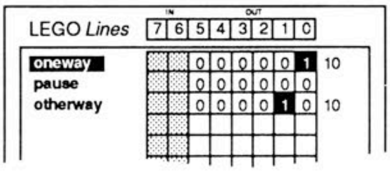
```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
```

Press **CTRL f8** to load a *Lines* program.



```
program name?
load:______
```

Type **FOUR** press **RETURN** to load program FOUR.



```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
oneway           0 0 0 0 0 1  10
pause            0 0 0 0 0 0
otherway         0 0 0 0 1 0  10
```

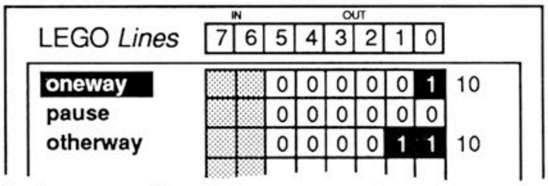
Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available, all groups could do this assignment simultaneously.



```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
oneway           0 0 0 0 0 1  10
pause            0 0 0 0 0 0
otherway         0 0 0 0 1 1  10
```

Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

Assignment: 2d

Introduction to computer control

A2d

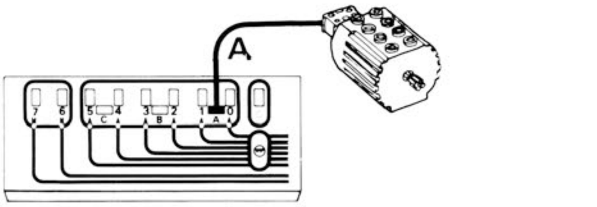
You will need

Computer system and LEGO *Lines* disk
A motor
Resources booklet R8, R10, R14
Program sheets

You are going to

Learn how to program a motor to reverse.

Connect a motor to the interface as shown. R8



A program is needed to raise a barrier, pause and then lower the barrier again.

The solution should operate when a button (TAB) is pressed. The LEGO *motor represents the barrier motor.*

Possible solution

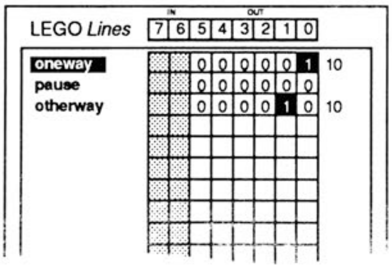
Press **CTRL f8** to load program.

Type **FOUR** press **RETURN** to load FOUR.

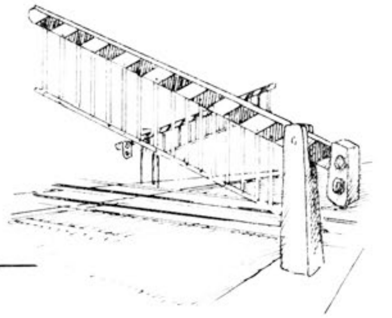
Press **TAB** to test program. R10

Can you change the program so that the barriers pause for at least 15 seconds before starting again? R14

Draw a structured flow diagram giving a solution to this problem and describe your solution.



LEGO *Lines*	7	6	5	4	3	2	1	0	
oneway			0	0	0	0	0	1	10
pause			0	0	0	0	0	0	
otherway			0	0	0	0	1	0	10



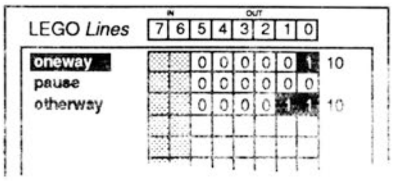
Investigation

Change the last line of the program so that the output bits 0 and 1 are both on.

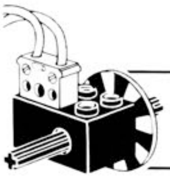
Press **COPY** to test new line. R10

Explain what happens. R19

Can you think of any everyday situations where this feature could be of any practical use? R20 R21



LEGO *Lines*	7	6	5	4	3	2	1	0	
oneway			0	0	0	0	0	1	10
pause			0	0	0	0	0	0	
otherway			0	0	0	0	1	1	10



a i s e c t

Assignment: 2e

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept The output bit pattern is used to control lamps and motors.

Skill The ability to design a simple sequence of instructions.

Attitude It is straightforward to control motors and lamps using a computer system.

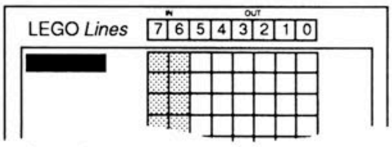
Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Resources booklet **R7**, **R8**, **R10**, **R11**, **R13**
Program sheets as required.

Sample program

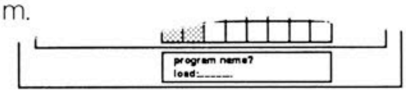
`FIVE`

Press `SHIFT BREAK` to load *Lines*.



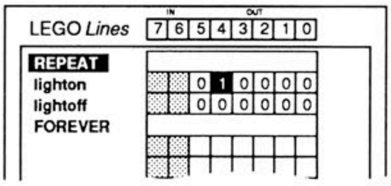
```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
```

Press `CTRL f8` to load a *Lines* program.



```
program name?
load:______
```

Type `FIVE` press `RETURN` to load program FIVE.



```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
REPEAT
lighton          0 1 0 0 0 0
lightoff         0 0 0 0 0 0
FOREVER
```

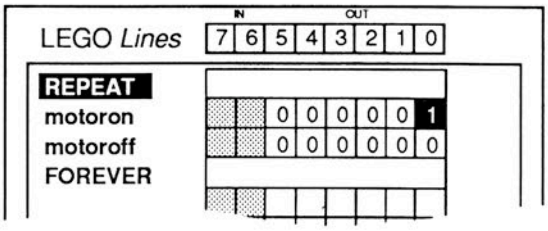
Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available, all groups could do this assignment simultaneously.



```
LEGO Lines   IN      OUT
             7 6 5 4 3 2 1 0
REPEAT
motoron          0 0 0 0 0 1
motoroff         0 0 0 0 0 0
FOREVER
```

Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

Assignment: 2e

Introduction to computer control

A2e

You will need

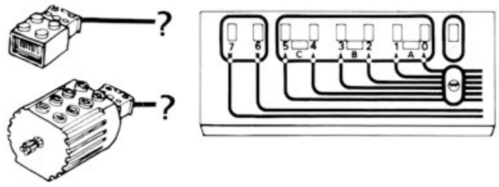
Computer system and LEGO *Lines* disk
A motor and two light bricks
Resources booklet R7, R8, R10, R11, R13
Program sheets

You are going to

Learn how to program a REPEAT FOREVER loop.

Use a motor and a light brick. You will have to work out from the programs where to connect them on the interface.

R8



A burglar alarm system requires a light which will flash on and of until a reset button is pressed.

Possble solution

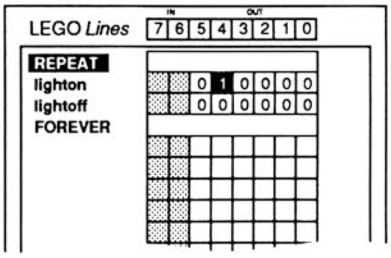
Press **CTRL f8** to load program.

Type **FIVE** press **RETURN** to load FIVE.

Connect the light brick to the correct output, once you have loaded the program.

Press **TAB** to test program.

R10



LEGO *Lines*	IN 7	6	OUT 5	4	3	2	1	0
REPEAT								
lighton	▨	▨	0	1	0	0	0	0
lightoff	▨	▨	0	0	0	0	0	0
FOREVER								
	▨	▨						
	▨	▨						
	▨	▨						
	▨	▨						
	▨	▨						

Investigation

Press **SPACE** to halt program.

R10

Press **SPACE** to continue program.

Press **ESCAPE** to end program.

The **ESCAPE** key acts as a reset button. When you press it, the program stops and the light no longer flashes.

Copy the instructions onto a Program sheet and describe what is happening.

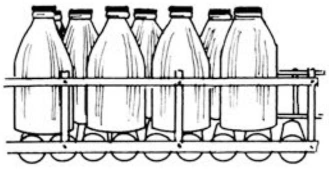
Design a program to control a machine which presses tops onto milk bottles as they move along a continuous conveyor belt.

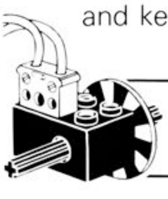
R13

What kind of device makes a conveyor move?
What kind of device can press a top onto a bottle?
How will the process of pressing the tops start and stop?

Use your answers to these questions to develop a solution to the problem using the LEGO materials.

Use the Program sheet to develop your solution and keep a record of your progress.





a | i | s | e | c | t

Assignment: 2f

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept The output bit pattern is used to control lamps and motors.

Skill The ability to design a simple sequence of instructions.

Attitude It is straightforward to control motors and lamps using a computer system.

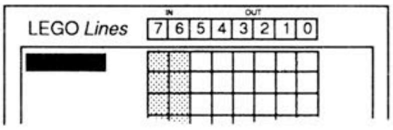
Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Resources booklet **R7**, **R8**, **R10**, **R11**, **R13**
Program sheets as required.

Sample program

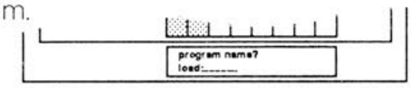
SIX

Press **SHIFT BREAK** to load *Lines*.



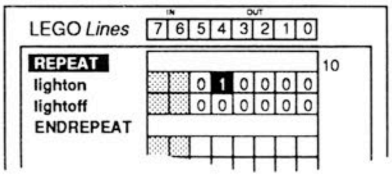
```
LEGO Lines   IN        OUT
             7 6 5 4 3 2 1 0
```

Press **CTRL f8** to load a *Lines* program.



```
program name?
load:______
```

Type **SIX** press **RETURN** to load program SIX.



```
LEGO Lines   IN        OUT
             7 6 5 4 3 2 1 0
REPEAT                         10
lighton          0 1 0 0 0 0
lightoff         0 0 0 0 0 0
ENDREPEAT
```

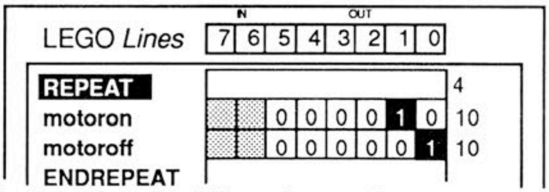
Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available, all groups could do this assignment simultaneously.



```
LEGO Lines   IN        OUT
             7 6 5 4 3 2 1 0
REPEAT                         4
motoron          0 0 0 0 1 0   10
motoroff         0 0 0 0 0 1   10
ENDREPEAT
```

Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

Assignment: 2f

You will need

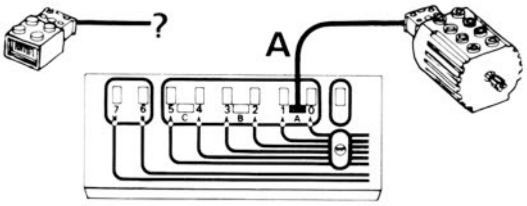
Computer system and LEGO *Lines* disk
A motor and two light bricks
Resources booklet R7, R8, R10, R11, R13
Program sheets

You are going to

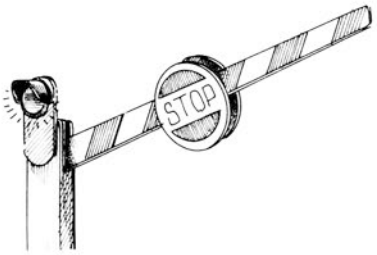
Learn how to program a REPEAT ENDREPEAT loop to run a set of instructions a number of times.

Use a motor and a light brick. Connect the motor to the interfaces as shown. You will have to work out from the programs where to connect the light brick.

R8



A level crossing barrier uses a flashing light to warn motorists it is closing. Design a program which will make this light flash 10 times.



Possble solution

Press **CTRL f8** to load program.

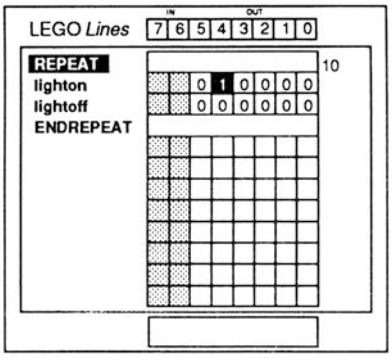
Type **SIX** press **RETURN** to load SIX.

Connect the light brick to the correct output, once you have loaded the program.

Press **TAB** to test program.

R10

Describe what happens in this program. How could the motor be included in this program?



LEGO *Lines*	IN		OUT						
	7	6	5	4	3	2	1	0	
REPEAT									10
lighton	▨	▨	0	1	0	0	0	0	
lightoff	▨	▨	0	0	0	0	0	0	
ENDREPEAT									
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							

A washing machine manufacturer requires a control program which will rotate the drum one way for 10 seconds and then rotate it the opposite way for 10 seconds. This sequence needs to be repeated 4 times.

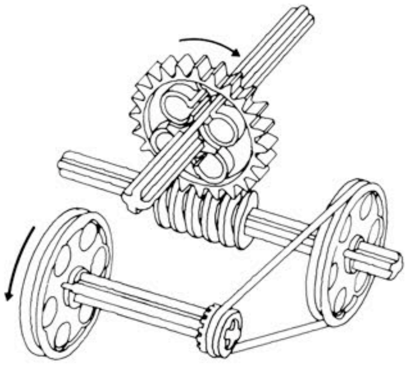
Using a light brick (*representing an on/off light*) and a LEGO motor (*representing the motor turning the drum*), write a program which solves this problem.

R13

Keep a record, on a Program sheet, of how your program develops.

Purpose: Assignment 3

To investigate mechanical control using a gearbox system.



Aims

Concept Gears and pulleys can change the speed, directions and torque of rotating shafts.

Concept Speed change is related to the relative number of teeth/ diameter of the gear or pulley system.

Skill Ability to interpret and use information to solve simple problems.

Skill Ability to apply mathematics to a practical problem.

Resources (each group)

LEGO model and leads
Construction Guide for model
Manual controller
Resources booklet **R6**, **R16–R21**
Stop clocks

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A2 (a–f)**.

A group should complete this assignment in the time allocated.

This assignment forms a very important basis for students' understanding of mechanical systems and mechanical control.

A record of the investigations in this assignment should be kept for future reference (especially for Assignments 6, 7 and 9).

This assignment requires a considerable amount of planning by the teacher. To avoid undue destruction of existing models, it would be desirable to be able to build gearboxes out of spare bits, or other LEGO Technic kits. A design is suggested on the student's assignment.

The output shaft speed of the gearbox should be:
speed of input motor × tooth ratio gear 1
× tooth ratio gear 2
and so on.

If a 16 tooth gear and a 24 tooth gear were used then the output shaft speed would be:
4000 × 1/16 × 1/24 rpm
= 10.42 rpm.

The actual output speed may differ significantly from this calculated speed because of friction and mechanical inertia. This may require some explanation.

Extension work

Identify everyday items which might make use of gearbox systems.

Investigate characteristics of:
Gear trains
Sprockets and chains (1090 A Ferris wheel)
Belt and pulley (1090 B Automatic door)
(**R16–R18**)

Troubleshooting

Has the sequence of instructions on the assignment been followed?

Calculations of shaft speed may prove too difficult.

Assignment: 3

Introduction to manual control

You will need

A model
A manual controller
Resources booklet R6, R16–21
Timing device (wristwatch)

You are going to

Learn how mechanical gears can control how quickly a motor turns a shaft.

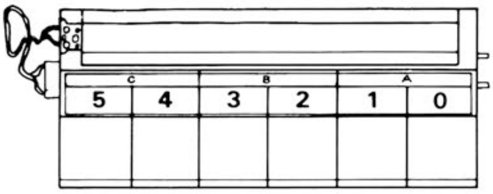
R6

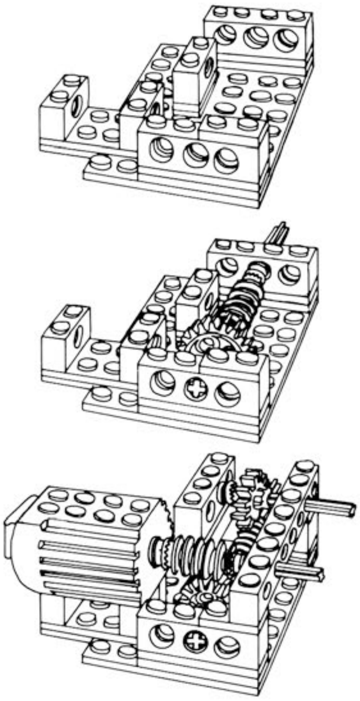
Connect the model to the manual controller.

Make sure the connections are properly made.

Turn the motor on.

The model mechanism (the output shaft) turns more slowly than the motor which is driving it (the input shaft). Why should this happen?





Build a gearbox like this one (you may need additional materials).

R18

Why do the output shafts of this gearbox have different speeds to the motor?

Discover if axles which turn quickly have a greater or smaller turning force than slower axles. Try slowing them down with your finger. Which is easier to stop?

Look at the section in the Resources booklet on gearboxes and drive mechanisms.

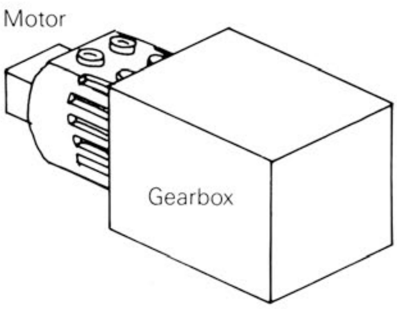
R16

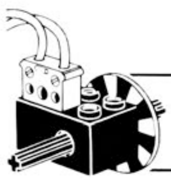
R18

How could you design a gearbox which will make your model move more slowly?

Use sketches and descriptions to record your ideas.

Build and test your design.





Assignment: 4a

Purpose: Assignment 4

To learn how to use the opto-sensor brick and to appreciate the limitations of human sensing and feedback.

Aims

Concept	The input of information needs a sensor.
Concept	Feedback is necessary for accurate control.
Attitude	It is easy to control a device using a computer system.

Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Opto-sensor brick with leads
Resources booklet **R7**, **R9**, **R10**, **R12**

Sample programs

```
MLB
COUNT
```

Notes

Groups of three students (maximum).

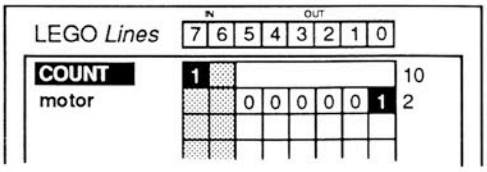
This part of the assignment should be completed in one half of the time allocated.

One half of the class may be working on this while the other works on **A4b**.

The opto-sensor brick responds to changes of light conditions. In the light, it could be either ON or OFF. This state can be 'flipped' by placing a finger over the sensor and removing it quickly. This feature should be explained to the students.

This assignment has been written so as to avoid stating the condition of the opto-sensor because it is indeterminate. The assignment is looking for *change* of light conditions.

The opto-sensor brick will work equally well in input bit 6 or 7 of the interface. They do not, of course, have any function if connected to any output bit.



```
                 IN        OUT
LEGO Lines      |7|6|5|4|3|2|1|0|

  COUNT         |1| |           |  10
  motor         | | |0|0|0|0|0|1|  2
```

Extension work

Gain further familiarity with input bits.

Explore uses of COUNT. In particular, this program could be used to give a more accurate measure of the gearbox speed calculated in Assignment 3.

Use the 4-segmented disc on the output shaft. Revise COUNT to count 400 sections on the disc.

Time this accurately (= 100 revs). Calculate number of revs per minute.

Troubleshooting

Check input bit connections and check one input bit is set to `any value` when using COUNT.

Check action of sensor.

Assignment: 4a

Sensing and feedback

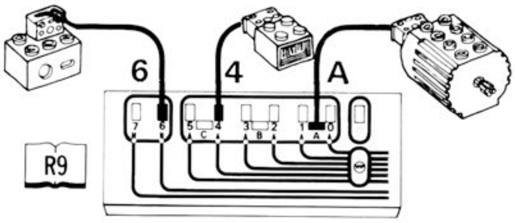
A4a

You will need

Computer system and LEGO *Lines* disk
A model with a light brick, a motor, and an opto-sensor brick
Resources booklet R7, R9–R12

You are going to

Learn how to use the opto-sensor brick.



R9

Connect the motor on the model to output bits A.

Connect the light brick and opto-sensor to the interface as shown.

On the interface bits 7 and 6 are input bits — that is, bits which let the computer system receive information from its environment.

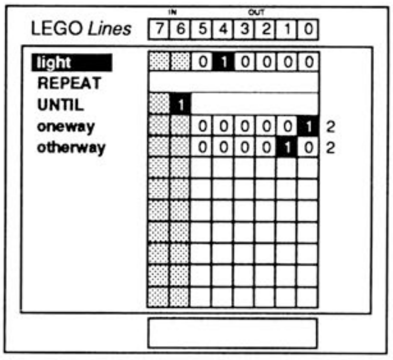
R10

Is the green indicator light for bit 6 on or off? Place your finger over the sensor slot — the indicator light will change. When you remove your finger it should return to its original state. Connect the opto-sensor to bit 7. Does it behave in the same way?

Describe what you think is happening.

Reconnect the opto-sensor to bit 6 and connect it to your model. Connect the light brick to the model so that it shines into the opto-sensor, but 3 cms away from it.

Load the program called **MLB**. This program is designed to wait for a signal from the opto-sensor before switching the motor on for 2 seconds.



LEGO *Lines*

	IN 7	6	OUT 5	4	3	2	1	0	
light	▨	▨	0	1	0	0	0	0	
REPEAT									
UNTIL	▨	1							
oneway	▨	▨	0	0	0	0	0	1	2
otherway	▨	▨	0	0	0	0	1	0	2

R9

Press **TAB** to run the program and use your finger to trigger the sensor.

Alter input sense

R11

Pressing ↓↓ will take the cursor block down to the third line. Press **f1** to alter bit 6.

Press **TAB** to run the program again.

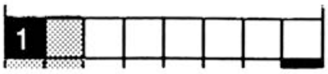
Can you explain why these two sets of instructions should behave differently?

Alter input line

Disconnect the opto-sensor from 6 and reconnect it to bit 7 on the interface.

Press ↓↓ and then press **SHIFT f1** to make bit 6 **any value**.

Press **f0** to alter the value of bit 7 to a 1 and press **TAB**.



▨ in bit 6 means 'any value'.

The sense of bit 6 will not matter when the UNTIL *looks for a signal.*

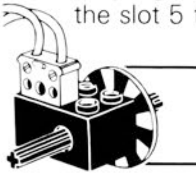
Counting input signals

R10

Load and run a program called **COUNT**.

This program waits for you to cover (and uncover) the sensor slot 10 times before continuing and turning the motor on.

Alter the program so that you only have to cover and uncover the slot 5 times before the motor is turned on.



a i s e c t

Purpose: Assignment 4

To learn how to use the opto-sensor brick and to appreciate the limitations of human sensing and feedback.

Aims

Concept The input of information needs a sensor.

Attitude It is easy to control a device using a computer system.

Resources (each group)

LEGO model and leads
Manual controller
Stop clocks
Resources booklet **R6**, **R12**

Notes

Groups of three students (maximum).

This part of the assignment should be completed in one half of the time allocated.

One half of the class may be working on this while the other works on **A4a**.

Part A of this assignment shows that the human opto-sensor operates quite quickly. The lack of control in part B arises from the slowness of the feedback loop. This contrasts with the computer which provides a fast feedback system.

There are, however, other concepts at work here. The human system has considerable inertia associated with it and the actual speed of the signal between the various parts of the system is negligible when compared to this delay.

Students should be encouraged to describe the tasks set in this assignment.

Extension work

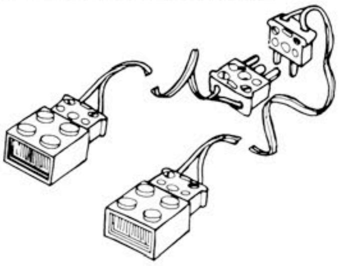
Students can be asked to look for other examples of feedback loops and to write descriptions of how they work.

Troubleshooting

The two-motor model may not be connected correctly.

The keyboard strip might not have been used to number keys correctly.

In this example, the lights should be connected by plugging one lead into the other before inserting into the manual controller.



Batteries may be dead — are extra ones available?

Assignment: 4b

Sensing and feedback

A4b

You will need

A model
Manual controller
Resources booklet R6, R12

You are going to

Learn about human sensing and feedback.

Part A

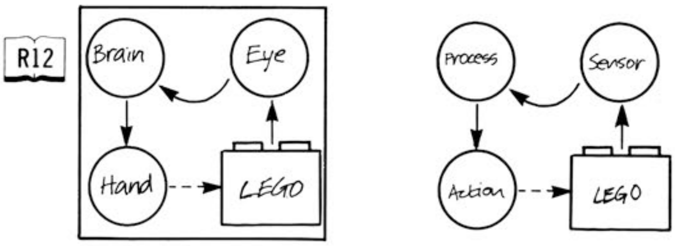
Decide on a task which fits your model (this should take about one minute to complete).

For example, use model 1090 A, the Ferris wheel.

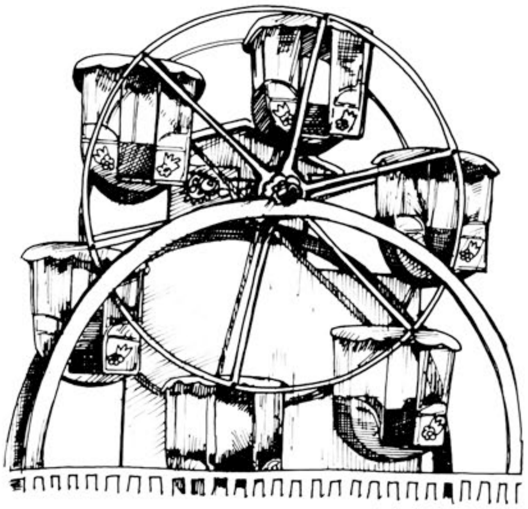
The task is to rotate the wheel so that each seat comes to rest at the bottom in turn so passengers can get on and off.

