Teachers Guide to SEQ

Control technology in the classroom

Paul Spurgeon

Table of Contents

1. Preface & acknowledgements

1.1. Preface

Children are always making things; sometimes things that work, but always things to play with. Exploring the world by constructing models is a powerful way of learning about it, and fun. My aim in developing the SEQ controller and associated materials has been to give children more opportunities to enjoy the pleasures of building something and then making it perform specific tasks: to allow them to have more control of their creations.

The name SEQ is an abbreviation of 'sequence'. SEQ actions are both entered and obeyed sequentially, one after another. If you pronounce it 'seek', it implies exploration and searching for solutions: problem solving.

A sequence of instructions implies that each instruction is obeyed one after another, with the effect of each instruction depending on the previous ones. Sequential actions can be more easily understood than when several things are happening at once. Later versions of SEQ may allow multiple actions to be programmed so they happen in parallel.

I have tried to make the way that SEQ works as natural and consistent as possible. Features that seem to be obviously missing may have been omitted in order to preserve these aims. At times the temptation to add features has been hard to resist: why not have two sensor inputs, allow SEQ to be connected to a computer, allow structured repeat loops, use a remote controller? Mostly the design constraints have been those of keeping the keyboard input consistent: every key is either a command, in which case it has no parameters, or an instruction, in which case it takes a single numeric parameter. The absence of any form of display for the stored instructions makes this consistency essential. I was also conscious of an overriding need to keep the cost to a minimum, so that SEQ could be seen as an affordable pre-computer alternative.

1.2. Acknowledgements

Many people have helped in supporting and guiding my thinking during SEQ's short gestation period: special thanks goes to Henry Liebling, who encouraged me to trial SEQ with children from 5-11; to Alison Kelly and Alison Blockly of Stuart Road Junior & Infants School, for allowing me to work with them and their pupils in an open and free way; to Paul Richardson for technical advice and encouragement; to Jon Coupland and the rest of the staff at CITE/ITMA, for allowing me the opportunity to explore new areas of the curriculum; to John Hunt for enthusiasm and ideas about using SEQ in a Primary Science context, to Roger Jones of Commotion and Steve Rogers of MESU for useful criticisms during pre-production trials; and lastly to my wife for considerable patience while I was engrossed in the project.

The biggest problem I had while writing the guide was in restraining myself from playing with SEQ and Technical LEGO to explore ideas. I hope that SEQ and this guide help others to discover the fascination of designing, making and controlling things.

Paul R. Spurgeon
August 1988

Revised & reformatted from original draft (but with obsolete Resources section removed) August 2020

ISBN 1-871641-00-4

2. How to use this guide

2.1. What to read

This guide gives some ideas about how to use SEQ in the classroom. It assumes you have access to a SEQ controller and appropriate construction materials or kits. It does not tell you how SEQ works: look at the **SEQ Manual** to find out how the various commands and instructions are used, and how to connect up various devices.

It is not intended to be read from beginning to end. It could be used for self-study, or as a basis for in-service, or a source book for ideas and materials. Hopefully it will be used in all these and other ways. You do not need to read it in order to use SEQ, or before you let your children use SEQ. But reading some sections should help you get started. If you are new to SEQ, electrical devices and making things, take it slowly: try one of the **Starters**.

Because SEQ can be used with a wide age and ability range, you may not find classroom materials that are directly suitable for your needs. Instead you will need to adapt the activities described to your own specific situation.

If you are planning to use SEQ for the first time with a group of children you will find it useful to read **Introducing SEQ**. You may find it useful to look at (and perhaps try out) some of the activities: **Starters**, **Builders**, **Projects**, **Techniques**, and **Devices**.

Once you and the children have some idea of what SEQ can do you will want to plan their learning experiences so that they develop the skills that you are interested in supporting. **SEQ in the Curriculum** and **Observing and Recording Progress** should enable you to design appropriate teaching materials and learning activities.

Before starting, glance at **Hints & Tips** so you are prepared for likely problems. Sources of equipment and classroom materials are given in the **Bibliography**. When you come across a word you don't understand, or suspect is being used in a special way, look in the **Glossary** for a plain language description.

When using SEQ you may wish to refer to **Appendix 1: SEQ Specification** for a technical description, or **Appendix 2: SEQ Features** for an outline of important characteristics. If you are familiar with Bigtrak, you may find **Appendix 3: Differences between SEQ and Bigtrak** useful, especially if you wish to use existing Bigtrak teaching materials. Don't look at **Appendix 4: SEQ & Program Structure** unless you are familiar with at least one programming language: it compares the SEQ control with other languages. **Appendix 5: Teaching Material Masters** contains originals that you can use to make domino and key cards, record sheets, posters, and other classroom materials.

3. Introducing SEQ

3.1. Aims & Objectives

3.1.1. Why SEQ?

The short answer is that SEQ provides an opportunity for children to solve problems in a concrete way, extending their knowledge of the world by developing the skills to break problems down into small, sequential steps, which they can test and repeat. SEQ provides a bridge between the concrete activity of making things, and the abstract activity of controlling things with a computer; a step that children often find hard. The way in which SEQ supports abstraction through concrete activity is described below.

A turtle is a 'device to think with' (see Seymour Papert's classic book 'Mindstorms'); SEQ is a general purpose computer controller, and gives children the opportunity to make their own devices to think with. Used with appropriate construction kits, SEQ allows children to invent, construct and control their own inventions and 'microworlds'.

SEQ can be used by children from five years old upwards. The objectives for any specific group of children will obviously need to take into account their age, ability and previous experience. SEQ fits into a progression of activities, from making and controlling a simple model to computer control of a sophisticated system.

3.1.2. Designing & Making

Making things is about exploring how aspects of the world work. Many ideas are encountered:

Strength and suitability of materials:
Is this strong enough? Is it too heavy?
Should it bend? Will it break?

Suitability of different structures:
Is it rigid? Does it move properly?
Will it fall over? Is it the right shape?

Aesthetic and pragmatic design decisions:
Does it look right? Does it do the job?
Is it ugly or beautiful? Am I pleased with it?

Problem solving skills:
How can I make this work? What parts do I need?
What ideas can I use? What's gone wrong?
How can I make this fit?

3.1.3. Mechanisms

Initially models are static; all movement is by play and imagination. Working models are more challenging, but introduce ideas of mechanism, change, structure & function. Wheels, joints, levers, gears, motors, hinges, chains and other devices are explored. Motive power is needed: usually this will be human, mechanical (elastic, clockwork, etc.) or electrical (battery powered motor). More ideas are met:

Cause and effect:

That fell off because it wasn't strong enough.
The motor stopped because the wire broke.

Mechanisms:

This chain drives the back wheels.
If you turn this the wheels change direction.

Friction:

I can't turn this because it's too tight.
The wheels haven't got enough grip.

Power, speed & energy:

It goes fast, but it can't climb slopes.
The batteries are almost flat: it's slowing down.

3.1.4. Control

Working models lead inevitably to a desire for control. This may be simply starting and stopping the model, using a switch or lever. Later it may be desired to modify a models' behaviour, by steering it or making it carry out a sequence of actions. Control is usually manual. More concepts are encountered:

Sequencing:

You need to go forwards, then you can turn left.
Lift the brick before you move the crane.

Current and desired state:

To get over there you'll need to move around the table.
It's not ready to turn yet.

Errors, accidents & feedback:

It's gone too far. The wheels fallen off.
Look out, it's near the end! Be careful, it's nearly there.

Controllability:

Can you turn it a small bit? The wheels are slipping.
I can't steer it.

3.1.5. Computer Control

Computer control sets out to extend these model building and exploratory activities by giving children the power to control their models with computers. Computer control allows sequences of actions to be repeated over and over again automatically. These sequences can be modified to correct, improve, or alter the actions carried out by the model, allowing for the first time the possibility of examining cause and effect in detail. In addition to clarifying concepts met in earlier work with models, children will encounter such ideas as:

Programming:

What instructions do I need to make it do that?

Debugging:

Why don't my instructions have the effect I intended?

Problem solving:

What does the computer need to do to make it work?

Hardware (models) & software interaction:

I'll turn the motor on for longer now I've changed the gears.
It moves forward until the switch hits something.

Children find the step from manual to computer control difficult. This may in part be due to the complexities of the control languages or programs that they are expected to use. Manual control may be a necessary preliminary activity, and this is supported by a wide range of available materials (e.g. LEGO Technic Manual Control kit 1039).

Experience with floor turtles and Bigtrak has shown the advantages of preliminary, concrete activities before starting to learn a programming language such as Logo. SEQ provides a bridge between manual control of models and computer control. Actions are entered and may be tested individually. The model can then be instructed to carry out these actions one after the other. If the sequence is wrong, actions can be deleted and re-entered. In this way children can solve control problems by entering sequences of actions without needing to know a programming language.

3.1.6. SEQ & Control

Initially children will enter a simple sequence of actions, and maybe repeat them with GO. Later they may need to use repeat (x2), or jump (JMP and JSW), to put loops into their programs. Ideas that will be met during these pre-computer activities include:

A stored program as a sequences of instructions:

When I press GO it does these instructions, then stops.

Commands, instructions and parameters:

Clear memory!
Forward: how much?

Output devices (e.g. motors, lights, buzzers):

The motor plugs in here, and the lights in here.
You make the light pulse by pressing this key.

Input sensors (e.g. switches, lights) :

You can plug a switch in here to make it stop automatically.

Writing (entering, teaching) programs:

Tell it to go forward, then turn, then forward again.

Running (executing, doing) programs:

Try it out: press GO.

Debugging: testing and correcting programs:

It went too far. Lets try a smaller number.

Modifying a models behaviour by changing programs:

We could make it go further by using larger numbers.

Flow of control (x2, JMP, JSW):

Using JMP 1 makes it go on doing it forever.

Conditional actions (JSW):

If you press this switch it will do something else.

Testing & acting on input from the real world:
It changes direction if the switch on the bumper hits a wall.

Feedback & motor control:
Using feedback from the wheels makes it more accurate.

Directional, pulsed and continuous power:
If it goes backwards instead of forwards, swap the leads around.
What's the difference between the Pulse and Out outputs?

Estimating & measuring; calibration:
How far does it go when you do forwards 1?
How much should I go to reach that wall?

3.2. Classroom Organization

3.2.1. Planning

How you organize your classroom will depend on several factors. It would be unrealistic (and unnecessary) to expect a class set of SEQ's and construction kits or materials, so it is likely that you will need to plan SEQ's use as one of a number of activities. Exactly how this works will depend on class size, children's abilities, the curriculum, and resources. Look at "SEQ in the curriculum" for some ideas about where it can contribute to existing work. It is hoped that SEQ will be used to extend and enhance the curriculum, rather than simply be added to it.

If you plan to use SEQ as part of a designing, making, and problem solving activity, then you may wish to organize things so that there is a progression. Some groups can be constructing models; others testing them using batteries and switches (the LEGO Switch Controller 1039 is excellent); others using SEQ with their models to develop control programs; and possibly culminating with one or more groups using a computer to control their models.

You must plan to allow plenty of time for children to explore, experiment and discover control and construction solutions. This may mean that their work needs to continue over several lessons: if this is so you will need to think carefully about the resource implications. If they construct complicated models, will they need to be dismantled so that other groups can use the kit, and re-made at the start of the following lesson? If so, how will they record what they have achieved so far? Consider photographs as a useful way of recording progress, plan for longer periods of work, or set tasks that should be completed within the allotted time.

The children will probably get most out of using SEQ if they work in small groups, three at most. This allows them to discuss ideas, collaborate and make decisions. They'll need a reasonably large space to work in too: models move, some of them quickly! A large, smooth floor area will be necessary for vehicles: much safer than having them fall off tables. Try to avoid corridors where people may accidentally tread on or trip over models.

Make sure the children know what they are going to do, why they are doing it, how long they have to do it in, where they can get help, and generally have a sense of purpose about their work. You may need to duplicate activities or workcards, record sheets, domino or key cards. Check the list of things that are required on the activity sheets.

3.2.2. Organization

Organize resources so that the children can see and obtain them easily, and know where they belong and what they are called (labelling storage helps). Make sure they know any safety rules you feel are appropriate (see "Hints & tips: Safety"). Try to ensure that there are sufficient leads, and plenty of output and input devices (motors, spare light bulbs, switches, sensors). If the children need to use tools there will be additional considerations.

Check the leads and plugs regularly. Remember to charge the batteries too: children (and you) will not want to lose time because the batteries are flat. Look at "Hints & tips: Batteries" for information about batteries. Regular maintenance could be carried out by children, yourself, a technician or other helper if you are lucky enough to have one. If you're sharing resources with other teachers you'll need to coordinate work so that clashes are avoided.

You will want to record the children's progress. Plan how this is to be done before starting, and make sure you have copies of all the record sheets or checklists required. Look at "Recording & assessment" for ideas and techniques. You will want to display work in progress and completed projects, and should plan for both wall and surface (shelf, table, floor) space where models and plans can be seen without getting damaged. Looking at mechanisms, designs and control programs can be a valuable learning activity and a source of inspiration.

Plan lessons so that there is time at the end not only for packing away (finding all the bits missing from kits), but, more importantly, an opportunity for children to record their work (if appropriate), reflect on what they have achieved, discuss problems, and share successes, failures and discoveries.

3.3. Getting Started

3.3.1. Basic Ideas

What's the minimum you need to know to start using SEQ? You could connect something (a buggy, go-cart, crane) to SEQ and just let them explore what the keys do. But quite early on it would be a good idea to draw their attention to the following:

Output sockets: they are all red. There is an important relationship between which sockets you connect things to, and which instructions make them work. The motor sockets (above the Forward and Left keys) use the Forward, Backward, Left or Right instructions. The Pulse and Out sockets use the keys directly below them.

Instructions and commands: Instruction keys are green; commands are red, and they are different.

Output instructions: Forward, Backward, Left, Right, Pulse, Out are the instructions to use when you want to make something happen (they may not need all of these to start with).

Programming and obeying the program: SEQ is either waiting for you to do something (give it a command or instruction), or obeying the instructions you have taught it. You can't give it commands (except STOP) or instructions while it's obeying instructions. GO makes SEQ start obeying instructions; STOP will stop it.

Memory: SEQ remembers all the instructions you give it, one after the other. Unless you press CM (or CE) new instructions will be added to SEQ's memory after all the old ones. Memory can get full. CM clears all memory, so you can start again.

Instructions: they all need a number. What do the numbers mean? (Best if they find this out). Output instruction numbers mean how much, how long, or how many.

Test and CE: Test makes SEQ do the last instruction in memory, CE gets rid of it from memory. Press CE if you make a mistake entering the numbers, and start again (including the instruction).

3.3.2. Key Sequences

Use the domino cards early on. You (or the children) could make and use a 'Breakthrough to literacy' sentence builder, to hold the domino or key cards in sequence. Watch (see "Observation") the children using SEQ. You can expect to notice unusual or unnecessary key press sequences. For example:

CM
GO
Forward
Forward 5
GO

You may feel that children should be guided to the most economical use of the keyboard, and want to correct them:

"You don't need/have to press GO before you give instructions"
"You only need to press Forwards once"

This isn't very important, so long as they don't become frustrated. For example:

CM
GO
Forward
Forward
GO

"It doesn't work!"

Small sample programs for them to use, modify and make their own are also useful. For example:

1 Forward 4
2 Backward 2
3 Jump 1

1 Pulse 5
2 Jump 1

1 Out 8
2 Jump 1

3.3.3. Techniques

There are many alternative ways of introducing SEQ:

Build something, using a kit or other materials
Use a complete model
Give a quick demo to a group

Work with a small group
Whole class introduction using OHP &/or charts
Give out lists of instructions
Circus: switches, SEQ, & computer control activities
Sow seeds: share information between groups

Working with small groups could be "You tell me what you want it to do & I'll program it" or "I'll help you when you get stuck". A fruitful technique is to put domino cards down as the group develop and change a program, making the instructions concrete & visible for discussion. Then they can be left to use the cards to plan and modify what they want to do.

Sharing information between groups or individuals so they get to know and use different instructions encourages talking, explanation and cooperative learning: they can use each other as resources, for example "Can you tell me how to make the light flash?"

SEQ is useful in testing models as they are being constructed, as well as when they're complete. Switches can also be used for manual control, or the battery connected directly to motors and lights.

Later on you may wish to talk about how to develop and test instruction sequences (see "Developing, testing & debugging"). Typically children will teach SEQ using an instruction, Test (and optional CE) cycle.

If you want them to start constructing models, you could provide them with some useful sub-assemblies. For example, give infants a motor complete with gear box, so they can put wheels directly on it and make different vehicles.

Another excellent way into constructing and controlling things is to use some of the kit manufacturer's instructions, and let the children build up some of the things they describe. Not only does this enable them to become familiar with some of the mechanisms and construction techniques that are needed with kits, but it also produces a working model that can be used to explore control sequences. The LEGO leaflets show how to build lots of different things, some of which can lead into (and be used with) the introductory activities.

3.3.4. Connections

If they are connecting their own models up, it would be a good idea to give a few guidelines for using leads:

Connect up one lead at a time. One end goes into SEQ's output sockets (red ones), the other end into a light or motor.

As each lead is connected, you could test it by giving SEQ one instruction to try.

If it doesn't work, several things could be wrong:

You used the wrong instruction
One of the plugs has dropped out
You forgot to press CM

Direction & polarity: if it goes in the 'wrong' direction, turn one of the plugs around.

Try to avoid a 'rats nest' of wires. They lead to wrong connections (and a possible short circuit of the battery, see "Hints: Safety"), and to a poor 'try it and see' approach to connecting things. Children should be able to explain which socket on SEQ is connected to what, and be able to follow the lead making the connection.

3.4. Using the Activities

3.4.1. Teaching Materials

At least three teaching approaches seem possible:

A structured introduction to each facility in SEQ

Making a model and using it with SEQ, learning about SEQ's features as the need arises

Identifying a problem, then developing and testing a solution

The latter two seem preferable. They allow SEQ to be used with a purpose, without introducing learning activities outside of the context provided by specific applications. When conceptual difficulties arise, or children are observed to lack a specific technique, it may be appropriate to use more skill based activities. Provided in the next five chapters are materials that can be used to support a variety of teaching and learning styles:

Starters provide structured control, construction and extension activities, giving a practical introduction to various control concepts.

Builders present more advanced control, construction and extension activities, developing concepts and techniques such as feedback, conditional control, and design decisions.

Projects give starting points for projects, with a design brief, ideas and extensions. A wide range of control and constructional skills and techniques can be used to design, develop and test solutions.

Techniques are specific control activities looking at designing and making control sequences, leading to an understanding of programming. Children may benefit from using some of these activities when they need to understand a specific control concept, such as repetition.

Devices are control and construction activities that look at attaching and using different input and output devices. They should prove useful when children wish to use an unfamiliar input or output device, or you want to extend the range of devices they have experienced.

3.4.2. Progression

The activities are not intended to provide tightly structured work for groups to pursue, but are to be taken as starting points to modify and adapt. Teachers and children are encouraged to identify their own problems that they wish to tackle, and to find alternative solutions to those described.

Depending on the age and ability of the children, you may wish to photocopy all or parts of the activities, using them unedited; or to rewrite them at a more suitable reading level, with more (or less) direction. You could photocopy and cut up each activity, making them into small workcards. Most activities are written around a theme or concept, and contain several tasks: these could be made into separate task cards.

The activities are not arranged in any particular order within each chapter. It is not necessary or desirable for them to be used one after the other. See "Progression" for information about planning routes through the materials.

Starter and Builder activities can be used to learn about control, using the control activities with pre-built models. Children can then go on to modify the models, perhaps exploring some of the construction activities. Alternatively, they can build the models first, using the construction activities, and then go on to

the control activities. Which approach you use will depend on their age and ability: construction is usually harder than control!

Notice that all the activities list what you need (excluding tools, wires, batteries, and so on), followed by any necessary skills or things children must know about before they can start the activity. Then there is a brief list of what they can expect to learn.

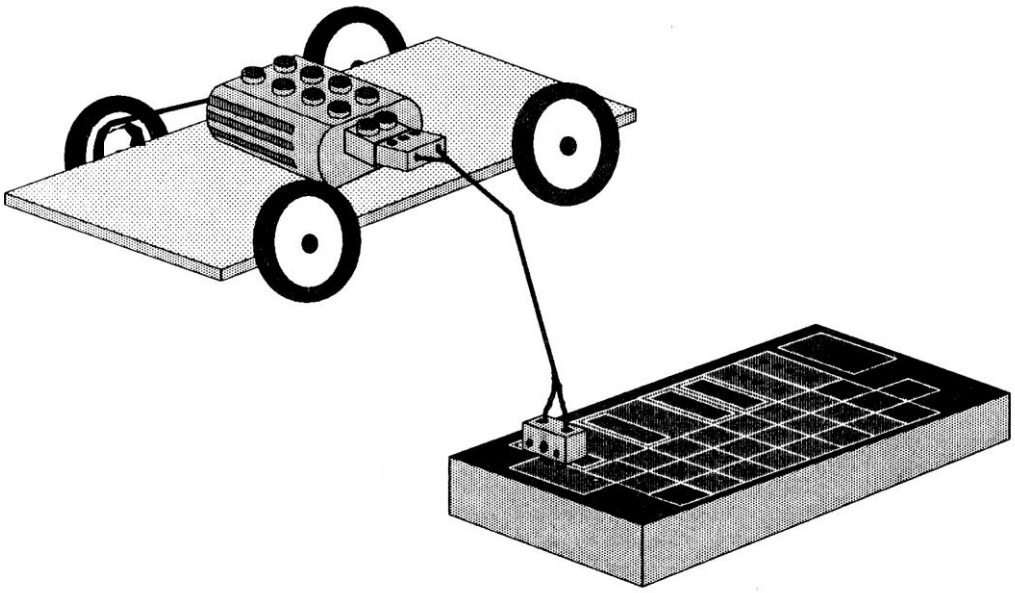
4. Starters

4.1. Go-carts: simple 1 motor vehicles

4.1.1. Introduction

A go-cart type vehicle with a single motor cannot be steered automatically, but is simple to use and understand. Children will have to aim the vehicle before pressing GO. The motor drives the go-cart forwards and backwards, and can be connected to the wheels by gears, a chain, or pulleys and an elastic band.

4.1.2. Example



4.1.3. Control activities

Needs: go-cart, SEQ
Learn about: estimating & measuring, using distance parameters.

How far does the go-cart move when you tell it to go Forward 1? How far does it go when you do Forward 10? Draw these distances on a piece of card, or use string.

Put the go-cart on the floor, and make a mark so you can remember where it starts from. Estimate how much you will need to tell the go-cart to go across the room. Check your estimate by using the card or string you made. Try out your estimate.

Get a friend to place an obstacle some distance away from the go-cart. You must try to make the go-cart move forward and stop just before it touches the obstacle. Take turns: the nearest to the obstacle wins.

Try making the go-cart move across the room, stop in front of an obstacle, and then returning back again to the start.

4.1.4. Construction activities

Needs: kit/materials to build go-cart, SEQ, stop clock
Learn about: gears & wheels, speed, power & acceleration, measuring, timing, recording, comparison & fair testing.

Make a go-cart. How fast does it go? Can you make it faster? Slower? Try different combinations of gears, pulleys or wheels. Do larger wheels make it slower?

Have a competition for the slowest/fastest go-cart design, then have a race. Time how long it takes your go-cart to cross the room.

Test your go-cart over obstacles. What is the steepest slope it will go up? Can you improve it's obstacle or slope climbing?

Does your go-cart go straight? If not, how can you make it travel in a straight line?

4.1.5. Extension activities

Add a light or beeper to the go-cart, and make it go up to someone and give them a surprise!

Develop an interesting program to make it look as though it's alive (so a cat might play with it). Try using the Jump instruction so it keeps on moving (JMP 1 is simplest).

Make a bulldozer from your go-cart. It should be able to push something (bean bag, ball) across the floor. What is the largest or heaviest thing it can push? How can you make it more powerful?

4.2. Buggies: simple 2 motor vehicles

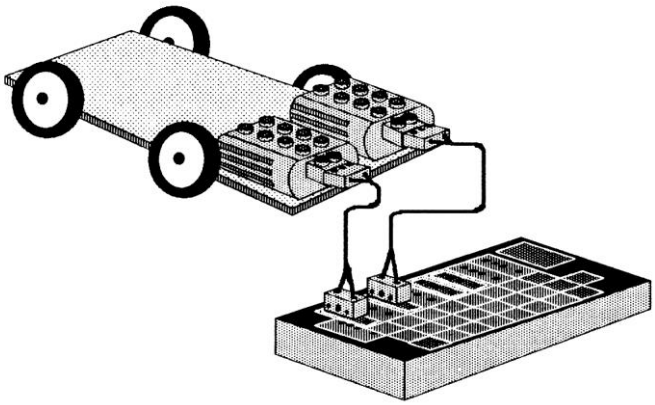
4.2.1. Introduction

Use two motors, one to drive each wheel. When both wheels turn forward, the buggy goes forwards; when both wheels turn backward, the buggy goes backwards.

Steering is achieved by making the wheels turn in opposite directions.

When connected to SEQ, the 'Upward' arrow may make the buggy turn: turn one of the motor plugs around. Or the upward arrow may make the buggy go backward: call the front the back, or reverse both leads. The 'Left' arrow may make the buggy turn right: swap or reverse both leads.

4.2.2. Example



4.2.3. Control activities

Needs: buggy & SEQ
Learn about: Sequence, planning & recording, estimation & measuring, current & desired state, prediction.

All Bigtrak and simple turtle activities. For example:

negotiate a maze
build your own maze, or try someone else's
can you program SEQ without trying the maze first?
play turtle football
involves designing and building a ball pusher
teach the buggy to draw a shape
square, circle, polygon, diamond
make the buggy go and collect something automatically (an "Egg race")
make a (sad or happy) buggy dance
repeat a sequence of movements
include pauses

Calibration:
How far is Forward 1, Forward 10, Forward 50?
How much is Left 1, Left 10, Left 50, a complete turn?

4.2.4. Construction activities

Needs: 2 motors, kit or materials to build buggy, SEQ
Must know about: using kit or materials to build structures
Learn about: gears, effect of wheel diameter

Faster and slower: make the fastest or slowest buggy. Use different gears or wheels.

Power: how steep a slope can it go up? Can this be improved: gears, wheel size & type.

Mobility: what sorts of rough surfaces & obstacles can it go over? How can you improve this? Try changing the wheel size, gears, ground clearance, chassis.

Accuracy: does it go straight? Turn accurately? How can this be improved?

4.2.5. Extension activities

Make the buggy into:

a load carrying lorry (how much will it carry?)
a bulldozer (push a ball through a goal)
a police car, fire engine, ambulance (add lights)
something aesthetically pleasing (disguise it)
an all-terrain vehicle, using tracks instead of wheels
an animal or insect, with moving parts (like a pull-along toy, see "Oscillation & repetition"):
duck (moving head), butterfly (flapping wings), dog (wagging tail)

Build a buggy obstacle course

Build a place where the buggy can have a purpose:

an urban environment (make it go from my house to yours, or to school)
a country park, with houses, a bridge, a farm, animals, faces (Postman Pat)

Add a bumper to the buggy: use feedback to control it's behaviour (see "Obstacle avoiding").