Use the manual controller and let each member of the group have a try. Make a note of the time it takes for each person to offload all the passengers.

Using the diagram, describe what is happening when a person performs this task (use the keywords ***sense***, ***process*** and ***act*** in your description).



Part B



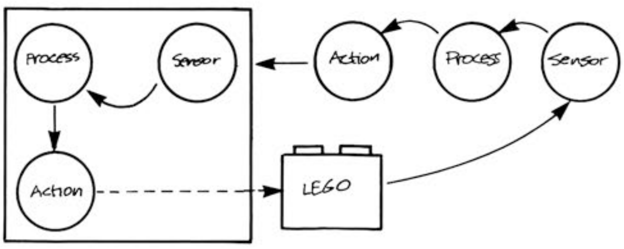
Whatever task you chose, you should now repeat it. This time, however, the person using the manual controller should have their eyes shut, so they cannot see the model.

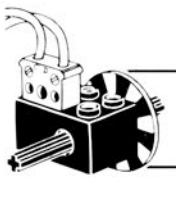
Another member of the group must now give instructions to the controller so the task can be performed. These might be 'Go forward', 'Back a bit' or 'Whoa!', whatever seems to work.

Again, you should make a note of the time it takes to complete the task.

How different were the times it took to complete the task?

Using the diagram, describe what happened (use the keyword ***feedback*** in your description). Try to explain why the tasks in part B took longer.





Purpose: Assignment 5

To learn the second set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept	Computers receive information through an input port.
Concept	Computers can be programmed to make decisions based on the input bit pattern.
Skill	The ability to design a simple sequence of instructions using the `REPEAT`, `UNTIL`, `IF`, `ENDIF` and `COUNT` keywords.
Attitude	It is straightforward to control devices using a computer system.

Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
A motor and a light brick with leads
Two opto-sensor bricks with leads
Resources booklet **R7–R12**

Sample program

```
SEVEN
```

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A6**.

A group should complete this assignment in the time allocated.

If sufficient computers are available, all groups could do this assignment simultaneously (interfaces will be required).

Assignments 5a–5f introduce further software techniques and they must be completed in the set sequence.

Hard copies are required to document the exercises. Students should use the following:

  Program sheets

  Structured flow diagrams from printouts (`CTRL f7`)

  Descriptions of each program.

From this point it is assumed that students know how to `LOAD` particular programs.

The focus of this assignment is on the new software commands. It is equally important that the relationship between what happens on the screen and on the interface is fully understood.

Extension work

Students can alter the set program.

Troubleshooting

Check that connections are not misplaced.

Difficulties may be experienced in understanding how the opto-sensor works.

In Assignment 5a the light brick is used to force an initial state but in other assignments 5b–5f it is recommended that students hold the sensor(s) so as to control its state.

Assignment: 5a

Computer control

A5a

You will need

Computer system and LEGO *Lines* disk
A motor, a light brick and an opto-sensor brick
Resources booklet R7–R10

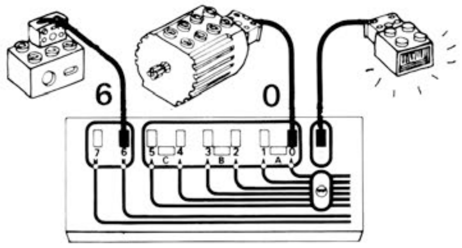
You are going to

Learn how to design a program which will REPEAT a sequence of instructions UNTIL an input message is sensed.

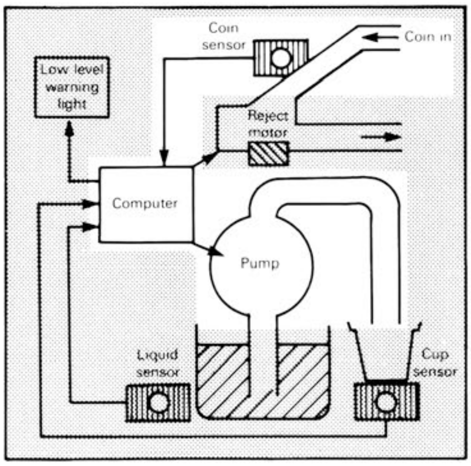
Connect the motor, the light brick and the opto-sensor brick to the interface as shown.

R8

R9



In a drinks dispensing system, a drink is to be pumped out only when a coin has been inserted. Design a program to solve this problem using the LEGO motor as the pump motor and the light brick and opto-sensor as a coin detector.



R9

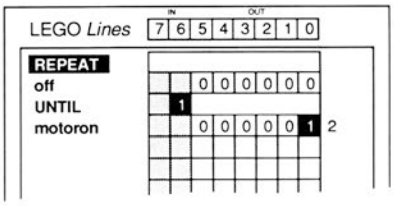
Check that the opto-sensor is connected to input bit 6.

R8

Connect a light brick to the 4 v output supply and arrange it so that when the light shines into the opto-sensor the indicator light for bit 6 is off.

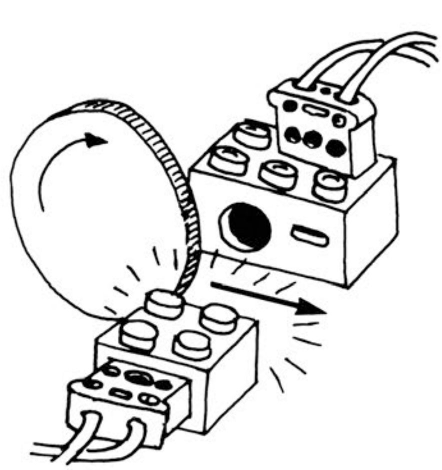
Load program SEVEN.

Press TAB to test the program. Cover the sensor to simulate a coin passing.



LEGO *Lines*

	IN 7	6	5	4	OUT 3	2	1	0	
REPEAT									
off			0	0	0	0	0	0	
UNTIL		1							
motoron			0	0	0	0	0	1	2



Use a hard copy of this program and describe what would happen in an actual drinks dispenser system once a coin had been detected.

Assignment: 5b

Purpose: Assignment 5

To learn the second set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept Computers receive information through an input port.

Concept Computers can be programmed to make decisions based on the input bit pattern.

Skill The ability to design a simple sequence of instructions using the `REPEAT`, `UNTIL`, `IF`, `ENDIF` and `COUNT` keywords.

Attitude It is straightforward to control devices using a computer system.

Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Two opto-sensor bricks with leads
Resources booklet **R7–R12**

Sample program

`EIGHT`

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A6**.

A group should complete this assignment in the time allocated.

If sufficient computers are available, all groups could do this assignment simultaneously (interfaces will be required).

Assignments 5a–5f introduce further software techniques and they must be completed in the set sequence.

Hard copies are required to document the exercises. Students should use the following:

Program sheets

Structured flow diagrams from printouts (`CTRL f7`)

Descriptions of each program.

From this point it is assumed that students know how to `LOAD` particular programs.

The focus of this assignment is on the new software commands. It is equally important that the relationship between what happens on the screen and on the interface is fully understood.

Extension work

Students can alter the set program.

Troubleshooting

Check that connections are not misplaced.

Difficulties may be experienced in understanding how the opto-sensor works.

In Assignment 5a the light brick is used to force an initial state but in other assignments 5b–5f it is recommended that students hold the sensor(s) so as to control its state.

Assignment: 5b

Computer control

A5b

You will need

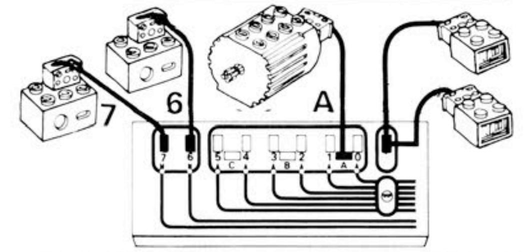
Computer system and LEGO *Lines* disk
A motor and two opto-sensor bricks and two light bricks
Resources booklet R7–R10

You are going to

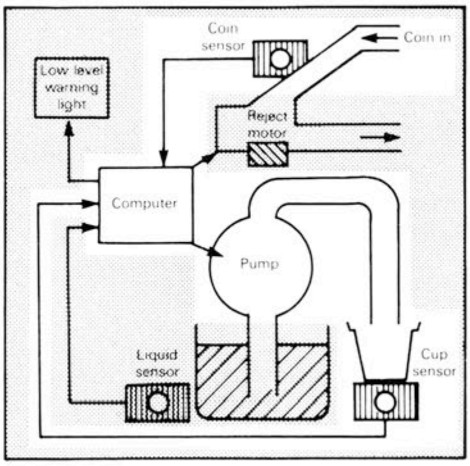
Learn how to design a program which requires two separate inputs to be sensed before an action is performed.

Connect the motor, the opto-sensors and the light bricks to the interface as shown.

R8-9



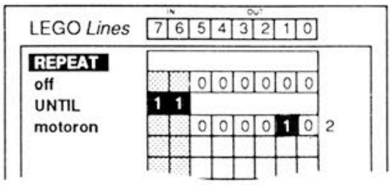
A program is required which will not dispense drinks until a coin is inserted *and* a cup is in place.



Connect the sensors to input bits 6 and 7. Make sure both indicator lights for bits 6 and 7 are off by placing your finger over the sensors as necessary, or by using the light brick as in **A5a**.

Connect the motor to output bits A.

Load program EIGHT.

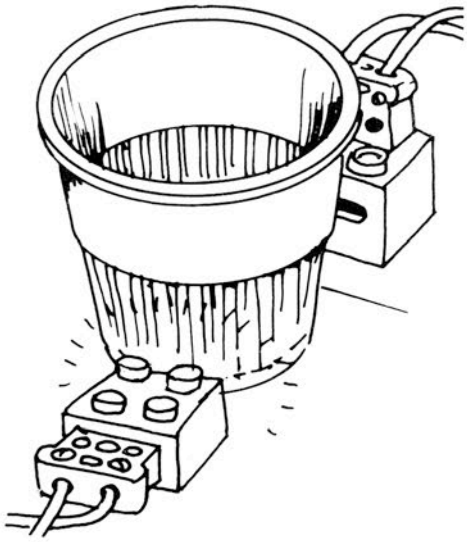


	7	6	5	4	3	2	1	0	
LEGO *Lines*	IN				OUT				
REPEAT									
off			0	0	0	0	0	0	
UNTIL	1	1							
motoron			0	0	0	0	1	0	2

The opto-sensors are used to sense the presence of a coin and a cup.

Press TAB to test the program.

Use a hard copy of the program and describe what would happen in an actual drinks dispenser if the program were run.



Assignment: 5c

Purpose: Assignment 5

To learn the second set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept Computers receive information through an input port.

Concept Computers can be programmed to make decisions based on the input bit pattern.

Skill The ability to design a simple sequence of instructions using the `REPEAT`, `UNTIL`, `IF`, `ENDIF` and `COUNT` keywords.

Attitude It is straightforward to control devices using a computer system.

Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
A motor and an opto-sensor brick with leads
Resources booklet **R7–R12**

Sample program

```
NINE
```

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A6**.

A group should complete this assignment in the time allocated.

If sufficient computers are available, all groups could do this assignment simultaneously (interfaces will be required).

Assignments 5a–5f introduce further software techniques and they must be completed in the set sequence.

Hard copies are required to document the exercises. Students should use the following:

  Program sheets

  Structured flow diagrams from printouts (`CTRL f`$_7$)

  Descriptions of each program.

From this point it is assumed that students know how to `LOAD` particular programs.

The focus of this assignment is on the new software commands. It is equally important that the relationship between what happens on the screen and on the interface is fully understood.

Extension work

Students can alter the set program.

Troubleshooting

Check that connections are not misplaced.

Difficulties may be experienced in understanding how the opto-sensor works.

In Assignment 5a the light brick is used to force an initial state but in other assignments 5b–5f it is recommended that students hold the sensor(s) so as to control its state.

Assignment: 5c

Computer control

You will need

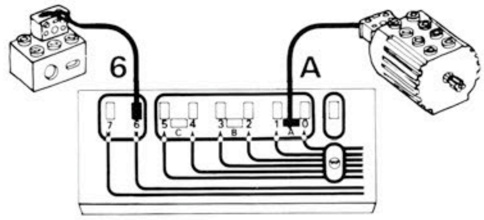
Computer system and LEGO *Lines* disk
A motor and an opto-sensor brick
Resources booklet R7–R10

You are going to

Learn how to design a program which will skip over a sequence of instructions until the right input is sensed.

Connect the motor and opto-sensor to the interface as shown.

R8-9

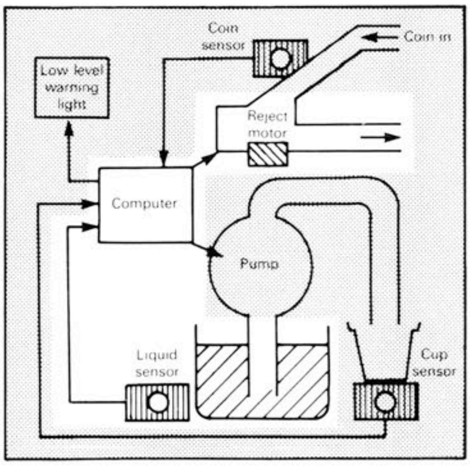


A program is required which will reject a coin if the liquid has been used up.

Condition the opto-sensor so that the indicator light on the interface is off at the start.

You could use a light brick (as before) or you could use your finger to activate the 'liquid sensor'.

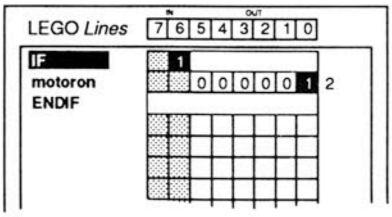
The LEGO *motor represents the motor which rejects the coin.*



Load program **NINE**.

Press **TAB** to test the program.

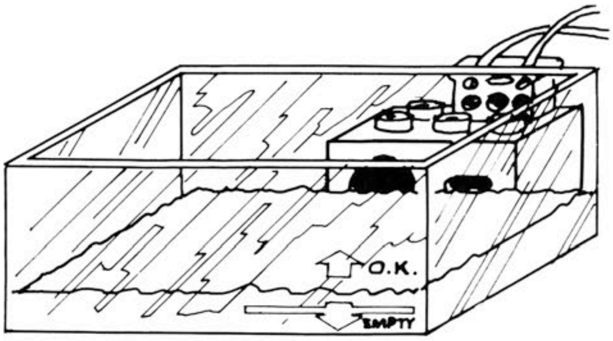
Obtain a hard copy of the program and describe how it controls the motor.

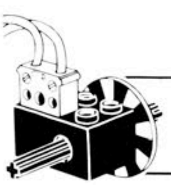


LEGO *Lines*

IN		OUT						
	7	6	5	4	3	2	1	0
IF		1						
motoron			0	0	0	0	0	1	2
ENDIF								

Note that the motor does not come on (to reject the coin) until the sensor detects that the liquid has been used up.





Purpose: Assignment 5

To learn the second set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept Computers receive information through an input port.

Concept Computers can be programmed to make decisions based on the input bit pattern.

Skill The ability to design a simple sequence of instructions using the `REPEAT`, `UNTIL`, `IF`, `ENDIF` and `COUNT` keywords.

Attitude It is straightforward to control devices using a computer system.

Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Two opto-sensor bricks with leads
Program sheets as required
Resources booklet **R7–R12**

Sample program

```
TEN
```

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A6**.

A group should complete this assignment in the time allocated.

If sufficient computers are available, all groups could do this assignment simultaneously (interfaces will be required).

Assignments 5a–5f introduce further software techniques and they must be completed in the set sequence.

Hard copies are required to document the exercises. Students should use the following:

Program sheets

Structured flow diagrams from printouts (`CTRL f7`)

Descriptions of each program.

From this point it is assumed that students know how to `LOAD` particular programs.

The focus of this assignment is on the new software commands. It is equally important that the relationship between what happens on the screen and on the interface is fully understood.

Extension work

Students can alter the set program.

Troubleshooting

Check that connections are not misplaced.

Difficulties may be experienced in understanding how the opto-sensor works.

In Assignment 5a the light brick is used to force an initial state but in other assignments 5b–5f it is recommended that students hold the sensor(s) so as to control its state.

Assignment: 5d

LEGO

Computer control

A5d

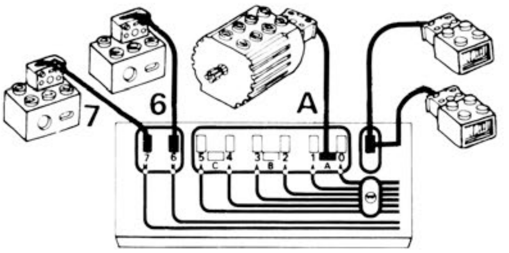
You will need

Computer system and LEGO *Lines* disk
A motor, two light bricks and two opto-sensor bricks
Resources booklet R7–R10

You are going to

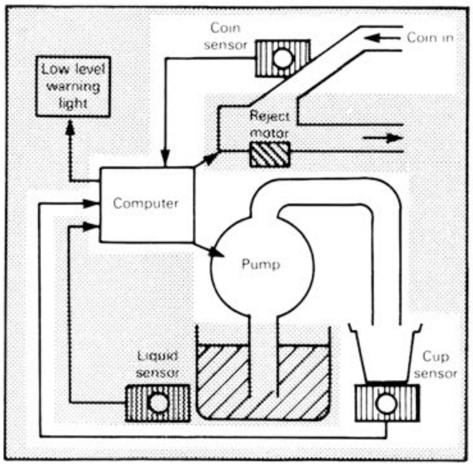
Learn how to design a program which will execute a sequence of instructions if two inputs are sensed.

Connect the light bricks, the motor and the opto-sensor bricks to the interface as shown.



A program is required which will only turn on the pump if a coin is inserted *and* a cup is in place.

Condition sensors so that the indicator lights on the interface are both off.

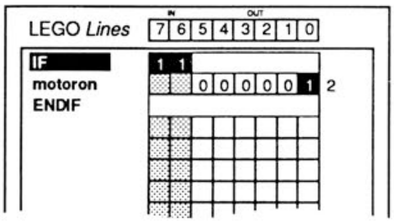


Load program **TEN**.

Press **TAB** to test program.

Describe what happens.

Can you account for why the program 'appears' not to work?



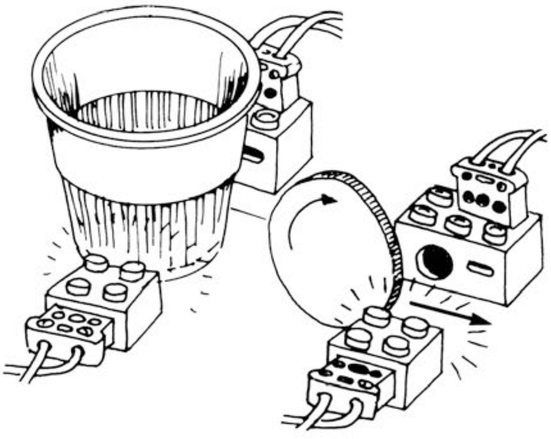
```
LEGO Lines    IN          OUT
              7 6 5 4 3 2 1 0
IF            1 1
motoron           0 0 0 0 0 1  2
ENDIF
```

Note

The motor should *not* come on.

Condition the sensors so that the motor does come on.

Describe how you did this.



Purpose: Assignment 5

To learn the second set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept Computers receive information through an input port.

Concept Computers can be programmed to make decisions based on the input bit pattern.

Skill The ability to design a simple sequence of instructions using the `REPEAT`, `UNTIL`, `IF`, `ENDIF` and `COUNT` keywords.

Attitude It is straightforward to control devices using a computer system.

Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
A motor and a light brick with leads
An opto-sensor brick with leads
Program sheets as required
Resources booklet **R7–R12**

Sample program

`ELEVEN`

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A6**.

A group should complete this assignment in the time allocated.

If sufficient computers are available, all groups could do this assignment simultaneously (interfaces will be required).

Assignments 5a–5f introduce further software techniques and they must be completed in the set sequence.

Hard copies are required to document the exercises. Students should use the following:

Program sheets

Structured flow diagrams from printouts (`CTRL f7`)

Descriptions of each program.

From this point it is assumed that students know how to `LOAD` particular programs.

The focus of this assignment is on the new software commands. It is equally important that the relationship between what happens on the screen and on the interface is fully understood.

Extension work

Students can alter the set program.

Troubleshooting

Check that connections are not misplaced.

Difficulties may be experienced in understanding how the opto-sensor works.

In Assignment 5a the light brick is used to force an initial state but in other assignments 5b–5f it is recommended that students hold the sensor(s) so as to control its state.

Assignment: 5e

Computer control

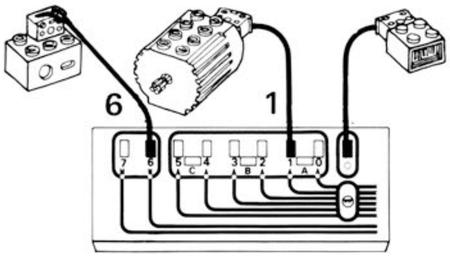
You will need

Computer system and LEGO *Lines* disk
A motor, a light brick and an opto-sensor brick
Resources booklet R7–R10

You are going to

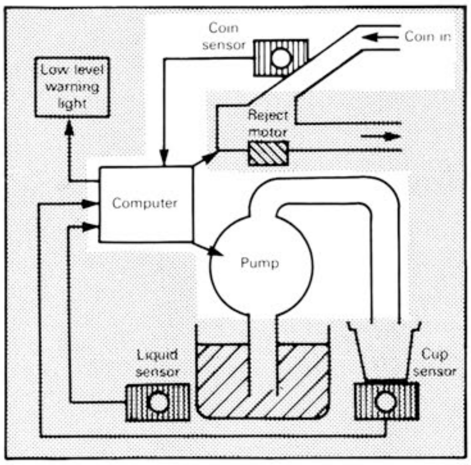
Learn how to design a program which continually looks for an input and, when that input is sensed, executes a sequence of instructions.

Connect the motor, light brick and opto-sensor brick to the interface as shown.



A program is required which continually looks for a coin and dispenses a drink when the coin is inserted.

Condition the sensor so that the indicator light on the interface is off.

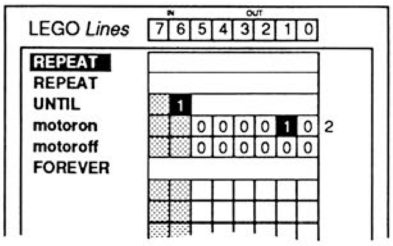


Load program **ELEVEN**.

Press **TAB** to test the program.

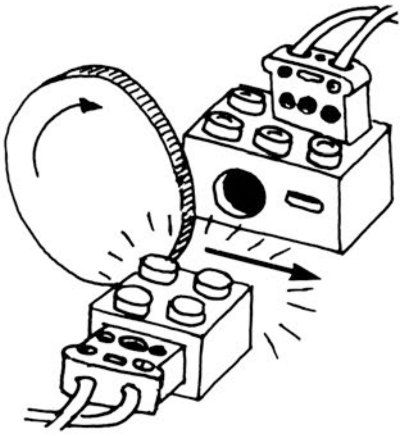
When you condition the sensor to be on, the program will turn the motor on, then off. You can condition the sensor to be off again to simulate waiting for a coin to be inserted.

This will be repeated until the **ESCAPE** key is pressed.



LEGO *Lines*	7 (IN)	6	5	4 (OUT)	3	2	1	0	
REPEAT									
REPEAT									
UNTIL		1							
motoron			0	0	0	0	1	0	2
motoroff			0	0	0	0	0	0	
FOREVER									

Copy this program onto a Program sheet and explain how it differs from previous examples in this assignment.



Assignment: 5f

Purpose: Assignment 5

To learn the second set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept	Computers receive information through an input port.
Concept	Computers can be programmed to make decisions based on the input bit pattern.
Skill	The ability to design a simple sequence of instructions using the `REPEAT`, `UNTIL`, `IF`, `ENDIF` and `COUNT` keywords.
Attitude	It is straightforward to control devices using a computer system.

Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Two opto-sensor bricks with leads
Program sheets as required
Resources booklet **R7–R12**

Sample program

```
TWELVE
```

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A6**.

A group should complete this assignment in the time allocated.

If sufficient computers are available, all groups could do this assignment simultaneously (interfaces will be required).

Assignments 5a–5f introduce further software techniques and they must be completed in the set sequence.

Hard copies are required to document the exercises. Students should use the following:

Program sheets

Structured flow diagrams from printouts (`CTRL f7`)

Descriptions of each program.

From this point it is assumed that students know how to `LOAD` particular programs.

The focus of this assignment is on the new software commands. It is equally important that the relationship between what happens on the screen and on the interface is fully understood.

Extension work

Students can alter the set program.

Troubleshooting

Check that connections are not misplaced.

Difficulties may be experienced in understanding how the opto-sensor works.

In Assignment 5a the light brick is used to force an initial state but in other assignments 5b–5f it is recommended that students hold the sensor(s) so as to control its state.

Assignment: 5f

Computer control

A5f

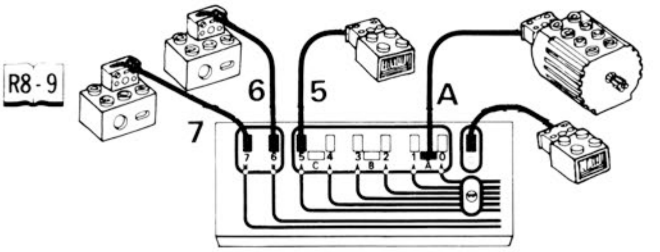
You will need

Computer system and LEGO *Lines* disk
A motor, two light bricks and two opto-sensor bricks
Resources booklet R7–R10

You are going to

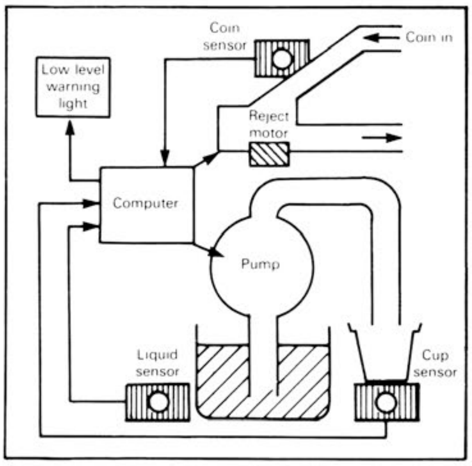
Learn how to design a program which executes a sequence of instructions when an input is sensed. As part of those instructions another input will be tested and, if it is on, the appropriate action will be performed.

Connect the motor, light bricks and opto-sensor bricks to the interface as shown.



A program is required which looks for a coin and dispenses a drink when the coin is inserted. The program should then check the liquid level and, if it is too low, a warning light should be lit.

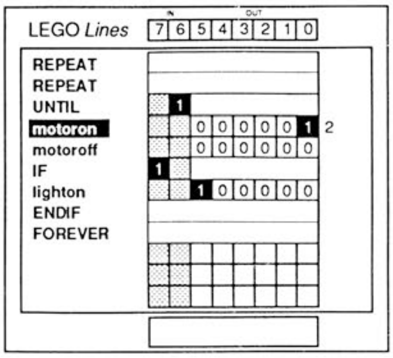
Condition the opto-sensors to be off.



Load program TWELVE.

Press TAB to run the program.

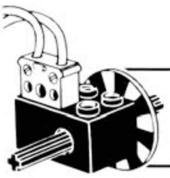
ESCAPE will stop the program.



LEGO *Lines*

	IN 7	6	5	4	3	2	1	OUT 0	
REPEAT									
REPEAT									
UNTIL		1							
motoron			0	0	0	0	0	1	2
motoroff			0	0	0	0	0	0	
IF	1								
lighton			1	0	0	0	0	0	
ENDIF									
FOREVER									

Copy this program onto a Program sheet and describe what would happen in an actual drinks dispenser if it were run.



Purpose: Assignment 6

To design and build a vehicle to be controlled by human feedback.

Aims

Concept Accurate control is dependent on feedback actions being implemented quickly relative to the speed of reaction of the system itself.

Skill The ability to design, test and evaluate a system.

Skill The ability to build a simple sequence of instructions to control a device.

Attitude The willingness and the confidence to solve problems.

Resources (each group)

A1 size paper (for drawing route)
LEGO 1090 Set (unassembled)
Manual controller
Resources booklet **R6**, **R15–R21**

Notes

A **teacher-led** activity for groups of three students.

One half of the class may be working on this assignment while the other works on the complementary assignment, **A5 (a–f)**.

A group should complete this assignment in the time allocated.

A possible approach to this assignment is to use the Resources booklet pages **R15–R17** to explore vehicle structures, drive mechanisms and methods of steering. Groups should be encouraged to explore ideas for themselves, select possible solutions and build prototypes for testing before creating their final model. A formal design session may be required (by both teachers and students) as part of this assignment.

Students should dismantle their vehicle at the end of the assignment, ready for the next group to start from scratch.

Troubleshooting

Check for efficient use of time.

Suitable vehicles are illustrated on the reverse of LEGO 1090 E Construction Guide or 1090 Series Guide.

Check for accurate record keeping (this is required for Assignment 7).

Assignment: 6

Manual control

A6

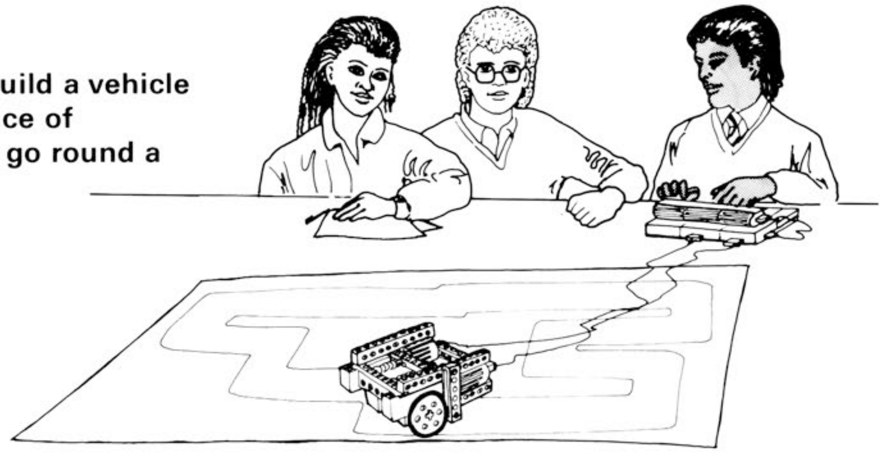
You will need

LEGO 1090 Kit
Manual controller
Resources booklet R6, R15–21
A1 sheet of paper

You are going to

Design and make a vehicle which you can then control.

Your group is required to build a vehicle and then to write a sequence of instructions to enable it to go round a course.



Building the vehicle

R15 shows some vehicle structures

R16 shows some drive mechanisms

R17 shows some steering mechanisms

Using these ideas, and ideas of your own, build the vehicle.

Draw a route

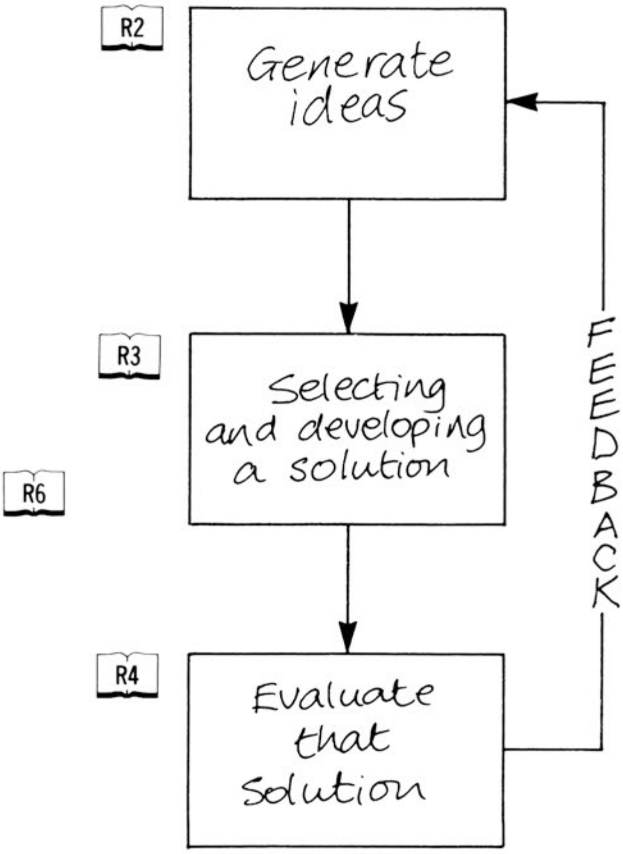
On the large piece of paper, draw a course for your vehicle to go round.

Test your model

Using the manual controller, test whether your vehicle is capable of going round at least part of the course. if it isn't, you will have to change your design until it can. This is the *feedback* part of the design process.

Write a program

Using a Program sheet, write a sequence of instructions which would let somebody else (from another group) make your vehicle follow the course you have drawn.



Purpose: Assignment 7

To formally introduce a process for solving problems.

Aims

Concept	The problem-solving process.
Skill	Using a model of the process for solving problems.
Skill	Using the information sheets.
Attitude	The willingness and the confidence to solve problems.

Resources (each group)

LEGO 1090 Set
Manual controller
Resources booklet

Resources (Teacher)

Demonstration buggy

Notes

A **teacher-led** activity for groups of three students.

The assignment should be completed in the time allocated.

This assignment is designed as an aide memoire to the problem-solving model. Emphasis should be placed on the process as set out in the Resources booklet, **R1–R5**.

A necessary startpoint is a discussion on the outcome of the previous assignment.

The problem set by the teacher in this assignment will be crucial in order to exemplify the value of the problem-solving process.

For example, the teacher could ask for thoughts on a vehicle to clear rubbish from a sandy beach. Alternatively, Assignment 6 could be explored in greater depth.

Constant reference should be made to the Resources booklet and the results of the previous assignment.

Extension work

Each group reports back to the class.

Troubleshooting

Allow sufficient time to develop the most appropriate solution.

Check for battery and connection problems.

Assignment: 7

Solving a problem

A7

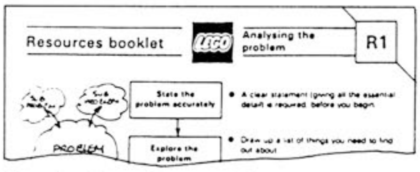
You will need

Resources booklet
Your results from Assignment 6.

This is a model which will help you solve a problem.

Actions | *Outcomes* | *Further explanations in*

Analyse the problem



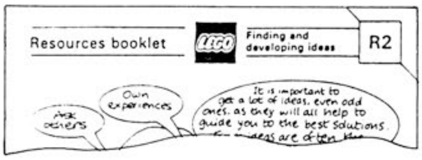
Resources booklet | Analysing the problem | R1

State the problem accurately
Explore the problem

- A clear statement (giving all the essential detail) is required before you begin.
- Draw up a list of things you need to find out about

Analysing the problem

Finding ideas

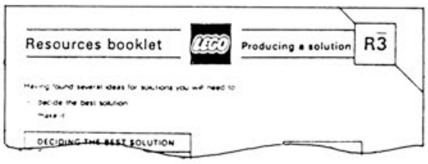


Resources booklet | Finding and developing ideas | R2

Ask others
Own experiences
It is important to get a lot of ideas, even odd ones, as they will all help to guide you to the best solutions.

Finding and developing ideas

Select and develop the best solution



Resources booklet | Producing a solution | R3

Having found several ideas for solutions you will need to

- Decide the best solution
- Make it

DECIDING THE BEST SOLUTION

Producing a solution

Evaluate your solution

Resources booklet | Evaluating the solution | R4

What does evaluation mean?

Evaluate means to decide the value of something or how [illegible]

Evaluating the solution

Communicate your work

Resources booklet | Communicating your solution | R5

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

Communicating a solution

Purpose: Assignment 8

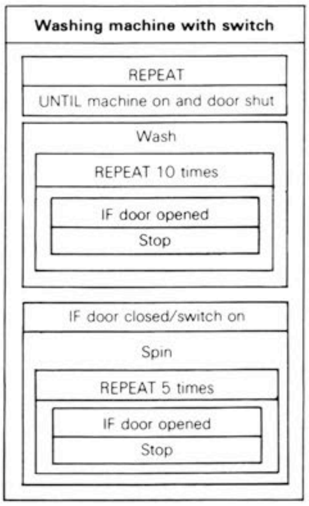
To apply the problem-solving strategy to a software environment and to learn the techniques necessary to produce a structured program.

Aims

Concept	Programming utilises the problems-solving strategy.
Concept	Structured programming results in successful and elegant solutions.
Skill	The ability to apply the problem-solving strategy.
Skill	The ability to use software commands in a LEGO *Lines* programming environment.
Attitude	The willingness and the confidence to solve problems.

Resources (each group)

Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* Program disk
Resources booklet **R1–R5**, **R10–R14**
1090 C Washing machine construction guide
Two light bricks with leads
Two opto-sensor bricks with leads
Program sheets as required



```
Washing machine with switch
    REPEAT
    UNTIL machine on and door shut
    Wash
        REPEAT 10 times
            IF door opened
            Stop
    IF door closed/switch on
    Spin
        REPEAT 5 times
            IF door opened
            Stop
```

Notes

Study in groups of three, teacher-led/group work.

The assignment should be completed in the time allocated.

The groups will have to build their own washing machine, 1090 C. (If Assignment 7 is not completed in a double lesson it may spill into the first part of an extra session together with the washing machine construction.)

The students will need to refer to **R13**, in particular, and the previous assignments to apply the programming techniques they have used before in Assignments 2–5. They will need to be guided through the development of a program using a ***structured flow diagram*** **R14**. Help for the teacher is found in *Introduction to the materials* under *Program design*.

More than one session is likely to be allocated to this activity.

Extension work

Further problems are set on the Assignment sheet.

Alternatively, groups could develop LEGO *Lines* programs for the vehicle built in Assignments 6 and 7.

Troubleshooting

The accuracy required by a programming environment may cause problems initially, in the precise use of keywords, for example. Students should refer to **R10–R14** to ensure they are using them correctly.

The addition of an emergency switch is an advanced problem. The structure flow diagram opposite provides one solution.

Assignment: 8

Designing a structured program

You will need

A washing machine model 1090 C
Computer system and LEGO *Lines* disk
Resources booklet R1–R5, R10–R14

You are going to

Use the problem-solving process to produce a structured program in order to control a washing machine.

Problem solving

A program is required which turns the motor and the indicator light of a washing machine on when the door is closed and the switch is on.

The washing machine should turn off when the door is opened.

Resources available

The Assignments you have already completed
Resources booklet

Further problems

Turn the washing machine on if the door is closed and the switch is on. First execute a wash cycle (and light) and then spin (and light).

Add an emergency switch facility which shuts the machine off.

Assignment: 9a

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill The ability to analyse a problem.

Skill The ability to search for information.

Skill The ability to formulate ideas for solutions.

Skill The ability to evaluate ideas against criteria and select the best.

Skill The ability to develop ideas into a working model and a software program.

Skill The ability to evaluate a solution against set criteria.

Skill The ability to communicate effectively using whatever means is most appropriate.

Attitude Problem-solving can be enjoyable and successful.

Attitude Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
   Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9a

Designing a programmed system

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

You are going to

Use the problem-solving process.

Sliding Door

A supermarket requires a sliding door system which will allow customers to pass in and out with ease but which remains shut at other times to save heat and energy.

As a development team for *HyperDoor (Electronic Doors) plc* your job is to prepare a report and demonstrate a solution to this problem.

Assignment: 9b

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9b

Designing a programmed system

A9b

You will need

- 1090 LEGO Set
- Computer system and LEGO *Lines* disk
- Resources booklet
- Ideas booklet
- Assignments you have completed

You are going to

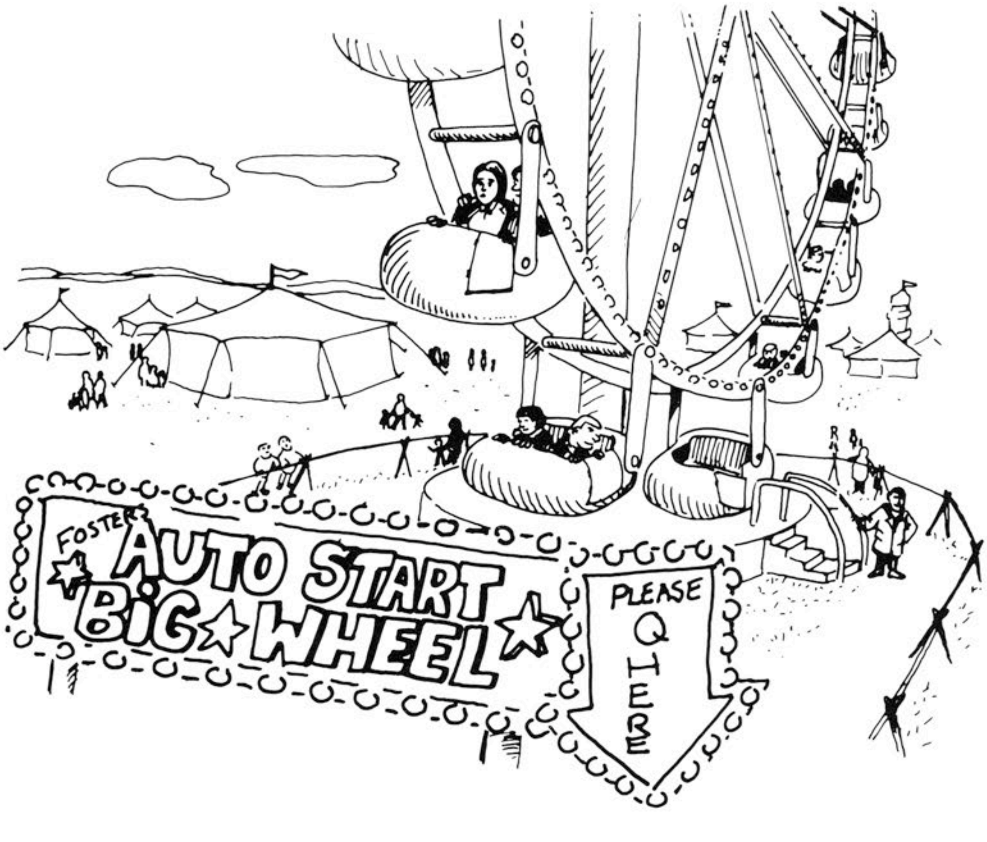
Use the problem-solving process.

Automatic Ferris wheel

A fairground equipment manufacturer has provided your design team with a contract to develop an automatic system for controlling a Ferris wheel ride.

The ride should be safer, more economic and more appealing than the present manual system.

You are required to provide a report and demonstration of a model of your best solution.



Classroom materials — Assignment: 9c

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill The ability to analyse a problem.

Skill The ability to search for information.

Skill The ability to formulate ideas for solutions.

Skill The ability to evaluate ideas against criteria and select the best.

Skill The ability to develop ideas into a working model and a software program.

Skill The ability to evaluate a solution against set criteria.

Skill The ability to communicate effectively using whatever means is most appropriate.

Attitude Problem-solving can be enjoyable and successful.

Attitude Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9c

Designing a programmed system

A9c

You will need

- 1090 LEGO Set
- Computer system and LEGO *Lines* disk
- Resources booklet
- Assignments you have completed

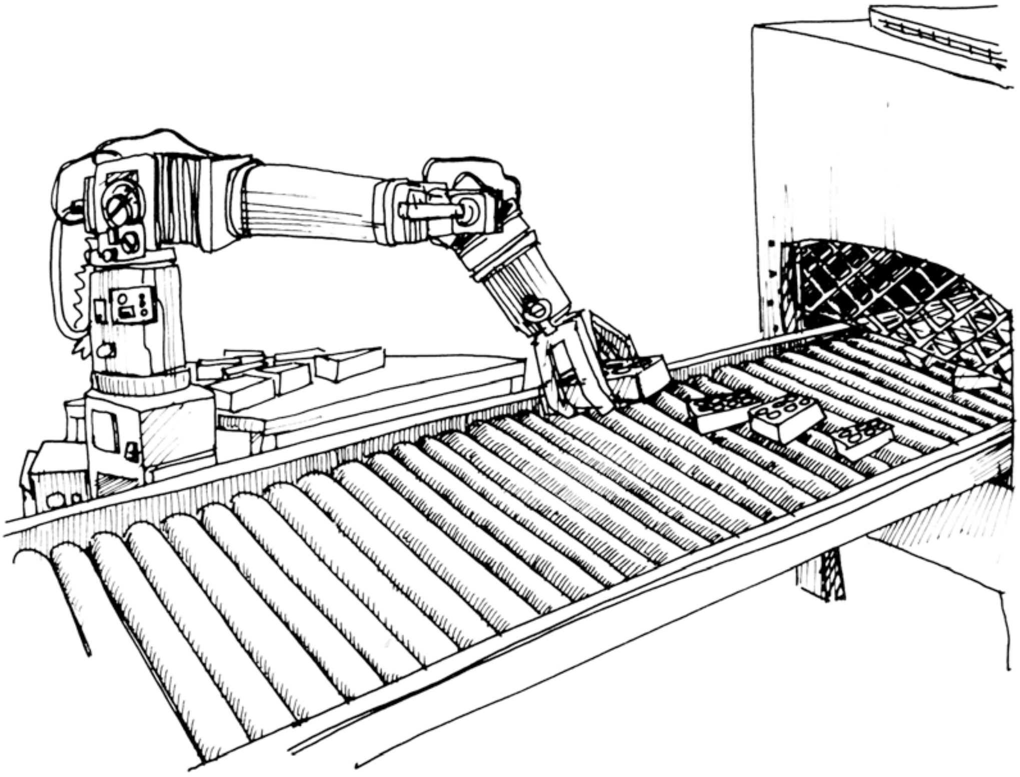
You are going to

Use the problem-solving process.

Robot Arm

A manufacturer requires a device which will pick LEGO bricks up from a conveyor belt and place them on a platform situated 10 cms from the conveyor.

Your team must provide a report and a model of this device.



Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9d

Designing a programmed system

A9d

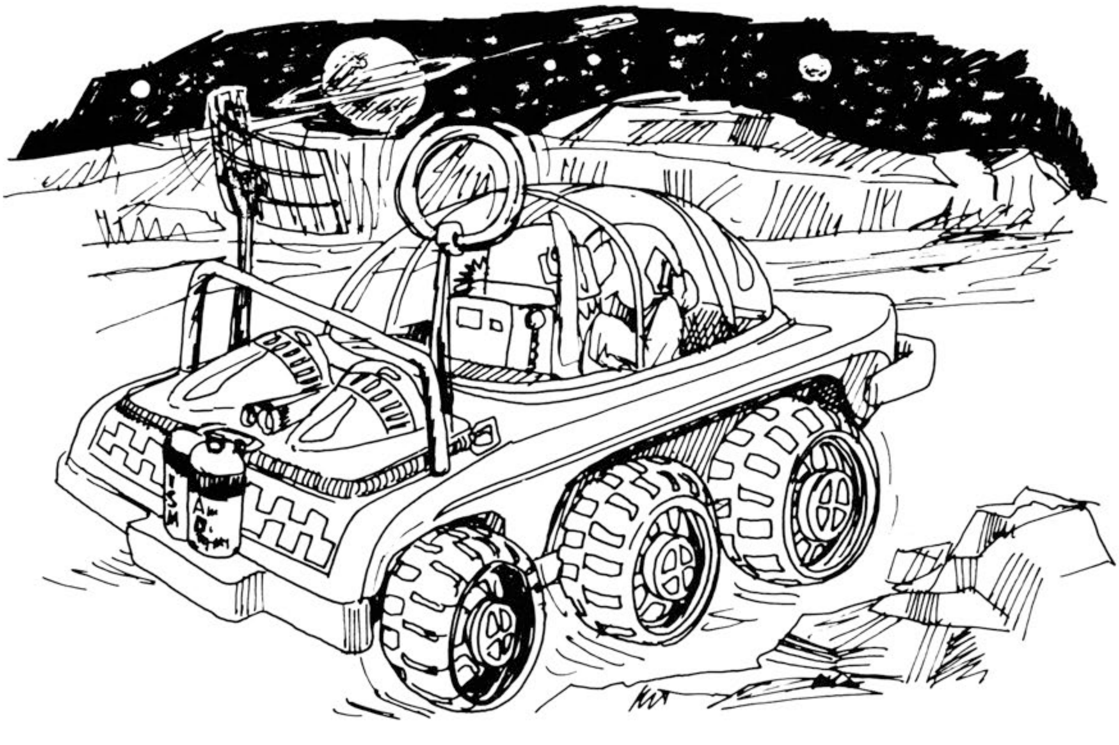
You will need
- 1090 LEGO Set
- Computer system and LEGO *Lines* disk
- Resources booklet
- Assignments you have completed

You are going to

Use the problem-solving process.

Space vehicle

Your development team must prepare a report and build a model of a mobile space vehicle. It should be capable of avoiding obstacles in its path.



Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill The ability to analyse a problem.

Skill The ability to search for information.

Skill The ability to formulate ideas for solutions.

Skill The ability to evaluate ideas against criteria and select the best.

Skill The ability to develop ideas into a working model and a software program.

Skill The ability to evaluate a solution against set criteria.

Skill The ability to communicate effectively using whatever means is most appropriate.

Attitude Problem-solving can be enjoyable and successful.

Attitude Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9e

Designing a programmed system

A9e

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

You are going to

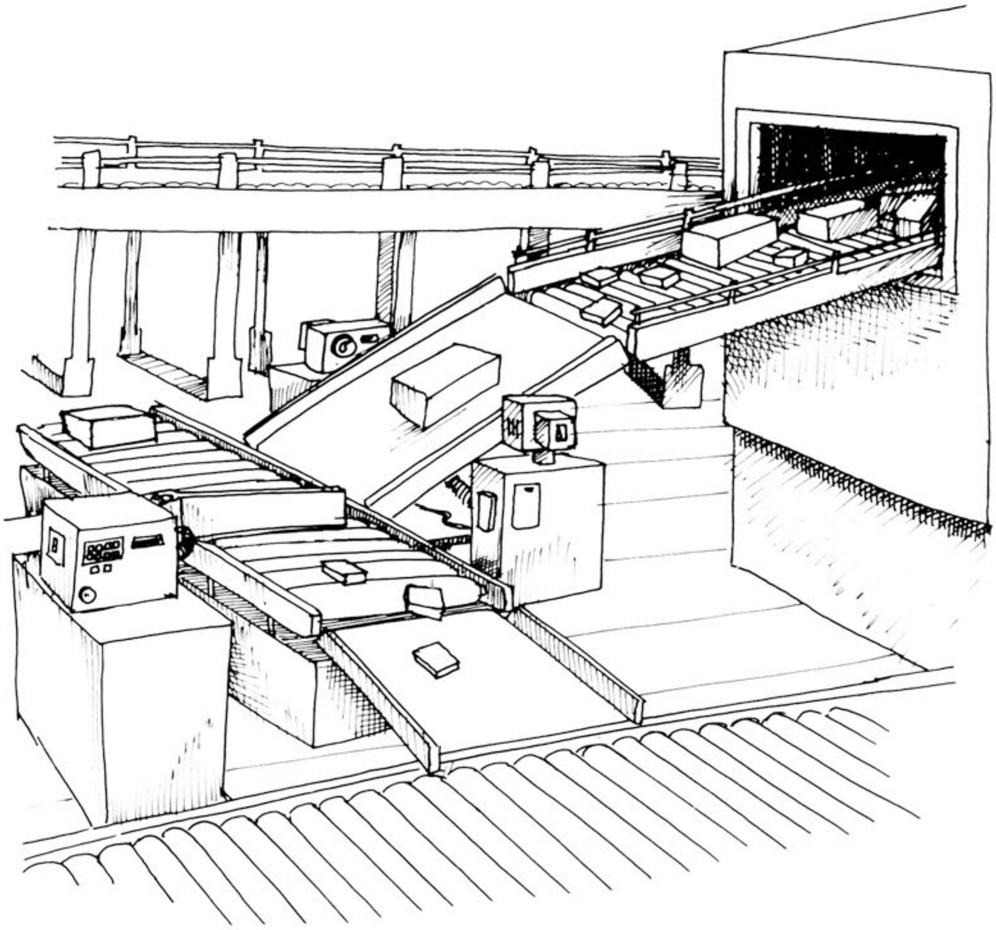
Use the problem-solving process.