Add feedback from each motor or wheel using sensors, to make it more accurate (see "Accurate buggies").

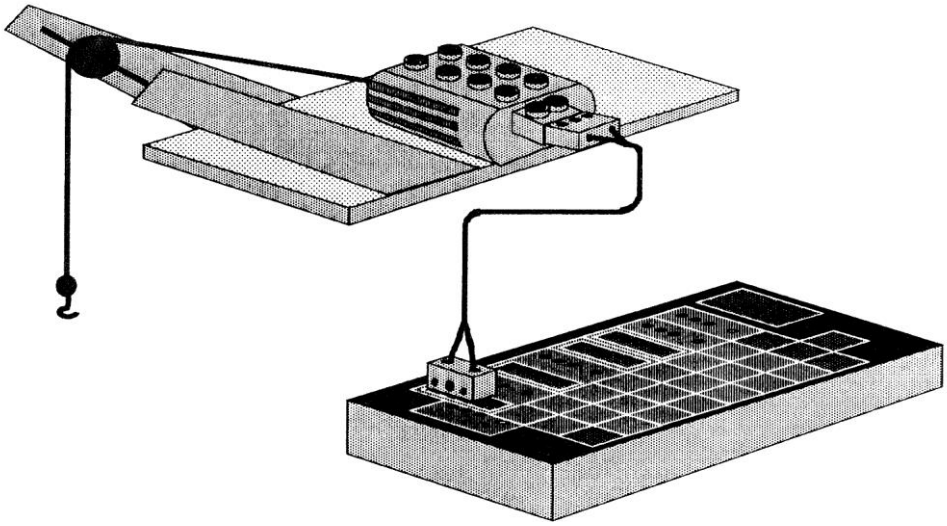
Add a pen to draw the buggy's path as it moves (see "Turtles").

4.3. Cranes

4.3.1. Introduction

A single motor can be used to raise and lower a hook on a crane. Turning the motor one way will wind the string up, turning the other way will lower it.

4.3.2. Example



4.3.3. Control activities

Needs: crane connected to SEQ
Learn about: sequence, estimating & measuring, current & desired state, calibration, prediction, planning & recording

Put the crane on a table. Lower the hook to the floor. How far down is it? How far can you raise a load?

Put some obstacles and platforms on the floor. Can you move a load from one platform to another?

Make up some tasks for the crane driver. Write programs to do the tasks. Give them to someone else to try. Are their programs the same as yours?

4.3.4. Construction activities

Needs: 1 motor, kit or materials to build crane, SEQ
Must know about: using kit or materials to build structures
Learn about: strength of materials, stability, gears & power, pulleys

Make a crane that you can use to lift things. Look at some pictures of cranes to get ideas.

How much can your crane lift? Can you make it lift a bigger load?

If the string breaks you will need to investigate the strength of different strings & threads.

If the crane topples over you will need to find ways of making it more stable.

If the motor stalls or slips, and can't lift the load you need more power. You could either change the gears, or use pulleys. Investigate different pulley arrangements and different gears. Does producing more power mean that the crane will lift the load more slowly?

4.3.5. Extension activities

Make a crane that can move along. It could use a single motor base (see "Go-carts"). Use independent motor control mode (see "Independent motor control") to move the crane along, lift a load, move it, then lower it again.

Make an automatic hook or grabber, so the crane will pick up a load when it is lowered, then lift it, and release it when it touches the ground again. Look at latches, toy cranes, arcade pick-up games, and real cranes for ideas.

Make a revolving platform so the crane can move loads around. You could drive this from another motor (it will need to be geared down a lot - why?), and use independent motor control mode (see "Independent motor control").

Add limit switches so that it's impossible to crash the hook into the crane jib. Attach them to the Stop input, or use the Switch input so you can program SEQ to take appropriate action (maybe sound an alarm).

Make a winch to raise and lower a load by winding the string onto an axle. Use feedback to count pulses from the winch shaft (see "Motor feedback"). Write a control sequence to raise the load, pause, then lower it again. Attach a sensor (see "Switch feedback") to stop or reverse the winch when the load has been lifted to the top. Can the winch be used as a lift? Use a switch as a lift request button.

Make a jib that you can raise and lower. You could use gears or pulleys, bu you will need quite a lot of power (why?). Use independent motor control mode to program the jib and lift loads. How does the position of the jib affect stability? What use is a jib?

Look at different sorts of cranes. Why are they different? What are their advantages and disadvantages? Cranes have many similarities with robot arms (see "Robot arms"), why?

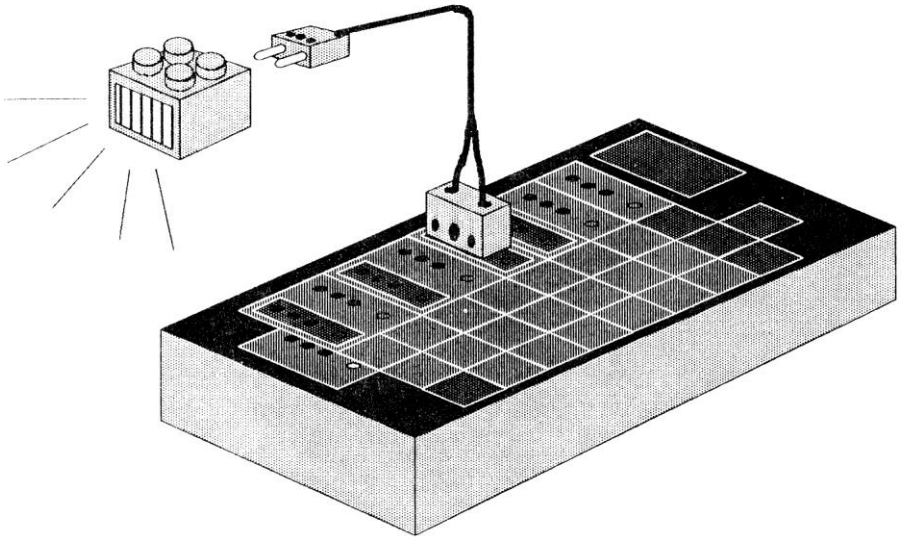
4.4. Flashing Lights

4.4.1. Introduction

A lighthouse, morse code flashers, Christmas tree lights, flashing beacons and pelican crossings can all be built using flashing lights, and messages sent using SEQ. A single light can be used, connected to the 'Pulse', 'Out', or motor outputs, for simple messages. Or several lights can be used, connected together or to different output sockets.

A motor can drive a platform with a light on it, but the wires will get twisted. An alternative is to use a motor to make a stationary light appear to flash by obscuring it, although the light is on continually.

4.4.2. Example



4.4.3. Control activities

Needs: single light, SEQ
Learn about: codes, counting, timing, observation (watching & listening), communication, recording.

One child programs a morse code message, the other reads it. How fast can you read a message? How can you slow it down? Use the 'Pause' instruction to add a delay between messages.

Find out about lighthouse signals, then make a lighthouse that gives a signal like one (for example the Fastnet Rock or Eddystone light). Can you change the program to make it like another one?

Can you flash the lights for a Christmas tree in time to a tune? Try jingle bells! What other tunes can you program?

4.4.4. Construction activities

Needs: lights, switches, batteries, construction kit or other materials, SEQ
Learn about: electric wiring, control & mechanical alternatives, resistance & loose connections, gears

Add extra lights. How many lights can SEQ work? How long can the leads be?

Make a turntable to make a light rotate slowly. How can you stop the wires getting twisted?

Make a screen with slits that can rotate around a light.

4.4.5. Extension activities

Make an automatic SOS flashing beacon. How will you make it start? Maybe you press 'GO', or you could attach an automatic switch to the 'Switch' input, so when the beacon is removed from it's holder it automatically goes off (the switch could be normally closed).

Make a morse code reader, that will recognize a simple code (see "Code recognizers").

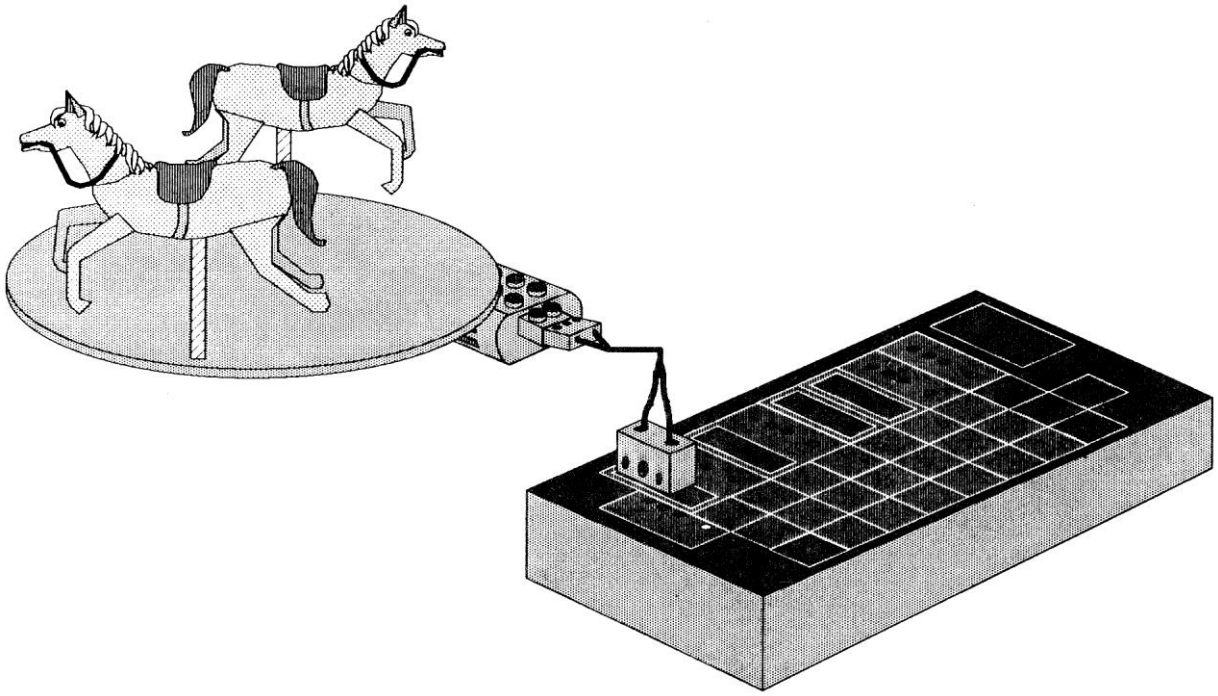
Make a lens holder to rotate around a light, like a real lighthouse. You may need several lenses: experiment in a dark place!

4.5. Fairground rides

4.5.1. Introduction

Fairground rides provide plenty of scope for imagination, innovation and exploration of a variety of mechanisms and control sequences. A single motor can be used to drive the ride. Ideas can come from a visit to the fair, or from playground apparatus.

4.5.2. Example



4.5.3. Control activities

Needs: carousel, merry-go-round or ferris wheel, SEQ
Learn about: sequence, planning, repetition, switch inputs

How long should a ride on the carousel be? How long should you allow between rides? Plan an interesting sequence for the carousel, and then program SEQ to carry these out automatically when you press GO.

You could add lights and buzzers to the carousel to make it more interesting. Or play some music while it is going around.

Use the Jump 1 instruction to make the ride repeat forever.

Add a switch which starts the ride automatically when a coin hits it (someone pays for the ride). Use the Jump on Switch instruction to wait until the switch is closed or open (for example JSW 1).

Add an emergency stop switch in case anyone or anything falls off.

4.5.4. Construction activities

Needs: kit/materials to construct rides, 1 motor, SEQ.
Learn about: gears, pulleys

Make a merry-go-round ride that goes slowly, using gears, or pulleys with an elastic band or chain drive. Add horses or cars to carry model people. What happens if it goes too fast?

Can you make a ride with seats that rotate as the ride rotates? They could be attached by gears or elastic bands to the centre, or perhaps rotate freely with off-centre weights.

Decorate your ride so that it looks attractive and realistic.

4.5.5. Extension activities

Make a merry-go-round so that the horses go up and down as it goes round.

Can you make a coin operated switch that works when the correct money is inserted, and ignores wrong coins? Use SEQ to let the customer know when the right money has been inserted. Could you make a SEQ program that requires two 10p coins before starting?

Visit a fair, or look at some pictures of fairground rides. Which ones could you make? How many motors are there on each ride, and how do the motors drive the ride around?

Look at some playground apparatus: swings, slides, see-saws. Which of these could be adapted to give an automatic ride if a motor was attached? Design a ride based on one of your ideas, then make a working model. Would it be exciting? Would it be safe?

4.6. Robots

4.6.1. Introduction

Robots are fascinating to build, control and watch, but difficult to perfect. However, it is possible to make simple robots that look like robot toys, and then to program them to behave in a more 'human' way. These activities focus on human biology, behaviour, and aesthetic aspects of making things.

Make a simple robot with arms, legs and a head, some or all of which move when the motor is run. Look at some of the construction kit robots or books (Richard Pawson's is recommended) for ideas.

4.6.2. Control activities

Needs: completed robot & SEQ
Learn about: empathy, movement & expression, sequence & repetition

Make the robot move in a realistic way. Watch someone walking or playing, and notice how they move. Try to make your robot walk, wave its arms, shake its head. Can you make it look comical or sad by changing the program? Get someone else to look at the robot's performance, and see if they laugh or feel sad.

Use the repeat and jump instructions to make the robot carry on performing. You could add lights for eyes; maybe you could make it wink?

4.6.3. Construction activities

Needs: materials/kit to make robot, 1 or 2 motors, SEQ
Must know about: using the materials or kit
Learn about: structure & function, aesthetics & decoration

Make an interesting robot. Thing about what it will be like. Draw some designs before you start. Maybe you could make it look fierce, strong, friendly, like some one you know, comical, like a monster, human, animal, or plant?

Think about how quickly or slowly your robot should move. If it is a graceful dancing robot, it should move slowly. If it's a fighting robot, it might have to move quickly. Use gears to alter the robot's speed.

4.6.4. Extension activities

How could you make the robot move? Can you make it walk, or put it on wheels, or even in a car.

Use motors to drive strings attached to a puppet. Animate the puppet to do a simple activity or sequence of activities: waving goodbye, dancing, or exercising.

Try making part of a robot instead of the whole thing. Make a robot hand, robot mouth, robot eye, or even a robot head. Look closely at a human hand, mouth, eye or head, and observe how it moves. Then design and build a model, and write a SEQ program to make it move realistically.

Look at parts of other animals: crab arms, elephant trunks, insect wings. Can you make working models of these?

See "Robot arms", "Walking machines", and "Animation" for more robotic activities and ideas.

4.7. Oscillation & repetition

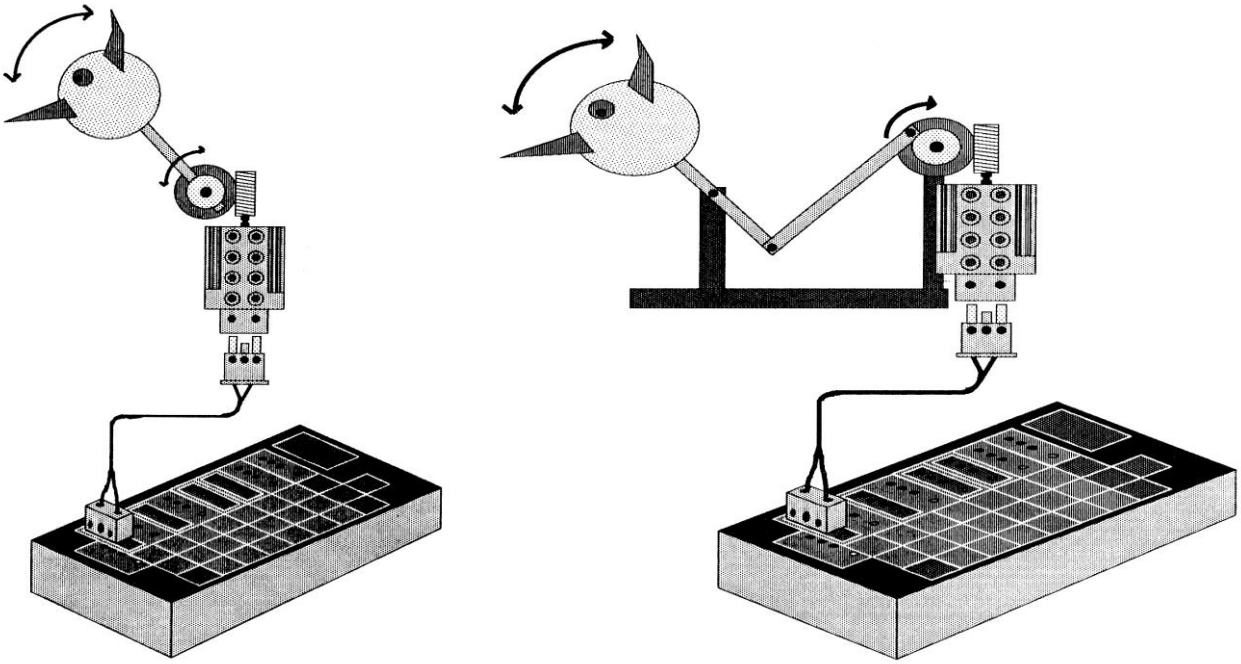
4.7.1. Introduction

Toys and models that oscillate and repeat motions can be made in two alternative ways. Traditionally they are constructed with cams and levers arranged to provide oscillation from a continually rotating drive. Using a programmable controller such as SEQ allows this mechanical complexity to be replaced by simpler mechanisms, and the program designed to produce the oscillations.

For example, a nodding head on a toy animal can be driven from a cam on the wheels. Alternatively a motor could drive the head forwards and backwards by SEQ moving a motor alternately forwards and backwards, perhaps with pauses between nods.

The design decision between these two alternatives is always present, and these activities provide some experience of the advantages and disadvantages of each approach. Apart from different constructional merits, the programmed alternative allows much more flexibility and variety of action and expression, but (at least with SEQ) you can't do more than one thing at a time).

4.7.2. Examples



4.7.3. Control activities

Needs: nodding head and flapping wing mechanisms, SEQ
Learn about: sequence, planning & recording, observation & empathy

Make the head nod forwards. How far can it go? Now make it nod backwards. Write a sequence of instructions that will make the head nod, pause, then do it again. Hint: use Jump 1 to do everything again.

Change the program so the head nods twice, then flash the eyes (use some lights). Can you make it nod wisely, or look sad (perhaps drop the head slowly, a bit at a time, then look up quickly)?

Make the wings flap. Can you write a program that makes them flap vigorously, or one which flaps them lazily?

4.7.4. Construction activities

Needs: materials/kit to build mechanism, 1 or 2 motors, SEQ, materials for decoration
Must know about: using gears, making structures with the materials or kit
Learn about: reciprocation, oscillation, levers & linkages

Make a model animal (dog, cat) with a head that nods. You could make it so that it moves along and nods its head at the same time; or so that it can move or nod its head, but not at the same time.

Make a model insect or bird that flaps its wings. Look at the "Levers and ducks?" case study in Chapter 9 of "Design & Technology 5-12" for some ideas.

4.7.5. Extension activities

Can you combine head nodding with wing flapping?

Make a model head that nods or shakes its head. Then program it to say "yes" or "no" by head nodding or shaking. You could use a switch input to SEQ to control whether it says yes or no, and attach the switch to an electric quiz game. The machine will tell you if you get the answer right or not!

Look at "Animation" for project ideas.

4.8. Doors & barriers

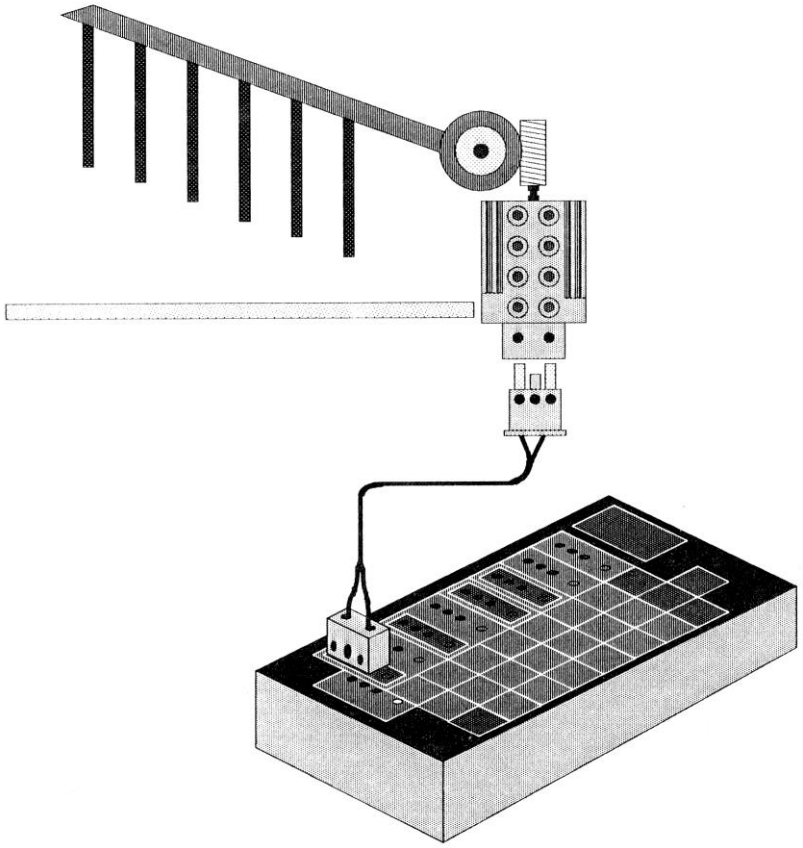
4.8.1. Introduction

There are many familiar examples of doors and barriers, some automatic (supermarkets, railway crossings); and a variety of mechanisms can be used. Automatic doors provide a useful introduction to linear motion, and to feedback in control.

Designs for constructing door and barriers can be found in many kits. Barriers usually have an arm that can be raised and lowered; doors often have a rack and pinion drive to slide the door open and closed.

Children frequently play with automatic doors, testing their response and trying to fool them. This could form the basis for an investigation into control mechanisms used in public places.

4.8.2. Example



4.8.3. Control activities

Needs: barrier or door model & SEQ, stop-clock
Learn about: estimating & measuring time

Make the door or barrier open and close. How long should it stay open? If it is used for a train, car or person, you will need to find out how long it takes to go through the door. Use a stop-clock, or estimate the time and try out your program.

Use the Pause instruction to make the door stay open for the required length of time, and then close automatically. Add a light and make it flash when the barrier is about to close (like a level crossing).

4.8.4. Construction activities

Needs: materials or kit to make door/barrier, 1 motor, SEQ
Learn about: mechanisms, gears & levers

Build a barrier like the ones used on railway level crossings. You may need to use some gears so the barrier opens and closes slowly, especially if the barrier is heavy.

Make a sliding door. Use a rack and pinion mechanism to drive the door open and closed. You could also try using levers.

4.8.5. Extension activities

Make an ordinary door that opens and closes. You could use gears or levers.

Add a switch (perhaps a pressure mat) so that the door opens automatically when someone or something comes up to it. Use the Switch input to SEQ, and write a program to wait until the switch is closed before opening the door, wait, close it, then start all over again (see "Automatic doors" and "Switch feedback").

Add a limit switch or switches so the barrier or door stops when it is fully open or closed. Think about the way in which you will instruct SEQ to operate using these switches before trying your ideas out.

5. Builders

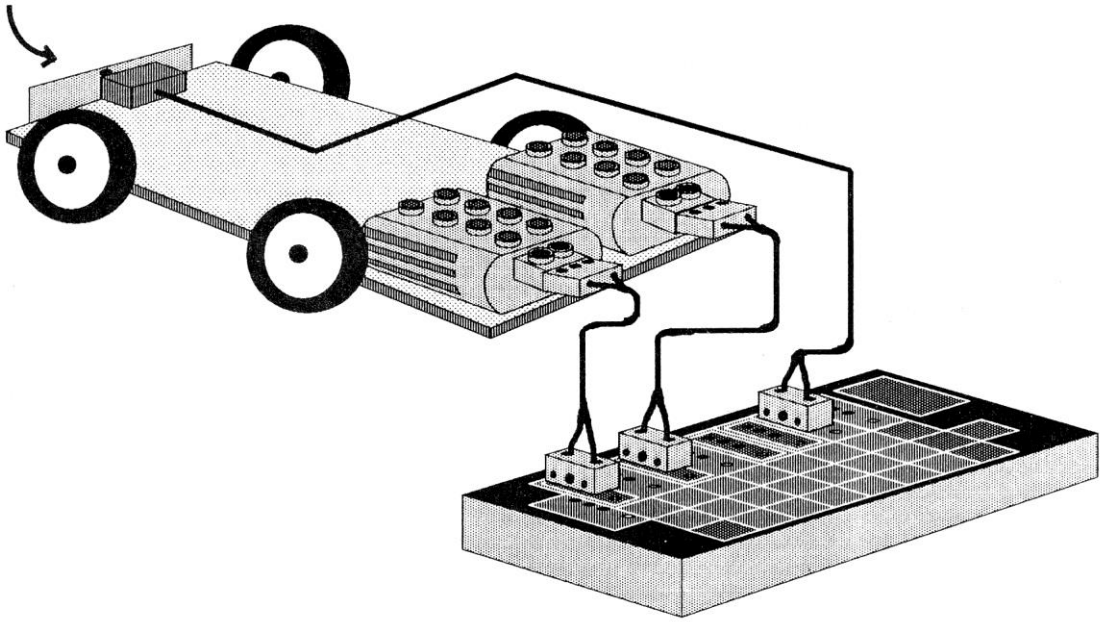
5.1. Obstacle avoiding - bumpers on buggies

5.1.1. Introduction

All the introductory activities use SEQ to control things in the real world without it having any idea of what is actually going on. When SEQ obeys the Forward instruction, the buggy could be stuck against a wall and it wouldn't be able to move. Adding a bumper that will operate a switch when the buggy hits something allows control programs to react to real world events, such as hitting a wall.

The bumper is arranged so that it presses a switch when it touches something. The switch is attached to SEQ's Switch input, and the JSW instruction can then be used to test whether the switch is pressed or not.

5.1.2. Example



5.1.3. Control activities

Needs: buggy with switch on bumper, SEQ (see "Switch feedback" for techniques)
Must know about: using a buggy, SEQ & JMP instruction (see "Repeating things")
Learn about: planning, strategy, flow of control

Make the buggy cross the room until it hits something, then sound an alarm to tell you its stuck.

Write a program so that the buggy will find its way around a wall placed in front of it. What should it do when the bumper hits the wall?

Write a program to make the buggy move continually around a room, avoiding obstacles, and not getting stuck. Think about what it should do when the bumper hits an object. What avoiding actions could it take? Which action is the most likely to help the buggy avoid being trapped in a corner?

5.1.4. Construction activities

Needs: buggy, SEQ, normally open push switch (see "Alternative switches and sensors" for details), LEGO touch sensor or similar
Must know about: making things
Learn about: switches & sensors

Put the switch on the front of the buggy. When the buggy bumps into something (a wall, chair leg, etc.) the switch should be pushed, completing the circuit. Attach the switch to SEQ's switch input socket (above the JSW key), and watch the light come on when the switch closes.

You may want to add something in front of the switch, on a hinge perhaps, so that there is a larger contact area.