Conveyor system

A manufacturer wishes to install a computerised conveyor system in the factory.

The system must be capable of sorting out large boxes on the conveyor from smaller ones.

Your team should build a model of the most appropriate solution, and submit a report on your work to the management.



Classroom materials

Assignment: 9f

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill The ability to analyse a problem.

Skill The ability to search for information.

Skill The ability to formulate ideas for solutions.

Skill The ability to evaluate ideas against criteria and select the best.

Skill The ability to develop ideas into a working model and a software program.

Skill The ability to evaluate a solution against set criteria.

Skill The ability to communicate effectively using whatever means is most appropriate.

Attitude Problem-solving can be enjoyable and successful.

Attitude Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9f

Designing a programmed system

A9f

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

You are going to

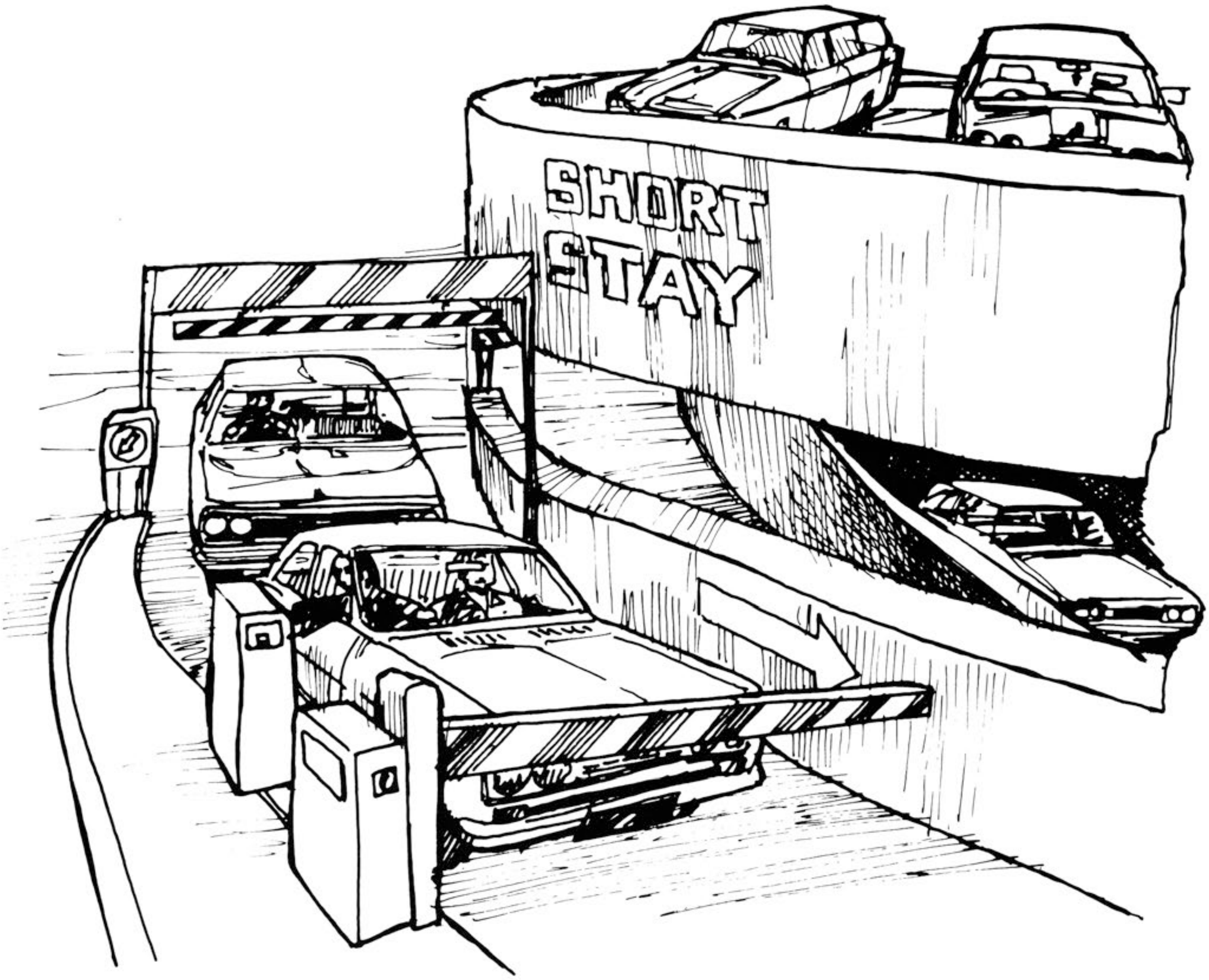
Use the problem-solving process.

Car Park Barrier

Design an automatic barrier for a multistorey car park.

Care must be taken to ensure that the system your team suggests is efficient and safe.

A model and a report are required.



a | i | s | e | c | t

Assignment: 9g

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9g

Designing a programmed system

A9g

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

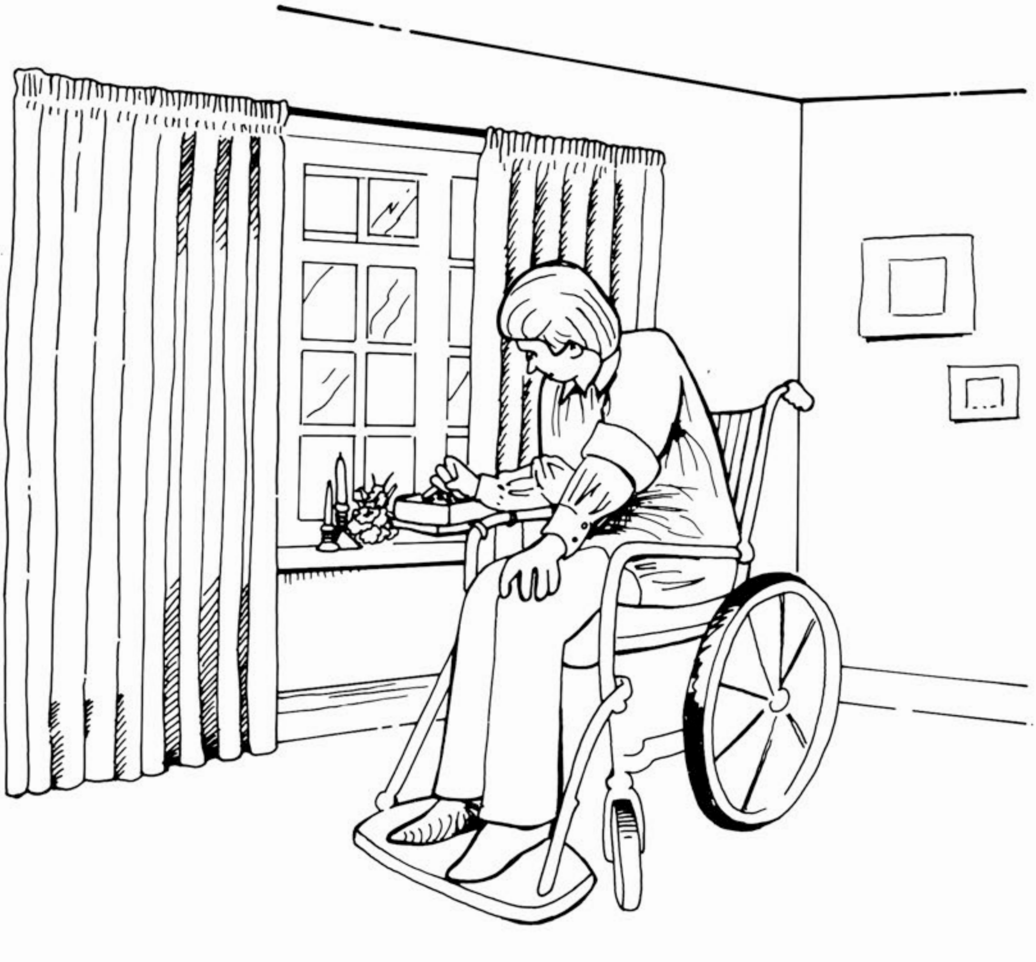
You are going to

Use the problem-solving process.

Automatic Curtains

Your team has been commissioned to design a set of curtains which open and close automatically. These would be particularly useful to elderly or disabled people.

Prepare a report of your work and a model of your solution.



Classroom materials Assignment: 9h

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9h

Designing a programmed system

A9h

You will need

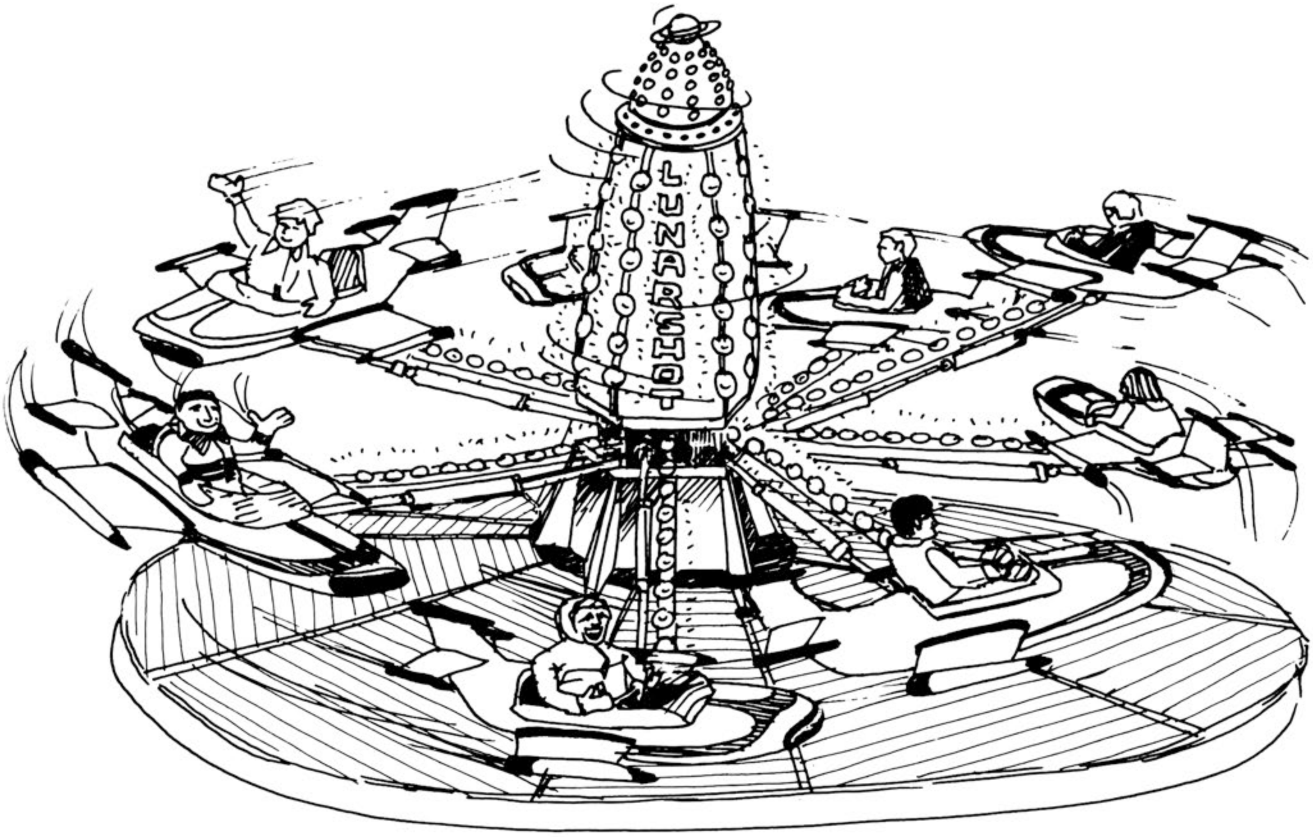
1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

You are going to

Use the problem-solving process.

Roundabout

A leisure park wishes to install an automatically controlled roundabout. They have commissioned your team to produce a working model and a report of your design.



a i s e c t

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9i

Designing a programmed system

A9i

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed
Plasticine block

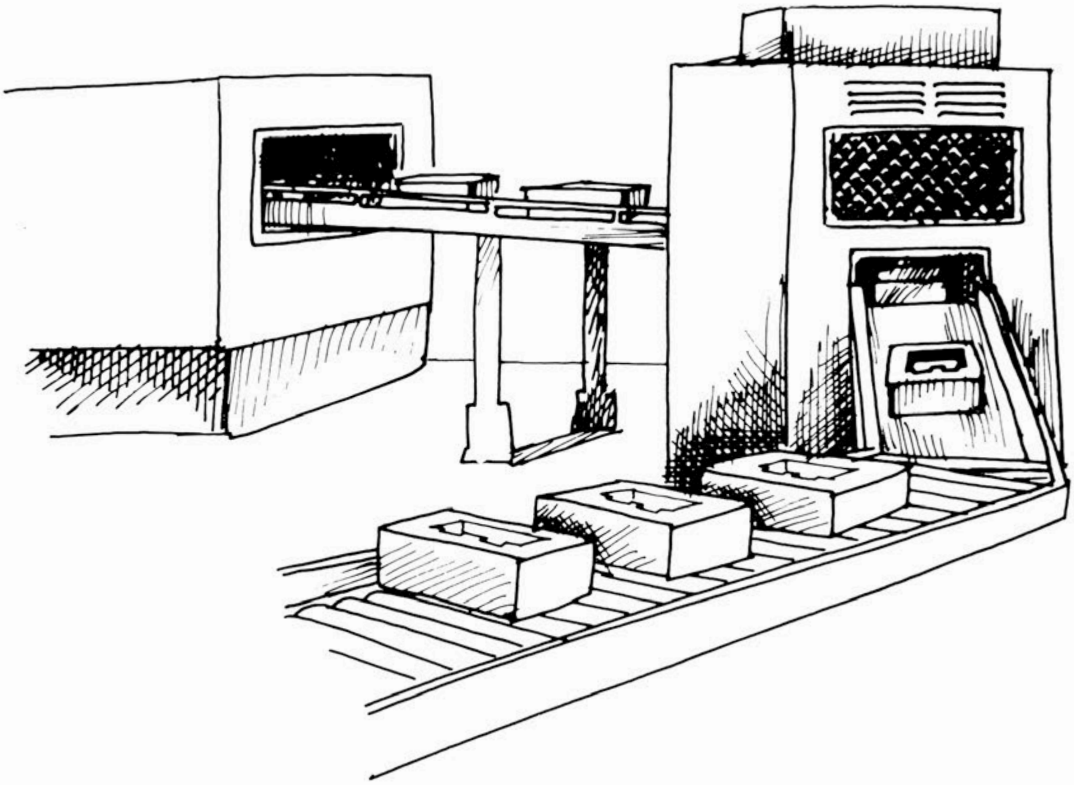
You are going to

Use the problem-solving process.

Pressing machine

Your team should design a device which will automatically stamp the pattern of a LEGO brick onto a block of plasticine when it is placed onto a platform.

A report and a working model are required.



Assignment: 9j

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
  Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9j

Designing a programmed system

A9j

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

You are going to

Use the problem-solving process.

Stage Effect

Your team has been commissioned to design a head to be mounted on a wall as part of a fantasy play. The head is required to 'come alive' each time one of the cast passes a certain spot.

Prepare a report of your ideas and a working model.

a i s e c t

Teacher's materials

Programmable Systems

Section 3

Resources booklet

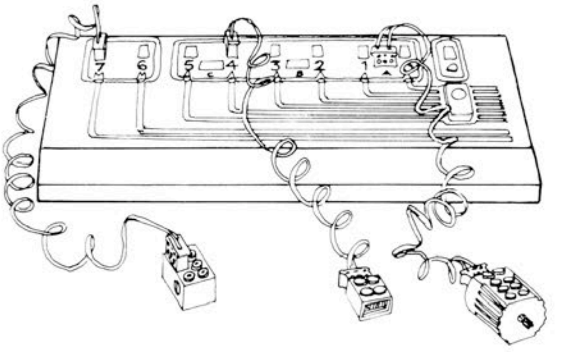
Resources booklet

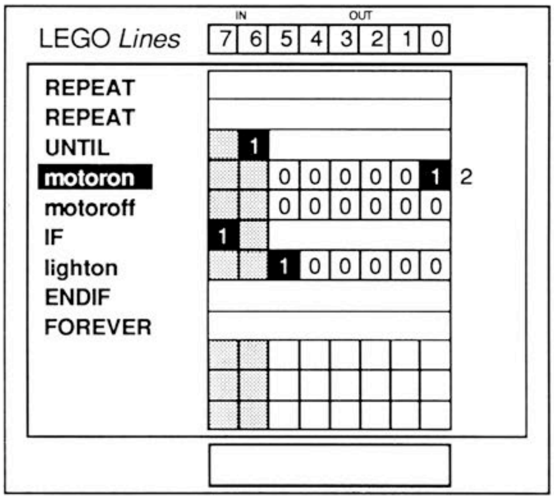
Programmable Systems LEGO Resources booklet

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.





LEGO *Lines*

	IN		OUT						
	7	6	5	4	3	2	1	0	
REPEAT									
REPEAT									
UNTIL	▨	1							
motoron	▨	▨	0	0	0	0	0	1	2
motoroff	▨	▨	0	0	0	0	0	0	
IF	1	▨							
lighton	▨	▨	1	0	0	0	0	0	
ENDIF									
FOREVER									
	▨	▨							
	▨	▨							
	▨	▨							

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

Page

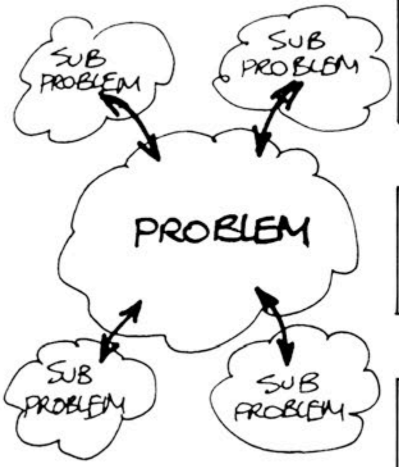
Organising yourself to solve problems

Using the equipment

Designing control programs using the LEGO materials

Analysing the problem

R1



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

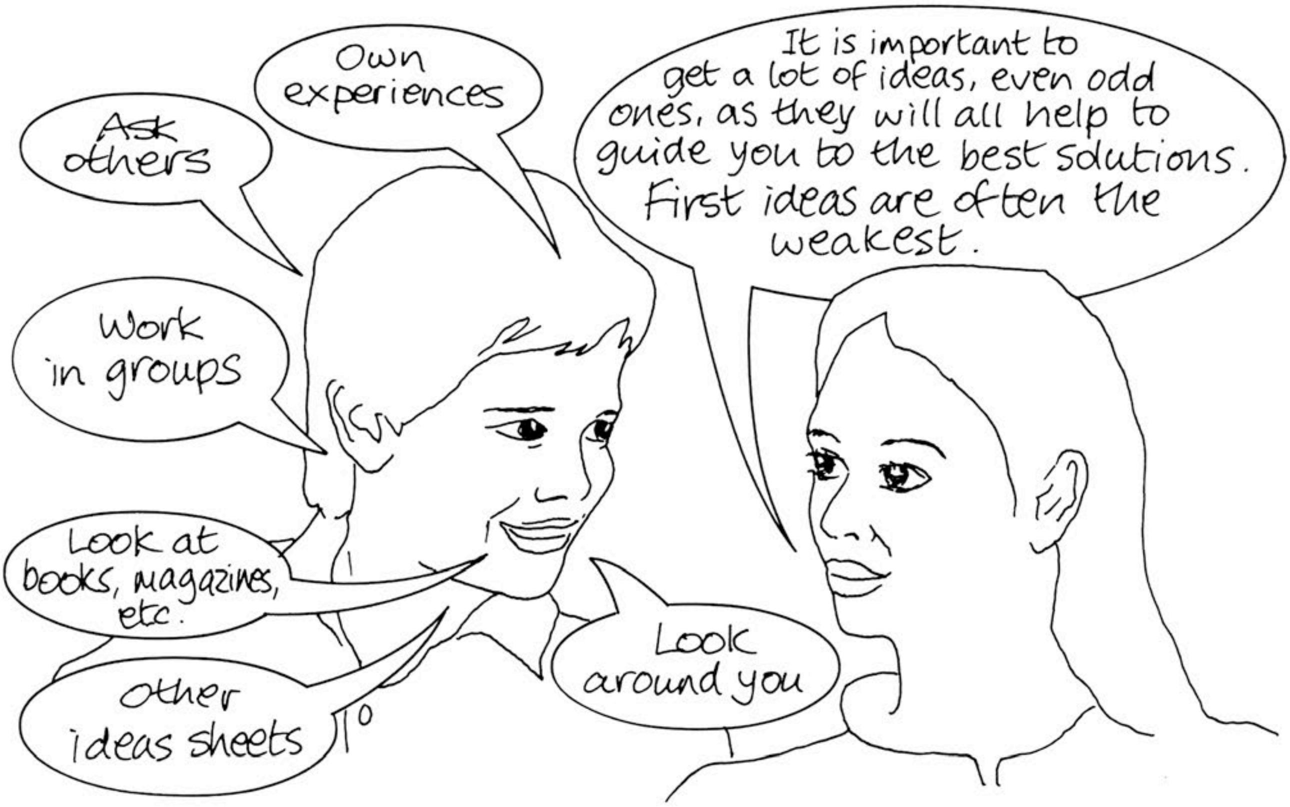
Ask yourself

See what others have done

Ask others

I AM GOING TO MAKE YOU A CAKE WHAT SORT DO YOU WANT?
I WANT A CHOCOLATE ONE
I WANT IT TO LOOK NICE
MAKE IT NICE AND BIG
I WANT ICING ON IT
I DON'T WANT IT TO COST TOO MUCH!

Finding and developing ideas


Ask others
Own experiences
It is important to get a lot of ideas, even odd ones, as they will all help to guide you to the best solutions. First ideas are often the weakest.
Work in groups
Look at books, magazines, etc.
Other ideas sheets
Look around you

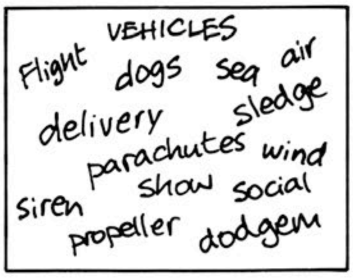
Working in groups to develop ideas

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

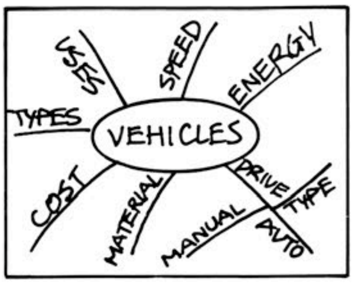


Mindmaps

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.

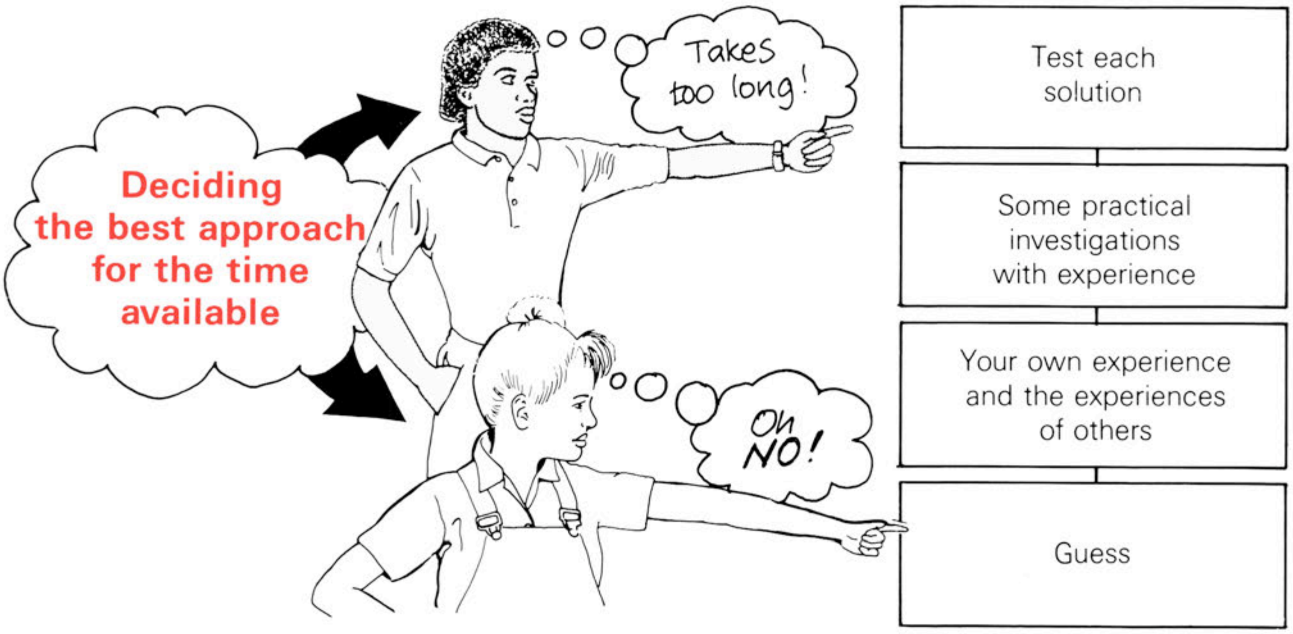
You will need

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.