How can you make the bumpers more sensitive? Or better at detecting objects before the buggy hits them?

Modify the bumper and switch arrangement so that you can drive the buggy around on a table without it falling off the edges. Be careful when testing it in case your design doesn't work: you may need to catch it quickly!

5.1.5. Extension activities

Make your own switch. Perhaps you could make feelers. Add an emergency stop switch in case the buggy tries to go under something that is too low.

Make a normally closed switch. Does this make it easier to program the buggy for the control problems above?

Add bumpers and switches at the side or back of the buggy and wire them all up together in parallel. Does this help the buggy to avoid being trapped in corners?

Use a light sensor instead of a switch (see "Switch feedback" & "Alternative switches and sensors"). Write a buggy program so that the shy buggy finds a dark corner and stays there.

5.2. Accurate buggies - using feedback

5.2.1. Introduction

You may have noticed that buggies do not always travel in straight lines when going forward: they will veer slowly to the right or left. Several things may cause this, and investigating sources of inaccuracy makes a worthwhile project.

Inaccuracy can be caused by more friction or stiffness in the motor drive on one side than the other. When the motors are turned on for the same length of time, the wheel on the stiffer drive will rotate by less than the other side, and the vehicle will veer off course.

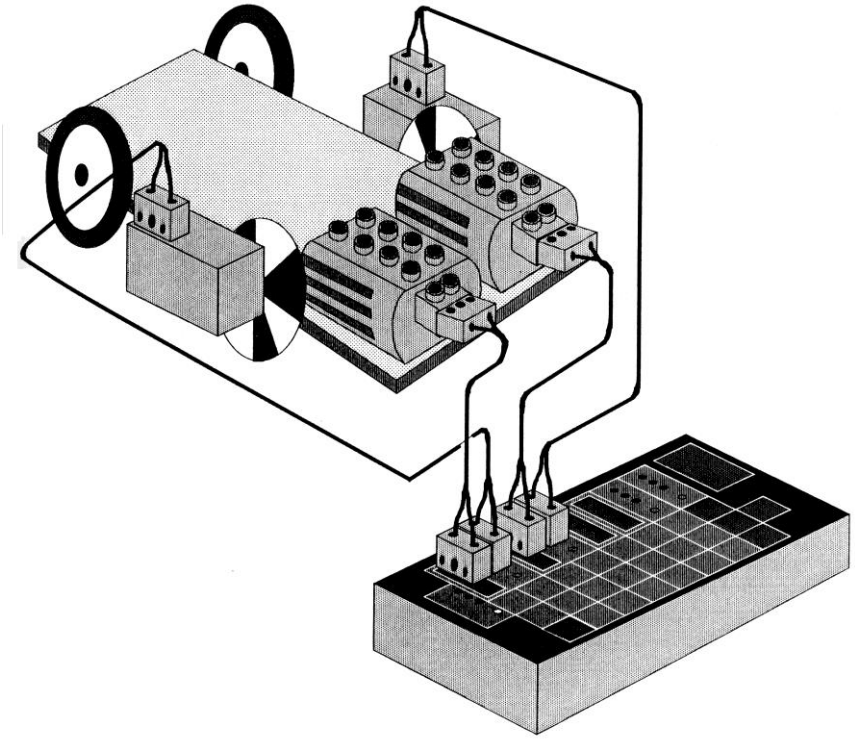
To compensate for this SEQ needs inputs from both wheels to tell how much they have rotated. Opto-sensors or other switches can be attached to the wheel shafts, and will provide on and off pulses as they rotate. SEQ counts these pulses (automatically) instead of turning the motors on for a fixed length of time. If sufficient pulses are counted from one wheel before the other, SEQ continues to drive the other motor until the wheel has rotated by the same amount.

For every unit of a motor instruction, SEQ turns on each motor for a fixed length of time. If a pulse is detected within this time, SEQ goes on to the next unit. Thus pulses must occur more quickly than the duration of each unit (approximately 0.1 seconds), and therefore buggies will travel a shorter distance (and turn by less) than without feedback.

If the buggy doesn't move far enough with the opto-sensors attached, you can use the STOP-number command to change the pulse count.

See "Motor feedback" for more details and activities.

5.2.2. Example



5.2.3. Control activities

Needs: buggy with opto-sensors, SEQ
Must know about: using of buggy & SEQ
Learn about: feedback, accuracy

Attach the buggy without the opto-sensors connected. Enter Forward 50 and press GO. Does the buggy go straight or turn slightly? Which way does it turn?

Attach the opto-sensors to SEQ, being careful to connect the sensor next to the motor that drives it! Press GO again. You should see the yellow input lights on SEQ flashing on and off as the opto-sensors rotate.

How far does the buggy go? Watch what happens as the buggy stops. You should notice that one motor continues for longer than the other, correcting any turning.

Press CM and enter this SEQ program:

```
Forward 10
Forward 10
Forward 10
Forward 10
Forward 10
```

Now press GO. Does the buggy move in a straight line now?

Try drawing a geometrical shape (square, triangle, circle). You should notice that the buggy turns by a different amount with the opto-sensors attached. Is the buggy more accurate with the opto-sensors connected or not?

5.2.4. Construction activities

Needs: buggy, opto-sensors, SEQ
Must know about: modifying mechanisms
Learn about: opto-sensors, backlash, rotation speed

The distance the buggy moves when using opto-sensors depends on where the sensors are attached. Try attaching them directly to the wheel shafts, or directly to the motors, or at any point in between. If you are using LEGO opto-sensors, you can also try using the other side of the shaft encoders (the small wheel with black and white stripes on it).

Which position gives best accuracy and control?

Opto-sensors are sensitive to light. Test if they will work in a bright light (sunlight, or near a lamp or torch). Make a screen to prevent bright lights spoiling the accuracy of your buggy.

5.2.5. Extension activities

Press and hold STOP, then press 2. You have changed SEQ's pulse count. Try the above activities again. Try pressing STOP-3, STOP-4 and so on. Which pulse count gives more accurate control, longest distances, or is the best compromise between accuracy and distance moved?

Make a list of things that could affect the buggy's accuracy. Which do you think are the most important? Find ways of testing the accuracy, and try to improve it. See "Turtles" for more ideas.

5.3. Robot arms

5.3.1. Introduction

Robots are involved in aspects of several of the other SEQ activities, for example: "Robots", "Oscillation & repetition", "Obstacle avoiding", "Turtles", and "Animation".

Making and controlling robot arms could form a complete project. Robot arms can also be combined with other projects, for example loading and unloading a vehicle, or picking things up from a conveyor belt.

The jointed arm is perhaps the most familiar robot arm design, but others are possible, and have their own advantages and disadvantages. The mechanism used at the end of the arm to pick up and hold things (the 'end effector') can also be designed in many ways. Robot arms can be fixed, or able to move around. Locomotion may be by using tracks or wheels (perhaps like some of the buggy designs); various leg and foot designs can also be used.

Feedback provides information about the position of the robot's joints, and allows the robot to sense the objects being manipulated.

There are lots of designs for robot arms that can be readily built. The number of motors available (and that SEQ can control) limits the number of joints the robot arm can have. The weight of various parts of the arm and the maximum load to be manipulated will determine the amount of power needed to move the arm. Transmitting this power down the arm gives plenty of scope for investigating different mechanisms.

Robot arms using two or more motors will need to be programmed in independent motor control mode (see "Independent motor control"), and provide a good introduction to the use of this mode.

The LEGO 1090 kit provides two motors and can be used to make several different robot arms. "The Robot Book" by Richard Pawson gives numerous designs for interesting robots using LEGO and Fischertechnik, and is also a useful source book for ideas and mechanisms.

In many respects the range of control and construction activities that can be carried out using robot arms is tremendous, and you should be encouraged to set your own problems and find ways of solving them.

Adding a switch or sensor to the grabber provides a good introduction to the need for and use of feedback in control. See "Switch feedback" for more information.

5.3.2. Control activities

Needs: 2 motor robot arm (for example LEGO 1090-E), SEQ
Must know about: using SEQ & loops
Learn about: sensors, switch feedback, loops & sequences

Write a program to make your robot arm pick up an object and drop it into a box. You will have to mark where the object should be placed.

Write a program using a loop (see "Repeating things"), so the robot arm gives a demonstration of all the things it can do, over and over again. You could use this in an exhibition.

Attach a sensor to the robot arm so it can tell when there is something to pick up. Then write a program so the arm moves a bit at a time until it finds an object, then picks it up and moves it. Can you adapt this program so the arm will find all the objects it can reach, and put them all into a box (a tidying-up robot)?

5.3.3. Construction activities

Needs: kit/materials to build robot, at least 2 motors, SEQ
Must know about: making and using mechanisms
Learn about: power & power transmission, friction, balance & levers, strength & weight.

Make a robot arm that can pick up a small object (building block, wooden brick). Decide how far you robot should be able to reach, and where the motors will be.

What is the heaviest thing your robot arm can lift? What is the largest and smallest thing your robot arm can grasp? Is the shape of the thing you are lifting important?

How could you improve the gripper or hand? Can you make it lift a table-tennis ball, and egg, or a pencil?

Add a switch to the gripper so you can tell when it has closed. Adapt the gripper so the switch only closes when it has actually picked something up. Now rewrite one of the control programs so the robot searches for things to pick up by grasping them (a 'snapper' robot).

5.3.4. Extension activities

Use a robot arm with some other models. You could pick things up from a conveyor belt, then place them on a lorry (or the other way around). You could team up with another group, and let their sensor tell your robot arm when to start. This could form the basis of an 'egg-race'.

It's easy to write a program for a robot arm to knock everything over. Can you devise a way of picking things up and putting them upright again? You may need to decide on what sort of things you are going to pick up: model people, skittles (a bowling alley robot), bricks?

Mount your robot arm on a buggy or other vehicle. You could team up with another group: they design the vehicle, you make the robot arm. But you'll need a lot of materials, at least four motors (two for the arm, and two for the buggy), and probably two SEQ's.

Design and build a fork lift truck. Use sensors so you can write a program to pick up a palette, then move it.

If you have made your own complete robot, you may like to add limit switches to the joints, so it is impossible for someone to make it move further than you intended. The limit switches can all be connected in parallel to the STOP input on SEQ.

5.4. Turtles

5.4.1. Introduction

Turtles are accurate vehicles (see "Accurate buggies") with a pen or pencil that can be raised and lowered. They can be programmed to draw by leaving a trail as they move. Turtles can usually be given simple instructions, a minimum set could be:

Forward a distance
Backward a distance
Turn left an angle
Turn right an angle
Pen up
Pen down

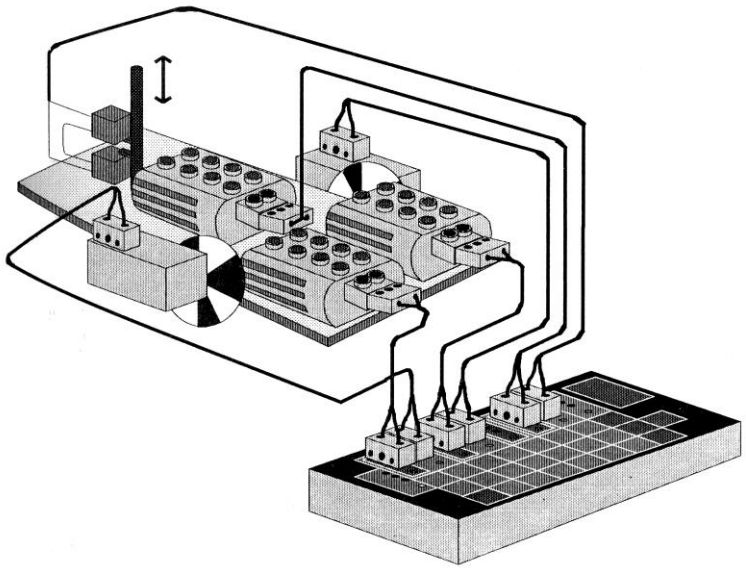
There are a large number of turtle activities available elsewhere. But they are not necessary: children and adults should be encouraged to set their own goals, and work toward achieving them.

Making a turtle, while not impossible, is not easy. However, programming one is, because the result of the program can be seen, and the programmer can 'walk through' the program if it doesn't work, by 'playing turtle'. See Christopher Schenk's excellent book, "Hands on, hands off" for lots of ideas about turtles, including games, pre-turtle and turtle activities. Many of the Bigtrak activities can also be used.

Once an accurate, steerable chassis has been developed (see "Accurate buggies" the only construction problem is to make a pen holder. This could be fixed, so a trail is left continually, or manual, so the pen could be raised and lowered by hand (perhaps during a Pause instruction, in response to a beep from SEQ). A more difficult and satisfactory solution would be to design a pen holding mechanism that allows the pen to be raised and lowered automatically, by SEQ instructions.

As both motor outputs are used for the buggy, only the Pulse and Out outputs are available for the pen mechanism. Unfortunately, these are not bidirectional, and don't have direct feedback, so some ingenuity is needed. One idea is to make a third motor drive the pen mechanism, and arrange a cam (or other rotary to linear conversion mechanism) to raise and lower the pen as the motor rotates in one direction. By adding one or two switches that are normally closed, but open when the pen is fully up or down, it is possible to use a simple loop for both pen up and down instructions.

5.4.2. Example



5.4.3. Control activities

Needs: turtle, SEQ, large sheets of paper, pens, domino or key cards, record sheets
Learn about: sequence, planning & recording, estimation & measuring, repetition, prediction, debugging

All Bigtrak, turtle and "Buggies" activities can be used. For example:

Draw a shape (rectangle, triangle, boat, diamond, house, star, circle.....)

Draw a dotted line, staircase, spaceship, garden, ...

Write a program to draw a shape. Now try to draw the same shape twice as big, three times as big, and half as big. Write down each program. What are the similarities & differences?

Experiment with simple repetition. Try this program:

```
Forward 5
Left 25
Jump 1
```

What shape does it draw? What effect does changing the amount of turn have? What effect does changing the amount of Forward distance have?

5.4.4. Construction activities

Needs: materials/kit to make turtle, 3 motors, 2 normally closed switches, SEQ
Must know about: making accurate vehicles (see "Accurate buggies", single switch feedback (see "Switch feedback" and "Obstacle avoiding"), rotary to linear mechanisms (see "Oscillation & repetition")
Learn about: switches & feedback, series & parallel

Make a mechanism that raises and lowers a pen holder as the motor rotates. You cannot use a rack and pinion, as the motor will rotate in the same direction all the time (plug it into the OUT socket on SEQ). Make sure the motor is geared down, so that you need at least OUT 5 to make the pen go from top to bottom.

Now add one of the normally closed switches to your mechanism so that it opens when the pen is fully up, and the other one so that it opens when the pen is fully down.

Connect the motor to the OUT socket. Connect the switches in series, one after the other. Try the following program:

```
OUT 1
JSW  1
```

Press GO. Is the pen up or down? Press GO again. Is it up or down now? You may need to adjust the pen and the switch positions. Explain how this program works.

Try some of the turtle control activities to test your pen mechanism. You should be able to adapt the program so you can raise and lower the pen with two instructions whenever you want to.

Try rewiring the switches in parallel. What happens now? Why?

Change your design so you can use normally open switches (in parallel). You may need to select inverse switch control mode using STOP-JSW (see "Switch feedback" for more details).

5.4.5. Extension activities

Try putting the pen in different places on the buggy. How does this alter the shapes that your programs draw? Where should the pen be positioned so that the turtle draws the 'best' geometric shapes?

All the usual turtle activities can be tried, but they will present a harder challenge than using Logo (or Dart) on a computer. For example, can you draw a spiral?

If you have a control interface to a computer, you could try writing programs to draw the same shapes, using Logo and using SEQ. How do they compare? What sort of things are hard to draw using SEQ, but easy in Logo? Why?

5.5. Code recognizers

5.5.1. Introduction

There are lots of ways that codes can be produced: see "Flashing lights" for some activities. But how can you build a machine that will recognize a code (say "SOS"), and ignore others?

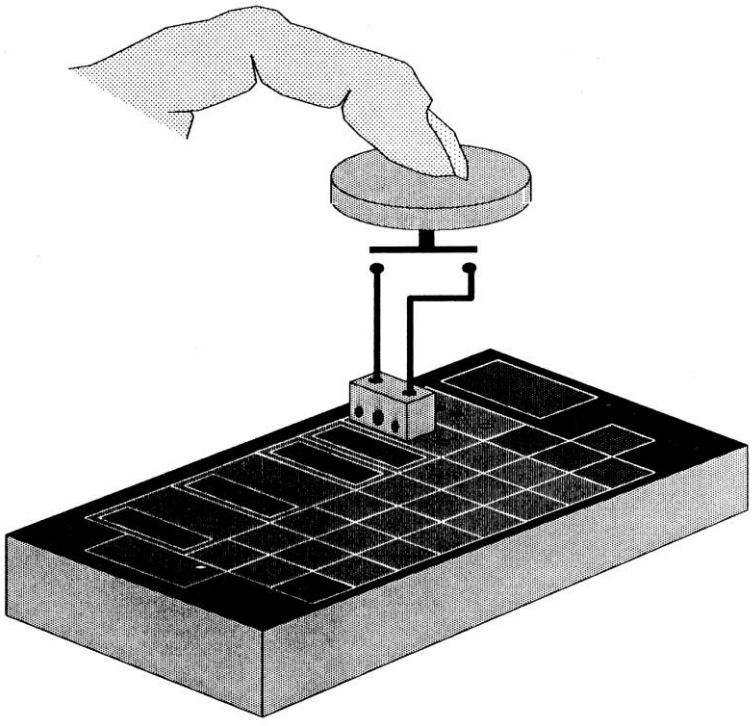
The switch input on SEQ can be used with a sensor to test an outside condition. For example, an opto-sensor could detect the flashes of a light. More simply, you can connect a morse code key (or even a switch) to SEQ and send it morse code messages directly. By writing control programs that expect a certain sequence and duration of on and off pulses a code recognizing machine can be devised.

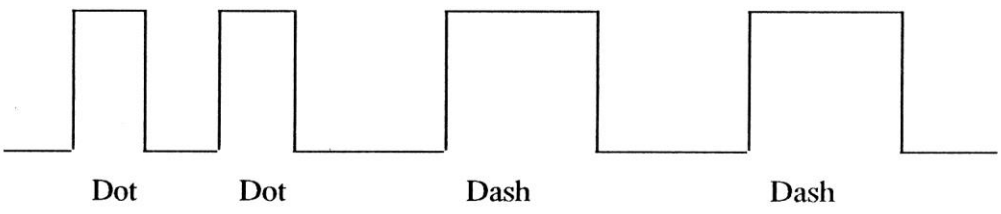
Warning: writing these programs is not too hard, but understanding them and finding out why they don't work is very demanding! If you can write these programs then you can consider yourself an expert SEQ programmer and logician.

You cannot program SEQ to count as it goes around a loop, so it is not possible to make a code recognizer that will discriminate between very long and very short switch input pulses. However, you can write small programs that will reject very short switch pulses, and accept all those over a minimum duration, or to ignore several short pulses.

The use of timing diagrams may be found helpful. It would also be worthwhile measuring the time SEQ takes to obey various instructions. Ten units is very approximately one second.

5.5.2. Example





5.5.3. Control activities

Needs: switch or opto-switch, SEQ
Must know about: using switches, JMP & JSW instructions
Learn about: control structures, timing

Connect the switch to SEQ's Switch input, and try this program:

```
1    JSW 3    jump if the switch is closed
2    JMP 1    go back and look at the switch again
3    Pulse 1
4    JMP 1    do all this forever
```

Press GO. What happens when you press the switch, or shine a light on the opto-sensor? Hint: if you are using the LEGO opto-sensor you should select inverse switch mode (STOP-JSW, see "Switch feedback").

How many beeps do you get each time the switch is pressed? What does this depend on?

This program detects when the switch is closed, but it doesn't matter how long it's closed for. Long switch closures cause lots of beeps.

Press STOP, CM, and try this program:

```
1    JSW 3    jump if the switch is closed
2    JMP 1    go back and look at the switch again
3    JSW 3    wait for the switch to open
4    Pulse 1
5    JMP 1    forever
```

How many beeps do you get for each switch closure now? How short or long must the pulse be to make SEQ give a beep?

Experiment with this program, altering the pause delays:

```
1    JSW 3    jump if the switch is closed
2    JMP 1    go back and look at the switch again
3    Pause 5      wait a bit
4    JSW 4    wait for the switch to open
5    Pulse 1
6    Pause 5
7    JMP 1    forever
```

Try to detect two pulses: you could write two lots of instructions (then the pulse lengths can be different), or use a x2 repeat loop. Use JMP 1 (or JSW 1) if your program detects an invalid code, and start again.

5.5.4. Construction activities

Needs: kit/materials, 1 motor, opto-sensor, SEQ
Must know about: making things, using input devices
Learn about: sensors, repeatability

Design a machine that scans bar-codes automatically, and can tell the difference between them. You could move the bar code and keep the sensor stationary, or the other way around.

Make a number of bar code cards. Try various widths of black and white stripes.

5.5.5. Extension activities

Combine several sequences to make a program that makes a beep after receiving 2 or 3 pulses. Can you extend this to make an SOS detector?

Adapt the bar code recognizer to make a sorting machine (see "Sorting and classifying").

Use your code recognizer as an electronic lock, so that it opens a door (see "Automatic doors").

5.6. Sorting and classifying

5.6.1. Introduction

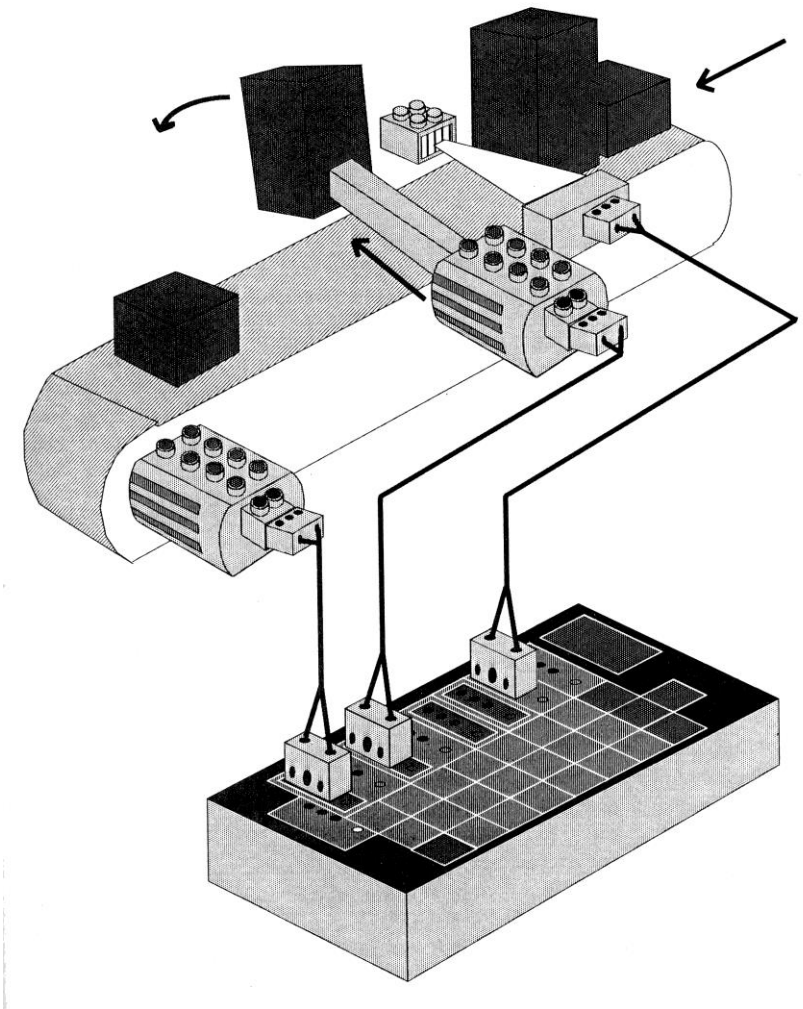
A typical sorting task uses a conveyor belt to move a variety of objects past a sensor. A mechanism uses the input from the sensor to select or reject the objects, perhaps putting them into different containers.

This kind of task makes a good 'egg race' activity for groups, perhaps in competition. The objects to be sorted should be well defined (for example, black & white table-tennis balls), and a range of mechanical, electrical and computer equipment provided.

Objects can be sorted by any physical property that can be detected using sensors, for example: size, shape, colour, whether they conduct electricity or are magnetic. See "Introduction to input devices" and "Alternative switches and sensors" for details.

Sorting can be carried out by mechanical or sensor controlled means, or by a combination of both. For example, sorting by size could be entirely mechanical, or a light sensor could be used to see how large the object was. Deciding which approach to use is a valuable lesson in designing and making: it is sometimes easier to modify a control solution than make mechanical changes.

5.6.2. Example



5.6.3. Control activities

Needs: completed conveyor belt from construction activity below, 2 opto or other sensors, SEQ, large & small wooden bricks (or alternatives)
Must know about: using JSW & JMP, independent motor control mode, making things
Learn about: decision making, timing, feedback

Make an 'arm' that moves in and out (look at "Oscillation & repetition"), so it can knock things off the conveyor belt.

Write a program to move the conveyor belt along. You will need to control each motor independently (press STOP-Left). Check that large bricks are detected by the sensor, and that small ones are not. Adjust the sensor if necessary.

Adapt your program so that when the sensor sees a large brick it rejects it by moving the arm and knocking the brick into a box. The small bricks can continue until they reach the end of the belt, and drop into another box.

Change your program so that the small bricks are rejected by the arm.

5.6.4. Construction activities

Needs: materials/kit to make conveyor, 1 motor
Must know about: making things
Learn about: mechanisms for decision making, modifying mechanical designs

Make a conveyor belt that can move three different sizes of brick (wood or plastic blocks perhaps). Add to your design so that large bricks are mechanically rejected by being knocked into a box at the side of the conveyor, but the small ones carry on to the end and drop into another box.

Can you modify your design so that small bricks are rejected and large ones accepted?

5.6.5. Extension activities

Use the conveyor belt and arm to sort objects with other properties. Try both mechanical and control methods of sorting them. Some properties you could investigate include:

light and dark
wide and narrow
heavy and light
upside-down or the right way up
insulators and conductors
magnetic or non-magnetic

Modify the sensors so that medium sized bricks are accepted or rejected, but small and large ones are not. (Hint: look at different types of switches).

Devise an automatic feed mechanism for the conveyor belt, so you don't have to keep putting objects on by hand. You may find it easier to feed round objects than bricks (but you'll have to adapt the conveyor belt so they don't fall off).

You don't have to use conveyor belts for sorting. Invent a mobile classifying vehicle. It could move along looking at things, and knocking them over if it doesn't like them! (see "Detectors & followers").

5.7. Steering

5.7.1. Introduction

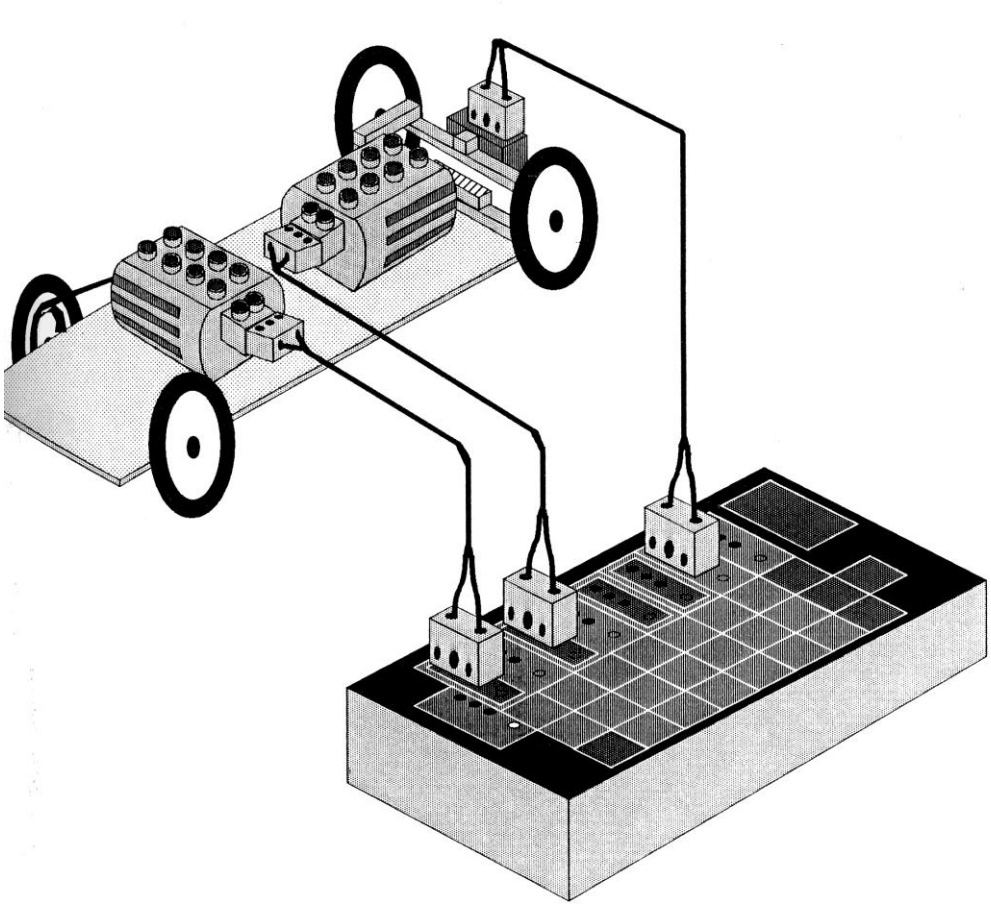
All the steerable vehicles in the other activities use two motors, one driving each wheel, to provide both forward and backward motion and to turn the vehicle by rotating the wheels in opposite directions. This is the simplest steering mechanism to make and operate, but it is not commonly used in the real world (why not?).

Steering is more familiar as a means of turning some of the wheels on a vehicle, so that when it moves it turns in the direction the wheels are pointing. One of the problems with automatic control of steering is making the vehicle go straight. While this could be solved mechanically, it is easier and more interesting to explore feedback as a way of controlling steering.

Two solutions to this problem are to use a switch to detect when the wheels are pointing straight ahead, or to use a sensor to count pulses as the steering is moved, and to move back by the same amount.

A variety of steering mechanisms can be used: perhaps the best known are rack and pinion (as in some cars); swinging beam (as used on soap-box carts); and single wheel (as used on on bicycles, tricycles and three wheeled cars.

5.7.2. Example



5.7.3. Control activities

Needs: vehicle with motor steering & switch, SEQ, record sheets
Must know about: JMP & JSW, independent motor control
Learn about: sensors, feedback & control, sub-procedures

Connect up the vehicle and select separate motor control. Turn the steering wheels to the left. Write a program to turn the wheels back to straight ahead. Make a note of it for later.

Now turn the wheels to the right. Write another program to turn them straight. Make a note of this too.

Now write a program to make the vehicle draw a shape (square, triangle, boat, house). Each time you turn the vehicle you'll need to straighten the wheels up again. Look at the two programs you have written: you can use them whenever you need to turn. Be careful though: any JSW or JMP instructions will have to jump to a different place in the program, so their numbers will be different.

Is it easier to program this vehicle or a buggy? Why?

5.7.4. Construction activities

Needs: materials/kit to make vehicle, 2 motors, 1 switch, 2 opto- or other sensors, SEQ
Must know about: feedback, constructing mechanisms
Learn about: accuracy & testing, evaluating designs

Make a vehicle with a mechanism that allows a motor to steer the front wheels. The second motor should drive the rear wheels, and should have an opto-sensor for feedback and accuracy (see "Accurate buggies - using feedback"). Arrange the switch so that it closes when the wheels are pointing straight ahead, but is open at all other times.

Test your design with this program (the values may need altering to suit your motors & gears). Don't forget to press Stop-Left first (see "Independent motor control")

```
1       Forward 10
2       Left 5
3       Right 1
4       JSW 6
5       JMP 3
6       Backward 10
```

Try some other programs: you can use any of the Bigtrak, buggy or turtle activities.

Remove the switch, and put an opto-sensor on the steering motor mechanism. Test with this program:

```
1       Forward 10
2       Left 5
3       Right 5
4       Backward 10
```

You may need to alter the pulse count per unit, depending on where you place the opto-sensor (see "Motor feedback"). Again, you could try any of the other activities to test the accuracy of your design.

Attach a pen or pencil and watch the shape you program draws. Why is it different to the shape a turtle or buggy draws?

Which approach, switch or opto-sensor, is the most satisfactory? Why? Does it depend on which steering mechanism you use? Try some different ideas and see.

5.7.5. Extension activities

Steering by turning some of the wheels means that the driven wheels must turn at different speeds, or slip. A differential is used to allow this to happen. Make a differential drive and see if it improves the accuracy of your design.

Find some other steering mechanisms to make. Add sensors so they can be controlled accurately.

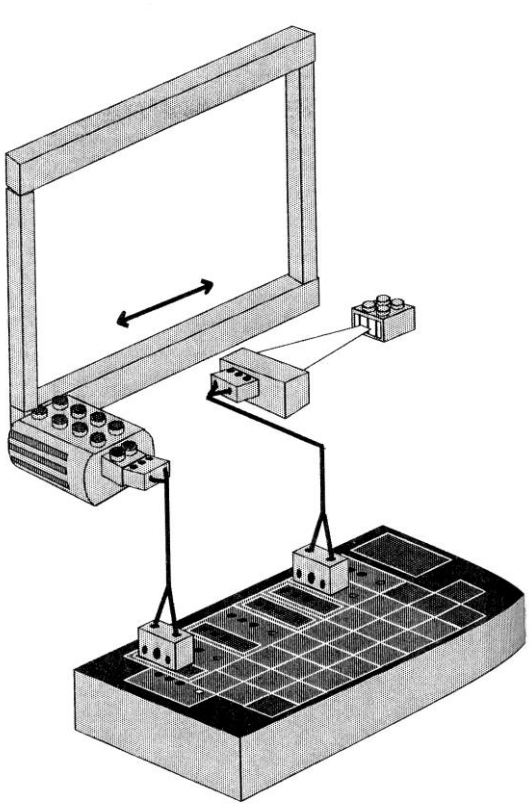
Use your best steering design to make an accurate alternative turtle, complete with pen raising and lower mechanism (see "Turtles"). Try out someone else's turtle programs on your vehicle. How does the meaning of Forward, Backward, Left & Right change?

5.8. Automatic doors

5.8.1. Introduction

Automatic doors provide a straightforward introduction to feedback and control mechanisms. They are familiar, the design aims are usually clear, they only need a single motor and sensor, and they can be easily tested.

5.8.2. Example



5.8.3. Control activities

Needs: motorized door (see "Doors & barriers"), 1 sensor, SEQ
Must know about: using JMP & Pause
Learn about: feedback & control, loops, timing

Connect up the door motor to the socket above the Forward key, and the sensor to the Switch input. Try this program:

```
1    JSW 3
2    JMP 1
3    Forward 10
```

You may need to turn the connection to the motor around if the door doesn't open when you work the sensor.

Now write a program so your door opens automatically when something (a train, a model, your hand) gets near it.

How long should the door stay open? Make it close after this time, provided there's nothing in the way.

5.8.4. Construction activities

Needs: materials/kit to make door, 1 motor, switches & sensors, SEQ
Must know about: making things
Learn about: sensors & sensitivity

Make an automatic door or barrier (see "Doors & barriers"). Decide on what sort of things should be able to open the door. Then experiment with different sensors to make the door open automatically. You could try to detect:

something hitting the door or standing near it
an object that blocks out the light
something carrying a magnet (like a cat-flap)

Use this program to test your door sensor:

1	JSW 3	jump if the switch is closed
2	JMP 1	otherwise go back & test it again
3	Forward 5	open the door
4	Backward 5	then close it
5	JMP 1	do this program forever

Is it possible for something to get past you sensor and hit the door? How can you improve your design?

Add a second sensor so you can use the door both ways.

5.8.5. Extension activities

Use single motor control and put an opto-sensor on the door motor mechanism, so you can control it more accurately.

Make a door that only opens if you know the 'key'. You could use a physical key, or use a code (see "Code recognizers").

Add an emergency open switch so that if the door closes on something (or someone) it opens again automatically.

Modify your door opening program so that it doesn't open the door when someone 'plays' with it by jumping in and out. Perhaps it should make a noise (sound an alarm) instead.

6. Projects

6.1. Animation

6.1.1. Design brief

Make a realistic model of a human, animal, insect or fantasy living creature. It should respond to external stimuli, such as noises, movement, or changes in light, in a realistic way. When there is no external stimulus it should still behave as if alive.

6.1.2. Ideas

Which parts will move? You will only have a few motor outputs, but they could drive more than one part at a time: for example, turning the head and raising the arm. Can you devise mechanisms that produce random, unpredictable actions? You will need an underlying mechanism, and suitable outer material to make it look realistic but allow movement. Look at puppets, fantasy creatures in films, dolls and children's toys for ideas. Response to stimuli could be positive (attending: moving or looking at the source) or negative (cringing, moving or looking away). Look at animals or humans for characteristic behaviour. The control program will need to be fairly complicated to produce movement that doesn't look to mechanical and repetitious. Develop small fragments of program first, then join them together.

6.1.3. Extension activities

Add a voice: use a cassette recorder and switch the remote control input on and off in response to stimuli. Make a 'random' switch input device that is sometimes closed and sometimes open. Use it to control the creature so it behaves unpredictably.

6.2. Clocks and counters

6.2.1. Design brief

Make a machine that keeps accurate time, and displays it in a readable way. It can show seconds and minutes, minutes and hours, or all three.

or

Make a machine that counts marbles, and displays the total.

6.2.2. Ideas

There are two aspects to these projects: the counting mechanism and the display. The counting could be mechanical, or a program in SEQ, or a combination of both. Displays could be like a traditional clock (with hands), an indicating wheel (rotating wheel with numbers), digital (look at some mechanical digital clocks). What is the most original display you can design?

6.2.3. Extension activities

Put an alarm on the clock or counter, so that when a pre-set time (or count) is reached an alarm rings.

Make the clock chime. The "chimes" could be mechanical, or programmed in SEQ, or a combination. Adapt the counter to make an electronic dice. How will you test to see if it is really random? Modify the counter so that you can reset it. Adapt the clock to make a stop-watch.

6.3. Walking machines

6.3.1. Design brief

Make a machine that can walk 2 metres along a level floor, pause for 30 seconds, then walk another 2 metres, without falling over. There must be no wheels in contact with the floor at any time. The machine should start in a standing position, without any human support.

6.3.2. Ideas

Stability is an important design consideration for walking machines: static stability occurs if the centre of gravity is always within the base area of the feet, dynamic stability will be needed if this is not true while walking. How will your machine know if it's off balance? Design and use some balance sensors so you can use feedback to determine what the control program should do to keep the machine balanced. How many legs will you use? You may wish to make the design brief more precise: two legged walking machines are much more difficult than three legged ones. Get ideas and inspiration by looking at humans, animals and insects walking.

6.3.3. Extension activities

Make a fast walking (or running) machine. Modify the design brief so that the robot must turn around after 2 metres and walk back to the starting point. Make a walking machine that is able to sense and negotiate obstacles (walk over or around them), or walk up and down steps.

6.4. Burglar alarms

6.4.1. Design brief

Make a burglar alarm to protect a small room from illegal entry. It should be designed to scare off criminals. It should not give false alarms when the switches move for short periods of time, and should reset itself after 2 minutes. Several 'valuable' items will be placed in the room, then the alarm will be tested by three attempts at breaking in and removing these items without setting the alarm off. The attempts will be at 5 minute intervals.

6.4.2. Ideas

There are three design components: the sensors, the alarm to scare burglars, and the control program. What kinds of switches and sensors will you use, where will they be positioned, and how will they be connected together? You could sense every possible entry point, or detect movement anywhere in the room. What devices will you use to scare off burglars?

6.4.3. Extension activities

Make an alarm suitable for use in a car or on a motorbike, to deter thieves. Adapt the alarm system so that valuable items (videos, hifi) can be protected. The system should be triggered by the removal of any item.

6.5. Locks and safes

6.5.1. Design brief

Make a lock that opens automatically when the 'key' is used, but otherwise stays shut.

or

Make a safe that protects a 'precious jewel'. It should sound an alarm if the jewel is stolen, but not if it is removed by it's owner.

6.5.2. Ideas

Locks and keys can be completely mechanical. Or the lock can be operated by SEQ, and the key could be the information that allows you to open it. Information could be a secret combination, or coded (for example, a bar code or magnetic strip). Will it be necessary to use the key to set the lock or close the safe?

6.5.3. Extension activities

Arrange your lock or safe so two people must cooperate to open it. Each person will have half the 'key'. Make a bank 'time' vault, that can only be opened during banking hours. Put SEQ and the batteries inside the safe as well, for added security. What will you do if the program goes wrong?

6.6. Detectors and followers

6.6.1. Design brief

Make a vehicle that will follow a line stuck on a contrasting surface. The line will be laid out as a maze, but will not bend by more than 90 degrees, or cross itself. Your vehicle should follow the line from start to finish.

or

Make a vehicle that will discover squares stuck on a contrasting floor. When it discovers one, it should stop and make a signal to indicate that it's found one. The operator should then be able to tell it to search for another (without picking it up or moving it).

6.6.2. Ideas

Both designs will need suitable sensors, driving and steering mechanisms. You could use a white line or square on a dark floor, or vice versa. Magnetic or physical line followers and detectors are also feasible. A major part of the line follower design is devising a suitable control program so that the vehicle can search for and find the line when it goes off course. Following a straight line would be a good start: then adapt the control program so it can go around bends in the line. The detector vehicle will need a search algorithm to look for things. But it mustn't get stuck in corners!

6.6.3. Extension activities

Make a vehicle that can follow a walled maze without getting stuck. Maybe it will reach the centre by trial and error. Change the detector vehicle into an automatic prospector. It should move around looking for objects (bricks, wood blocks, balls) and picking them up.

6.7. Drawing machines

6.7.1. Design brief

Make a machine that draws automatically. It should be possible to make it draw different things. The machine must not have any wheels that move along the ground, and the body of the machine should remain in one place as it draws.

6.7.2. Ideas

You could move the paper and keep the pen still, move the pen and keep the paper still, or move both. Many of the devices from the introductory and extension activities can be adapted to draw. The condition at the end of the design brief excludes buggies and other vehicles. The drawing could be modified by changing the control program, or mechanically.

6.7.3. Extension activities

Adapt your machine so that it draws a copy of a shape that you give it, for example a triangle or circle made out of plastic. It should be possible to enlarge or reduce the original, so a bigger or smaller copy is made. Adapt your machine so it can be used as a drill or router. A drilling machine should produce a pattern of holes, a router will cut out the shape. Use a relatively soft material, for example stiff cardboard. Make a harmonograph or spirograph machine, that automatically draws interesting, curving patterns. It should be possible to make different patterns by altering the machine or program slightly.

6.8. Music machines

6.8.1. Design brief

Make a machine that will play music. The music should last at least 20 seconds before it repeats itself. It should be possible to change the tune.

6.8.2. Ideas

Music machines can be considered in two parts, the note (pitch and duration) selecting mechanism, and the parts that sound each note. Note selection could be mechanical, or by changing the program, or by a combination of both. Methods for producing sounds include striking (a hammer), plucking (strings), blowing (an electric air pump and variable length pipe?), and vibrating.

6.8.3. Extension activities

Adapt your machine so that it can play a tune with at least two different sorts of notes in it. They could be longer and shorter, louder and softer, or of a different quality (timbre). Modify your machine so that the music can be played faster or slower, under manual control. Identify a certain style of tune: waltz, cha-cha-cha, etc. Make a rhythm machine, preferably one that can play several different rhythms.

6.9. Changing gear

6.9.1. Design brief

Make a vehicle that changes gear automatically. It should start in a low gear, then change up if it's going fast enough. If it's in the high gear and going too slowly, it should change down. Test it over some steep hills and slopes.

6.9.2. Ideas

Three major aspects: making a gearbox that can change gear electrically, using one of SEQ's outputs; devising a speed sensor to detect how fast the car is going; and designing a suitable control program to change up and down as the speed varies. Gear changing could be achieved using a solenoid, or linear motor drive. The speed sensor could be a switch that's on when the speed is sufficient (centrifugal?) and off otherwise, or you could look for fast or slow pulses (easier to make, but needs a more complicated control program).

6.9.3. Extension activities

Make the vehicle go automatically into reverse if it gets stuck. Make an all-terrain vehicle, adding features that allow it to cope with varying conditions automatically.

7. Techniques

7.1. Sequences

7.1.1. Introduction

The idea of a sequence of instructions is central to SEQ.

Two concepts are involved:

one instruction is obeyed at a time, the next is not begun until the previous one has been completed.
and
the effect of an instruction depends upon what happened as a result of the previous one ('state').

Use of the domino cards, key cards, and recording sheets will help make the instructions concrete, and allow discussion of the sequences as they are developed and tested.

7.1.2. Sequence activity 1

Needs: 1 light & SEQ, morse code book
Learn about: sequences, codes

Attach a light to the Out output. Here are two short sequences for SEQ:

Sequence 1:
Out 10
Pause 5
Out 5

Sequence 2:
Out 5
Pause 5
Out 10

Show someone these two sequences and the light, but don't let them see which instructions you give SEQ. Can they guess which sequence you used? Make up and try out some different sequences.

Make the light flash three times, then wait 0.5 seconds, then flash twice.

Make the light flash out the morse code for SOS. Press GO to do it again. Write a morse code message; see if your friend can understand it!

7.1.3. Sequence activity 2

Needs: buggy & SEQ (see "Buggies" for more details)
Learn about: sequence, state, prediction

Here are two sequences for you to try with the buggy and SEQ:

Sequence 1:

Forward 4
Right 15
Forward 3
Right 15
Forward 3

Sequence 2:

Right 15
Forward 4
Right 15
Forward 3
Right 15
Forward 3

Where do you think the buggy will end up? Try the buggy from the same position with each program. What is the difference between the programs? Can you adjust the buggy before it starts so it ends in the same place for each program?

7.1.4. Extension activities

Bigtrak activity cards are useful; the following all provide details for structured activities that will help develop the idea of sequences:

5: Draw a crossroads on a piece of paper, with the letters A-D down each arm, and R in the middle. Write programs to make the buggy go from A to B, A to C, spell out BAD, CAB, and ABRACADABRA.

6: Program the buggy to draw letter shapes or your initials.

7: Program the buggy to make the same shapes as some plane shapes (triangle, square, hexagon, circle). Try different sizes.

9: Shuffle domino cards, arrange them in a program, predict where the buggy will end up, then try the program out.

12: Make up a village of shops, and a shopping list. Send the buggy shopping.

13: Make up a street of numbered shops and make the buggy (the postman) deliver letters to the shops in order, or down each side in turn.

14: Making up an adventure for the buggy to visit places on a model village, with streets, crossings and ponds.

15: Writing a program to knock down numbered 'skittles' arranged in different ways.

16: Writing a program to trace out symmetrical patterns.

7.2. Designing & making

7.2.1. Writing programs

The process of developing, testing and debugging control programs is in essence no different from any other design process. In general, design can be considered as:

1 Identify the problem or need
2 Research: identifying the possibilities and difficulties
3 Generating preliminary ideas
4 Selecting appropriate ideas from those generated
5 Making or implementing the selected ideas
6 Testing and evaluating the solution

As a result of testing and evaluating, some or all of the previous steps will need to be reconsidered. Testing may require revision of how the idea was implemented (stage 5), or of the ideas (stages 3 and 4).

Problem solving with control is similar:

1 Identify clearly the problem, task or need.
2 Research: identify instructions, inputs and outputs
3 Think up possible solutions and express them abstractly as algorithms: 'ways of doing it'.
4 Select appropriate algorithms.
5 Write a sequence of instructions (the program) to carry out the selected algorithm.
6 Test and evaluate the program.

As with design, testing control solutions may require revision of the program (how the algorithm was implemented, stage 5), or of the algorithm (how the problem was to be solved, stages 3 and 4).

It is not always necessary to impose such a formal structure on the design-making process, but children (and adults) should be aware of these stages, and able to use them to structure their problem solving activities when appropriate.

One of the difficulties and advantages of control solutions to problems is that trying out alternatives is easy and quicker to do, because things don't have to be physically made or modified using materials and tools. This is an advantage because many more design ideas can be explored before one is selected, and a difficulty because time can be wasted looking at many different control solutions at random, in the hope that the 'right' one will be found by luck. There is a distinction between debugging (a thoughtful, considered activity) and hacking (suck it and see). Hopefully, structured play falls between these two extremes.

7.2.2. Programming SEQ

A specific difficulty with SEQ is that it is not possible to examine a program once it has been entered. Domino or key cards, or written instructions will be found invaluable, as the abstract program can then be seen, examined, followed (as SEQ obeys it), justified, discussed and argued about.

SEQ encourages incremental development of sequences of instructions, not altogether a good thing as planning can be avoided. However, the Clear Error key (CE) can be used to revise the sequence as it is developed and tested, together with the Test key. Again, domino or key cards, or written notes, should be

encouraged. Later on the program can be simplified if required: for example, successive small Forward (and Backward) instructions can be added together into one larger Forward (or Backward) instruction.

7.2.3. Design & make activity 1

Needs: buggy, SEQ, domino or key cards
Must know about: using buggy a bit
Learn about: predicting, sequencing

Use domino or key cards to show this program:

Forward 5
Backward 5
Backward 5
Forward 5

Now enter the program into SEQ. Where do you think the buggy will end up when you press GO? Try it.

Press CE to clear the last instruction, and take away the last domino or key card. Where will the buggy end up now? Try it.

Keep pressing CE, taking away cards, and predicting where the buggy will go.

Try this game with someone else. First you need to make up a program with domino cards. Then enter the program into SEQ. Then your friend has to say what the buggy will do. Keep press CE, take away a domino card, and ask them again.

7.2.4. Design & make activity 2

Needs: buggy, SEQ, domino or key cards, blocks for maze
Must know about: using buggy a bit
Learn about: incremental development & testing

Make a maze for the buggy. Mark the start and finish. Press CM to clear its memory. Put the buggy at the start of the maze, and mark where it is with a block or bean bag.

1 What do you want the buggy to do next? Which instruction do you want? Find a domino or key card for the instruction, then press the keys on SEQ. Press the Test key (a tick). Did the buggy do what you wanted?

2 If it went wrong, press CE to remove the last instruction, take the domino or key card away, move the buggy back to the marker, and try again. Go back to step 1.

3 If it was ok, move the marker to where the buggy is now. Has the buggy reached the end of the maze? If not, go back to step 1.

4 Move the buggy back to the beginning of the maze, press GO and watch it do all the instructions on the cards.

7.2.5. Design & make activity 3

Needs: buggy, SEQ, blocks for maze, domino or key cards, record sheets
Must know about: using buggy a bit
Learn about: planning, estimating & measuring, recording, testing, prediction

Make a maze. Put the buggy at the start. Think of (and write down) at least two different ways of making the buggy go from the start to the end. Describe one of them to a friend. They must plan (and write down) a sequence of SEQ instructions to do what you asked. You should enter these instructions into SEQ, and test them.

If the buggy doesn't reach the end, your friend must look at the planned sequence, and modify it. Try again, until it works.

Write down your working maze program, but put one deliberate mistake in it. Find someone else to work with. Can they spot the mistake? Let them try to find the bug: watch what they do and how they discover it.

7.2.6. Extension activities

Make a buggy draw a simple shape (triangle, square, rectangle, boat). How many different programs can you write that draw the same shape? Can you draw it backwards? What are the similarities and differences between these programs?

Make a list of bugs you have found and fixed or write a story about fixing a bug.

7.3. Repeating things

7.3.1. Introduction

SEQ's repeat instruction is useful if you wish to do the same sequence of instructions twice. This can save laborious and error prone entry of the same instruction sequences, and also economizes on SEQ's memory.

The repeat instruction repeats as many of the previous instructions as you wish. Children (and adults) will find it easier to have a physical representation of the program: use the domino or key cards, or write down the instructions on paper.

Repeating counts back from the current instruction, using the number entered as the number to count back by. It doesn't count the repeat instruction!

The Jump instruction lets you repeat instructions forever, or at least until you press STOP, disconnect the batteries, or they run out. This can be useful if a model has to run continually, for example a clock or metronome.

It is necessary to realize that SEQ's instructions are numbered so that you can tell it which instruction to jump to. The first instruction is 1, the last 40. Jumping to instruction 1 repeats every instruction in SEQ before the jump instruction, forever.

7.3.2. Repetition activity 1

Needs: 1 light & SEQ
Must know about: SEQ output instructions
Learn about: counting on and back, repeating sequences, rhythm

Attach a light to the Pulse output. Try this SEQ program:

```
Pulse 1
Pulse 2
Pause 5
x2 3
```

How many pulses did you count? Can you explain why? Change the repeat number to 2. How many pulses will you see and hear now?

Make SEQ flash the light twice, wait half a second, and then do it again. Use the x2 instruction.

Make an interesting rhythm using a combination of Pulse, Pause & Repeat instructions.

7.3.3. Repetition activity 2

Needs: 1 light & SEQ
Pre-knowledge: SEQ output instructions
Learn about: numbering instructions, rhythm

Attach a light to the Pulse output. Try this SEQ program:

```
Pulse 1
Pulse 2
Pause 5
Jump 1
```

How many pulses did you count? Can you explain why? Change the jump number to 2. How many pulses will you see and hear now?

Make SEQ flash the light twice, wait half a second, and then do it again. Use the Jump instruction.

Make an interesting rhythm using a combination of Pulse, Pause, Repeat & Jump instructions.

7.3.4. Repetition activity 3

Needs: buggy & SEQ (see "Buggies" for more details)
Must know about: simple buggy control
Learn about: definite & indefinite repetition

Try this SEQ buggy program:

```
Forward 6
Right 15
Forward 6
Right 15
x2 4
```

What shape does the buggy draw on the floor? Change the program so the buggy ends up at its starting point.

What happens if you change the repeat instruction to x2 2? Make the buggy draw a triangle or hexagon using repeat instructions.

Try this program:

```
Forward 3
Right 2
Jump 1
```

What shape does the buggy draw now? Try changing distance it goes forward, or the amount it turns, and see what happens to the shape.