Deciding the best approach for the time available
Takes too long!
Oh NO!
Test each solution
Some practical investigations with experience
Your own experience and the experiences of others
Guess

Evaluating the solution

R4

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

Communicating your solution

R5

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)

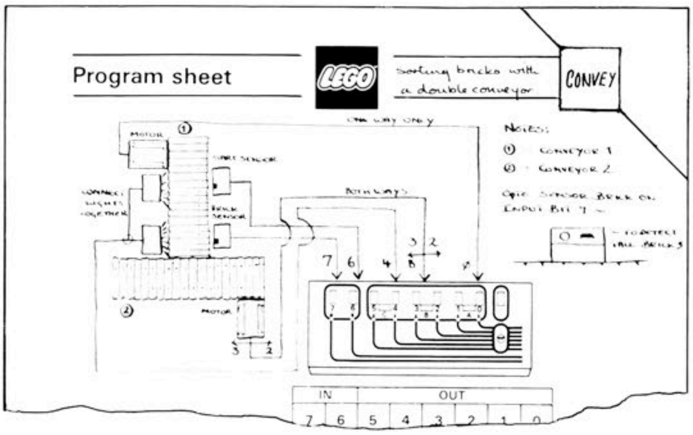
Content of talk

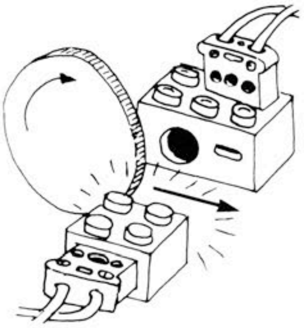
The problem
The criteria used
Solutions considered
The solution chosen
The reasons for choice
Development problems

Hints

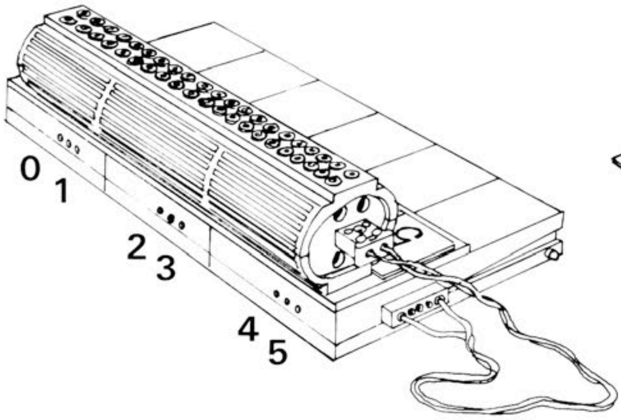
Keep the task simple
Talk to the whole audience
Do not speak too fast
Speak loud enough to be heard

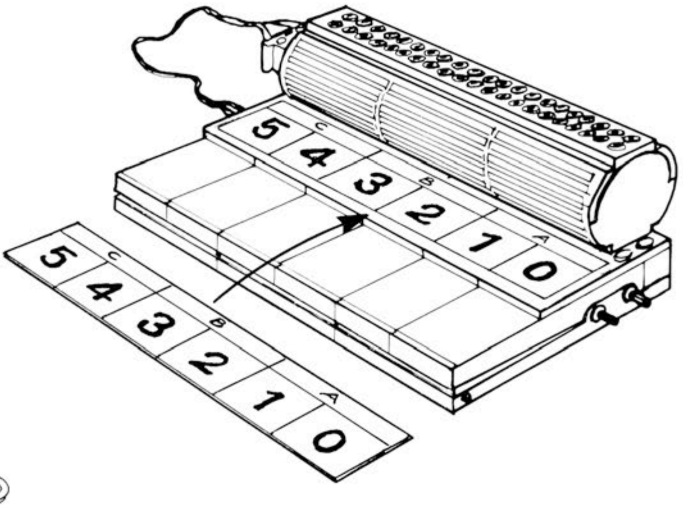
Remember: sketches and diagrams are worth many words.





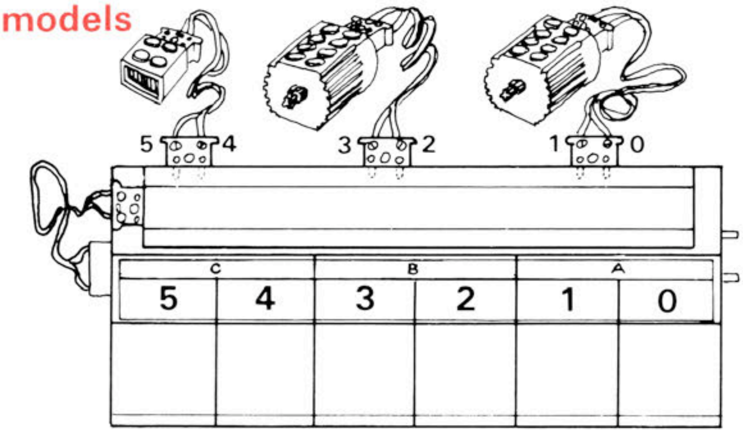
Manual controller



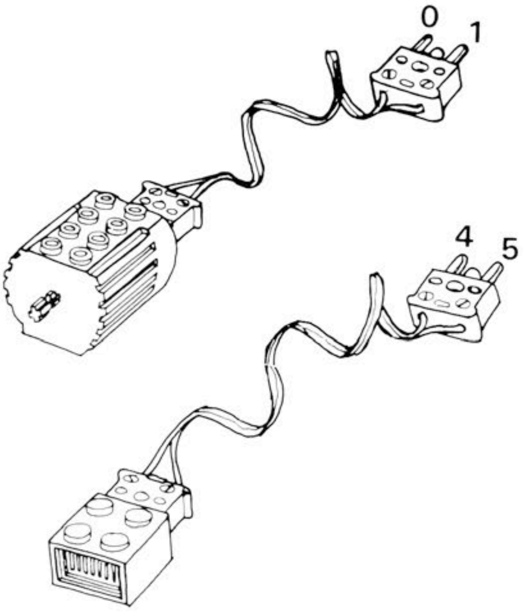


Put keyboard strip over keys

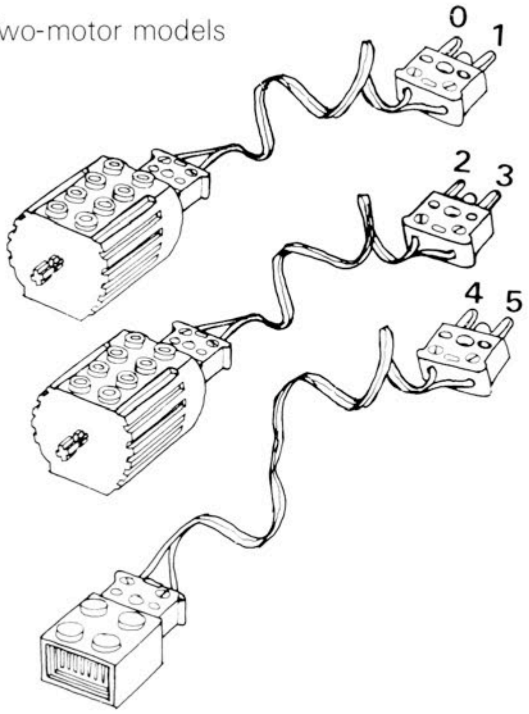
Connecting models



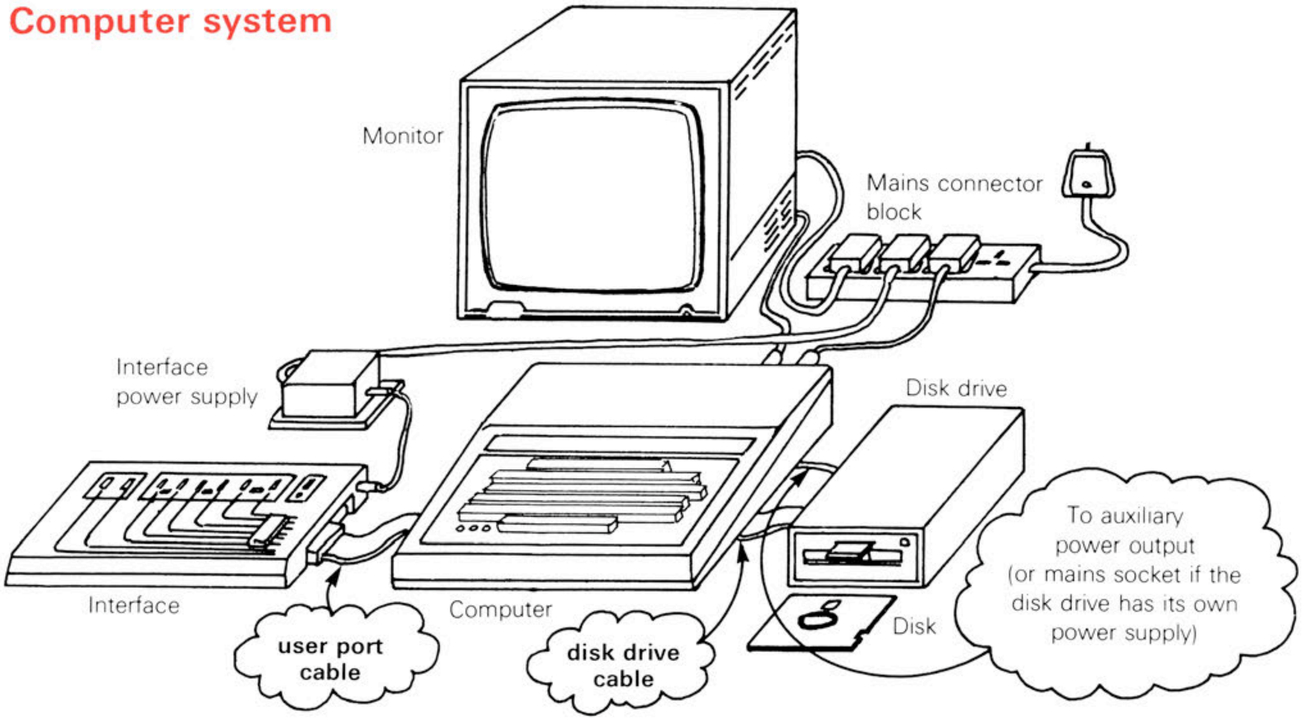
One-motor models



Two-motor models

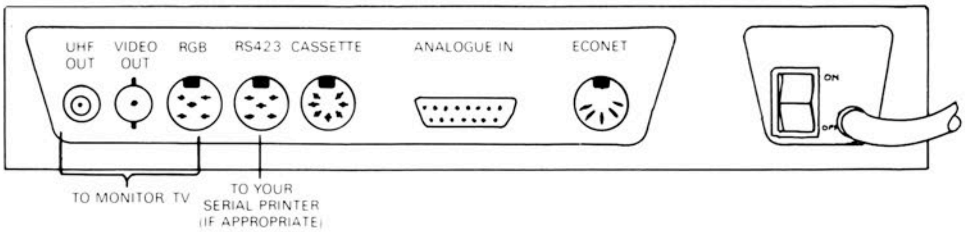


Computer system

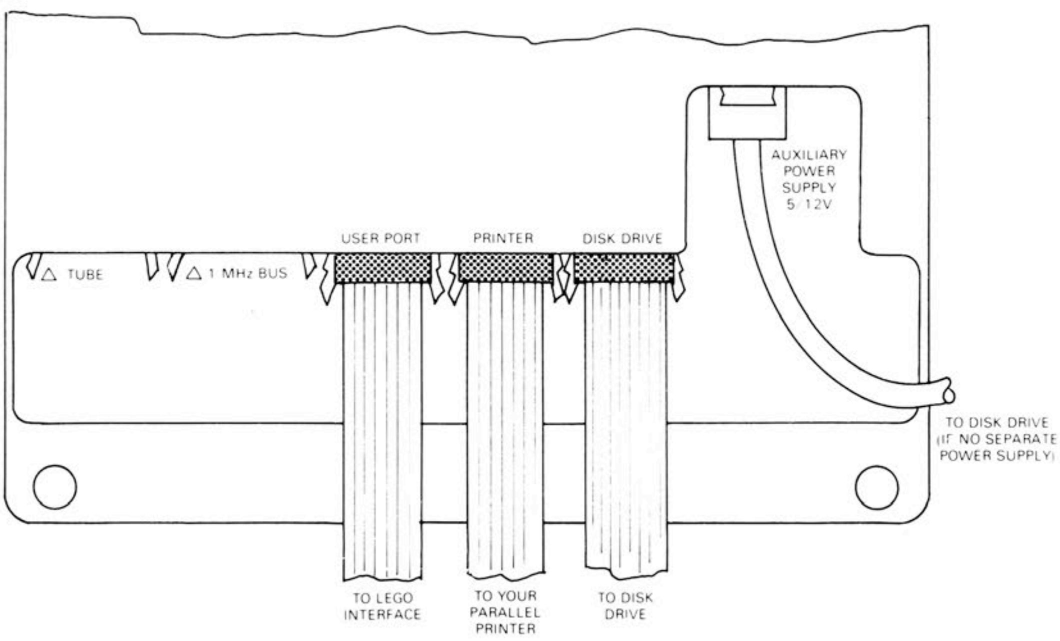


BBC B connections

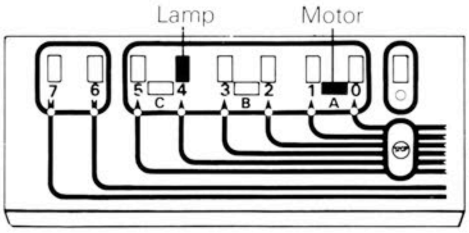
Rear



Underside



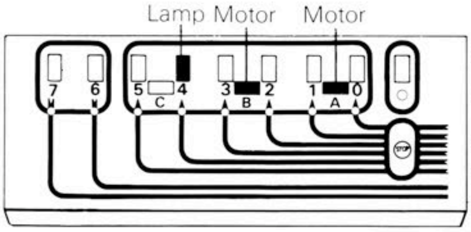
Interface connections for One motor models



4 v
power supply

STOP — Press
to switch
off all outputs

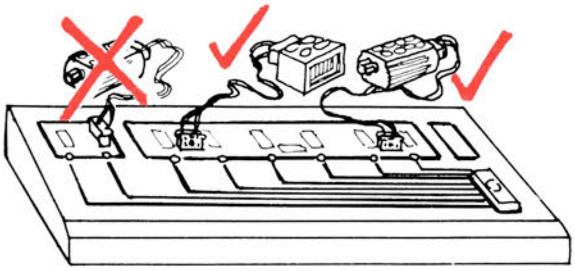
Interface connections for Two motor models

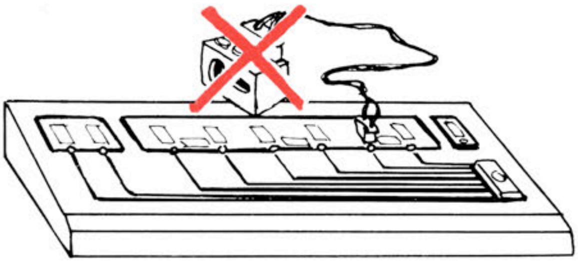


4 v
power supply

STOP — Press
to switch
off all outputs

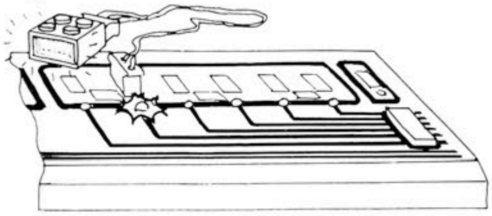
Output connections



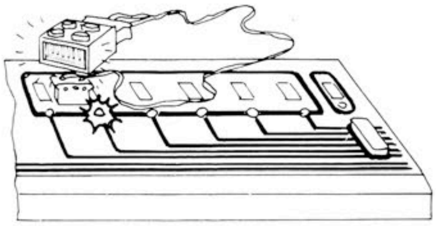


Use of output connections

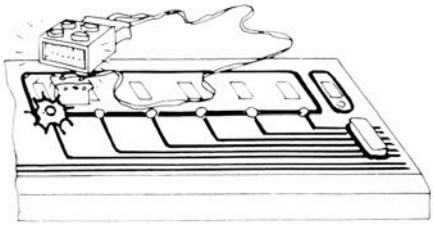
Light on



Light on

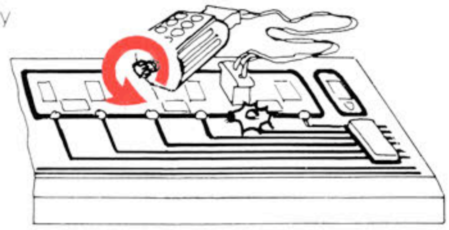


Light on

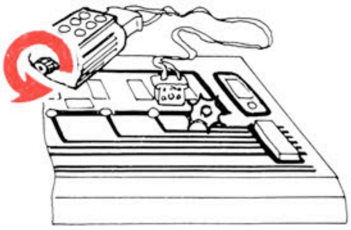


Motor on

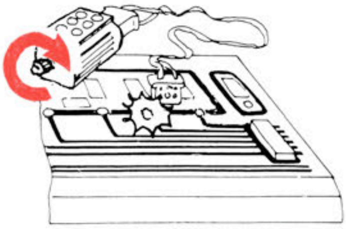
One way only



Motor on

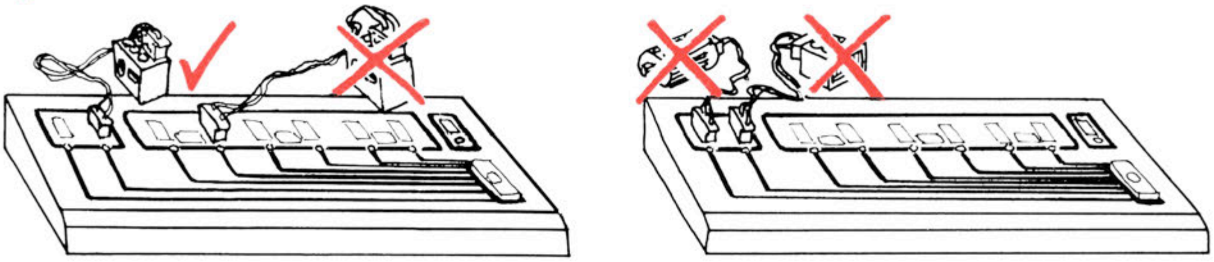


One way



Other way

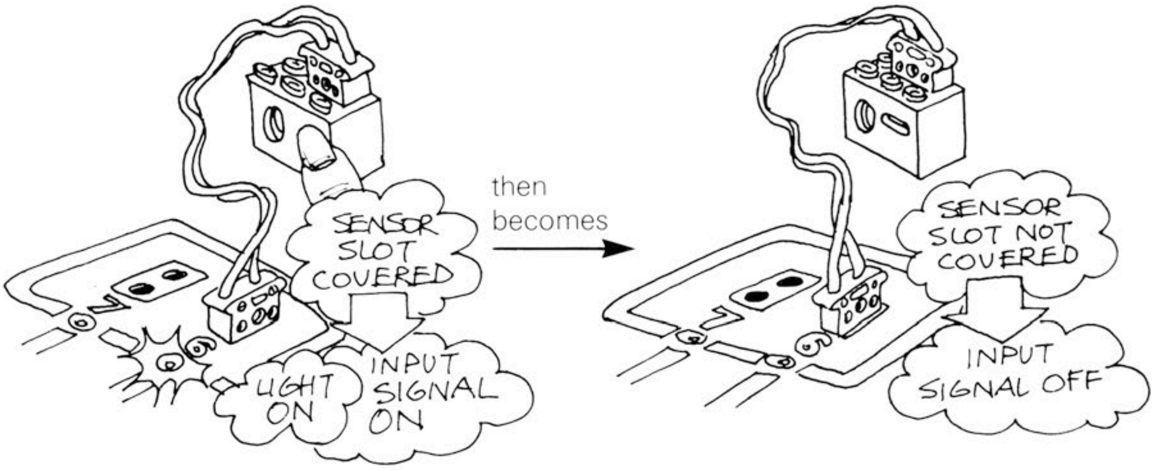
Input connections



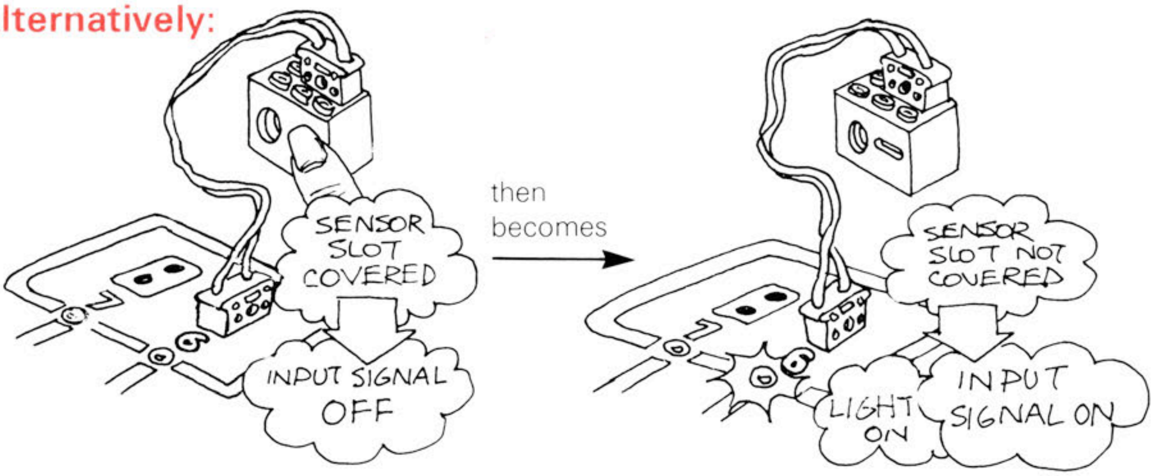
Opto-sensor brick

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.



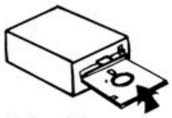
Alternatively:



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

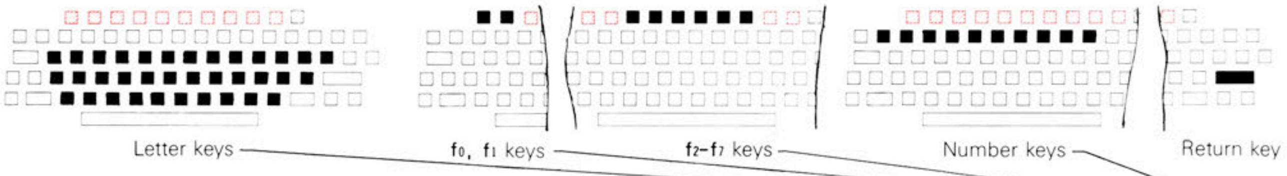
3 Autoboot program SHIFT BREAK.

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

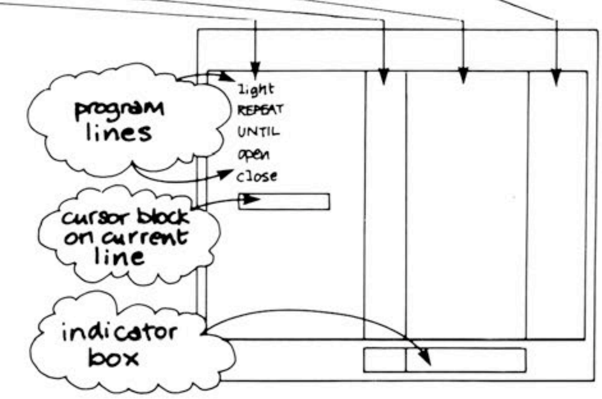
Write	**Set**	**Set**	**Set**	**Move**
label or keyword	input bits	output bits	time or count	to next line



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)

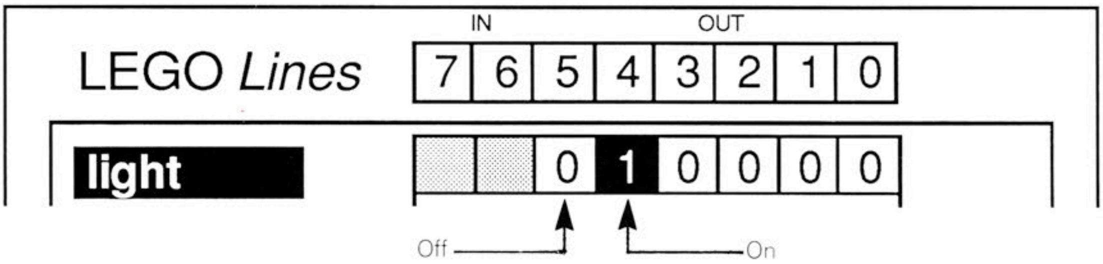
Print	CTRL f_7
Disk directory	CTRL f_3

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

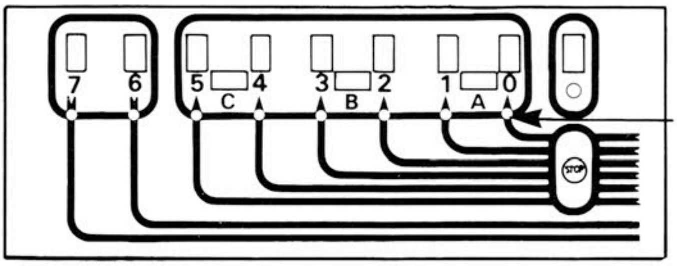
Physical control over the models is achieved by using a pattern of signals, made up of 8 bits, which link the model and the computer together via the interface.