7.3.5. Extension activities

Unlike Bigtrak, you can use as many repeat instructions in a SEQ program as you wish. SEQ won't get confused, but you might!

You can repeat a repeat instruction if you wish (this will then repeat the sequence once more). Or you can repeat all the previously repeated instructions, so the sequence will be repeated 4 times. Experiment with these carefully, keeping a note of what happens. Use a short sequence to repeat.

With 40 instructions in SEQ, what is the longest terminating sequence you can write using repeat instructions? Estimate how long it will take to finish if a 1 second instruction is repeated.

7.4. Independent motor control

7.4.1. Introduction

When SEQ is first connected it is in normal motor control mode, so that both motors turn together when Forward, Backward, Left or Right instructions are obeyed. If you build a model with two motors operating different things (for example, a crane with one motor raising and lowering the hook, and the other turning the base around), you will want to control the motors independently.

Pressing and holding the Stop key, and then pressing the Left key, selects independent motor control mode. Now the Forward and Backward instructions control the motor attached to the socket above these keys, and the Left and Right instructions control the other motor, independently of each other.

In normal motor control mode SEQ obeys the Forward and Backward instructions by timing (or counting pulses) 8 times the actual amount entered: for example, Forward 1 takes as long as Left 8. This allows accurate turning, but reasonable forward and backward movement.

In independent motor control no multiplying occurs, and all instruction values are taken literally. This makes both motors behave consistently.

Normal motor control mode can be selected by pressing Stop and then the Forward key, or by disconnecting SEQ and plugging it in again.

Note that normal motor control mode is the only example of SEQ doing more than one thing at a time: every other instruction is obeyed sequentially, whereas motors are controlled both at once, in parallel.

7.4.2. Independent motor control activity 1

Needs: crane or robot arm with 2 motors (see "Cranes" and "Robot arms" for details), SEQ
Must know about: using SEQ with 2 motors vehicles
Learn about: sequence, planning, independent control

Press Stop-Forward, and then CM. SEQ is now in normal motor control mode. Try using the Forward, Backward, Left and Right keys to control the crane or robot arm. What happens? Is it easy to write a program to do what you want?

Press Stop-Left, and then CM. SEQ is now in independent motor control mode. Try using the Forward and Backward instructions: which motor moves? Now try using the Left and Right instructions: which motor moves now?

Write a program to pick up an object and move it to another place, avoiding an obstacle (perhaps a wall). Plan out the sequence of moves first: which motor will you need to move first?

Write another program to move the object back again. How does this program compare with the first one? How many different programs can you write that do the same thing?

7.4.3. Independent motor control activity 2

Needs: buggy & SEQ
Must know about: using buggy & SEQ
Learn about: sequence, independent control, scaling

Press Stop-Forward to select normal motor control mode, then try this SEQ buggy program and note what happens:

```
Forward 1
Left 8
Forward 2
```

Now press Stop-Left to select independent motor control mode, press CM, and try the same program. What happens now?

Press CM and try this program:

```
Forward 8
Right 8
Backward 8
Right 8
Forward 16
Right 16
```

This is the closest program in independent motor control mode to the first one in normal motor control mode. Notice that each motor moves separately from the the other, so the buggy moves in a crab-wise fashion.

Also notice that Forward 1 has been replaced by Forward 8 and Right 8, and Forward 2 has been replaced by Forward 16 and Right 16. What happens if you use the same numbers?

Write a small buggy program in normal motor control mode. Then rewrite it in independent motor control mode. Which mode is easier to use with buggies? Why?

Why do you think SEQ multiplies the Forward and Backward distances by 8 in normal motor control mode?

7.4.4. Extension activities

Calibrate SEQ. How long (time it) is Forward 50, Backward 50, Left 50 and Right 50 in normal motor mode and independent motor mode? Check this for other values: 1 to 10, 20, 30, etc. Or calibrate SEQ with a specific model attached, and measure how far parts move or turn in each mode. Make a chart so you can write programs to make the model do specific tasks more easily.

You can mix independent mode and normal mode instructions in the same program, by pressing Stop-Forward or Stop-Left before entering instructions. Use a buggy and write a program that draws a square, but use independent motor control instructions to turn the corners using 1 motor only. Attach a pen or pencil to the buggy to see the shape (see "Turtles"): how does it compare to a square drawn in normal motor control mode?

7.5. Switch feedback

7.5.1. Introduction

The only input that can control which instruction is obeyed next is the Switch input, directly above the JSW key. By using JSW instructions the program can look at the state of the real world, test what is happening, and take appropriate action. The program can loop until something happens, or while something is ok it can continue looping. (See "Appendix 4: Program structure & SEQ" for more information on loop structures).

The Stop input does exactly that, halting SEQ's program completely. This can be used for limit switch inputs, to prevent the mechanism continuing to operate when some pre-set condition occurrs (for example, a crane jib moving so it might topple over).

SEQ does not remember what happens at the Switch input. The only instruction that looks at the state of the Switch input is JSW, and it only looks at it briefly. This is called polling. So your program might miss a switch event, unless it looks at the Switch input frequently, or you arrange the switch input device so that pulses last a reasonable length of time.

Unlike the Switch input, SEQ responds to the STOP input immediately, regardless of what instruction is being carried out. That is why it can be used as an emergency stop switch: SEQ stops even when in the middle of an instruction. The Stop input behaves as an interrupt.

Normally when SEQ obeys a JSW instruction it tests the switch input; if the switch is closed (light on), it will jump to the instruction number given. If the switch is open (light off), it will ignore the jump, and go on to the next instruction. Jumps (like JMP) can be forward (to an instruction yet to be entered) or backward (to one that already exists).

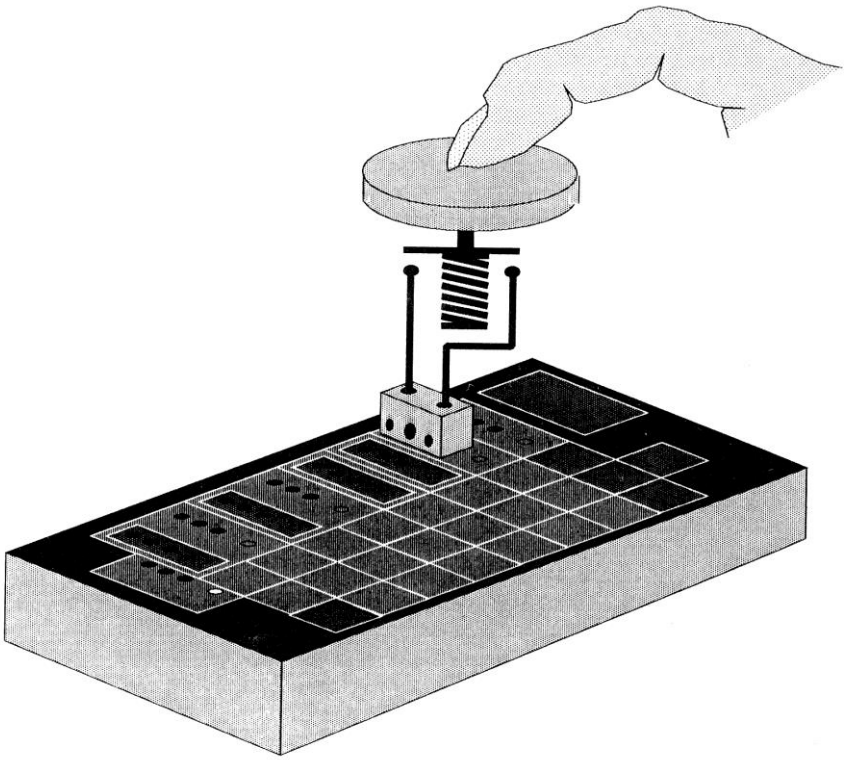
In inverse switch mode SEQ obeys the JSW instruction in the opposite way, by jumping when the switch is open, and ignoring the instruction when it is closed. This can be useful if you find the following instruction sequence necessary:

```
1       Pulse 2
2       JSW   4       stop looping if the switch is closed
3       JMP   1       otherwise carry on waiting
```

Using inverse switch mode replaces this with:

```
1       Pulse 2
2       JSW   1       loop if the switch is open
```

All the activities can be carried out using any switch or sensor: opto-sensors, touch sensors, etc. instead of ordinary switches (see "Introduction to input devices").



7.5.2. Switch feedback activity 1

Needs: 1 normally open switch, SEQ
Must know about: JMP instruction (see "Repeating things")
Learn about: feedback, switches, testing at the end of loops

Attach the switch to the switch input socket (above JSW). Notice that the light below the socket comes on when you press the switch, and goes off when you let go. Enter this program:

```
1       Pulse 1
2       JSW 1
```

Press and hold the switch, and press GO. What happens? Let go of the switch. What happens now? Try again, but this time don't hold the switch down when you press GO.

Try this program and see if it behaves differently:

```
1       Pulse 1
2       Pause 10
3       JSW 1
```

Change the first program so that the Pulse light flashes while the switch is not pressed, and stops as soon as you press it. (Hint: try using the JMP instruction as well).

Describe what SEQ does when it meets a JSW instruction.

7.5.3. Switch feedback activity 2

Needs: 1 normally open switch, SEQ
Must know about: JMP, testing at the end of loops
Learn about: testing at the beginning of a loop

Attach the switch to SEQ's Switch input. Try this program:

```
1       JSW 6
2       Pulse 1
3       Pause 5
4       Pulse 1
5       JMP 1
```

What difference does putting the JSW instruction at the beginning of the loop make?

7.5.4. Switch feedback activity 3

Needs: 1 normally open switch, SEQ
Must know about: testing at the beginning & end of loops
Learn about: polling and interrupts

Connect the switch to the Switch input and try this program:

```
1       Pause 5
2       Out 1
3       JSW 5
4       JMP 1
5       Pulse 1
6       JMP 1
```

Can you press the switch and let go without letting SEQ beep? When does SEQ look at the switch position? Try making the Pause shorter.

Connect the switch to the STOP input. Does this make any difference?

7.5.5. Switch feedback activity 4

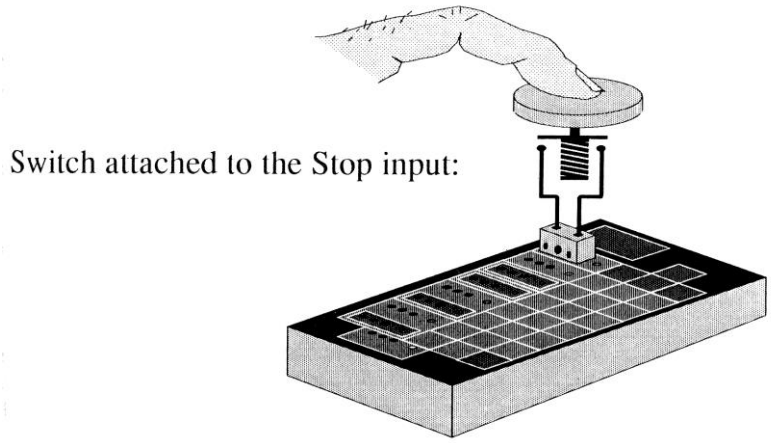
Needs: 1 normally open switch, SEQ
Must know about: testing at the beginning & end of loops
Learn about: inverse switch mode

Attach the switch to SEQ's Switch input. Select inverse switch mode by pressing and holding the STOP key, and pressing the JSW key. Try the programs in activities 1 and 2 again.

Describe what SEQ does when it meets a JSW instruction in inverse switch mode. Compare this with what it does in normal switch mode.

7.5.6. Switch feedback activity 5

Needs: 1 normally open switch, SEQ
Must know about: using JMP instruction
Learn about: using STOP input



Attach the switch to SEQ's STOP input, above the STOP key. Press the switch: the STOP light should come on. Let go of the switch: the light should go off. Enter this program:

```
1       Pulse 2
2       JMP   1
```

Press GO, then press the switch. What happens? Press and hold the switch, then press GO again. What happens now?

Is there any difference between pressing the STOP key on SEQ and pressing the switch?

7.5.7. Extension activities

Try the above activities using a normally closed switch (see "Introduction to input devices").

Put the switches you have been using on a buggy, and make it avoid obstacles (see " Buggies with bumpers").

Design a burglar alarm program, using a switch under a mat, or mounted on a window or door (see "Burglar alarms").

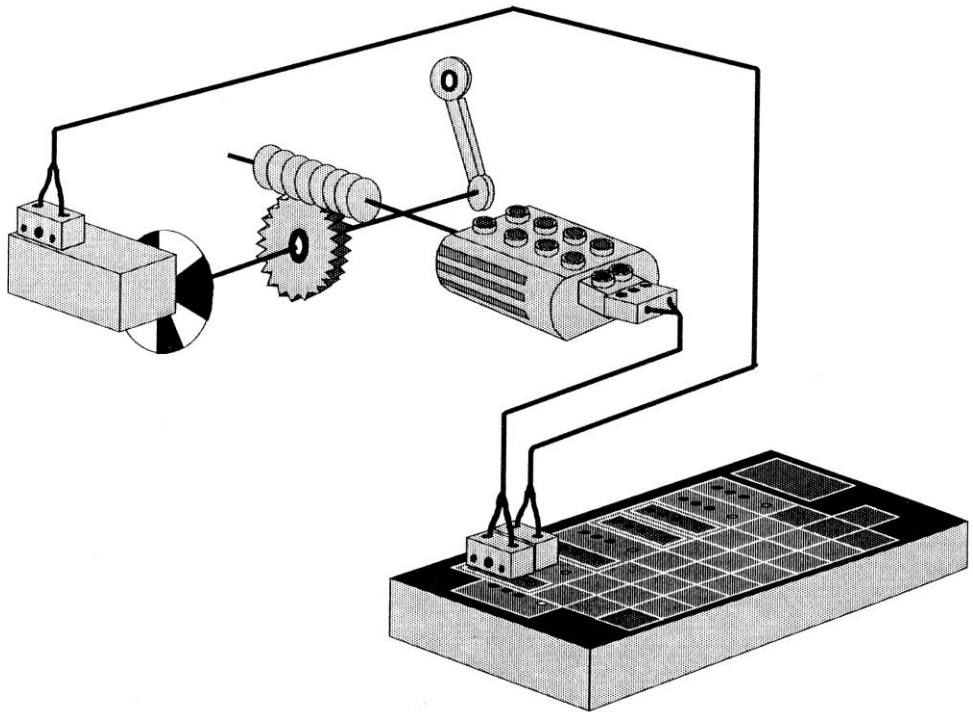
7.6. Motor feedback

7.6.1. Introduction

Normally SEQ interprets motor instructions (Forward, Backward, Left, Right) by turning them on for a fixed duration. This is an accurately controlled length of time; but varying mechanical resistance, friction, loading, backlash, slipping, inertia, etc. may mean that the actual amount of rotation produced by the motor may vary undesirably. Feedback from motors allows SEQ to count pulses as the motor rotates, ensuring accuracy and repeatability.

These pulses can be provided by an opto-sensor or other switch/sensor arrangement (see "Making switches and other sensors"). The opto-sensor consists of a light sensitive switch and a shaft encoder (a disk with black and white segments, LEGO call this a 'counting disk'). The shaft encoder is mounted on an axle driven by the motor, and the light sensitive switch placed so it can react to the black and white segments.

When the opto-sensor is connected to an input on SEQ the yellow input light should go on and off as the shaft rotates. SEQ will count these pulses if they occur more often than the duration of one motor unit (approximately 0.1 seconds). If they occur more often than approximately once every 10 milliseconds (100 pulses/second), SEQ may miss some of them, and count rotation inaccurately. The STOP-number command can be used to change the pulse count per unit, allowing the opto-sensor to be mounted directly on a motor, or on a slower shaft after speed reduction.



7.6.2. Motor feedback activity 1

Needs: 1 motor & arm, reduction drive, opto-sensor, SEQ
Must know about: using a single motor (see "Independent motor control"), repetition, changing motor gearing
Learn about: feedback, pulses

Connect up the motor to the socket above the Forward key, but leave the opto-sensor unconnected. Select single motor mode (STOP-Left) and enter this SEQ program:

Forward 2
Backward 2
JMP 1

Press GO. Watch the extreme positions of the arm as the program repeats. Leave the motor running for a while. Do the extreme positions remain the same? If not, why do you think they change? Try slowing the motor down with your finger: what happens to the extreme positions now?

Connect the opto-sensor to the socket above the Backward key. This is the feedback socket for the motor you are using. The yellow light should flash on and off as the sensor rotates.

Try the same program now. Watch the extreme positions. Try slowing the motor down. Does adjusting the distance between the opto-sensor and the shaft encoder make any difference? If the surrounding light is very bright it may upset the sensor: try shading it.

Experiment by changing the gearing. Try placing the opto-sensor on the motor shaft, or on a shaft that goes slower (or faster) than the motor shaft.

Investigate the effect of changing the pulse count: press and hold the STOP key, then press 2 (or any number from 1-9). What happens?

If you are using the LEGO opto-sensor, turn the shaft encoder disk around and use the side with 8 black segments instead of 4. What difference does this make?

7.6.3. Motor feedback activity 2

Needs: robot arm using 2 motors & opto-sensors, SEQ
Must know about: use of separate motor control (see "Independent motor control")
Learn about: feedback, planning sequences

Attach the robot arm motors and opto-sensors to SEQ. Select independent motor control mode. Write a program to make the arm move, pick up an object, move it somewhere else, and drop it.

Then try to adapt your program so it works with the opto-sensors disconnected. You should be able to use the same instruction sequence, but may need different values.

Which program is more reliable? Why?

7.6.4. Extension activities

"Accurate buggies - using feedback" provides a number of control and constructional activities using opto-sensors on buggies.

Make your own shaft encoder. You could use fewer (or more) black and white segments, or make it larger. Or make a shaft encoder with holes in it and a light shining through the holes. Try it with some of the activities above.

Make your own shaft encoding switch (see "Making switches and other input devices").

8. Devices

8.1. Output devices

8.1.1. Introduction

An output device can be connected to any of SEQ's four output sockets, and will convert SEQ's electrical outputs into some other form of energy, typically:

motion	motors, solenoids
sound:	buzzers
light:	bulbs, LEGO light brick
magnetism:	electro-magnets

These and other output devices will work when connected to SEQ's output sockets, but a continual path (circuit) from one pin of the socket, via the device, back to the other pin, must be provided for the electricity. All the output sockets are red.

Some output devices (such as lights) work the same regardless of which way around they are connected to electricity. Other devices (like motors) behave differently, depending on the direction of electricity flow. For example, reversing the direction of electricity through a motor will change the rotation direction.

Electricity flows from one socket on SEQ, via the output device, to the other. Different instructions determine the direction of electricity flow on the two motor sockets, and the lights under each socket indicate the direction of flow. The Pulse and Out sockets produce electricity of one polarity only.

In independent motor control mode, Forward and Right give the same polarity as that from the Pulse and Out sockets; Backward and Left give the opposite polarity.

Electricity can be continuous, and the output device will be on continuously; or it can be intermittent, and the output device will go on and off. The two motor outputs and the OUT output on SEQ are continuous, but the Pulse output is intermittent, allowing devices to be turned on and off a programmed number of times.

8.1.2. Serial & parallel

Several output devices can be connected together to the same output socket on SEQ. There are two ways to connect devices: serial and parallel.

Serial connection means that the devices are connected together one after the other, in a daisy-chain fashion, and the first and last one connected to SEQ's output socket. Electricity (voltage) will be shared between each device, and it may not be sufficient to drive each one properly. For example, several lights in series will each appear dimmer than one light.

Parallel connection means that each device is connected directly to SEQ's output socket, side by side with the others. Each device can receive the full amount of electricity (voltage), but SEQ may not be able to provide sufficient electrical power (current). For example, several lights in parallel will be as bright as one light, unless they overload the output socket, in which case SEQ turns them all off.

Note that it is not very easy to use LEGO plugs to make a series circuit; Fischertechnik ones are ok.

8.1.3. Output features

Features of SEQ's four outputs are summarized below:

Two main MOTOR outputs:
- bidirectional (both polarities)
- continuous
- feedback possible
- four instructions (Forward, Backward, Left, Right)
- can operate both motors together, or independently
- can set feedback pulse to instruction step ratio

PULSE output:
- single direction (polarity) only
- pulses produced at 0.1 sec. intervals
- no direct feedback (could use SWITCH input)
- single instruction (Pulse)

OUT output:
- single direction (polarity) only
- continuous
- no direct feedback (could use SWITCH input)
- single instruction (Out)

8.1.4. Output devices activity 1

Needs: go-cart & 1 motor (see "Go-carts - simple 1 motor vehicles"), 1 light, SEQ
Learn about: electricity: direction & polarity

Connect the go-cart to SEQ. Write a program to make it go forwards, then backwards to its start position. Carefully unplug the motor lead from SEQ, and turn it around. Can you explain what happens when you try your program now?

Carefully unplug the lead from the motor, and turn it around. Try your program again. Can you explain what happens?

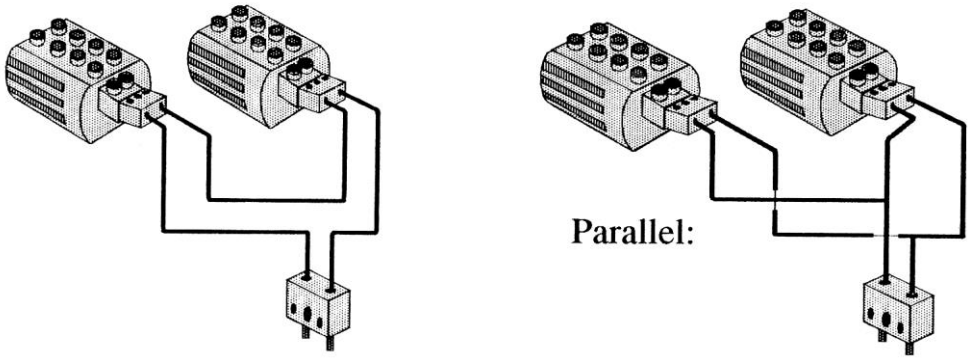
Unplug the motor and connect the light to SEQ. Try your program again. Then turn the plug around at either end and try the program again. What do you notice?

If you have any other output devices you should repeat this exercise with each of them. Which ones behave differently when they are plugged in different ways around?

8.1.5. Output devices activity 2

Needs: lots of motors and lights, leads, SEQ
Learn about: electricity: serial and parallel

How many motors can SEQ work connected to the same socket? How many lights? Try connecting them in parallel, (easy with LEGO plugs, as they stack on each other). Try connecting them in series.



Try connecting them all to the OUT socket on SEQ. Then try using one of the motor sockets.

What happens when you connect too many things to one socket?

8.1.6. Output devices activity 3

Needs: go-cart & 1 motor (see "Go-carts - simple 1 motor vehicles"), SEQ
Must know about: simple sequences, using Pause instruction
Learn about: electricity: pulsed & continuous.

Connect the go-cart to the left motor output (above the Left key). Try this program:

```
Left 1
Pause 5
Left 1
Pause 5
Left 1
Pause 5
```

Connect the go-cart to the Pulse output (above the Pulse key). Write a program to make it behave the same as before.

Connect the go-cart to the Out output (above the OUT key). Write another program to make it behave as before.

Connect the go-cart to the right motor output (above the Forward key). Try this program:

```
Forward 1
Pause 5
Forward 1
Pause 5
Forward 1
Pause 5
```

Can you write programs to make it behave the same when connected to the Pause and Out outputs?

8.1.7. Extension activities

"Independent motor control" looks at how two motors or other output devices can be controlled separately from SEQ's motor outputs. "Motor feedback" looks at using motors with feedback, and adjusting the pulse ratio. "Alternative output devices" gives details of making and using other devices.

8.2. Input devices

8.2.1. Introduction

An input device can be connected to any of SEQ's four input sockets, and will convert some physical change in the real world into an electrical change, typically in resistance. All the input sockets are green (including the battery input). Typical forms of physical energy and examples of sensors include:

mechanical	switches
light	opto-sensors, light dependent resistors
magnetic	reed switches, hall effect switches
heat	thermistor (temperature dependent resistor)

These and other input devices will work when connected to SEQ's input sockets, but a continual path (circuit) from one pin of the socket, via the device, back to the other pin, must be provided. When electricity flows along this path the switch or sensor is called closed, and the indicator light below the socket will come on.

Most simple input devices (switches, LEGO opto-sensors) work the same regardless of which way around they are connected to electricity. Other devices (for example hall effect switches, semiconductor and integrated circuit based devices) must be connected the right way around to work properly, and may even be damaged if connected wrongly. The top pin of all the input sockets is always positive, and the lower pin varies between ground (0 volts) and positive, depending on the state of the input sensor.

SEQ tries to send an electric current through each input device, and detects the voltage across it. If there is a low resistance path (or a short circuit) between the two pins of an input socket, the voltage on the lower pin will be the same as that on the top pin (positive), and the input will be regarded as being on (or the switch as closed). If there is a high resistance (or open circuit, or nothing connected) between the two pins, there will be a low or zero voltage on the lower pin, and the input will be regarded as being off (or the switch as open).

Two of SEQ's input sockets are used by the motor movement instructions, Forward, Backward, Left and Right, to count pulses for more accurate control. Sensors attached to these sockets should be arranged so that they provide pulses when SEQ rotates each motor. See "Motor feedback" for more details.

The third input socket is used by the JSW instruction to determine which instruction SEQ should obey next. Using this socket lets you write programs that react to changes in the real world, using feedback. The other input socket is used for an emergency stop, and is connected in parallel with the Stop key. See "Switch feedback" for more details on both these inputs.

8.2.2. Sensors & switches

The simplest sensor is a switch, which is always on or off. SEQ can use sensors that are not fully on or off, provided they have a low and high resistance state. Circuits inside SEQ judge whether the sensor's resistance is low enough to be called on, or high enough to be called off.

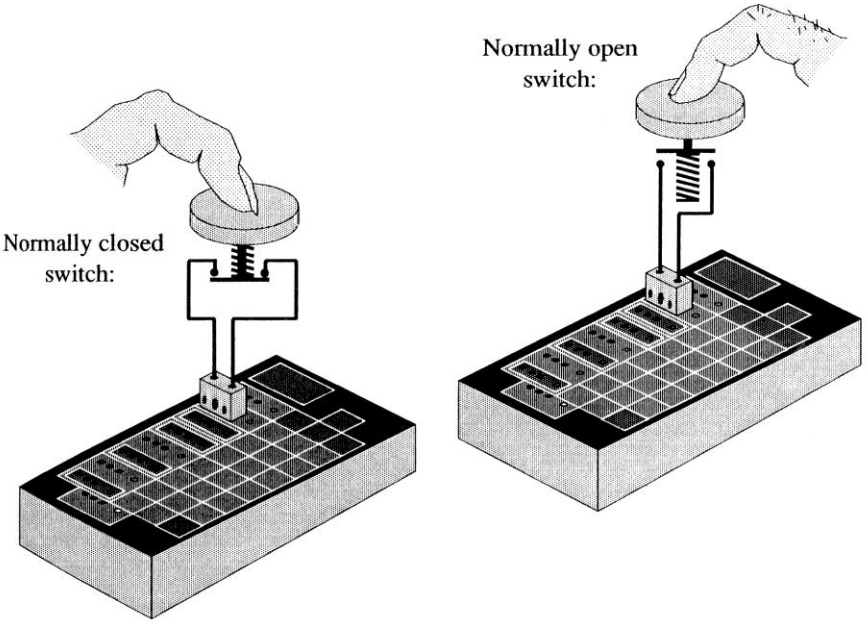
Switches are lots of fun, and provide an excellent introduction to concepts of feedback, electrical circuits, sensors, and logic. You can invent and make your own switches easily, or use a wide variety of existing switches and sensors.