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.



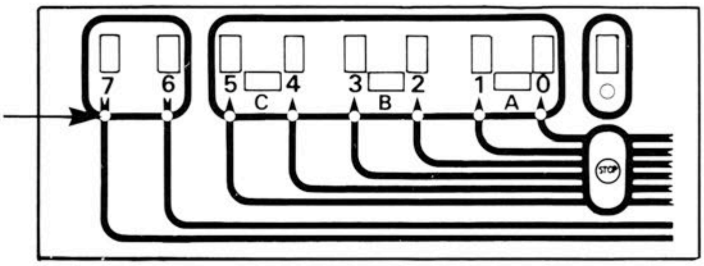
The LEGO *Lines* program is designed to use six of the eight bits as **output** bits. These control the motor and lamp devices from the LEGO set. These are numbered 0-5 on the interface and the indicator lights will show whether they are set on or off at any time.



Output bits indicator lights

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.

Input bits indicator lights

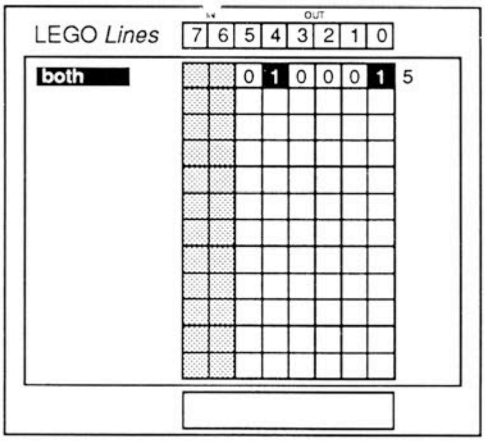


LEGO *Lines* makes use of these bits to physically control the LEGO devices.

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.



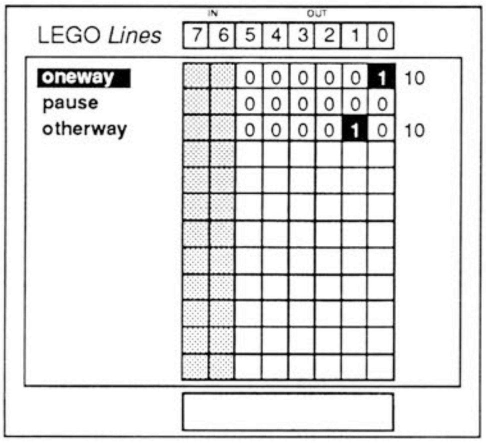
LEGO *Lines*

	IN		OUT						
	7	6	5	4	3	2	1	0	
both			0	1	0	0	0	1	5

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.



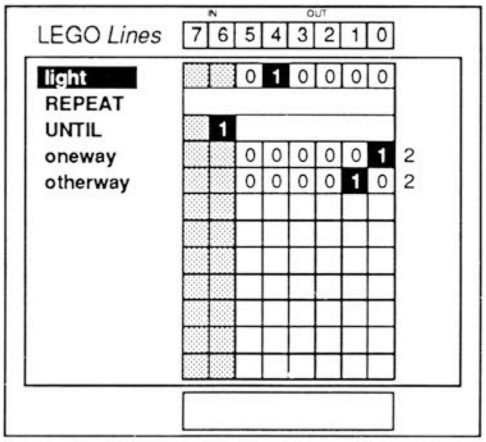
LEGO *Lines*

	IN		OUT						
	7	6	5	4	3	2	1	0	
oneway			0	0	0	0	0	1	10
pause			0	0	0	0	0	0	
otherway			0	0	0	0	1	0	10

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.



LEGO *Lines*

	IN		OUT						
	7	6	5	4	3	2	1	0	
light			0	1	0	0	0	0	
REPEAT									
UNTIL		1							
oneway			0	0	0	0	0	1	2
otherway			0	0	0	0	1	0	2

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.

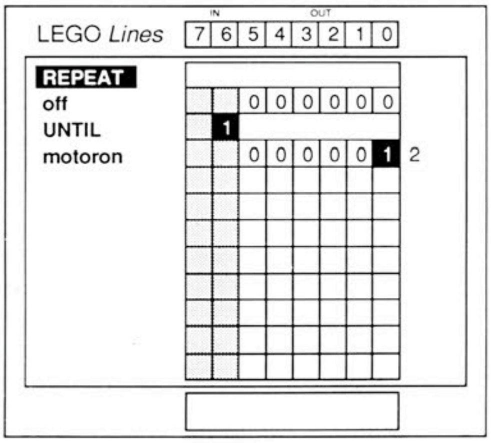
The LEGO *Lines* program has a number of reserved words which must not be used as labels because they are words associated with its important decision-making capability. They will always appear in capitals on the screen to distinguish them from labels, which always appear in lower case.

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



LEGO *Lines*

	IN 7	6	5	OUT 4	3	2	1	0	
REPEAT									
off			0	0	0	0	0	0	
UNTIL		1							
motoron			0	0	0	0	0	1	2

Program "SEVEN"

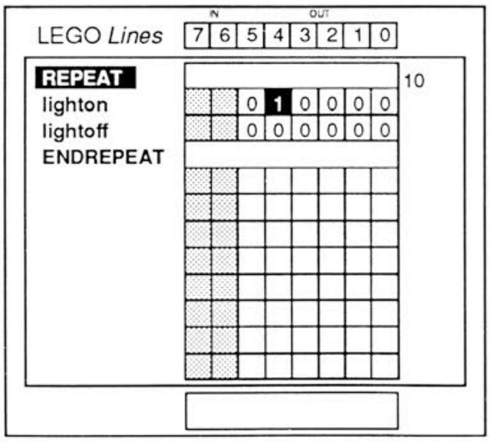
Lines will either be repeated or ignored **UNTIL** a certain input pattern is recognised by the program.

The input pattern on the **UNTIL** line is set using **f0**, **f1** and, to set **any value, SHIFT f0** or **SHIFT f1**.

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



LEGO *Lines*

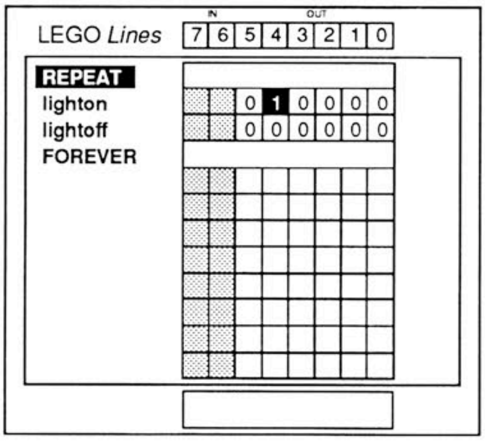
	IN 7	6	5	OUT 4	3	2	1	0	
REPEAT									10
lighton			0	1	0	0	0	0	
lightoff			0	0	0	0	0	0	
ENDREPEAT									

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



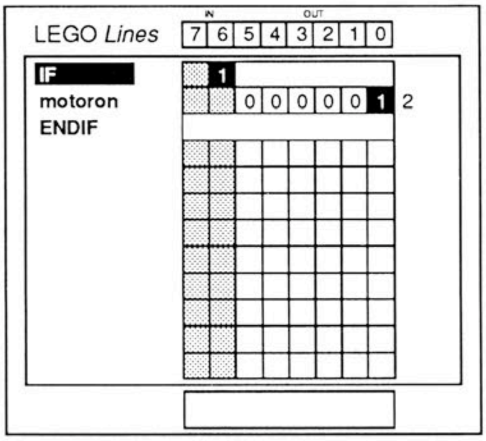
	IN		OUT					
LEGO *Lines*	7	6	5	4	3	2	1	0
REPEAT								
lighton			0	1	0	0	0	0
lightoff			0	0	0	0	0	0
FOREVER								

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



	IN		OUT						
LEGO *Lines*	7	6	5	4	3	2	1	0	
IF		1							
motoron			0	0	0	0	0	1	2
ENDIF									

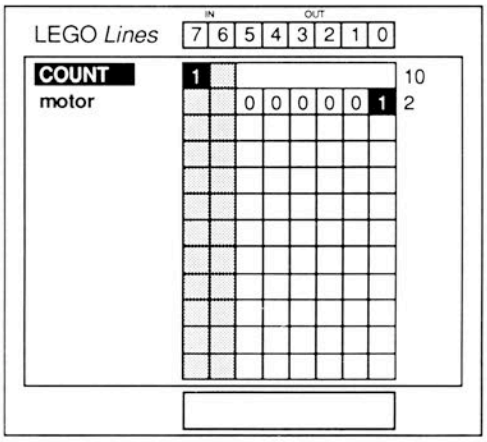
Program "NINE"

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



	IN		OUT						
LEGO *Lines*	7	6	5	4	3	2	1	0	
COUNT	1								10
motor			0	0	0	0	0	1	2

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

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

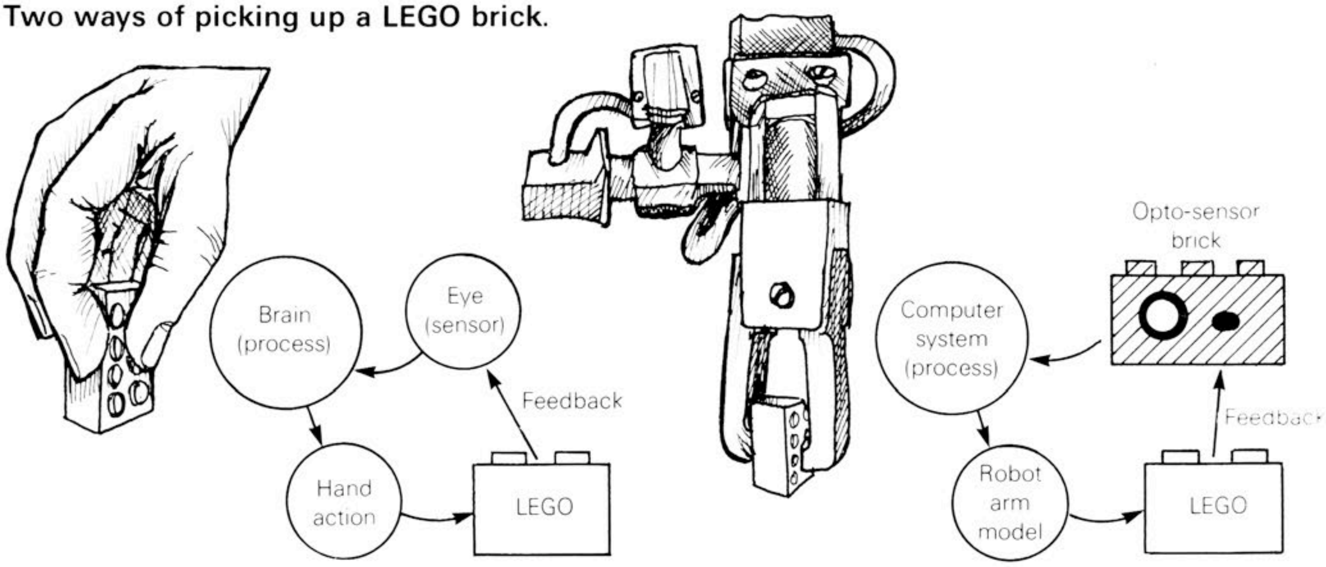
First write down the sentence about feedback.

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.

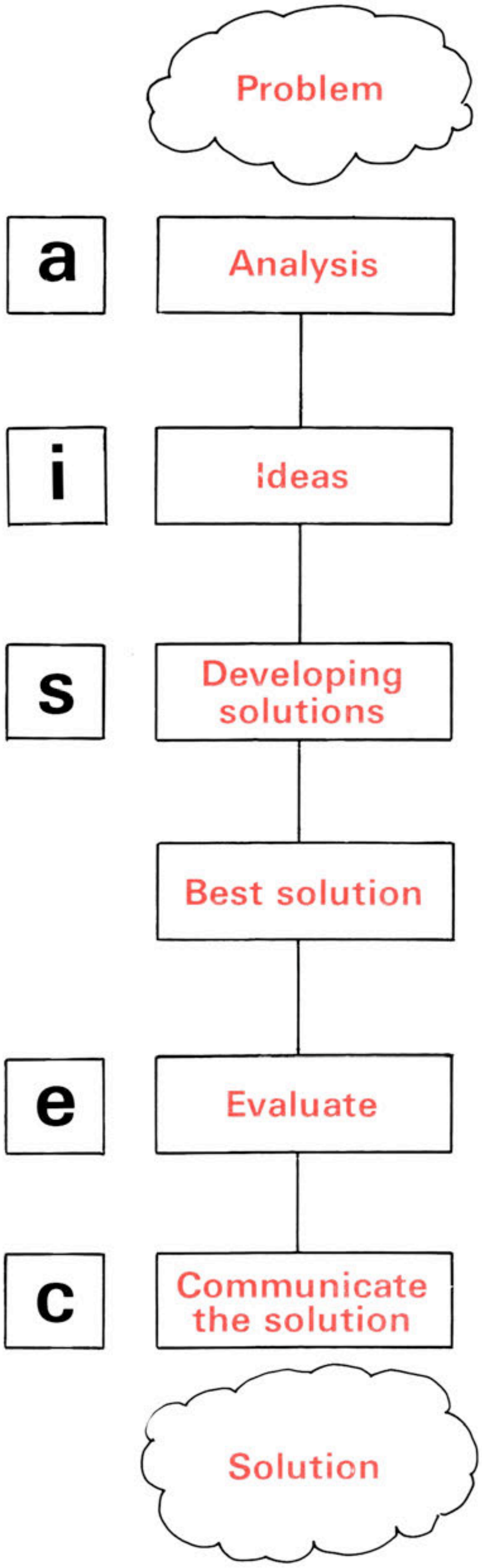
The computer system acts on the information it receives from the opto-sensor.

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.



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.

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.

Use a structured flow diagram (Resources booklet **R14**) to produce solutions.

Choose the best solution and translate the diagram into a program which you can test on the computer.

Judge whether the program achieves what you required it to do.

State what the problem was and describe your solution.

Structured flow diagrams (*sfd's*)

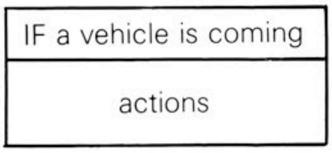
In order to construct a structured flow diagram it is necessary to take a sentence which describes a solution to a problem and identify:

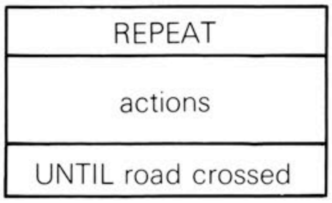
The ***sequence*** of events
The ***actions*** which make up these events
The ***tests*** which decide whether or not those actions take place
The ***loops*** in which actions or sequences of actions take place.

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

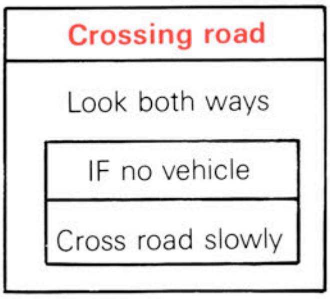
Sequence	Look both ways — if no vehicles — cross the road slowly
Actions	Look both ways — cross the road slowly
Tests	If there are no vehicles coming in either direction
Loops	—

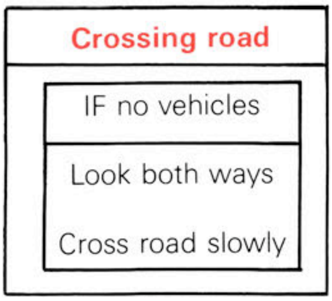
In a structured flow diagram the idea is to place actions into specially shaped boxes





For the example of crossing the road:

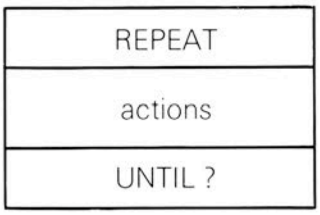




Structured flow diagrams (*sfd*'s)

R14

Loops



```
REPEAT
    actions
UNTIL ?
```

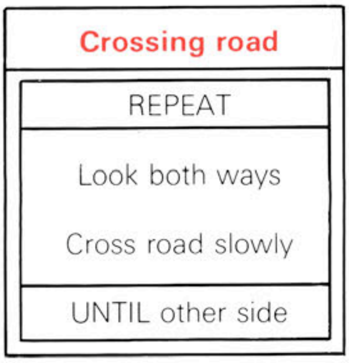
There are a number of ways to organise the condition (?) which finishes the REPEAT/UNTIL loop.

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

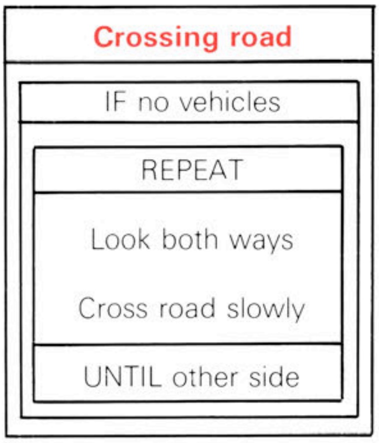
In the example:



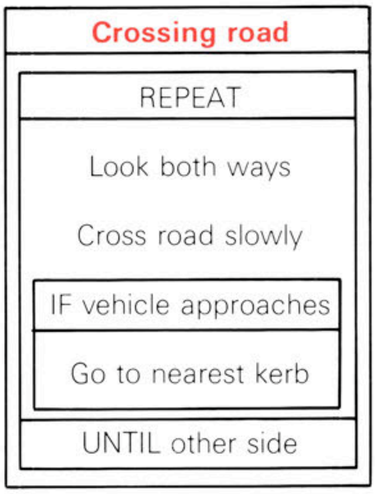
```
Crossing road
    REPEAT
        Look both ways
        Cross road slowly
    UNTIL other side
```

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
```

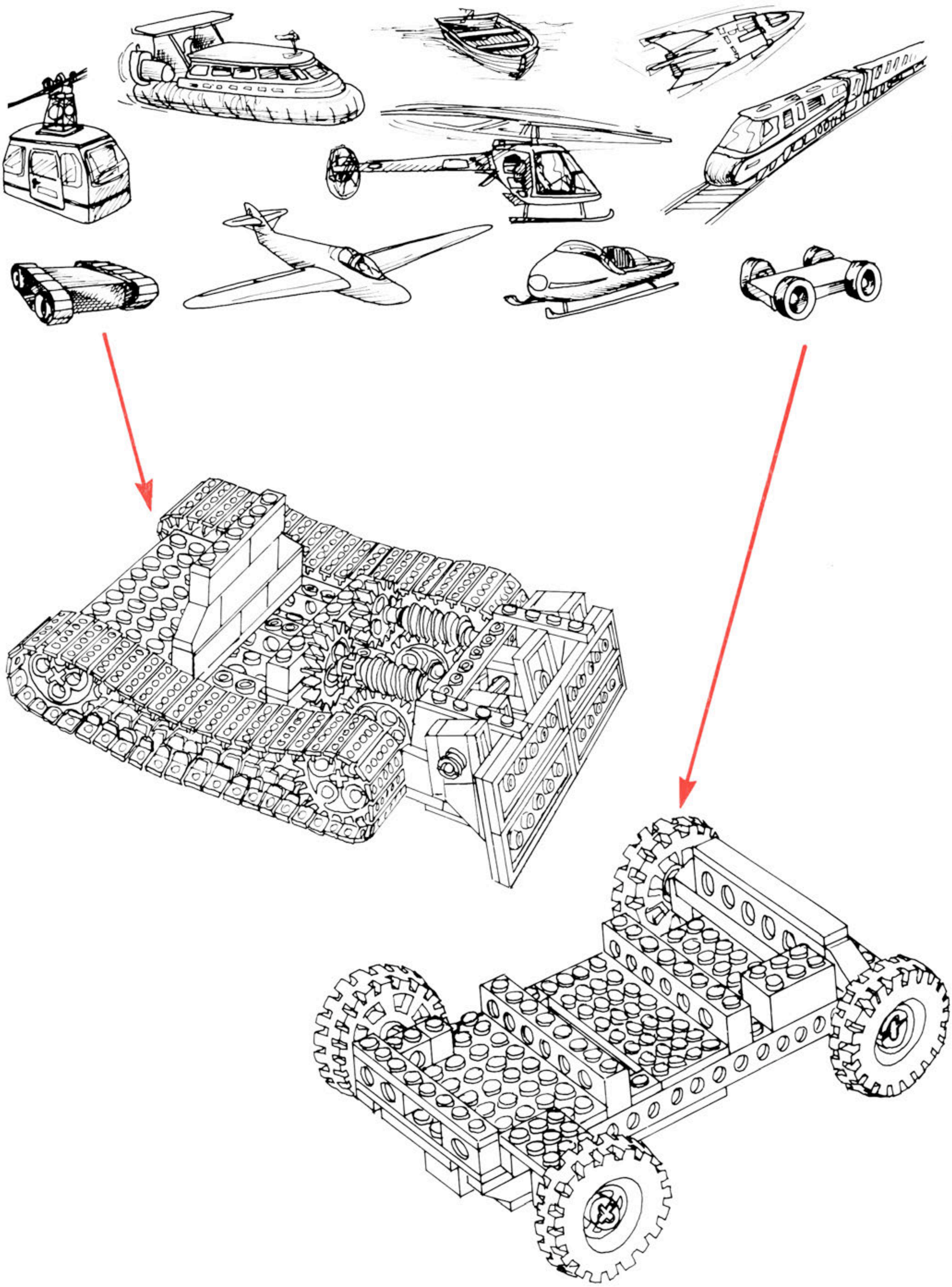