There are two main types of switch: those that are normally open, and those normally closed. Both work by completing and breaking a circuit, the only difference is the position the switch is in when nothing is happening.

Several switches can be connected together. Switches can be connected in parallel, side by side, so that any switch can be closed for electricity to flow. Or they can be connected in series, one after the other, so all the switches must be closed for electricity to flow. Which arrangement you use will depend on what you want your program to detect.

Note that it is not very easy to use LEGO plugs to make a series circuit; Fischertechnik ones are ok.

8.2.3. Switch types



8.2.4. Input devices activity 1

Needs: 1 normally open & 1 normally closed switch, SEQ
Must know about: using JMP instruction
Learn about: normally open & closed switches

Connect the normally open switch to the Switch socket (above the JSW key). Press the switch: the light below the socket should come on. Let go: it should go out.

Try connecting the switch to the other input sockets: there is one above the Backward key, one above the Right key, and one above the Stop key.

Repeat this with the normally closed switch. What's the difference?

Plug a switch into the Switch socket again. Try this program:

```
1       Pulse 2
2       JSW   1
```

Try it with the switch pressed before you start, and afterwards. Now try the other switch. What's the difference?

8.2.5. Input devices activity 2

Needs: 2 normally open switches, SEQ
Must know about: using JMP, connecting wires to things
Learn about: series & parallel circuits

Connect the two normally open switches together so that you need to press both of them for this program to beep:

```
1    JSW 3
2    JMP 1
3    Pulse 1
4    JMP 1
```

Draw a diagram of how they are connected: they should be connected in series.

Now reconnect the switches so that you can press either of them to get the same program to beep. Draw this: they should be connected in parallel.

8.2.6. Input devices activity 3

Needs: light dependent resistor, LEGO opto sensor, SEQ
Must know about: using switches, doing experiments
Learn about: on & off thresholds, hysteresis

Connect one of the sensors to SEQ's Switch input, and point it at a source of light. Make sure the light level stays the same, and that the sensor is held firmly in position. Fix a piece of paper under the sensor.

Using a long thin rod (or a transparent ruler), slowly push an obstacle towards the sensor. As soon as the Switch socket indicator changes, stop moving it and mark on the paper where the obstacle is.

Now move it slowly away from the sensor until the indicator changes back again. Mark this position in a different colour.

Repeat this several times until you are confident of the two positions that you have marked. SEQ has a built in on and off threshold for all its inputs. The exact change-over point will depend on the light level, shape of the obstacle, and the sensor used.

You might like to try moving the obstacle at different angles to the sensor, to investigate its sensitivity.

Try another sensor.

Note that the LEGO opto-brick is off (high resistance) when it is receiving a lot of light, whereas the light dependent resistor is on (low resistance). Also the LEGO opto-brick can detect reflected light (it contains an invisible, infra-red light emitter), so it is on when pointing at white materials, and off when pointing at black ones, provided there isn't too much outside light.

8.2.7. Extension activities

Repeat activity 2 with two normally closed switches. Rewrite the control programs so they behave the same as before.

Use switches to control what your program does. Look at "Switch feedback" for activities.

Experiment with the LEGO opto-sensor and the shaft encoder wheel (with black and white segments). Try using it in a dark place, and with different levels of outside light. See "Motor feedback" for more activities.

Make your own normally open and normally closed switches, or use something different, like a reed switch or thermistor (see "Alternative switches and sensors").

8.3. Alternative output devices

8.3.1. Introduction

SEQ already has two different sorts of output device built-in: light emitting diodes, and a buzzer.

Any output device can be successfully connected to SEQ provided it satisfies a few simple requirements. It must not draw a current of more than 1 amp continuously, or SEQ will regard it as a short circuit, and close the output down. However, SEQ can supply a peak of 2 amps for short periods of time, for devices such as motors with a high start-up current. The two motor sockets are driven in either direction with a voltage 1.5 volts below the battery supply; the two other outputs are single polarity, and receive a slightly higher voltage, 0.9 volts below battery supply level.

Care should be taken in selecting appropriate output devices and batteries. You should provide SEQ with batteries capable of supplying approximately 1 to 1.5 volts more than the output device's voltage. For example, if you are using 4.5 volt motors, use four 1.5 volt cells (rechargeable nicad cells give 1.2 volts, which is also ok). Reduce the voltage and the output device will either give reduced output (slower motors, dimmer bulbs), or not operate at all). Increase the voltage and you may get increased output (faster motors, brighter bulbs), but you may also burn out or permanently damage devices. Do not operate 4.5 volt motors and bulbs from 12 volts!

Total available power is limited by the batteries (their voltage, and how much current they can supply), and SEQ's 1 amp maximum output. If you need more power, use SEQ to control a more powerful energy source indirectly, for example by using a relay.

8.3.2. Movement

Most of the things controlled by SEQ are mechanical, so the most common output device is the motor. Motors work by passing electricity through a coil or coils, producing a magnetic field which reacts with small magnets to produce motion. SEQ's outputs will drive a wide range of DC motors, providing they work at between 4.5 and 12 volts, and draw less than 1 amp continuously. It has not been designed to drive AC or stepper motors.

Motors draw a variable current, depending on their load. If SEQ is driving a motor and it stalls, it may draw over 1 amp, and SEQ will turn off the output for the duration of that instruction. Also note that drawing large currents from small capacity batteries will cause the voltage to drop: if it drops below 4.5 volts SEQ will stop operating and all your instructions will be lost.

Another problem you may experience with motors occurs if you ask SEQ to rapidly turn them in opposite directions alternately. At the end of each motor output instruction SEQ carries out a powered motor stop for a short period (around 0.1 seconds), then leaves the output in a high impedance, off state. If the motor continues to turn (through mechanical inertia, or by some external force) and SEQ obeys the next instruction

and tries to turn it in the opposite direction, a large current surge will probably mean SEQ will think it has detected a short circuit, and stop outputting. An instruction may appear to have been skipped.

Other mechanical devices that can be driven include solenoids and counters. Solenoids consist of a coil of wire: when current flows through the coil a magnetic field is generated, which pulls a metal rod into the coil (usually against a spring). They can be used to produce linear motion directly, instead of converting rotary (motor produced) motion into a linear form. Solenoids are the opposite of mechanical switches.

Electro-mechanical counters are just solenoids with a mechanical counter attached. They can be connected to the Pulse or Out output sockets, as they only need a short pulse of power to operate. It's quite an interesting and challenging task to make your own counter (see "Clocks and counters"). Surprisingly, some solenoids and counters designed to operate from 24 volts work at 12 volts: if in doubt, plug it into SEQ and try!

Meters (at least the non-electronic types) are mechanical too: it might be instructive to connect a voltmeter to SEQ's outputs, or an ammeter in series with a load (motor, light).

8.3.3. Light

There's an enormous range of light bulbs that can be connected directly to SEQ. The only thing to watch is that you don't connect a low voltage bulb when using high battery voltages, or you'll blow it straight away. If this is a problem, connecting two or more bulbs in series will divide the voltage between them.

More recently light has been produced from electricity using light emitting diodes (leds). SEQ's indicators are all leds. They usually operate from a low voltage (typically 2 volts), so you must wire a resistor in series, but some are constant current and don't need an additional resistor. Unlike light bulbs, leds only produce light when connected the right way around. Their advantages include the fact that they take little power, can be small, and don't get hot.

A special type of led is an infra-red emitter. These are commonly used for remote control of televisions, videos (and even turtles). They are used in conjunction with an infra-red receiver, which could be connected to SEQ's Switch input with a suitable interface.

8.3.4. Heat

Most other output devices produce heat as a by-product. Perhaps the most controllable source of small amounts of heat is a light bulb. Resistors of a suitable power rating could also be used: at 6 volts use a 10 ohm, 5 watt resistor; at 12 volts use a 15 ohm 10 watt resistor.

8.3.5. Sound

SEQ already contains a beeper, but you can connect others (and maybe turn SEQ's off with STOP-Pulse). Note that you will need to use a buzzer that produces a tone when connected to DC; some buzzers expect to receive an AC signal. Electric door bells are the most instructive, as they can be taken apart and the mechanism examined. Then you can set about inventing you own noise (or tune) making machine! Use motors or solenoids to get motion, then make the moving parts strike things (bells, gongs, strings): get inspiration from musical instruments (see "Music machines").

8.3.6. Magnetism

Make your own by winding a coil. Use reasonably fine wire so that the final resistance of your coil is over 20 ohms (the higher the resistance, the smaller the magnetic field). But be careful to use a long enough length of wire so that you don't end up making a fuse! If you use a metal core the magnetic field will be increased considerably (although you may get some residual magnetism). You can also get coils from old equipment: relays, solenoids, etc. Check for magnetic fields with a small compass, nails or iron filings (messy).

You can use the coil as an electro-magnet, for example to lift things (on a crane or robot arm), to move something (like making your own solenoid), or even to operate a reed switch (making your own relay).

8.3.7. Electricity

Why use an output to produce electricity? One reason would be in order to control an output device that needs more power than SEQ can provide.

SEQ could be used to control an electric car (anyone got an old Sinclair C5?), but the motors can't be connected directly to SEQ's output. Instead relays can be used. Relays contain a coil that produces a magnetic field when current flow through it, and the magnetic field pulls a mechanism which behaves as a switch. This switch can be used to control another electric circuit, with its own power source (more powerful, at a higher voltage or current) and output devices (powerful electric motors).

Selecting appropriate relays is a matter of keeping within the maximum output current (1 amp) and voltage (12 volts). You will also need to make sure that the current rating of the relay contacts is adequate for the circuit you wish to control.

8.3.8. Alternative output activity 1

Needs: six 1.5 volt batteries, a variety of disposable bulbs (1.5v, 6v, 12v), SEQ
Must know about: connecting up circuits
Learn about: voltage ratings

Connect SEQ up to three batteries. Try this program:

```
1    Pulse 1
2    Out 5
3    Jump 1
```

Connect each bulb in turn to the Pulse output. Make a note of the brightness of each bulb. Do any of them stop working? Repeat, but connect them to the Out output.

Disconnect SEQ & the bulbs, and connect SEQ up to four batteries. Repeat the above, noting the brightness of each bulb, and any that fail.

Continue with five and then six batteries. Explain your observations.

8.3.9. Alternative output activity 2

Needs: 1 or 2 motors, SEQ
Must know about: using JMP instructions

Learn about: inertia, motor current demands & characteristics

Try this program with no load on the motors:

1 Forward 1
2 Backward 1
3 Forward 1
4 Jump 1

Watch the motors and SEQ's motor indicators carefully. Does SEQ miss any instructions? Try loading the motors.

Try putting a Pause between instructions 1 and 2, and between 2 and 3. You may also need one after instruction 3.

8.3.10. Extension activities

Collect examples of output devices that produce movement, heat, light, sound, magnetism and electricity. Try each one with SEQ.

Make your own electro-magnet. Use thin wire, and wind enough to give a reasonable size coil. Test it (use a compass or iron filings on paper). Convert your electro-magnet into a relay, a buzzer, or solenoid.

Make a solenoid operated clock. Use an elastic band or weights for the main power source, and the solenoid attached to SEQ to control the escapement. Is this easier or better than making a completely mechanical clock? See "Clocks & counters" for project ideas.

8.4. Alternative switches and sensors

8.4.1. Introduction

Any input device can be connected to SEQ provided it satisfies a few simple requirements. It's on resistance must be less than around 2000 ohms, and it's off resistance must be more than around 2400 ohms. Inside SEQ each sensor is connected directly to 5 volts (at the top pin of each socket), and in series with a 100 ohm resistor (via the bottom pin of each socket) to ground (0 volts). Therefore any sensor will be provided with a maximum current of 50 milliamps, and should be selected to withstand this.

Several types of sensor can be used: on-off sensors, resistive sensors, or semiconductor sensors.

On-off sensors are usually described as switches: they have a very low on resistance, and a high off resistance. Any switch can be directly connected to SEQ. Current consumption can be reduced (and battery life slightly extended) by placing a 1000 ohm resistor in series with each switch.

Resistive sensors can be connected provided that their change of resistance includes the change-over range of 2000 to 2400 ohms. They must also be able to tolerate the current provided by SEQ, up to 50 milliamps. Usually appropriate resistive sensors can be selected from manufacturers' data sheets.

Most semiconductor sensors will require additional circuitry, usually a few resistors and an additional connection to ground, before they can be connected to SEQ. Consult an electronics textbook for more information, or look at manufacturers' data sheets for interface details.

Switch sensors can be bought or made to detect a wide variety of physical changes. Designing and making sensors can be interesting and rewarding, involving mechanical, electrical, and control techniques.

8.4.2. Movement

These either use the motion to directly produce an electrical change (a switch), or convert the mechanical energy into another, intermediate form of energy, which is then used to produce an electrical change (such as an opto-sensor, where movement is converted into light levels, then into electricity).

Any mechanical switch can be adapted to detect mechanical change. Motion can be linear or rotary. Here are some example sensors; many more are possible:

Linear motion:

- Amplify or reduce movement, perhaps using levers
- Simple push switch, possibly spring loaded
- Slide switch
- Optical switch, using the motion to cover and uncover the light
- Convert to rotary motion, using rack and pinion

Rotary motion:

- Covert to linear, using a cam or rack and pinion
- Use a toothed wheel to hit a small (micro-) switch
- Slotted wheel or shaft encoder and opto-sensor
- Mount a magnet on the shaft and sense with a reed switch
- Use a motor as a generator

Other mechanical changes that can be used include:

Positional changes:

- Mercury tilt switch (be careful not to break the switch: mercury is poisonous)
- Level switches: put a magnet on a float and use a reed switch, or use a micro-switch
- Place a micro-switch near the part that moves

Pressure:

- Spring loaded switch, so that the pressure has to overcome the spring's strength
- Membrane key pads
- Use a burglar alarm pressure mat
- LEGO touch sensor brick

Flow (of liquids or gases):

- Convert the flow to rotary motion, using a water wheel or impeller (like a windmill)

8.4.3. Light

Light dependent resistors are cheap and work very well. The familiar ORP12 can be directly connected to SEQ: it has an on resistance of less than 2000 ohms at reasonably bright light levels, and an off resistance of over 10 megohms in the dark.

The LEGO opto-brick is special: it responds to external light levels, but can also use reflected light.

8.4.4. Heat

Thermistors are temperature dependent resistors, and can be obtained with a wide range of characteristics. The main constraint when using them with SEQ is to ensure that the 50 milliamps (maximum) current from SEQ does not heat the (sometimes very) small thermistor bead. Self-heating will give strange and unexpected results.

Choose a thermistor that has a resistance of around 2200 ohms at the temperature you're interested in.

8.4.5. Sound

Sound detectors are difficult to make because the amount of energy carried by sound is usually very small, and so must be amplified before changes can be detected. You could experiment with a diaphragm: use a sensitive micro-switch, or an opto-sensor and mirror.

8.4.6. Magnetism

A reed switch consists of a small pair of metal contacts encapsulated in glass: the movement can be seen if you look closely. The contacts close (or change over) when a magnetic field is created. Magnets can be used: as they are brought near to the switch it closes, and opens as the magnet is taken away.

8.4.7. Electricity

Why would you need a sensor that responds to electrical changes? Why not use the electricity directly? One reason might be that the electricity is not of the right voltage or type (it could be AC instead of DC). Another reason would be to isolate SEQ from other equipment, so neither can be damaged.

Electricity sensors usually work by converting the electricity into another form of energy, which is then converted back into an electrical change. Any output device could be used (see "Introduction to output devices" and "Alternative output devices").

One possibility is to adapt reed switches. Reed relays work by placing the reed switch inside a coil. When electricity flows in the coil, it produces a magnetic field, which closes the reed switch contacts. Sealed reed relays are also available.

8.4.8. Alternative input activity 1

Uses: 2 leads with bare ends, SEQ, range of materials
Pre-knowledge: use of jump instructions
Concepts encountered: conductors & insulators

Attach two wires to SEQ's Switch input socket (directly above the JSW key). Enter this program:

```
1    Pulse 1
2    JSW 1
3    Pause 5
4    Pulse 1
5    JSW 1
6    JMP 3
```

Press GO. Watch the light below the Switch socket, and listen. Touch or short the two ends of the wire together. What happens? Move the ends apart again. What happens now?

Use the two wires to test materials to see if they let electricity through or not. Make a list of electricity conductors (they let electricity through), and insulators (they don't). Is air a conductor? What would happen if it was?

8.4.9. Alternative input activity 2

Uses: buggy, pieces of metal, wires, springs, elastic bands, SEQ
Pre-knowledge: some construction skill
Concepts encountered: switch mechanisms

Make a switch that could be used as a bumper on the buggy. The switch should close when something pushes against it, and open again when released. Use two pieces of metal, one moving and one fixed. The moving piece of metal might be attached to a hinge.

You could use a spring, bendy metal, or an elastic band to pull the moving part back again.

Connect the switch to SEQ's Switch input. Does the light come on when the bumper is pressed, and go off when it it released?

Try this (or write your own obstacle avoiding program):

```
1       Forward 1
2       JSW 4
3       JMP 1
4       Backward 1
5       Left 10
6       JMP 1
```

Look at "Obstacle avoiding" for more activities.

8.4.10. Extension activities

Many of the more advanced activities use switches or other input devices for feedback and control. Design your own switches for use with:

"Obstacle avoiding"
"Accurate buggies"
"Robot arms" (sensing objects & position)
"Turtles" (the pen up-down mechanism
"Code recognizers"
"Automatic doors"

If you have access to a test meter that will measure resistance, you could use a variable resistor (ideally 2000 to 2500 ohms, although 0 - 5000 would be ok) to find out the exact on and off resistance for each input on SEQ. Set the variable resistor to its highest value, and connect it to each input in turn. Slowly decrease the resistance until the indicator comes on. Disconnect and measure the on threshold resistance using the meter. Re-connect, and slowly increase the resistance until the indicator goes off. Disconnect and measure the off threshold resistance. The exact on and off resistance will vary slightly with battery voltage. so try to keep this constant.

9. SEQ in the curriculum

9.1. Introduction

9.1.1. What & where?

What can SEQ contribute to the curriculum? This guide does not define exactly how SEQ fits into the curriculum at your school. That is for you to determine. This chapter identifies areas of the curriculum where it can contribute: it should be clear that SEQ's problem solving activities are cross-curricular. Your aim as a teacher may be to find ways of integrating technology (and SEQ in particular) into your existing curriculum, or you may feel that certain curriculum areas are non-existent and need developing.

9.1.2. Age & ability

What age and ability is necessary for SEQ to be used successfully? Again, this is not defined here. But children in infant, junior and secondary schools have used SEQ to control aspects of their environment, working at their own level, and learning has taken place. Children of nursery age and below have been shown to enjoy controlling their environment, and less able or handicapped children can be given experiences and opportunities suiting their own level of ability. Skilled teachers will no doubt push the use of SEQ beyond those described here.

Clearly it is unreasonable to expect very young children to design and make complex electro-mechanical mechanisms, and then write sophisticated control programs. But time and time again I have been impressed and surprised at the achievements of young children when give sufficient structure and guidance in using the technology. Certainly, infant children can develop sequential control programs of considerable length and complexity when given a working model connected to SEQ, and are adept at inventing imaginative problems to solve. A planned progression of designing, making and control activities will help.

9.1.3. Problem solving

One of the characteristics of control solutions to problems is that they can be modified more quickly than mechanical ones. For example, an SOS generating program can be quickly altered to send a different message, but modifying a mechanical SOS machine would take much longer. This has the advantage that many more ideas and hypotheses can be tried out in a given time. But with this increased freedom comes a danger: it is sometimes tempting to 'play' randomly with a control program that is not working, hoping to accidentally hit on the solution required.

Of course, play is a necessary part of learning, but the permutations allowed in a control program are many: most of them meaningless, hard to understand and follow, and do not produce the required effect. It's like playing with LEGO: you need to do it to become familiar with the pieces, but until someone shows you (or you look at pictures of LEGO models) you'll need to play for a very long time before discovering all the mechanisms you need to make a fork lift truck. Structured learning experiences need to be provided: both for model making and for writing control programs.

An intermediate strategy that often proves successful is "trial and error". Here there is conscious (often verbalized) thought about what is wrong and what to try next: it can be applied equally well to construction and control problems. Failure leads to success and learning. Indeed, if something works first time a valuable learning opportunities will have been missed: we (should) learn by reflecting on our mistakes.

The need to plan and think arises from complexity: linear sequences of instructions are reasonably easy to follow, especially if they are written down. But as soon as you start using repeat loops, jumps, or conditional

jumps (on a switch input, using feedback), programs can become hard to follow and test unless they're well structured. Simply playing with a control sequence does not mean that a satisfactory (or 'correct') solution to the task in hand will be achieved. A more organized approach is necessary.

9.1.4. Infant schools

SEQ can therefore be used in the infant classroom, where it would be expected to extend the children's experiences, providing an opportunity for them to control aspects of their immediate environment. One of the most noticeable things about infants using SEQ is the amount, quality and variety of language that occurs (see "Observing children"). Many mathematical and scientific concepts are encountered, as described below. And using SEQ is a concrete activity, giving children observable outcomes: a tool to think with. Foundations laid at infant level can be built on during later years, especially in craft, design and technology.

Of course, one of the really nice aspects of using SEQ is that it allows infants to use their existing skills with LEGO (or other construction kits), then making the models do things, under their own control.

Infants using SEQ will also develop new insights that will help them explore machine and computer myths, and extend their psychological and physical perceptions of the world about them (see Sherry Turkle's fascinating book "The second self" for lots of examples of children talking about computers and machines). These include deep questions such as "Are smart machines alive?", "How are machines and computers different from people?", and "What makes a computer smart?".

9.1.5. Junior schools

At junior level SEQ can be used to bridge concrete and abstract thinking. High level skills (hypothesizing, planning, designing, observing, recording) are natural extensions to concrete activities: making and doing. The abstract becomes concrete, and directly accessible. For example, a planned control sequence is entered into SEQ. When GO is pressed, the abstract, invisible program has observable, concrete effects: the motors turn, lights flash, noises are heard. So children can discuss (and argue) about why things happen: cause and effect. And they can experiment with the cause (their program), hypothesizing about what will happen as a result.

SEQ also allows juniors to build upon and extend their physical and mechanical skills and abilities: too often the ability to make things at an infant level is dismissed as 'play' in junior classrooms. Here too is an opportunity to prevent gradual polarization of the sexes: at home boys may be given technical things to play with and make, but girls may not receive the same encouragement. How many girls in top junior classes know how to use gears and pulleys, or can wire up a battery, buzzer and switch? How many girls feel in control of modern technology, rather than threatened by it?

9.1.6. Secondary schools

Much of the above also applies to children in secondary schools, if only because most junior schools are not yet providing their children with designing and making control experiences. Compensatory education should be provided as early as possible in secondary schools: in I.T. Awareness courses, Science, or Craft design technology subjects.

In later years the main use of SEQ will be found in craft design technology and computing courses: for too long children have either had no experience of controlling things in the real world (including things they make), or have struggled with electronic control without feedback (working blind), or have had to wrestle with computer programming at a low level, with inadequate time and knowledge.

9.1.7. Further & higher education

Many students in FE and HE will not have experienced using control to solve technological problems, so again much of the above applies. They can gain technical confidence and competence, while at the same time being able to reflect on their own thought processes.

Computers are often quoted as being part of the 'post-industrial' revolution. Many aspects of contemporary and future society are being shaped by technology, and it is important that all students (as future citizens) understand who is really in control of the technology. Can programs or machines be trusted? Do things have to be designed in a particular way? What are the alternatives?

It is apparent that SEQ could be used as an example of a sequential controller. Students following technology or industry related courses could take modules on robotics, control technology, or designing and making things, based on a selection of the more challenging SEQ materials.

Students who are training to become teachers have an additional bonus when using SEQ (or other computers) in classrooms: it is a tool which can be used to investigate and illuminate learning processes. By encouraging children to engage in (and talk about) problem solving activities, both students and children gain insight into thinking and learning. As with Logo, SEQ makes the abstract concrete, accessible and visible.

9.2. Language

9.2.1. Introduction

Teachers who have introduced control technology into their classrooms always seem to remark on the way in which quality (in addition to quantity) of both oral and written language is improved. The extent to which you wish to accept or use this depends on your own situation, and the age and ability of the children. One thing you ought to do is to listen to children talking as they work (see "Observation").

Children are observed to use oral and written language to sort out their own problems. As you listen, you will hear them use language to discuss their problems, describe possible solutions, talk about how they should be constructed, relate their observations as they test, develop and refine solutions. Many teachers will arrange classroom activities so that children work in groups on a common task: here language has a functional purpose. Children in groups will talk more: the ideal size seems to be three. But don't forget to give them opportunities to work alone, and time to reflect.

If one of your concerns is language development (remember 'Language across the curriculum'?), you will want to focus at least part of your observations of the children and the tasks that they do on their use of language. Or you may wish to use language as an indicator of children's understanding of the designing and making process: after all, how do we know what children are thinking or understand?

Traditionally, teachers question and children offer answers ("Guess what I'm thinking"). In designing and making there are no right answers, and the design processes are often more important than the end product. Therefore children should be encouraged to make explicit their own thinking processes (thinking out loud - we do it anyway). There are plenty of accessible books on oracy and literacy in education: it is not appropriate for this guide to review them. However, some hints as to the kinds of language skills teachers might expect are in order.

9.2.2. Skills & categories

Language encompasses thinking, learning and communication skills. Language activities should enable children to effectively describe, record, justify, persuade, tell stories, report, argue, express feelings, express opinions, explain, speculate and hypothesize, and reflect. Speech reflects thinking: textual markers can be identified that indicate the type of thinking, and the kind of discourse that the speaker wishes to engage in.

Some of the sorts of language you should be able to identify include:

predicting and hypothesizing
observing
sequencing
aesthetic judgment
decision making
explanation

You could classify the following speech structures as hypothetical, experimental, argumentational, operational or expositional, then use your classification when listening or talking to your pupils (after Terry Phillips "Beyond lip-service"):

"What about...", "How about..."
"Say...", "If...", "Could...", "Might..."

"I remember..."
"Once..."
"It reminds me of..."

"Yes but...", "Yes well..."
"....will it?", "...don't they?"

"This...", "That...", "These...", "Those...", "It..."

"Where...", "What...", "Who...", "Which...", "When..."

Both the hypothetical and experimental modes encourage children to turn from immediate activity and reflect, hypothesize, evaluate and order (in other words to become actively involved in their own learning). Expositional style is most used by teachers and pupils in whole class 'discussion' (question & answer sequences). Operational style focuses the child's attention on the context and encourages action.

9.2.3. Functional language

Children develop their own functional language to describe what each key does and means. This may vary, depending on what they are controlling, and should be encouraged. Use the blank or partially blank overlays so they can mark symbols or words representing the associated action.

One class of infants quickly decided to call the Forward key 'upwards' and both Left and Right 'sidewards'. Later they used 'inwards' and 'outwards' for Left and Right: further observation would have shown how far they consistently applied these names. Other concepts are associated with opposite word pairs:

above and below
forward and backward
left and right
faster and slower

Simple questions with deep and difficult answers can be anticipated: "Please Miss, what's a gear?". Naming things begins to give you power over them, and can lead to concept development. Naming and identifying parts is important, and enables discussion without vagueness or confusion. You may find the Glossary helpful for technical terms, but I suspect it was of most use in helping to refine my ideas and concepts: another example of the process being more important than the end product!

9.3. Mathematics

9.3.1. Early experiences

Like using Logo, working with SEQ designing, writing and testing control programs is an activity where mathematical thinking is made visible and constantly encouraged. There are many opportunities for mathematical aspects of these activities to be made explicit and extended, from introductory activities with infant children to advanced work at secondary level and beyond.

Starting with SEQ's instructions, the values that are given with each instruction take meaning from their action. Children discover that Forward 1 is less than Forward 2, and Backward 1 is opposite to Forward 1. Number line ideas can be introduced by, for example, a buggy actually moving along one.

Programming something to achieve what you want involves estimating and measuring. Initially the numbers used with each instruction have no meaning or units. How far is Forward 20? How much does the buggy turn when you do Left 10? A sense of discovery (formally: calibration) exists as children begin to get a 'feel' for these numbers when used with a specific buggy or other model. And then they find that the meaning of numbers depends on the context: use a different model, and they may mean different things.

Any activity that involves turning left or right ought to lead to an increased grasp of the concept of angle, without (at least initially) problems caused by introducing (arbitrary) units of degrees.

One aspect of SEQ that may be usefully studies is the way in which each instruction is timed (unless you use feedback). So children could look upon distance travelled or angle turned as a representation of time: a time line or clock (see "Clocks and counters"). Ideas about the relationship between speed, time and distance can also be developed.

9.3.2. Combinations

There's lots of mathematics in using several instruction instead of one, or joining successive instructions together to make a single one. Young children seem to adopt two strategies when programming a buggy to follow a maze: either go as far as you can in a straight line, then turn, or go forward just 1 unit, then another, and so on, sometimes turning. If they use the latter approach, SEQ's memory (40 instructions) soon fills up.

Two techniques can be introduced at this stage: use domino cards (or some other means of recording instructions), and then look for instructions that can be joined together. In one infant classroom, blank looks were the only response to the question:

"How can you make Forward 1 and Forward 2 and Forward 3 into one instruction?"

But the same children were able to investigate problems such as:

"How could you do Forward 6 in two goes?"
or "How many different ways can you go Forward 6?"

These investigations were made concrete by trying them with SEQ and a buggy. Children learn to count on if the buggy doesn't go far enough, and back if it goes to far. And they can explore which numbers can be combined to make a single instruction with the same effect:

Forward and Forward
Backward and Backward
Forward and Backward
Left and Left
Right and Right
Left and Right

These are effectively signed numbers. Why can't you combine Forward and Left into one instruction?

9.3.3. Concepts

Designing, making and controlling things can involve using, or learning about, lots of mathematical concepts. Gears, pulleys and levers can be used experimentally, leading to an intuitive feel for multiplication, division and ratio; or more formally, so the necessary mechanism is designed or analyzed by calculation. Even using wheels involves ideas of circumference, radius and diameter. Which goes faster, a buggy with large or small wheels?

There is lots of two dimensional geometry in the buggy or turtle world, as described in many books about children using Logo. When constructing robots, cranes, and other mechanisms this geometry extends to three dimensions. Working with these models can help children's spatial awareness, leading to more formal techniques in later years. Calculating the optimum path for a robot arm (or even the shape of the volume the arm can reach) needs advanced mathematics: but programming a robot can give practical approximations to these later concepts.

Central to controlling things is the idea of an algorithm and a sequence of instructions that implements it. There are analogies between mathematical algorithms (for example, for solving equations) and computational ones, but it's the processes involved in designing, implementing and testing algorithms and programs that is at the heart of doing mathematics, computing, and problem solving. As in craft design technology, the problem solving skills are held to be important.

9.4. Science

9.4.1. Introduction

Science and craft design technology have a unique relationship: CDT activities can pose problems that can only be solved by scientific investigation, and CDT provides concrete, real world applications for abstract, theoretical scientific concepts. SEQ activities bridge both science and craft design technology, bringing together physical and control solutions to problems.

Links between SEQ activities and science are of two kinds: process based (experimenting, observing, testing, measuring, calibrating) and skill/knowledge based (physics, chemistry, biology, computing). What makes science into designing and making with SEQ is the design process; the application of scientific concepts to real world problems.

How you make use of these cross-curricula links will depend on your school and available (or planned) opportunities. In a primary school it may be possible to organize work so that science and technology

projects can proceed at the same time, or be combined, or a progression of ideas and concepts for each year could be planned. Some primary schools take a theme based approach to the curriculum: this allows SEQ to be integrated in a meaningful way, instead of being used out of context, for its own sake. Primary science provides many opportunities for control and technology based activities using SEQ.

At secondary school level, if the curriculum is more rigid, it will be important to ensure that children have encountered necessary concepts in their science work before using SEQ activities in other areas of the curriculum, such as craft design technology or computing/IT. Alternatively it may be possible to use SEQ in specific science activities: mechanics (forces, friction, levers), electricity, magnetism, or energy for example.

9.4.2. The scientific process

Science is about questioning and experimentation. Traditional science applies a 'scientific method':

Aims:
what you wanted to find out
Method:
how you set about finding it out
Results:
what actually happened
Conclusion:
what you discovered

Unfortunately it seems that this is not how real scientists work or make discoveries: it's more a rationalization of science after doing it. However, it can be used as a model which, if not adhered to too religiously, can provide a basis for science in the classroom (adapted from "Encouraging primary science"):

Define the problem:
list parts of the problem
identify constraints
hypothesize and predict

Identify activities which might solve the problem:
experiments
investigations
visits

Gather resources and collect information:
books, discussion

Discuss procedures:
fair tests
practical difficulties

Select promising solutions and try them:
observe, estimate & measure
communicate
record

Draw conclusions:
classify and look for patterns
collect sufficient evidence
don't make hasty judgments

revise hypothesis, make inferences

This should not be seen as a linear, sequential process. It is analogous to the design cycle (see "Craft design technology), or writing programs (see "Developing, testing & debugging"). Children should be expected (and encouraged) to use their observations to modify solutions and procedures.

It would be useful and informative to look through some of the activities, identifying those that contribute most clearly to an understanding of scientific processes.

9.4.3. Concepts

SEQ activities provide opportunities to learn about and use a wide range of scientific concepts. These can be classified along traditional lines: physics, chemistry, biology, computing; but the activities can also be used to blur and cross these subject boundaries, the appropriate technology (or mix of technologies) being selected according to the task.

Three areas involving scientific concepts, ideas, facts and understanding occur again and again in using SEQ: energy, mechanics, materials:

Energy:

Forms of energy:
- Mechanical
- Heat
- Light
- Sound
- Electrical
- Magnetic

Using energy (energy technology):
- Power sources
- Output devices (actuators): motors, lights, electro-magnets
- Input devices (transducers): sensors, switches
- Transmitting energy: mechanisms, electric circuits, optics

Mechanics:

Mechanisms:
- levers, pulleys, springs, cams, linkages

Friction (desirable and undesirable)

Forces:
- inertia
- gravity

Structures:
- Strength, stability, rigidity
- Form and function: links with the natural world

Materials:

Properties and limitations:
- strength, flexibility, weight (mass)
- electrical properties (resistance or conduction)

Processing and changing:
- tools, treatment, joining, finishing

You can use this either way: carry out the SEQ activities and discover the meaning of (for example) power, or carry out science experiments so you come to understand of power in order to use it when using SEQ to solve real world problems. A combination of the two would be ideal.

9.5. Craft design technology

9.5.1. Designing & making

Design technology has come to mean problem solving by a process variously described as the design cycle, design loop, design process, design line, etc. The idea is to break the problem solving process down into stages. Put simply:

Think
Design
Make
Test
Improve

Or more formally:

The design brief:
- setting the problem

Investigation and research:
- asking questions about the problem

Solutions:
- finding ideas and choosing the best one

Realization:
- making the selected project

Testing and evaluation:
- how well does the project solve the problem?

At each stage various design decisions will need to be made. As this design process is followed, pupils will use and enhance their technological skills. Technology includes understanding about and knowing how and when to use:

Materials:
- visual, tactile and physical properties
- how they can be joined, shaped and finished

Structures:
- form and function, forces
- static structures: strength and stability
- mechanisms: motion, gears, pulleys, levers

Energy:
- sources of energy
- storing, transferring and using energy

Control:
- mechanical, electrical and computer control

The art (or science) of designing and making is in skilfully combining different technologies in order to produce satisfactory solutions.

SEQ's contribution to this area of the curriculum is twofold: firstly it enables the technology of control to be explored and used in problem solving activities, and secondly it allows the design process to be made more explicit.

9.5.2. Exploring control technology

The technology of control has been difficult to integrate into the design process, and consequently has often been avoided. This has been partly due to the need to use computers and learn a programming language, combined with the need to know idiosyncratic, low level details about how the computer communicates with an interface, and hence to the real world. Frequently there are so many layers that the designer has had to work through that it has been difficult to see any relationship between real world events and the control program.

SEQ provides transparency, so control programs are seen as directly interacting with output and input devices. Connections to motors, lights and switches can be altered without complicated reprogramming. Control sequences can be built up and tested incrementally, or designed, implemented and tested.

Using SEQ to learn about control technology will give an understanding of motors, switches, sensors, and feedback, and how these devices can be controlled. The concepts of sequential instruction execution, repetition, testing input conditions and making decisions, and timing can all be developed. As these ideas are assimilated, children should begin to extend their options when it comes to generating alternative solutions to problems: control is another technology that can be used instead of, or in conjunction with, other technologies. The goal is that they should be able to make design decisions based on the advantages and disadvantages of using control technology in problem solving.

9.5.3. Enhancing the design process

This is made possible by children being able to go around the design cycle more often with control solutions to problems than with mechanical or electrical ones. It is relatively quick and easy to modify a control sequence, test the modification, evaluate it, then to form a better idea of requirements, or of an alternative way to solve the problem.

Changing the mechanical design after making and testing the first prototype is time consuming and sometimes frustrating, so the design cycle becomes a design line. The testing and evaluating parts of the design cycle are carried out once only, instead of being used to inform and modify earlier decisions.

This approach views writing a control program as a making activity. Indeed, it is instructive to look at the design cycle from two viewpoints:

Physical realization (how to make it):

Specification of requirements (analysis)
Exploring physical ideas (planning)

Implementing physical ideas (making)
Testing what's been made (evaluation)

Control realization (how to control it)

Specification of requirements (analysis)
Exploring control sequences (planning)
Implementing control sequences (programming)
Testing control sequences (debugging)

Note that these are not two independent activities: the viewpoints are interdependent and children should be encouraged to jump from one view to the other, in order that they may come to use physical and control technologies together. Design decisions made from a physical (how to make it) viewpoint will affect how it can be controlled, and control decisions will affect how it can be made. It is interesting and instructive to compare the traditional programming cycle (see "Developing, testing & debugging") with the design cycle.

The HMI report "Craft, design and technology from 5 to 16" provides a useful overview of aims in this area, together with objectives for pupils at age 11 and 16. Primary schools will find "Design & technology 5-12" an excellent introduction to the place of designing and making in the primary school, together with lots of ideas, activities, case studies, and ways of planning a school policy for design and technology. Other useful books for primary and secondary use are given in the bibliography.

9.6. Computing

9.6.1. Introduction

SEQ could be used in computer 'awareness' or information technology courses, to show the application of control technology and computers to real world problems, and to give practical experience of how these control problems are tackled. A robotics or control technology module of work could be made up by selecting appropriate introductory and advanced activities. If resources are limited, using SEQ could be part of a 'circus' of activities. Students should be aware of the advantages and limitations of using a simple sequential controller, and be able to judge when computer control would be needed (for example, when the control program becomes too complex).

SEQ could be used in computer studies courses, as an introductions to sequential program execution, control structures, repeat loops, and conditional statements (see "Appendix 4: Program structure & SEQ"), and also to the difficulty of following spaghetti loops. Designing, developing and testing control programs is a more concrete activity than programming a computer to add numbers or process text, and therefore provides an excellent introduction to program design techniques (see "Developing, designing & debugging").

At a more advanced level, SEQ's functions appear fairly simple: you press instruction keys, and it remembers your program. You give it commands, and it obeys them. But it would be a non-trivial project to write a program to emulate this action, because of the way in which SEQ allows key presses, and the lack of a return key! Internally, SEQ is a finite state machine, and it would be interesting to see if students could identify the (relatively) few states, then write a simulator. If you used Logo, the turtle could behave as the vehicle SEQ drives. Or if you used Control Logo, you could connect the same devices as used with SEQ, and the simulator would allow SEQ programs to be reproduced exactly!

One of the motivations behind developing SEQ was to give students a more accessible way into computer control, without the need to 'twiddle bits' at a low level. It was not intended to be used as an introduction to machine code programming! Clearly students would be expected to progress to the 'how it works' aspects of computer control at some stage, after they have seen and used it at a higher level.

9.7. In-service

9.7.1. Planning

If you are new to some of the ideas and techniques in this guide, you could find that you're not alone! Perhaps you could get together with a group of colleagues and work together. Such in-service activities could be school based, in which case they will also be useful in helping to formulate a school policy for control technology, or held outside the school. You may wish to involve teachers who you know are already doing things in this area of the curriculum; or advisors, advisory teachers, or even college lecturers.

Look at "Design & technology 5-12" by Pat Williams and David Jinks, an excellent and readable book which discusses the place of design technology in the primary curriculum, classroom resources, teaching techniques, and developing a design technology policy. It also contains a number of case studies, and each chapter ends with a number of points for discussion, so it could be easily adapted for use on an in-service course.

Also worth looking for is the teacher training pack "Posing and solving problems using control technology", written during MEP, jointly with British Schools Technology. It contains a complete in-service course, including multi-media materials, course reader, tutor guidelines, resource list, materials on classroom management, information, and workcards. All the materials can be freely copied for use in schools and colleges; the only catch is that it is probably out of print. Each LEA (try your CDT or Computer advisor) should have a copy, as should most teacher training institutions.

Case studies are always interesting: Roy Richardson's article (Primary control technology in action) in "Teaching & learning with robots" describes a curriculum development project involving in-service work with teachers who had not done any control before, and details their experiences. He says:

"Teachers need to leave a course with a number of simple problems which they themselves have solved and are confident enough to try out with a small group of children."

In-service providers could use or adapt SEQ's materials to form the basis for an introductory course in control technology for teachers. Allow plenty of time for practical work: only by doing designing and making will teachers have sufficient opportunities to understand the processes involved.

9.7.2. Action

Of course, it is not necessary to do all the activities. But teachers should be encouraged to do one or two of the introductory (Starter) activities, try at least one of the more advanced ones (Builder activities, using feedback), and then set themselves a small project (possibly one of the project ideas). Merely looking at or doing the "Techniques" and "Devices" activities is not to be recommended, as there will be no apparent purpose to the work.

As a last resort, if you can't find anyone else to work with, it is possible to use this guide (especially the pupil materials) as self-study activities.

A final word: get on with it! You do not have to know all the answers, because designing and making is a process, with no one right way or outcome. Use your professional skills as a teacher to judge the processes: it is the children's task to take responsibility for finding out about aspects of the technology.

10. Observing & recording progress

10.1. Progression

10.1.1. Approaches

Progression implies some sort of sequence in the learning experiences of children. It presupposes a linear model of learning (or at best spiral), contrary to experience. And yet if there is no perceived sense of direction and purpose in children's activities they will seem aimless.

Problem solving approaches to learning lend themselves to a personal sense of progression. Traditionally, the teacher presents or poses a problem, hopefully by demonstration rather than description. Observation leads children to investigate many aspects of the problem. Ideas are suggested, experiments tried, results discussed, further opportunities explored. The results of such a process can be represented by a curriculum web, but it should develop with the children's ideas and perceptions, rather than being drawn up by the teacher beforehand. There are no right solutions, but a sense of cooperative learning and discovery, where findings should be challenged and defended.

Other approaches to problem solving stress the top-down approach. The whole problem is broken up and structured into (hopefully) more manageable chunks. Death by a thousand worksheets (or workcards) is common (especially in secondary schools). Yet they provide a convenient way for teachers to apply top-down problem solving to classroom organization: divide and rule. Make this, try that, what if.... The "What if ... " questions have gained respectability as a powerful learning technique. But unfortunately it is too often the teacher that is posing these questions, instead of the child.

10.1.2. Problem solving

An alternative approach, and one I would wish to encourage, is to provide a problem solving environment, where children can start (like teachers) from existing knowledge, and take responsibility for, and control of, their own learning.

In this situation the role of the teacher is challenged and changes. Some of the more mundane roles include resource provider, assessor and enabler. But the teacher is in a position to monitor the child's learning. From observation (see "Observing children") you will see when a child does not have enough knowledge, or lacks some technique, that is necessary in order to solve the task in hand or proceed further with investigations. Then new concepts should be introduced.

For example, children will be able to solve simple control problems with sequential programs. They will need to know some of the six output instructions, and how to use SEQ commands. But there will come a time when there is a need for repetition, not just to save instructions, but to solve certain types of problem. The order in which these concepts will be required is not predetermined, but depends on the task.

You may also observe that children's interest is flagging ,and they feel nothing more can be achieved with the techniques they know. Then new ideas are needed to extend and challenge existing concepts. The "Techniques" and "Devices" activities have been written not to be used before allowing children to work on real control problems, but to provide support in child-centred work when they are needed.

At all times there is a tension between working at your own level, and extending and building on that ability/skill. Boredom results from too easy a task; frustration from too difficult a one.

Children should be encouraged to take responsibility for their process skills: to detect when they seem to be missing or misunderstanding a skill, concept or idea that is necessary in solving the current activity.

10.1.3. Control technology progression

A straightforward progression in control technology would seem to start with simple electrical circuits using switches to control output devices, then use sensors instead of the switches, then use SEQ as a programmable sequencer, and lastly to move on to programming a computer. However, it all depends on what you want the children to learn.

It is not necessary to use switch controllers before using SEQ, although they can be worthwhile. Indeed, the LEGO switch controller can and should be used with models to highlight the differences between manual (with human feedback) and program control (with and without sensor feedback). And there will come a time when SEQ seems inadequate for a programming task: then computer control should be introduced.

Eventually children should be able to select the appropriate tool (manual, electronic, SEQ, computer) for any particular task, according to the design needs. But it would be confusing to introduce them all at the same time.

Some of the SEQ activities lead naturally into work with a floor turtle, and hence into Logo. If Control Logo and a suitable interface is available, children can program the same devices that they use with SEQ, perhaps writing more sophisticated programs.

10.1.4. SEQ progression

An obvious progression of ideas is represented by the order of both "Techniques" and "Devices" activities. But again it is not necessary to use the materials in this order; they should be introduced as and when the problem currently being solved needs them.

However, in order to avoid frustration, careful structuring of children's problem solving activities will be needed. Assessment and observation will help you to determine children's competence. The Starter activities are not in any particular order, and children may wish to try most of them before trying more advanced, Builder activities. Or they may find a particular topic of special interest and wish to follow it through from a Starter activity, via more advanced ones, to one of the project ideas. Example topics are easy to think up (but these should not be regarded as rigid routes through the materials):

Robots:

Introduction		"Robots"
	&/or	"Oscillation & repetition"
Extension:		"Robot arms"
Project:		"Walking machines"
	&/or	"Animation"

Vehicles:

Introduction:		"Buggies -simple 2 motor vehicles"
Extension:		"Obstacle avoiding - bumpers on buggies"
	&/or	"Accurate buggies - using feedback"
	&/or	"Turtles"
Project:		"Changing gear"

Doors:

Introduction:		"Doors & barriers"
Extension:		"Automatic doors"
Project:		"Locks & safes"
	&/or	"Burglar alarms"

Pulses:

Introduction:	"Flashing lights"
	"Oscillation & repetition"
Extension:	"Code recognizers"
Project:	"Clocks and counters"

10.2. Observing children

10.2.1. Getting started

Why observe your children working? If you are introducing new materials and approaches, perhaps altering your teaching style, changing their curriculum, trying to encourage the development of different skills, or concerned about some aspect of their work, you should observe what your pupils are doing.

Decide on a specific aspect of the children's activity to focus on before you start. What do you want to find out about? It could be the quality and type of language, whether they discuss, argue or ignore each other when working in groups, how they go about developing and testing a control sequence, or some other part of the activity that you feel concerned about or interested in observing.

You may wish to ask a colleague to observe while you teach, but this isn't necessary: you can observe your own class.

Think about what you will record: speech, interaction, the programs they write, behaviour. Plan for short observation periods: use a notepad, maybe tell the children to get on without you. You may want to tell them what you are doing, to put them at ease, or involve them in the observation process.

Children can observe themselves: they can note down what they are doing, when they reach certain points in their activities, and so on. All you need to do is to provide them with some structure for their observations: a worksheet with questions for example.

Give them time. Don't fall prey to the teachers' temptation to 'help them along'. Allow them to make mistakes while you are observing, and watch how they cope with them. After an observation period, ask the children questions about what they were doing, thinking, and feeling. A group or class discussion afterwards can be useful (and is easier to tape record). Problem solving and thinking are time consuming processes.

10.2.2. Techniques

You can make use of a variety of observation techniques and aids: tape recording, video recording, drawing diagrams instead of writing, inventing your own coding scheme (or learning a published one). Tape recording isn't always easy, especially if there is a lot of background noise. Also it takes a long time to review afterwards. You could listen for (and maybe count) SEQ's beeps though!

Photographs make an excellent record of work, both for you and the children. Take them during work to record processes and intermediate stages, as well as of the finished product. Use them afterwards: get the children to tell you (or present to the class) what they did. Or you could use video: a bit intrusive during the thinking and experimental phases of work, but good for working models!

Make a plan of the classroom, and use it to show patterns of movement, or verbal interaction.

Always have a small notebook at the ready, and write down any 'telling phrases' you hear, showing sudden insights or misconceptions.

As one observer wrote:

> "It was during an observation session that I noticed one group moving their buggy back to the start position (of the maze they were working on) by giving SEQ simple instructions, instead of just lifting it up and putting it back at the start. They were feeling more 'in control', and gaining confidence in their new found skills."

After observing try to find time away from the class. Reflect on what you have just seen. Expand any cryptic notes while events are fresh in your mind. Later you can think about the implications of what you've found out, and what you are going to do about it.

10.3. Recording & assessment

10.3.1. Introduction

Recording and assessment will depend on your curriculum needs, aims and objectives, level and ability of pupils, and so on. It would be inappropriate and undesirable to record and assess a child's use of SEQ independently from the curriculum context. Here is a paraphrase of the conceptual framework for design and technology developed by the APU ("Understanding design and technology") that may provide a useful starting point.

10.3.2. Skills

These tend to overlap, and to follow each other cyclically and repeatedly. Primary children show evidence of these skills whilst engaged in activities that may not be instantly recognized as technological. They can be described as such when they rely on concepts drawn from the "knowledge" part of the framework, described below.

Each item can be read as "Is the pupil able to ....", or "Each child should be able to ....".

Investigation:

recognize design problems
perceive how well something meets stated needs
look for information and resources
carry out appropriate experiments and observations
judge reliability & relevance of information & resources
balance knowledge, analysis & judgment to reach conclusions

Invention:

initiate & develop ideas & images of proposed things or systems

manipulation of these ideas & images
think of, adapt & select alternative configurations
express & communicate ideas & images in various ways
examine coherence & integrity of ideas with respect to the problem

Implementation:

plan a practical activity & see it through
select appropriately from available resources
use tools, instruments, materials, components, & energy resources
monitor & measure effects of operations & control outcomes

Evaluation:

discern context & criteria of use of the design
select appropriate measures & means of measuring performance
judge merit of solution with respect to criteria
distinguish between & prioritize differing needs
appraise efficacy of design activities

10.3.3. Knowledge

There are three strands of technological concepts. Each item should be read as "Does the pupil know about ...." or "Each child should know about...."

Control:

static & dynamic systems
mechanical, electrical & computer control
open and closed loops (repetition & feedback)

Energy:

sources, costs & forms
methods for storage & transmission
efficiency & conversion

Materials:

sources & costs
properties & limitations
methods for processing, manipulating & connecting

10.3.4. Values

Criteria for decision making involves making value judgments. Phrases such as "right solution", "good design" or better approach" raise questions that can be answered in different ways depending on the criteria: technical, economic, aesthetic or moral. Children should be able to recognize different these different values. They usually overlap, and seldom appear in isolation. Each item should be read as "Does the pupil appreciate and apply concepts of ...." or "Each child should be able to appreciate and apply concepts of...."

Technical:

efficiency & comparison of input and output
robustness
flexibility & sensitivity to change
precision, fit & fitness to purpose
confidence & reliability

Economic:

distinction between user, intrinsic & exchange value

distinction between value, price & cost
comparison of marginal costs of products
effects of supply & demand on availability & price

Aesthetic:

structures, proportion & colours found in the natural & man-made world
importance of aesthetics in communication & self-expression
inter-relationship between workmanship, tools & products

Moral:

impact of technology on the natural environment
responsibility for the natural environment
inter-relationship between man-made world & religious, social, economic & political philosophies
needs of individuals in societies, & meeting them
importance of ethical values in evaluating the effects of technology

10.3.5. Profiles

You could use the skills, knowledge & values criteria above to make your own matrix for each unit of activity you wish to assess. Start each entry with "Can pupils..."

For comparison, the Design Council (see "Design and primary education") offers the following broad headings, and suggests that teachers write a list of questions based on these criteria that can form the basis of individual profiles for each child.

Sensibility

aesthetic & artistic
constructive critical analysis
appreciation of the material world around us

Observation and representation:

intelligent, imaginative & precise observation
drawing, talking, making & writing as means of representation
visualization & the use of signs & symbols

Knowledge and skills:

use of a range of materials
changing the form, shape, colour and texture of materials
safe & correct use of tools
ability to organize working environments
ability to seek & judge information & resources

Problem solving & creativity:

use creativity to tackle practical & functional problems
learn from failures by re-thinking & re-assessing designs
work collaboratively on problems, & evaluate solutions

Language:

be able to discuss and write about problem solving activities
have the vocabulary to discuss designing & making

Coherence:

use different sorts of knowledge & skills together in solving practical problems

Assessment of an activity or skill should focus on what the child does during the complete design processes, and not merely on the end product. Hence the importance of observation.

11.Hints & tips

11.1. When it doesn’t work

11.1.1. Problems?

General: try again; substitute an alternative part; check the wires; read the manual; ask someone. Most problems are caused by broken or loose wires and connections, weak or old batteries, or by trying to do something that SEQ doesn't understand.

If SEQ's green ON light does not come on:

is the battery connected the right way around?
is the battery flat?
are the leads from the battery to the plug ok?
is there a loose connection at the plug end?
is there a loose connection in the battery holder?