```
Crossing road
    REPEAT
        Look both ways
        Cross road slowly
        IF vehicle approaches
            Go to nearest kerb
    UNTIL other side
```

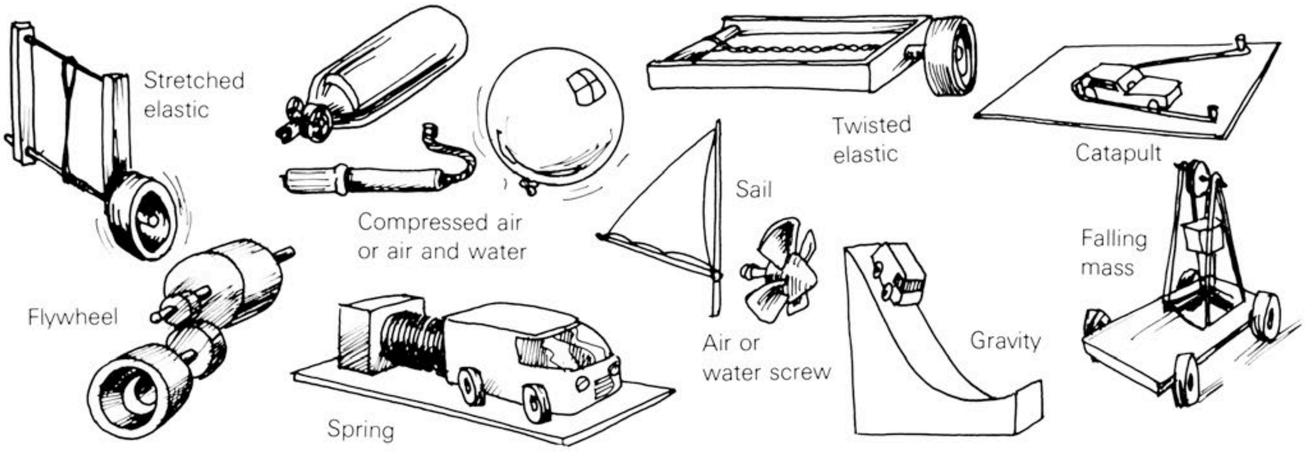
Loops or tests which are within other loops or tests are described as *nested*.

Vehicles — ideas on structures

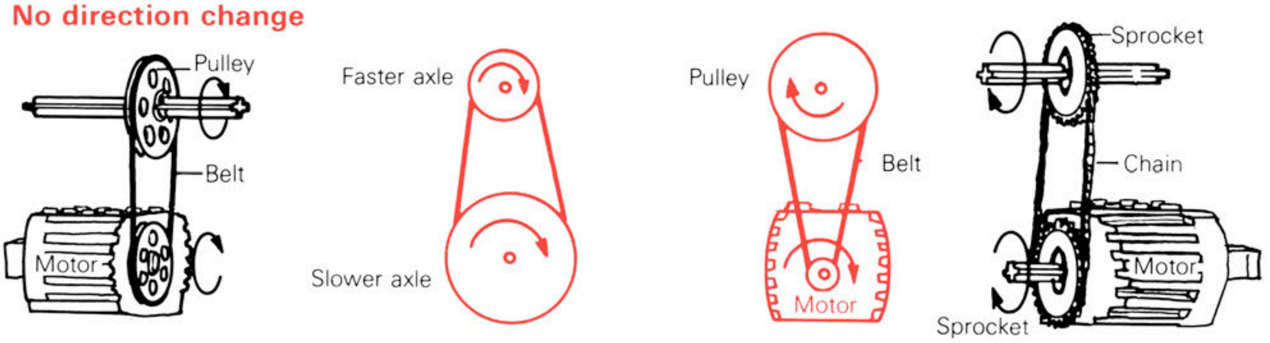
R15



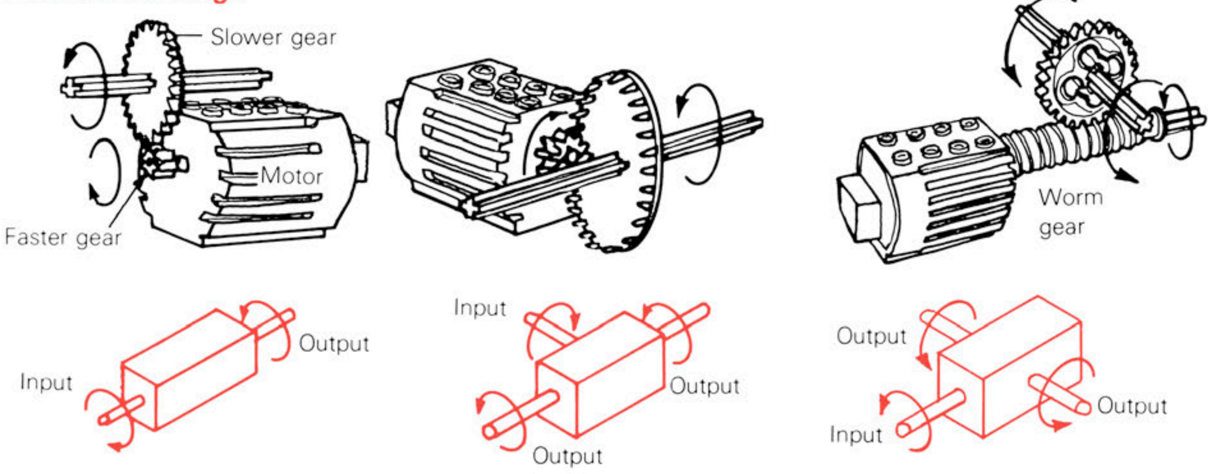
Vehicles — ideas on drive mechanisms R16



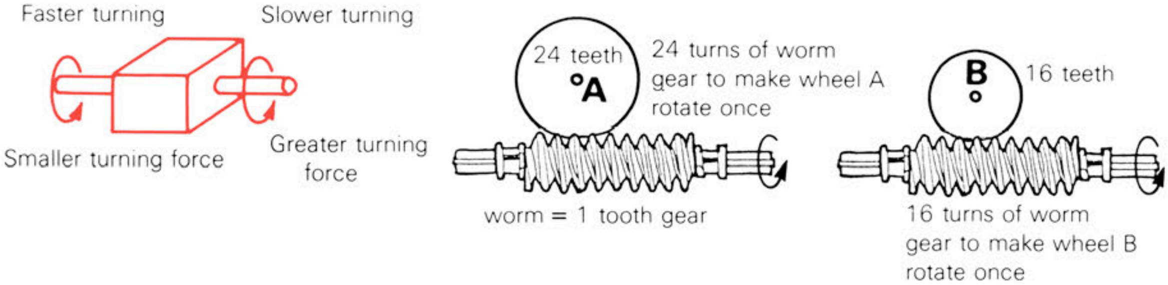
No direction change



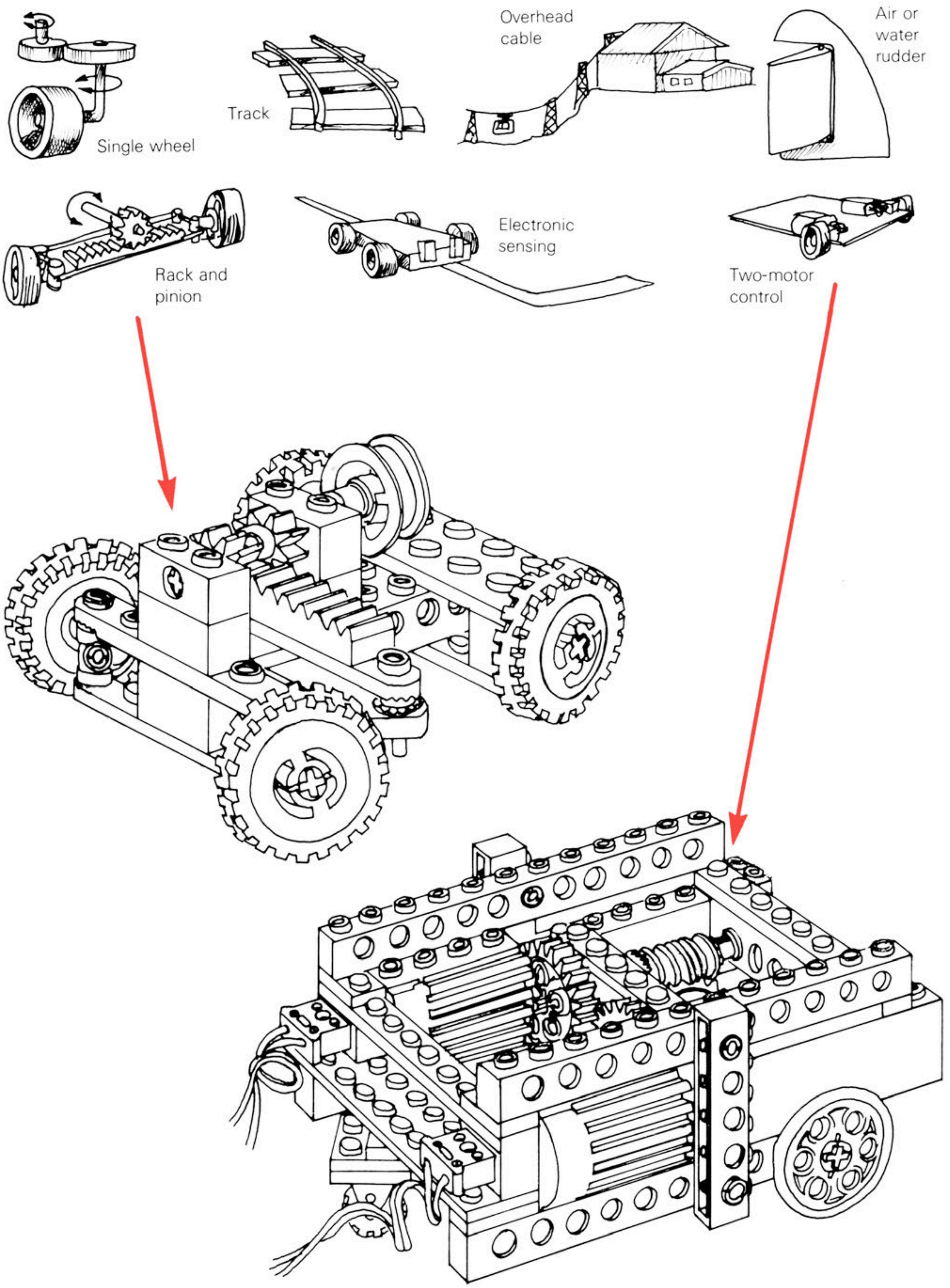
Direction change



Change of speed



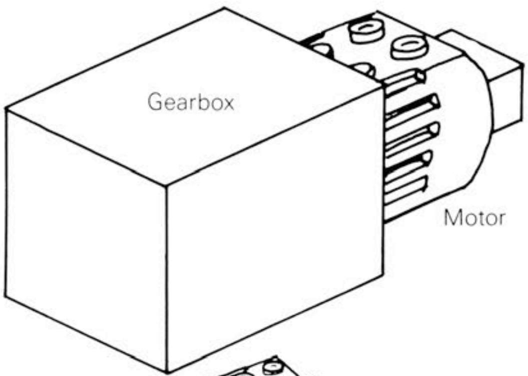
Vehicles — ideas on steering mechanisms

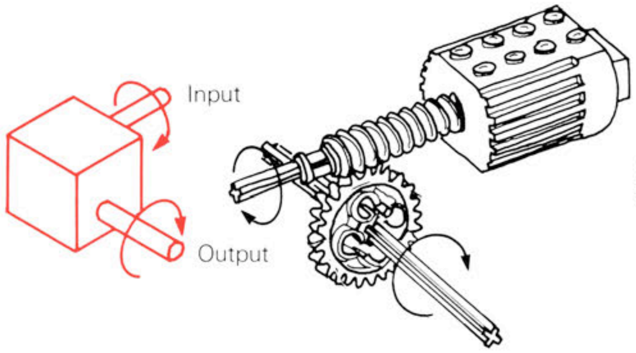


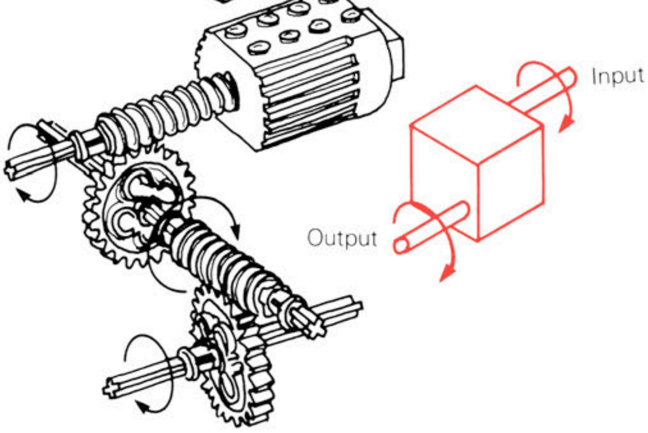
Vehicles — ideas on gearboxes

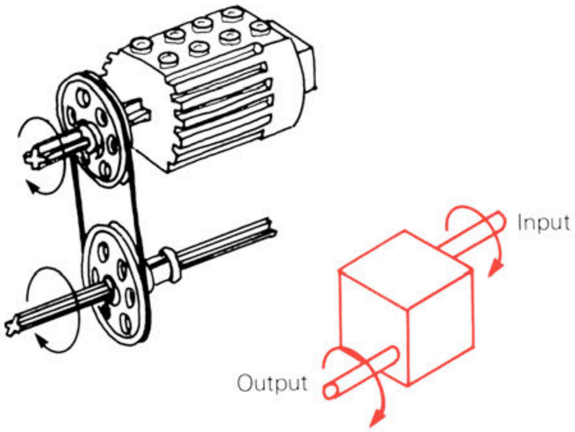
Gearboxes can be used to:

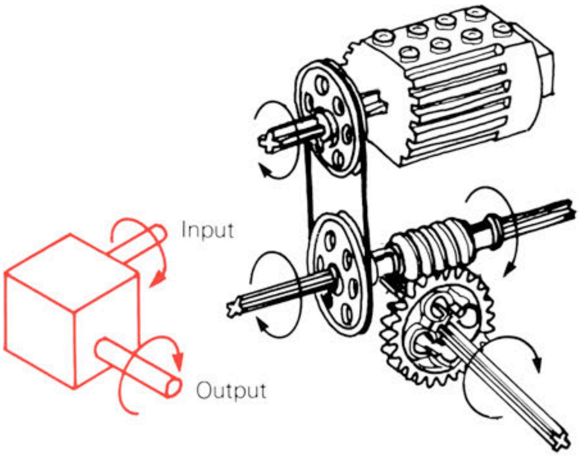
— change speed
— change direction
— change turning force.



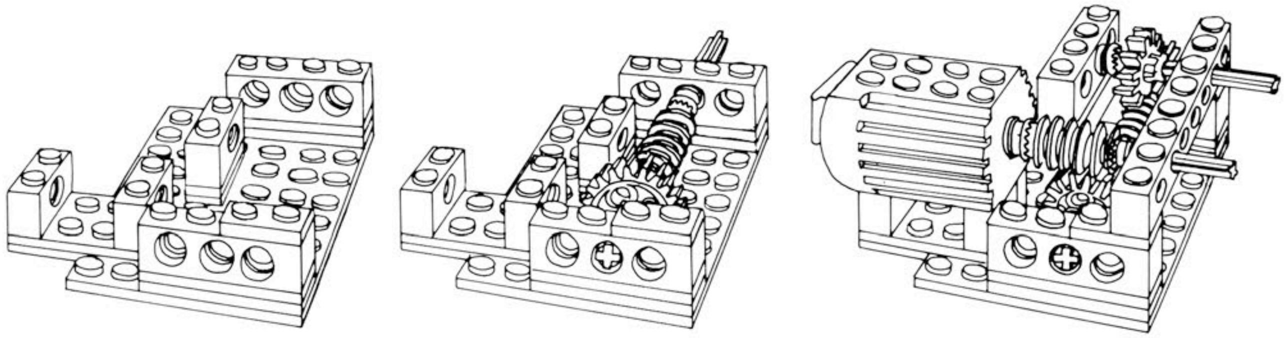


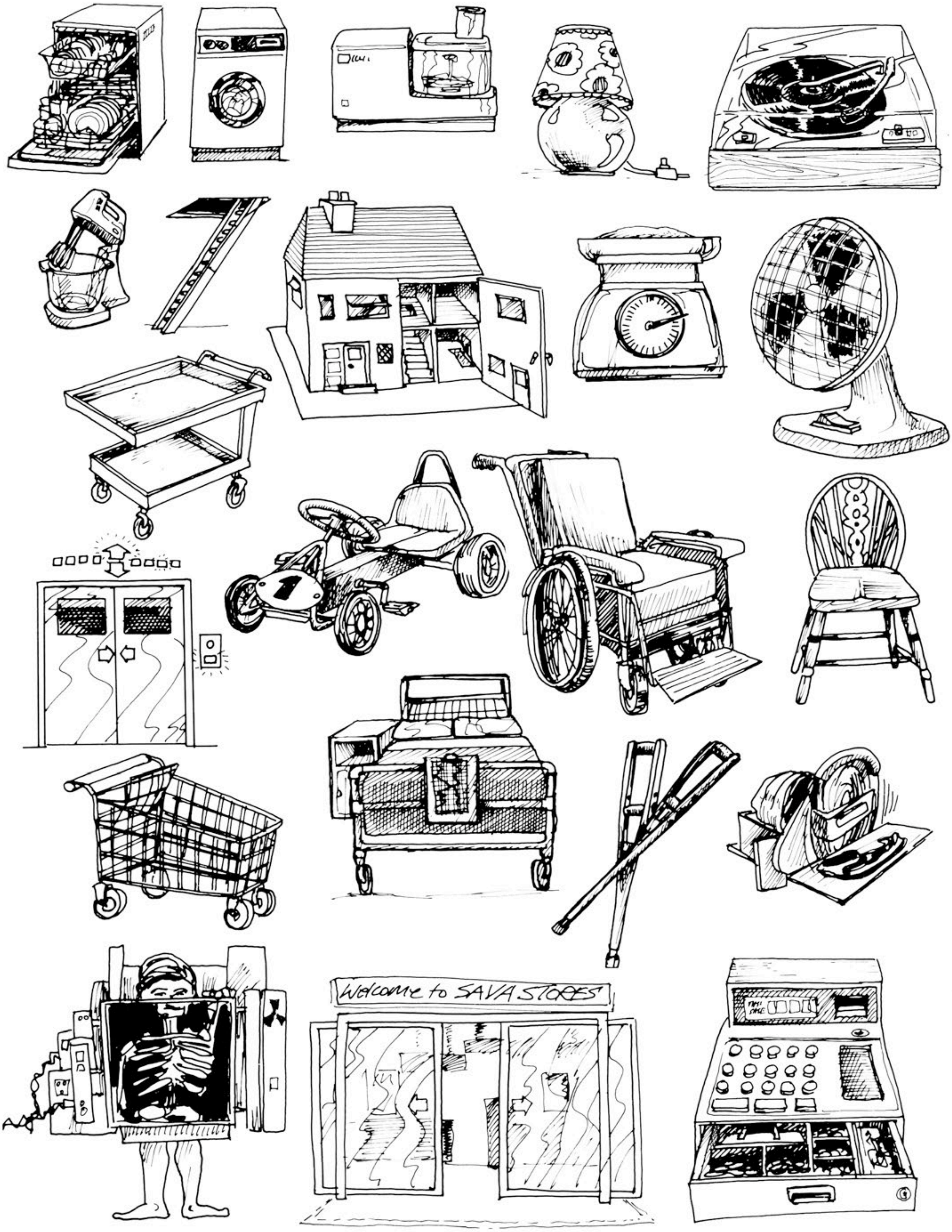


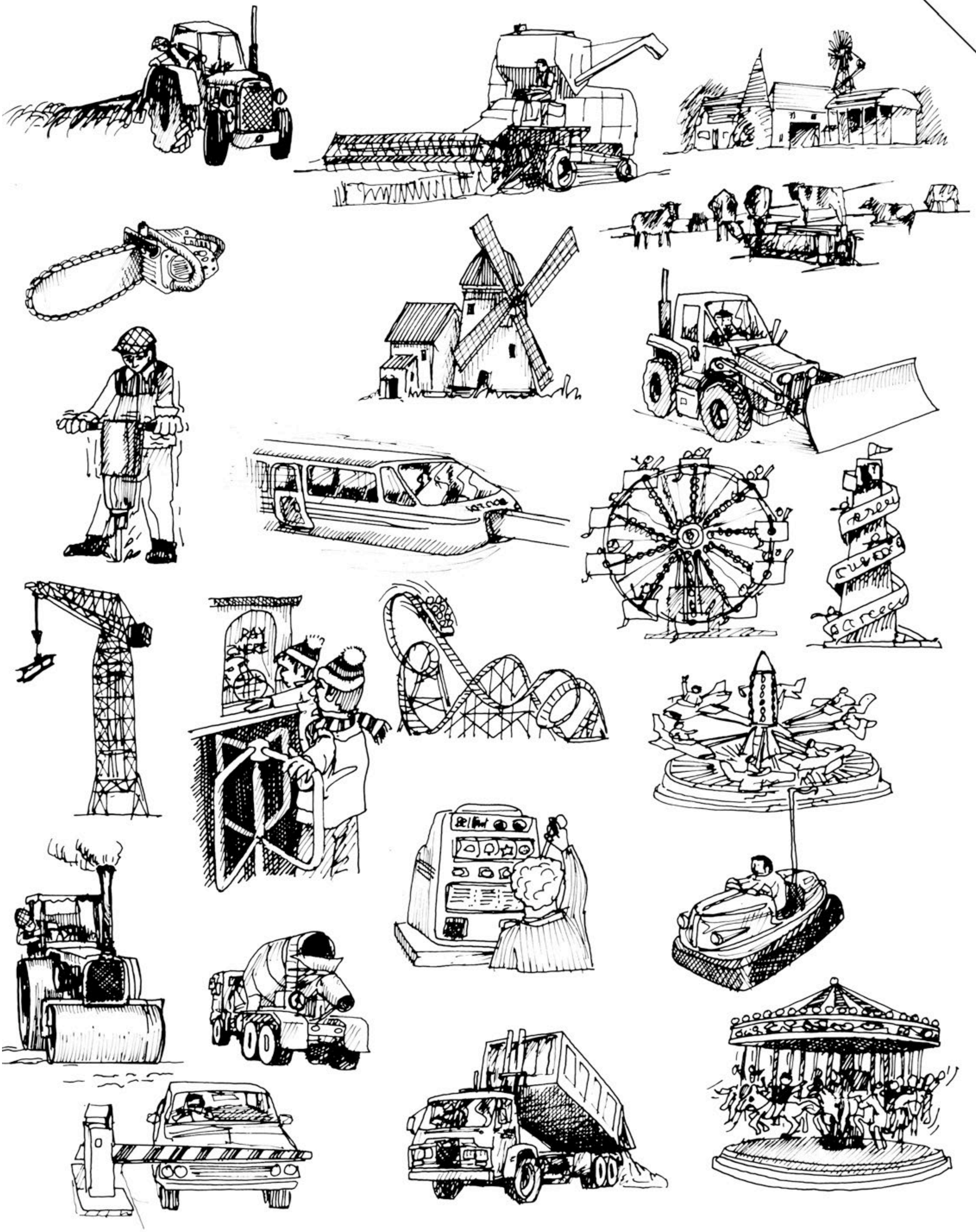


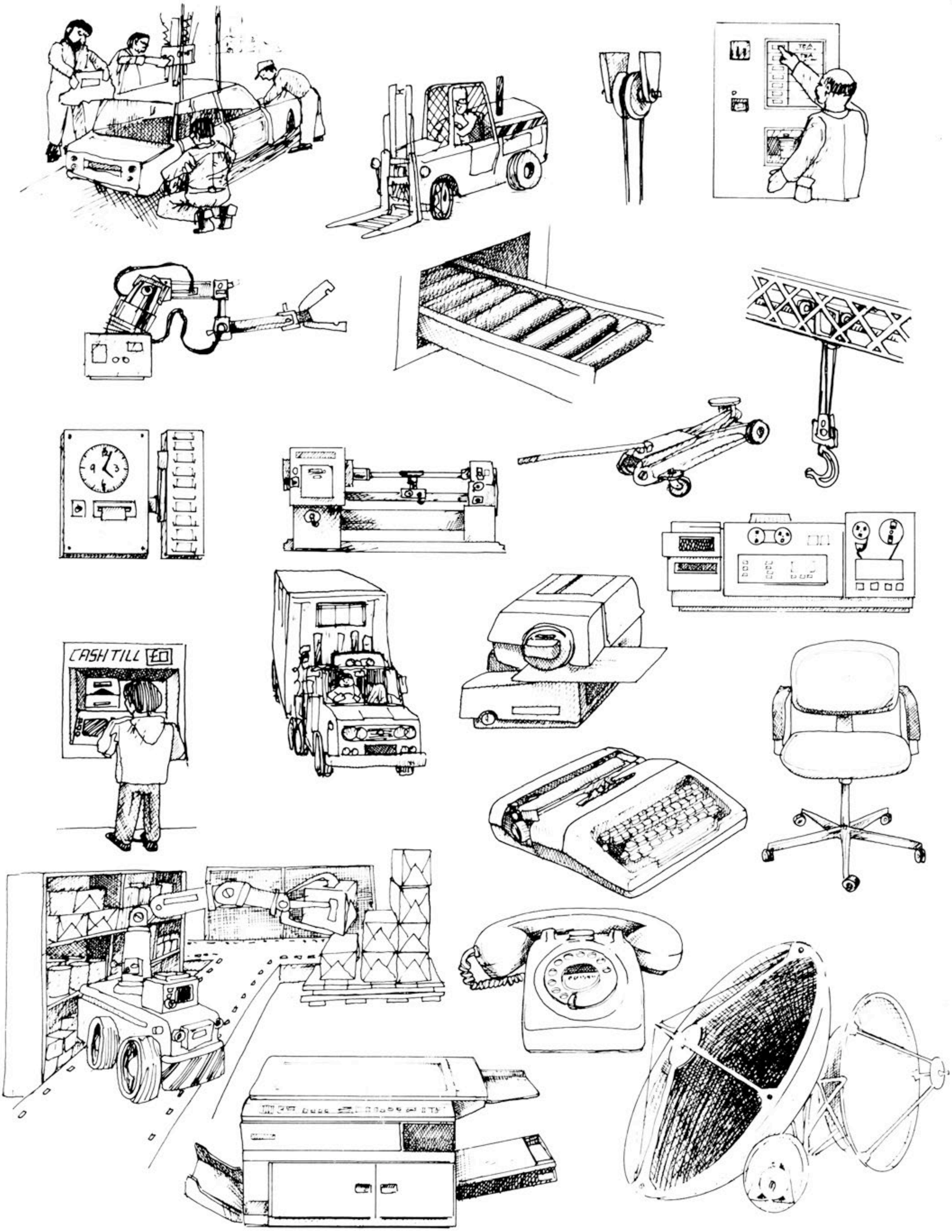


Gearbox for use in Assignment 3




1
Welcome to SAVA STORES




TEA
CASH TILL
£0

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**

Section 4

Technical reference

Introduction

This section is to enable you to get the most out of the LEGO ***Lines*** software environment. The first part is designed for the teacher who will need assistance in administering the computer resource.

It will help in providing an understanding of the different programs which make up LEGO ***Lines***, in using different types of equipment and in the preparation of disks for use in the classroom.

The second part of this section is intended for those with an interest or need to look more closely at how LEGO ***Lines*** works. ***Lines*** is written in BBC BASIC and a list of procedures and functions is given which enables you to emulate the way ***Lines*** communicates with the LEGO interface through your own BASIC programs.

The LEGO *Lines* program suite

LEGO ***Lines*** is not a single program but a collection of integrated programs written in BBC BASIC.

This collection (or ***suite***) is made up as follows:

`!BOOT` This program is the one which allows you to start the activity merely by pressing **SHIFT BREAK** instead of using a series of commands (not always easy to remember and very frustrating if you get them wrong). When you see autoboot in the text, this is the file which is executed. You will see that it takes care of the memory space required and calls the next program.

Note: if you intend using a serial printer with LEGO ***Lines***, you will need to alter this file (see pages 4.2–4.3).

`TITLE` This draws the title page you see as the first LEGO ***Lines*** screen. In the background, it is assembling some machine code to set up the User Port and to cope with the different graphics facilities found in the BBC B and Master Series microcomputers.

`LINDIS` From the title page you are led into the ***Lines*** display which this program creates.

`LINES` The main program of the suite, this is used for the input, editing and execution of instructions which form a ***Lines*** control program.

`SCCHAR` This program draws the reduced-size characters which are used on the ***Lines*** display.

`LLOAD` This program (called by pressing **CTRL f8**) handles the loading of ***Lines*** control programs from the disk.

`LSAVE` This program (called by pressing **CTRL f9**) handles the saving of the currently displayed ***Lines*** control program to the disk.

`PROUT` This program (called by pressing **CTRL f7**) handles the printing of the current control program on any parallel printer (see pages 4.2–4.3 if you are using a serial printer).

`CAT` This program (called by pressing **CTRL f3**) displays a list (***catalogue***) of the programs on the disk you are using. Control programs created by Lines appear as:

L.*name*

`BOXDIS` This program (called by pressing **CTRL f4**) displays the structure of the current ***Lines*** control program in a form very similar to a structured flow diagram.

DISCUT This program is a utility which enables you to make use of double or dual disk drives (see page 4.3). It allows you to specify which disk drive you wish to use in loading and saving operations. It is separate from the *Lines* suite in that, if it is to be used, it must be run before starting a *Lines* session; it cannot be called during the activity.

It is used in the following way:

Insert the master program disk into drive **0**
Type **CHAIN "DISCUT"** and press **RETURN**

This displays an information screen which prompts you to enter the drive number you wish to use.

Enter the number of the drive (**1**–**3** only) and press **RETURN**

The screen will confirm your selection.

You may now press **SHIFT BREAK** to start your *Lines* session.

Note: there are two errors which may occur when using this program:

The program writes the drive number of the drive to be used back to the master program disk (to be called at any point). If the write-protect sticker has not been removed from the disk, the program will fail, displaying the message:

```
disk error
```

If you are using a single drive, and this program has been run to set the drive to a number other than **0**, the *Lines* program will fail with the message:

```
disk error
```

DISCNO This is a data file, on the master program disk, which stores the number of the drive selected in **DISCUT**. This is referenced in subsequent loading and saving operations.

Using LEGO *Lines* with different types of equipment

The LEGO *Lines* software is designed to be used on a variety of different configurations. These are outlined below, together with actions you need to take to enable it to be run in your special circumstances.

Different microcomputers

LEGO *Lines* has been tested and will run successfully on the current range of BBC microcomputers: the BBC B, the BBC B+ and the Master Series (operating in DFS). All must, however, be capable of using 5.25 inch disk drives in 40-track format (the standard configuration).

Different BASICs

LEGO *Lines* has been tested with BBC BASIC version 1 and version 2.

Using a serial printer with LEGO *Lines*

The print utility (called by pressing **CTRL f₇**) assumes that you wish to print the current program onto a parallel printer attached to the printer port under the machine. If, however, you wish to use a serial printer, connected to the RS423 DIN socket at the back of the machine, you will need to rebuild the **!BOOT** file, to accommodate this.

Insert the master program disk in drive **0**

Type ***ACCESS !BOOT** and then press **RETURN**
Type ***BUILD !BOOT** and then press **RETURN**

You will then be prompted with line numbers

On line **1** type ***BA.** and then press **RETURN**	(calls BASIC)
On line **2** type ***FX5,2** and then press **RETURN**	(initialises serial port for printing)
On line **3** type ***FX6,0** and then press **RETURN**	
On line **4** type **PAGE=&1900** and press **RETURN**	(sets the correct memory location)
On line **5** type **CH. "TITLE"**and then press **RETURN**	(calls the next program)
On line **6** press **ESCAPE** to finish building **!BOOT**	(it is written back to your master disk as soon as you press **ESCAPE**)

Finally, protect this new **!BOOT** by locking it.

Type ***ACCESS !BOOT L** and press **RETURN**

Using multiple-disk drives

The most common disk drive in use is the single disk drive, reading and writing in 40-track format. *Lines* assumes that this is being used unless it is instructed otherwise (by running program **DISCUT**, described on page 4.2). While this type of drive may be the most common, it is not exclusive and you may be asked to use other types, the two most likely being:

a *double* disk drive (single slot but capable of reading both sides of the disk)

or a *dual* disk drive (two slots reading both sides of both disks)

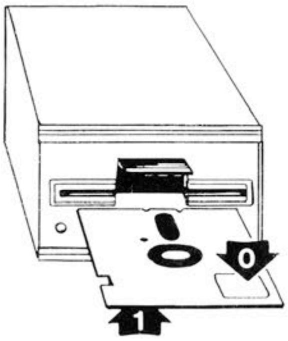
The drive numbers shown on the diagram are those you would select if you were using program **DISCUT** (described on page 4.2). To gain the maximum advantage from this configuration, you should adopt the following procedure:

Insert the master program disk in drive **0**

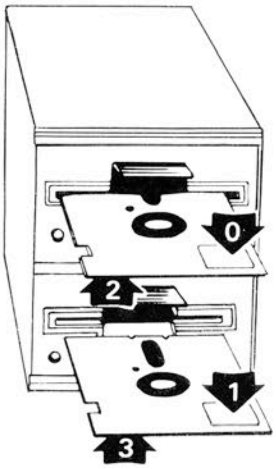
Run program **DISCUT** as described, selecting a drive other than drive **0** for your data disk (or data side)

Press **SHIFT BREAK** to autoboot LEGO *Lines*

You are now loading and saving programs on the specified data disk, leaving the master untouched, and removing some of the need for preparation you would otherwise require.



A double disk drive



A dual disk drive

Note: either of the drive types described may be of the ***switchable*** variety, enabling them to read and write disks in either 40- or 80-track format. (If the switch is correctly set and your disk is formatted to 80-track, it is possible to save programs on a data disk on that 80-track drive.)

The drive numbers shown may differ on various drives. Always check the manufacturers' documentation for confirmation.

Preparing disks for use in the classroom

We recommend the following regime:

- *As soon as you receive the pack* — make **backup** copies of the two disks supplied and put the originals in the safe (or other secure place).
- *Designate disks for use by each group* — make sure they are **format**ted (remember that the master disk is supplied in 40-track format).
- *Copy the LEGO Lines software to these disks* — using the **backup** procedure to make a master for each group.
- *Before a session on the materials* — **copy** the appropriate sample program or programs (always outlined in the teacher's commentary pages in the *Classroom materials* section) onto these group master disks.
- *After a session* — **wipe** these programs (taking care not to destroy any of the LEGO *Lines* software, or any programs created by the group) off the disk.
- *If in doubt as to what is on a disk* — take a **catalogue** of the disk and make your decisions on the basis of what you find.

Here is a brief guide to these operations:

Formatting a disk

When you purchase a disk, it is just a blank surface, incapable of being used. Formatting 'stamps' a pattern onto the disk surface to allow the disk to be read and to be written to. The procedure used to format the disk differs according to which Disk Filing System your machine uses. This fundamental procedure is invariably well documented in the appropriate *DFS User Guide* and what follows describes the operation using a single disk drive:

Insert your utilities disk (if one is supplied with the computer) into the drive

Type ***FORM40** and press **RETURN**

Remove your utilities disk and insert the disk to be formatted

In response to the prompt, type **Y**

Formatting will now begin

When it is complete, you will be returned to the BASIC arrow prompt.

Note: formatting creates a blank, ordered disk ready for use. It can be used to clear old disks to enable them to be reused. The danger in this is, of course, that you may accidentally wipe off some useful data. In this case you could wipe off the utilities disk — a disaster! Whenever you use the utilities disk, make sure that it is protected by a write-protect sticker (if you attempt to format over this, you will fail with an error message).

Making a backup copy

This operation takes the contents of one disk and makes an identical copy onto another disk. Again, the procedure will differ according to which Disk Filing System you are using and it will be described in detail in the accompanying documentation. Here is the procedure for backing up on a single drive system:

Organise two disks, your ***source*** (the master you are copying) and the ***destination*** (the disk, formatted, you are copying onto)

Type ***ENABLE** and then press **RETURN**
Type ***BACKUP 0 0** and then press **RETURN**

You should then just follow the instructions on the screen until the backup is complete, when you will return to the BASIC arrow prompt.

Like formatting, backing up will simply dump things onto a disk and, if you get your disks out of sequence, it can dump the wrong information onto the wrong disk. Again, you should ensure that your source disk is protected by a write-protect sticker so, even if you get the disks muddled, you cannot do any harm.

Copying individual files

Backing up takes an image of the whole disk and dumps it to another disk, not always what is required. You may have something valuable on your destination disk (in this case LEGO *Lines*!) and need to add another program to it. This requires the use of the **COPY** command. Using a single drive, follow these instructions:

> Organise two disks, the ***source*** (containing the program you wish to copy) and the ***destination*** (the disk, already containing useful data, you want it transferred to)
>
> Type ***COPY 0 0** L.*name* and press **RETURN**
>
> (all *Lines* programs have this format, for example **L.CONVEY**)

Again, just follow screen instructions on inserting the correct disk at the correct time and wait for the arrow prompt to confirm that the copy has been effected successfully.

Wiping unwanted programs off the disk

While formatting clears the whole surface of the disk, you will need at times to remove single files from a disk, without disturbing the useful data still on it. This is effected by using the **WIPE** command. Using a single drive, follow these instructions:

> Insert the disk containing the unwanted programs in the drive
>
> Type ***WIPE** *.* and press **RETURN**
>
> (the *.* calls every file on the disk for your decision)

Everything on the disk is now presented to you by name, for you to decide whether to wipe it off or not:

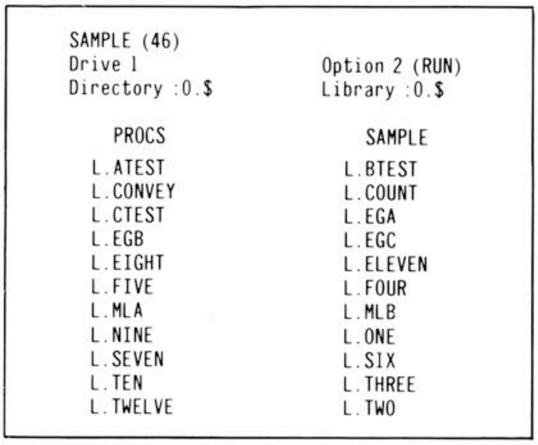
> Press **Y** to wipe a file off the disk
> Press anything else to retain it on the disk

Taking a catalogue of the contents of a disk

The **CAT** command provides you with a list of everything on a particular disk.

> Insert the disk into drive **0**
> Type ***CAT** and press **RETURN**

You will receive a catalogue of the disk contents.



```
SAMPLE (46)
Drive 1                          Option 2 (RUN)
Directory :0.$                   Library :0.$

     PROCS                            SAMPLE

  L.ATEST                          L.BTEST
  L.CONVEY                         L.COUNT
  L.CTEST                          L.EGA
  L.EGB                            L.EGC
  L.EIGHT                          L.ELEVEN
  L.FIVE                           L.FOUR
  L.MLA                            L.MLB
  L.NINE                           L.ONE
  L.SEVEN                          L.SIX
  L.TEN                            L.THREE
  L.TWELVE                         L.TWO
```

Directory of sample program disk

LEGO *Lines* and BBC BASIC

LEGO ***Lines*** is written in BBC BASIC and forms an integrated suite of programs. There are, however, opportunities for going beyond the scope of the current version of ***Lines***, requiring the use of ***Lines*** procedures for communicating with the interface and beyond. These have been incorporated on the Sample programs disk under the program title, **PROCS**. A demonstration program, **SAMPLE**, is also included to illustrate the procedures in action. These two programs are described here, together with the action required to merge them into a single demonstration program.

Program PROCS

This program starts with line **1000** to enable you to load it over your own BASIC program (or program **SAMPLE**) without the necessity for rekeying.

The actual code is given, together with a brief description of what is happening.

```
1000 DEFPROCsetupinoutbits
1010 A% = &97 : X% = &62 : Y% = 63
1020 CALL &FFF4
1030 ENDPROC
1040 REM
```

This procedure sets up the User Port to consist of two input bits (6 and 7) and six output bits (0–5).

```
1050 DEFPROCoutputbits(A%)
1060 B% = 0
1070 FOR I% = 5 TO 0 STEP - 1
1080 C% = 10 ↑ I%
1090 IF A% DIV C% = 1 B% = B% + 2 ↑ I% : A% = A% MOD C%
1100 NEXT
1110 PROCoutput(B%)
1120 ENDPROC
1130 REM
```

This procedure allows you to specify an output pattern as a binary expression of six characters, just as ***Lines*** does. Note that this pattern is converted to a decimal number and only then is it output to the User Port by **PROCoutput(B%)**.

Example **outputbits(001100)** which will set output bits 2 and 3 on.

```
1140 DEFPROCsetbiton(A%)
1150 IF A% < 0 OR A% > 5 ENDPROC
1160 E% = 2 ↑ A% : PROCreadoutputbits
1170 PROCoutput(Y% OR E%)
1180 ENDPROC
```

This procedure sets on the particular output bit specified. The status of the other bits is not altered (that is, if they were on before then they stay on). Note that **PROCreadoutputbits** calls in the status of the other output bits before the new bit pattern is output to the User Port.

Example **setbiton(5)** which will set bit 5 on, maintaining the status of the others.

```
1190 DEFPROCsetbitoff(A%)
1200 IF A% < 0 OR A% > 5 ENDPROC
1210 E% = 2 ↑ A% : PROCreadoutputbits
1220 PROCoutput(Y% OR E% EOR E%)
1230 ENDPROC
1240 REM
```

This procedure performs in exactly the same way as the **setbiton**, but now it sets a specified output bit off, leaving the status of the other bits unaltered.

Example **setbitoff(5)** which will set bit 5 off, maintaining the status of the others.

```
1250 DEFPROCmotor(A$)
1260 PROCremovepadding
1270 motor% = ASC(LEFT$(A$,1)) - 65
1280 IF motor% < 0 OR motor% > 2 ENDPROC
1290 B$ = RIGHT$(A$,LEN(A$) - 1)
1300 IF B$ <> "RIGHT" AND B$ <> "LEFT" AND B$ <> "OFF" ENDPROC
1310 bit% = 2 * motor% : PROCreadouputbits
1320 IF B$ = "OFF" Y% = Y% OR 2 ↑ bit% OR 2 ↑ (bit% + 1) EOR 2 ↑
     bit% EOR 2 ↑ (bit% + 1) : PROCoutput(Y%) : ENDPROC
1330 IF B$ = "RIGHT" right% = 0 ELSE right% = 1
1340 bit% = bit% + right%
1350 IF right% = 0 bit2% = bit% + 1 ELSE bit2% = bit% - 1
1360 Y% = Y% OR 2 ↑ bit% OR 2 ↑ bit2% EOR 2 ↑ bit2%
1370 PROCoutput(Y%)
1380 ENDPROC
1390 REM
```

This procedure allows you to specify the bi-directional output bits A, B or C on the interface instead of just a single output bit. This enables you to set a motor off or to set it in one direction or another. **PROCremovepadding** converts the expression you enter in the brackets to two arguments, the output bit being specified (output bits A, B or C only) and the instruction (OFF, RIGHT or LEFT). The entry in the brackets must be enclosed in double quotes and the arguments must be in capitals. The separator is currently read as being either a space, a comma or a semi-colon (see **PROCremovepadding** line **1780**) but you may alter this to your own specification.

Example **motor("A,LEFT")** which sets output bits A to rotate a motor to the left.

```
1400 DEFPROCcountfor(loops%,channel%)
1410 IF channel% <> 6 AND channel% <> 7 ENDPROC
1420 FOR I% = 1 TO loops%
1430 status% = FNread(channel%)
1440 REPEAT
1450 UNTIL FNread(channel%) <> status%
1460 NEXT
1470 ENDPROC
1480 REM
```

This procedure emulates the **COUNT** function of *Lines*, allowing you to specify a number of signals to be counted (**loops%**) and the input channel being read (**channel%**). It calls the function **read(channel%)** to determine whether there has been a change of state on the appropriate input bit.

Example **countfor(15,6)** which counts 15 changes of state on input bit 6.

```
1490 DEFFNreadcount(channel%)
1500 IF channel% <> 6 AND channel% <> 7 ENDPROC
1510 status% = FNread(channel%)
1520 timeout% = TIME + 50
1530 REPEAT
1540 UNTIL FNread(channel%) <> status% OR TIME > timeout%
1550 IF TIME > timeout% = 0 ELSE = 1
1560 REM
```

This is a function which is used to determine whether a change of state has been encountered on an input channel. When it has encountered an off state and an on state, it registers that a count has taken place. Function **read** actually reads the on or off status of the input bit being tested.

The timeout detects a 'jam' in that no change is taking place for this length of time (half a second — 50 x 1/100th of a second).

```
1570 DEFPROCreadoutputbits
1580 A% = &96 : X% = &60 : !&70 = USR(&FFF4) : Y% = ?&72 AND 63
1590 ENDPROC
1600 REM
```

This procedure reads the status of the output bits (0–5) and returns this as a decimal number into the variable **Y%**.

Example if **Y% = 3** then output bits 0 and 1 (only) are set on.

```
1610 DEFFNread(D%)
1620 A% = &96 : X% = &60 : !&70 = USR(&FFF4) : Y% = ?&72 AND
     192 : E% = 2 ↑ D%
1630 IF (Y% AND E%) = E% = 1 ELSE = 0
1640 REM
```

This function reads the status of an input bit (**D%** = 6 or 7.) It returns the value **1** if it is on, or value **0** if it is off.

```
1650 DEFPROCoutput(Y%)
1660 A% = &97 : X% = &60 : CALL &FFF4
1670 ENDPROC
1680 REM
```

This procedure is called whenever you wish to send a bit pattern to the User Port. It writes a decimal value to the port, sending or denying power to the specified output bits.

Example **output(5)** which sends power to output bits 0 and 2 only.

```
1690 DEFPROCwait(time)
1700 T% = TIME + (time * 100)
1710 REPEAT
1720 UNTIL T% < TIME
1730 ENDPROC
```

This procedure allows you to build in a time delay between actions. This delay, the variable time, is entered as the number of seconds you require the program to wait.

Example **wait(10.5)** which halts program execution for ten and a half seconds.

```
1740 DEFPROCremovepadding
1750 C$ = " "
1760 FOR I% = 1 TO LEN(A$)
1770 B$ = MID$(A$,I%,1)
1780 IF B$ <> " " AND B$ <> "," AND B$ <> ";" C$ = C$ + B$
1790 NEXT
1800 A$ = C$
1810 ENDPROC
```

This procedure is called by **PROCmotor** and converts the input (**A$**) to the required format: the output bits and the action to be performed as a single expression, to be used in lines **1270** and **1290** of **PROCmotor**.

Example if **motor** passes **A$ = "B;RIGHT"** to **removepadding**, **A$** will be converted to **BRIGHT** for use in **motor**.

Note: line **1780** determines which characters (currently space, comma and semi-colon) can be used as separators. You can alter this line if you have a preference for a different character.

Program SAMPLE

Merging SAMPLE with PROCS

This program utilises the procedures to demonstrate their use in BASIC. Before it can be used, it must be merged with the procedures contained in the program **PROCS**.

Type **LOAD "SAMPLE"** and press **RETURN**

SAMPLE has line numbers below those of **PROCS**, this is important and requires it to be loaded first.

Type **PRINT ~TOP-2** and press **RETURN**

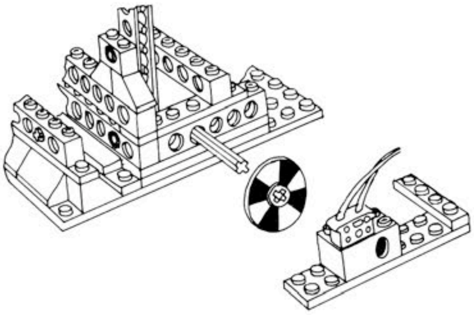
This statement prints out the ***top of memory address***, where the second program is to be loaded.

Type ***LOAD PROCS** ***address*** and press **RETURN**

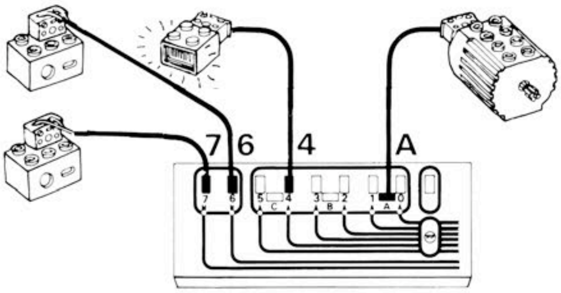
Note: you use ***LOAD** instead of just **LOAD. PROCS** does not go in double quotes and the address is what was returned by the previous print statement.

Using the sample program

The program requires the use of model 1090 B the Automatic Door. You will have to add an opto-sensor brick and segmented disc to the gearbox shaft as shown in the diagram.



The motor should be connected to output bits A. The light brick should be connected to output bit 4. The opto-sensor detecting the status of the door (open or closed) should be connected to input bit 7. Finally, the opto-sensor which you have added (utilising the segmented disk) should be connected to input bit 6 (the sensor should be set to 0 initially).



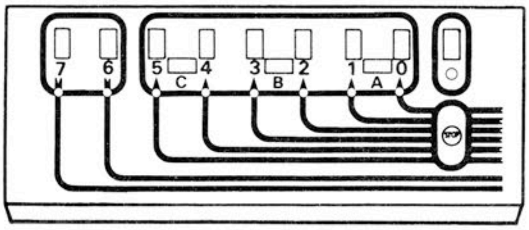
Program SAMPLE

```
100 REM PROGRAM SAMPLE
101 REM Demonstrates LEGO Lines
102 REM BBC BASIC procedures for use
103 REM with model 1090B Automatic
104 REM Door
105 REM
200 PROCsetupinoutbits
201 REM: Sets up User Port
202 REM
210 REPEAT
211 REM: Start of loop
212 REM
220 PROCsetbiton(1)
221 REM: bit 1 closes door
222 REM
230 REPEAT:UNTIL FNread(7) = 1
231 REM: Looks for a change in bit 7
232 REM  (from 0 to 1)
233 REM
240 PROCmotor("A,OFF")
241 REM: Turns motor off
242 REM
250 PROCwait(2)
251 REM: Waits 2 seconds
252 REM
260 PROCoutputbits(010001)
261 REM: Turns light on and turns
262 REM motor on to open door
263 REM
270 REPEAT:UNTIL FNreadcount(6) = 0
271 REM: Opens door until timeout
272 REM
280 PROCoutputbits(000000)
281 REM: Turns output bits off
282 REM
290 UNTIL FALSE
291 REM: End of loop
292 REM
300 END
```

Section 5

Duplication masters

Program sheet



	IN		OUT						
	7	6	5	4	3	2	1	0	
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							
	▨	▨							

Assessment

a Analysis

Stating the problem accurately
Exploring the problem
Developing deeper understanding
Developing criteria

s Finding and developing the best solution

Evaluating each solution
Choosing the best solution
Modelling the best solutions

c Communication

Discussion with others
Listening to other's ideas
Drawing sketches and diagrams
Writing a report
Giving a talk

i Finding and developing ideas

Discussion with others
Looking and thinking up ideas
Investigating ideas
Finding several solutions

e Evaluation

Using the criteria to carry out tests
Analysing the tests' results
Making modifications when necessary

t Teamwork

Contributing ideas
Sharing work equally
Working cooperatively

Assignment	*Notes*

Assignment	*Notes*

Assignment	*Notes*

Assignment	*Notes*

Assignment	*Notes*

BBC 'B' MICROCOMPUTER

Lines © 1986 LEGO Group	SHIFT f0	SHIFT f1	◀SHIFT f	CTRL f3	CTRL f4	CTRL f5	CTRL f6	CTRL f7	CTRL f8	CTRL f9	CONTROL SOFTWARE
	ANY VALUE	ANY VALUE	CTRL f▶	DISK DIRECTORY	MAIN/BOXED DISPLAY	END	WIPE	PRINT	LOAD	SAVE	
	7	6	5	4	3	2	1	0	INSERT LINE	DELETE LINE	

Lines © 1986 LEGO Group	SHIFT f0	SHIFT f1	◀SHIFT f	CTRL f3	CTRL f4	CTRL f5	CTRL f6	CTRL f7	CTRL f8	CTRL f9	CONTROL SOFTWARE
	ANY VALUE	ANY VALUE	CTRL f▶	DISK DIRECTORY	MAIN/BOXED DISPLAY	END	WIPE	PRINT	LOAD	SAVE	
	7	6	5	4	3	2	1	0	INSERT LINE	DELETE LINE	

Lines © 1986 LEGO Group	SHIFT f0	SHIFT f1	◀SHIFT f	CTRL f3	CTRL f4	CTRL f5	CTRL f6	CTRL f7	CTRL f8	CTRL f9	CONTROL SOFTWARE
	ANY VALUE	ANY VALUE	CTRL f▶	DISK DIRECTORY	MAIN/BOXED DISPLAY	END	WIPE	PRINT	LOAD	SAVE	
	7	6	5	4	3	2	1	0	INSERT LINE	DELETE LINE	

Lines © 1986 LEGO Group	SHIFT f0	SHIFT f1	◀SHIFT f	CTRL f3	CTRL f4	CTRL f5	CTRL f6	CTRL f7	CTRL f8	CTRL f9	CONTROL SOFTWARE
	ANY VALUE	ANY VALUE	CTRL f▶	DISK DIRECTORY	MAIN/BOXED DISPLAY	END	WIPE	PRINT	LOAD	SAVE	
	7	6	5	4	3	2	1	0	INSERT LINE	DELETE LINE	

BBC MASTER SERIES MICROCOMPUTER

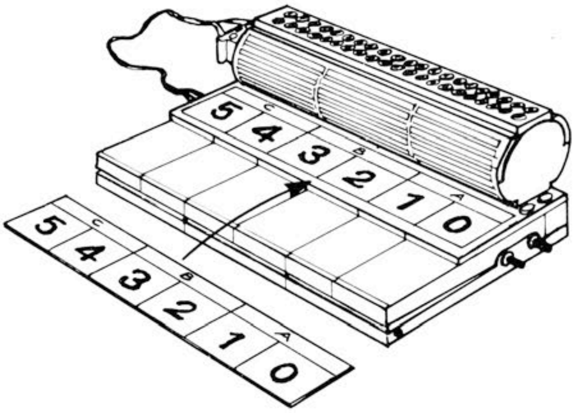
c		b		a	© 1986 LEGO Group
5	4	3	2	1	0

c		b		a	© 1986 LEGO Group
5	4	3	2	1	0

c		b		a	© 1986 LEGO Group
5	4	3	2	1	0

c		b		a	© 1986 LEGO Group
5	4	3	2	1	0

c		b		a	© 1986 LEGO Group
5	4	3	2	1	0



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

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