If SEQ's green light is on, but it doesn't respond:

disconnect and try again
have you turned the beep off (STOP-Pulse)?
is it obeying a program? (press STOP)
is it waiting for an instruction parameter? (press CE)

If SEQ behaves erratically:

is the battery weak? (try another)
are the leads from the battery to the plug loose?
is there a loose connection at the plug end?
is there a loose connection in the battery holder?
have the plugs become squashed or dirty?
are the output devices taking too much current? (disconnect them)

The outputs light up but nothing happens:

is anything plugged in?
are the leads loose or broken?
is there a loose connection at either end?
does the output device work on its own? (connect directly to a battery)

Only one motor works:

are you in independent motor control mode? (STOP-Forward)
check the leads and plugs (swap them)
check the motors (swap them)

The switch input light changes when the sensor does, but the program doesn't work:

are you in the right switch mode (STOP-JSW)

If SEQ refuses to obey commands:

is it on?
is is running a program? (GO, CE, CM, Test don't work: press STOP)
is it waiting for an instruction parameter? (GO, STOP, Test don't work: press CE)
is it empty? (GO, STOP, CE, Test won't work)
perhaps it is (if you pressed GO or Test) but your instructions don't do anything!

If SEQ makes a funny beep:

is the memory full? (40 instructions)
loose battery connections, leads or plugs?
look up the beep messages in the manual
is the battery weak? (try another)
are the output devices taking too much current? (disconnect them)

If SEQ refuse to obey instructions in your program:

are there any short circuits? (disconnect outputs one at a time)
are you using loops? (use domino cards to 'walk-through' your program: maybe it's stuck in a loop)
perhaps it never gets to them (eg 1 JMP 1)
are you in the right mode? (see STOP-modes in SEQ manual)

11.2. Safety

11.2.1. Rules

With a little thought, using SEQ will be an accident free experience (except for the control programs, perhaps). A few simple rules (which apply to any classroom based electrical or mechanical activity) should be made clear to the children:

1 Never connect anything to the mains.

2 If wires start to get hot, you have created a short circuit somewhere: disconnect the battery at once.

3 Before pressing GO or Test check that no one could be hurt by moving parts. Hair, fingers and eyes are most vulnerable.

4 Do not try to stop a rapidly moving part with your hand. You might get a friction burn or bruise. Use the STOP key, or disconnect the battery.

5 Do not leave SEQ controlling a model with no-one around, unless you are sure both your design and program are safe.

6 Take care with crocodile clips, sharp wires, etc.

7 Keep water away from SEQ, batteries, wires and sensors.

8 Be careful when using tools.

Teachers may wish to extend and adapt existing safety rules to includes these. Depending on the kit and materials your children are using, there may be additional safety considerations that should apply (especially with junk modelling).

Note that batteries, in particular rechargeable nicads, can supply lots of current. If you use leads to short them out they will melt the plastic insulation (which can cause a nasty burn) and possibly glow red-hot (perhaps causing a fire). You may wish to add some rules about how to avoid short-circuits. It's quite easy to accidentally connect LEGO leads so you get a short: maybe children should be told to connect everything up first, and then carefully connect the battery?

11.3. Wires

11.3.1. Loose connections

Poor connections can waste lots of time and be very frustrating for children. You might like to regularly check the condition of all the leads: look at the plug on each end, check for loose connections (don't pull too hard though). Frayed, cut or squashed insulation can weaken a lead, increasing it's resistance and eventually cause it to fail. Small (grub) screws can fall out of the plugs and get lost easily: check they are tight.

LEGO, Fischertechnik and other 2.5 mm plugs are made with four small cuts between the end prongs. These allow the plug to be squeezed together as it's inserted into a socket, gripping the socket so it doesn't fall out too easily. After repeated use the plug can become permanently squashed, fitting into a socket loosely, and easily dropping out. Take a Stanley knife (a blunt one is best!) and carefully push it down between the plug's segments, making a small gap. If you do this to the battery plugs, take care not to short them with the knife. Test the plug for fit.

You need long leads if you're using something that moves very far. But how do you stop them getting tangled up? Use spiral cable wrapping, cable ties, plastic bag ties, or multicore (4 or 8 core) cable. Short wires are useful if the model is static, and they are less messy. Perhaps a 'cable tidy' would be a good design project?

When experimenting with different input and output devices, leads with a plug for SEQ on one end and crocodile clips on the other would be useful. Or make up a set of 'jump' leads with crocodile clips on both ends. Crocodile clips allow you to make quick and reliable (but temporary) connections.

11.4. Batteries

11.4.1. Choice

Match battery voltage to output (and input) devices, or you'll either blow the device, or it won't work reliably. For example, a 12 volt battery will blow a 6 volt bulb, and blow or damage a 6 volt motor. In general, using a device at a higher than recommended voltage will drastically shorten its life, whilst a lower voltage will dramatically increase it, especially for light bulbs. You can use 3 volt motors with a 4.5 volt supply to SEQ, because the actual motor output voltage on SEQ is approximately 1.5 volts below battery voltage (see "Alternative output devices").

SEQ may get slightly warm when you use 9 to 12 volts. This is normal. Six volts is recommended as the best compromise between power, heat and range of devices that can be used. So use four 1.5 volt cells, or a 6 volt battery (eg PJ996).

You could use AA cells, but they can't supply much current, and won't last long. I recommend C types: they are a good compromise between size, weight, cost and available current. You could use D types, or lead-acid batteries, or even a mains power supply, if you were not concerned with size, safety or cost.

Developing and testing control sequences takes a lot of time. This may be expensive if you have to buy lots of batteries. I recommend that you use rechargeable nicad (nickel-cadmium: it's what they're made of) cells. A fully charged nicad cell produces 1.25 volts, falling to around 1.2 volts. Only when they are nearly exhausted does the voltage drop below 1.2 volts. C type nicads are rated at 2 amp-hours. How long they will last depends on how long they are connected to SEQ, and how much time they spend actually driving motors and other output devices.

As a guide, with no input or output devices, SEQ would last for between 15 and 20 hours with a fresh set of C type nicads. It would also beep once every 5 minutes to tell you it was still connected. If it was driving a 1 amp motor continually, it would last less than 2 hours. Practical experience has shown that SEQ can be used with a group of infants writing control programs using LEGO buggies (2 motors) for about 4 hours. This was spread over two successive afternoon sessions.

11.4.2. Maintenance

You should take note of the manufacturer’s instructions when recharging nicad cells, or you may shorten their life and charge capacity. In general, you should 'cycle' the cells: make sure they are exhausted before charging them for the time recommended (on the battery charger's instructions). Note that the capacity of a cell depends on temperature: winter temperatures mean reduced capacity (just like car batteries).

When batteries get low, SEQ accepts instructions, but when you press GO the motors don't work, and it will sometimes make strange beeping noises. This is because some devices (especially motors) draw a large current as they start up. This surge can cause the battery supply voltage to fall (especially if you're using low capacity batteries). If it drops below 4.5 volts SEQ may stop altogether, and when the voltage rises again (because SEQ's stopped), SEQ will assume it's just been connected and re-initialize itself. Thus you'll lose your program. Keep some fresh batteries at hand to avoid lost teaching time.

Take care that the batteries (especially nicads) are not shorted. This will damage the cells, but worse damage may be caused by the very large current. Wires can get hot and burn (see the section on safety above).

Battery holders are of variable quality: it is most frustrating for children to write a long and carefully planned control sequence, only to lose the program because the batteries moved in the holder and power was lost (briefly) from SEQ. When it's reconnected, it clears its memory. So try to ensure that each cell makes a good contact with the others, and with the battery holder contacts. They may need to be cleaned (use fine emery paper) from time to time.

11.5. Motors

11.5.1. Reversing problems

Alternately driving a motor in opposite directions may cause problems if the motor doesn't slow to a halt before the next instruction. If this happens, you'll have to insert a Pause instruction between each direction reversal, to allow the motor time to stop. Very severe cases of this behaviour can cause SEQ to become

confused, which either results in the program stopping, or occasionally clearing the program completely. This only occurs with poor quality, noisy motors.

11.6. Feedback

11.6.1. Thresholds

If the feedback sensor is providing a signal close to one of the thresholds (on or off), then it may drop below (or rise above) that threshold when an output device (motor, light) is turned on, due to a small voltage drop in SEQ. Adjust the sensor so the level is further away from this threshold. If it is a sensor used in the Switch input, put a pause in the program to allow the sensor to settle before testing it.

Don't be caught out be inverse switch mode (STOP-JSW): you could even press this by accident, as the JSW key is next to STOP.

11.7. Programming tips

11.7.1. Techniques

Have you noticed that SEQ allows you to change your mind about an instruction if you haven't given it a number (parameter)? You don't have to use CE unless you have pressed a number (or two) (although you can if you like).

Don't forget CM! If you don't press CM, new instructions are added after the old ones. If the program jumped back to the start, the new ones can never be obeyed.

If you're testing a program with a model or device, put JMP 1 as the last instruction (perhaps with a pause). This saves you from having to press GO lots of times.

Put beeps (Pulse) in the program so you know which instructions are being done and where the program has got to (useful in programs that repeat a lot, jump, or jump on a switch input).

Adjust the position of your model by adding one instruction to SEQ, pressing Test to make it do it (as often as needed), then CE to cancel it. Repeat this (with different instructions) as often as you like.

Finally: stop annoying people by pressing STOP-Pulse to turn the beeper off when your program's running ok, if it produces pulses for an output device. But don't be surprised by the missing beeps when you press STOP and start to program it again.

12. Conclusion

SEQ marks the start of an educational development process. I hope that SEQ and the associated materials will stimulate and enable curriculum change, leading to more problem solving, child-centred, activities in schools. I expect there will be in-service sessions when teachers experiment with and explore SEQ's potential; followed by classroom based observation of children using the materials and learning to be in control of their environment.

As teachers and children begin to use SEQ I hope and expect that they will discover other exciting and valuable activities, and it will be used in more imaginative ways than I ever thought of. If you find something exciting going on, please write to me. I will collect anything of interest together and produce an occasional newsletter for all SEQ users.

I hope too that SEQ will be used as a tool that allows learning to be made more explicit; computers can also be used in this way. Initial teacher training and in-service courses could use SEQ activities to illuminate learning processes: making abstract thinking concrete helps both the learner and the teacher.

Doubtless there will be successors to SEQ. An obvious one would doubtless be called PAR: users would be able to enter instructions to be obeyed in parallel. Or a larger version could be produced, more suitable for handicapped children. Maybe several could be linked together. Other, less obvious directions for development may be suggested by classroom experience or curriculum need: write if you have specific suggestions.

Finally, I hope you have as much fun using SEQ as I had in designing, making and testing it. I gained lots of pleasure from making robots walk (and watching them fall over), and buggies draw (not always what I intended): it's nice to have an excuse to play!

Send any comments, views or suggestions to:

Paul R. Spurgeon
~~ProCom,~~
~~5 Churchill Road,~~
~~Tavistock,~~
~~Devon PL19 9BU~~
paul.spurgeon@blueyonder.co.uk

13.Bibliography

Bigtrak plus by M.D. Meredith & B.I.Briggs, published by CET in 1982, ISBN0861840747

Report about schools' and children's experiences using Bigtrak for a wide variety of activities. Lots of transcriptions of children engaged in problem solving (useful if you intend to observe SEQ activities). Plenty of practical ideas too.

CDT for GCSE by Peter Toft, published by Heinemann in 1987, ISBN0435759906

GCSE textbook taking the design process approach, strongly visual, with lots of small activities. Useful short sections on structures and movement, but no control technology.

CDT projects and approaches by David Barlex and Richard Kimbell, published by MacMillan in 1986, ISBN0333366204

GCSE textbook, describing the design process from an industrial/commercial viewpoint. Includes practical construction activities and sample exam questions. Nothing on control technology.

Control technology: Pupils' Assignments by G.J.Fox and D.F.Marshall, published by Hodder & Stoughton in 1974, ISBN 0340364068

Short experiments illustrating control technology concepts. Suitable for 4th & 5th year secondary students. See Teachers' Handbook below.

Control technology: Teachers' Handbook by G.J.Fox and D.F.Marshall, published by Hodder & Stoughton in 1975, ISBN 034036405

Thorough introduction to control, including structures, gears, linear motion, & simple electrical circuits (transistors). Part of the Project Technology series. Rather didactic assignments, but suitable for background reading.

Craft, design and technology from 5 to 16 (Curriculum Matters 9) by DES/HMI, published by HMSO in 1987, ISBN0112706428

Intended to stimulate discussion about the craft, design and technology curriculum, this short booklet summarizes current thinking about the aims and objectives of this area of the curriculum. Includes sections on teaching styles and learning processes, the school environment, resources, progression and assessment.

Design and primary education by the Design Council's Primary Education Working Party, published by The Design Council in 1987, ISBN0850722128

An excellent overview of design related activities in primary schools. Reviews current practice, discusses the place of design activities in the primary curriculum, and suggests ways in which they can be developed and assessed. A very useful source of checklists and detailed descriptions.

Design and technology 5-12 by Pat Williams and David Jinks, published by Falmer in 1985, ISBN1850000492

> Very readable book intended for teachers, in-service providers and curriculum planners. Discusses the place of design technology in the primary curriculum, and gives classroom resources, teaching techniques, case studies, and an approach to developing a school design technology policy.

Designing and making by Louis Brough, published by The East Midlands and Yorkshire Forum of Advisers in Craft, Design and Technology, in 1985

> Sub-titled "Learning through craft, design, technology" this short but stimulating book covers both primary and secondary activities. Based around photographs of children's work, it outlines the structure of this area of the curriculum, from early experiences to GCSE level. A good source of ideas too.

Encouraging primary science by George Raper and John Stringer, published by Cassel in 1987, ISBN0304313726

> A complete manual for the primary science curriculum. Starts with an introduction to children's thinking and primary science, discusses ways of developing primary science and a school policy, and includes activities, resources, notes on classroom organization, and safety. Could provide material for an in-service course. Includes sections on the relationship between CDT and primary science.

Equal opportunities in craft, design and technology, published by the Equal Opportunities Commission in 1983, ISBN0905829719

> Critically examines sex discrimination in secondary school CDT, and places this in a contrasting social context. Describes the impact of attitudes, organization, teaching methods, and the curriculum. Checklists and recommendations. Sources of information and resources.

Exploring primary science & technology with microcomputers edited by Jan Stewart, published by CET for MEP as Readers 5 in 1985, ISBN0861841360

> An anthology of articles, giving an historical perspective to computing and control technology in the primary classroom. Includes:
>
> "Teaching electronics to primary school children"
> "Why include control technology in the primary curriculum?"
> "Electronic building bricks"
> "Of mice, minibeasts and micros"

Hands on: hands off by Christopher Schenk, published by A & C Black in 1986, ISBN0713627077

> Visually exciting, full of ideas for on- and off- computer activities with children. Includes Logo and turtle activities, information handling, and other computer applications.

Ideas for egg races & other practical problem solving activities, published by the British Association for the Advancement of Science, ISBN0900902078

Over seventy tried and tested 'egg race' ideas. Objectives are followed by equipment, rules, and judging criteria. The ideas could be used for projects too.

Infant and First Schools - a teacher training pack, published by CET for MEP.

The course reader includes an article by Linda Spear describing children's experiences using Bigtrak, together with curriculum links, initial activities, the teacher's role, and record keeping. There is also a set of 16 Bigtrak pupil activity cards, and teacher's notes suggesting activities with Bigtrak.

Language & learning: an interactional perspective, edited by Gordon Wells & John Nicholls, published by Falmer in 1985, ISBN 1-85000-028-X

Fascinating collection of papers describing the role of language in various aspects of the educational process. The papers are practical rather than theoretical, and contain ideas that can be adapted and used in the classroom. Terry Phillips' paper, "Beyond lip-service: discourse development after the age of nine" is especially relevant to teachers wishing to observe and encourage higher level thinking skills.

Microworlds: adventures with Logo, by Richard Noss, Clare Smallman & Michael Thorne, published by Hutchinson in 1985, ISBN 0091611113.

An exciting, fascinating and visually stimulating book packed with plenty of Logo activities, starting points and ideas. About half the book devoted to programming with a turtle.

Mindstorms: children, computers and powerful ideas, by Seymour Papert, published by Harvester Press in 1980, ISBN0855271639

The classic Logo text: useful to get a feel for the vision behind Logo, but doesn't teach Logo or give classroom ideas. Looks at children using turtles, and includes some reflections on the concepts and metaphors of gears.

Posing and solving problems using control technology - a teacher training pack, published by CET for MEP & BST in 1985.

This pack contains a wide range of in-service materials, including pupil activities, resource lists, information about materials & kits, articles, a detailed example of an in-service course, and a booklet on classroom management. All the materials may be freely copied for use in schools and colleges, and should be available from LEA's or Teacher Training Institutions.

Programmable systems: pack by BST/MEP, published by LEGO in 1986, ISBN1869953002

Materials specifically developed for use with the LEGO Technic 1090 kit, a BBC computer, and the LEGO interface. Uses a program called "Lines". Some of the materials can be adapted for use with SEQ, allowing a SEQ to computer progression. The pack contains:

- Classroom materials
- Teachers' materials
- Resources booklet

Science 5-13, published by Macdonald in 1972.

There are many useful books in this range, including:
Science, models and toys (Stage 3)
Science from toys (Stages 1 and 2)
Structures and forces (Stages 1 and 2)
Time (Stages 1 and 2)

Starting CDT by Keith Good, published by Heinemann in 1987, ISBN0435754009

Secondary school textbook. Very visual, with clear text. Takes a design process approach, and includes lots of practical ideas and activities. Strong on construction, but very little on control technology.

Teaching & learning with robots edited by Colin Terry & Peter Thomas, published by Croom Helm in 1988, ISBN0709943180

An up-to-date anthology of articles by many of the leading control technology educationalists. Contributions include the use of control technology in primary and secondary schools, examples of current practice, together with glances at the future of robotics in education.

Technology in schools by John Cave, published by Routledge in 1986, ISBN0710207328

Sub-titled "A handbook of practical approaches and ideas", this book contains detailed descriptions of control technology (pneumatics, hydraulics & electronics), manufacturing techniques, product modelling, and microelectronics. Good background reading.

The robot book by Richard Pawson, published by Winward in 1985, ISBN0711204144

Excellent, visual and stimulating book by an acknowledged expert in the field. In two sections: the first describing various aspects of robotics; and the second giving lots of practical robot projects, mostly using Technical LEGO or Fischertechnik. Each project has detailed construction photographs and drawing, plus a description of the design. Although short BASIC programs are used to control the robot designs, SEQ could also be used.

The second self: computers and the human spirit by Sherry Turkle, published by Granada in 1984, ISBN0246125683

A readable and fascinating account of research into children's perceptions of computers, including computer 'toys', robots and turtles. Shows how computer technology is altering the way in which we think about ourselves, our relationships with other humans, and our feelings about mind and machines. Lots of observations and interviews with children.

Understanding design and technology discussion document by APU, published by APU/DES in 1982.

This paper analyses the subject under three headings: Skills (investigation, invention, implementation, evaluation), Knowledge (control, energy, materials), and Values (technical, economic, aesthetic, moral).

14. Glossary

"When I use a word," Humpty Dumpty said, in rather a scornful tone, "it means just what I choose it to mean - neither more nor less."

(Lewis Carroll, "Through the Looking-Glass")

AC - alternating current. Electricity without a fixed polarity: the direction of electricity flow changes (often 50 times a second). Contrast with DC.

Actuator - transducer that converts electric energy into movement, usually linear. A solenoid is an actuator.

Algorithm - a structured description, usually sequential, of how to solve a problem or task. The plan of action that is to be achieved by programming.

Bidirectional - literally 'moving both ways'. SEQ's motor outputs are bidirectional, because the output voltage can be reversed by using appropriate instructions (eg Forwards & Backwards). Thus a motor can be driven both clockwise and anti-clockwise, depending on the program. Unidirectional outputs do not allow this.

Bug - an unexpected and undesirable instruction or instruction sequence, causing the program not to behave as was intended. Some bugs can be considered features if they are not very undesirable, and intentions modified. Bugs are removed by debugging.

Buggy - vehicle with two main wheels, independently driven by two motors, allowing it to be steered by driving the wheels in opposite directions. Accurate buggies are sometimes called turtles.

Command - an imperative, meaning 'do some action immediately'. There are five SEQ commands (GO, STOP, CM, CE, Test), all of which have an immediate action. Unlike instructions, commands are obeyed immediately.

Communication - transfer of information from one place to another. It is expected that the information receiver must have some understanding of the transmitters' intention for communication to have taken place.

Computer - device (usually but not necessarily electrical) with inputs and outputs, containing a stored program of instructions which are obeyed either sequentially (eg SEQ) or in parallel.

Concept - abstraction about perception. Often used as a convenient term denoting a handle (name) representing a particular way of thinking about classes of things. Concepts can be considered as facts together with ideas, combined by understanding.

Condition - something that can be tested. SEQ tests the condition of sensors attached to the switch input when it obeys the JSW instruction. Conditions are used to control loops: either when they should stop, or when then should continue.

Conductor - material allowing electricity to flow through itself. All metals are conductors. Contrast with insulator.

Control - Vague term to describe one thing influencing the behaviour of another, usually consciously, deliberately or causally. Children control SEQ (by programming it with instructions, and giving it commands), SEQ controls whatever it's outputs are connected to (by obeying its program), SEQ's inputs

control how it behaves (by using feedback). SEQ understands four control instructions: Pause, Repeat, Jump, and Jump-on-switch.

Control technology - designing and making systems, including input and output devices, their connections to a controller (computer, SEQ), and writing the control program.

Craft - the skills needed to make things. Craft in schools once meant learning and practicing skills without any purpose, for their own sake. A more enlightened view would be to include craft as one of the skills needed in the design and making process.

Current - a flow of electricity, measured in amps. Current can be converted by output devices into other forms of energy, for example, movement, sound or light.

DC - direct current. Electricity with a fixed polarity and flow direction. A battery always provides DC. Contrast with AC.

Debugging - the act of removing bugs. Less prosaically, looking at your mistakes in order to correct them. Debugging inevitably means lots of reflective thought and discussion about you intentions (the design), the way in which you planned to achieve the desired result (the algorithm), the instructions you used (the program), and the observed results. Such focused thinking combines several high level skills: analysis, observation, evaluation, synthesis, logical argument. Some claim that it is only by focusing on and learning from our mistakes that we progress: success is satisfying, but not educationally profitable if achieved immediately. Contrast with hacking.

Designing - a way of problem solving. Matching a set of requirements and a way of meeting them, or an acceptable compromise. Various educationalists see the design process in terms of a cycle or spiral: problem clarification & definition, generation of ideas and potential solutions, making or experimenting with selected solutions, testing these solutions, revising original ideas.

Device - vague term including components or parts used to convert one form of energy into another (more properly called transducers). For example, a motor, switch, light or buzzer.

Electricity - form of energy, provided by conversion of other energy forms (batteries convert chemical energy), and realized by the flow of electrons in materials. The potential to cause this flow is called a voltage. The actual flow is called a current.

Energy - examples are electrical, mechanical (kinetic, potential), heat, light, sound, magnetic, chemical, atomic (nuclear).

Fact - item of information generally agreed to be true. Understanding facts means applying concepts.

Feedback -.using a resultant event or action to control the cause of the event or action. Thus SEQ can use feedback from opto-sensors attached to motors to sense their movement and count how much they have rotated by, and stop the motors when sufficient rotation has been detected. In the absence of feedback, SEQ is blind to the world, and just assumes everything is working ok: it just turns the motor on for a programmed length of time.

Gear - a toothed wheel that can mesh with other gears, often with a different diameter and/or number of teeth. Gears can be used to change rotational speed, direction, turning power (torque), and type of motion (rotary to/from linear). Gears using chains are really examples of pulleys.

Hacking - changing a program randomly, with little thought, in the hope of fixing bugs. Contrast with debugging.

Idea - hypothesis, belief or way of thinking about something. Agreed or well thought out ideas about facts are concepts.

Information - abstract characterization of a physical message or signal, denoting its meaning. Communication involves transferring information from one place to another. Sensors communicate with SEQ (in one direction), sending electric signals carrying information about the physical environment.

Input - a socket on SEQ were a switch or other input device can be connected. SEQ uses inputs to control how its behaviour by sensing events in the real world. There are two motor feedback inputs, used by the Forward, Backward, Left and Right instructions; and two switch inputs, one used by the Jump-on-switch instruction, and one used as an emergency stop.

Instruction - something to be followed and obeyed. When writing instructions it is not expected that they will be obeyed immediately, but will be used at some time in the future. SEQ understands ten different instructions; six output instructions, and four control instructions. SEQ instructions are held in its memory, and obeyed sequentially when the GO command is given.

Insulator - material that does not allow electricity to flow through itself. Plastic, glass and rubber are examples of insulators. Contrast with conductors.

Interface - the boundary between two different sorts of things. In control technology, refers to electronics allowing different sorts of devices to communicate with each other.

Interrupt - an event or signal that gets immediate attention. The STOP input on SEQ is an interrupt input. Contrast with polling.

Iteration - Doing something more than once in order to achieve some result. In computer programs, iteration may be achieved looping. Also applied to the designing and making process, in physical construction or programming, when going around the design cycle.

Logo - a programming language widely used in education, invented by Seymour Papert. Part of the language can use a screen or floor turtle to draw shapes.

Loop - sequence of instructions that is repeated by jumping back to the beginning again. Loops can repeat a fixed number of times (twice using x2); repeat forever, using JMP; or repeat until a condition is met, using JSW. Also called iteration.

Mechanism - an abstract description or name of a mechanical way of achieving a specific task. Mechanisms are constructed by using levers, gears, cams, and pulleys. For example, the rack and pinion is a mechanism that is constructed with gears. The mechanical equivalent of an algorithm.

Mode - a way of working. Used to denote a specific way of working when more than one is possible. Thus a Forward instruction can be interpreted by SEQ in two different ways, depending on the current mode: normal motor control mode, or independent motor control mode.

Model - a representation, usually scaled down, of a design. Construction of a prototype of a machine, using the same or similar mechanisms; sometimes to test a design.

Motor - device that converts electrical energy into mechanical energy, in the form of rotational motion. The direction of rotation usually depends on the polarity of the power source. Motors usually turn at high speed, and gears are used to reduce the speed and increase power.

Output - a socket on SEQ which can be programmed to produce electricity, thus any driving motors, lights, etc. that are connected to the output. There are two bidirectional and two unidirectional outputs on SEQ. Six output instructions control SEQ's output sockets.

Parallel (1) - method of connecting electrical devices. All leads from one side of each device are connected together, and all leads from the other side are also connected; the devices are connected side by side. All devices experience the same voltage, but the current flow is shared between them in inverse proportion to their resistance. Contrast with serial.

Parallel (2) -method of computation when a computer obeys more than one instruction at a time (SEQ is not a parallel computer). Contrast with sequential (SEQ is).

Parameter - value given to an instruction. A number from 1 to 99 which determines what the instruction does when it is obeyed.

Polarity - positive or negative. Used to describe the direction in which DC electricity will flow. Electricity flows from negative to positive. Reversing the polarity can damage some devices. Reversing the polarity to a motor makes it change direction, but has no effect on a light bulb.

Polling - examining the status of an input frequently in order to discover if it has changed. You can write a JSW loop (for example, 1 JSW 1) to poll the input. Contrast with an interrupt.

Problem solving - using intelligence to achieve a non-trivial goal.

Process (1) - a process produces outputs according to the values of its inputs. In computing, processes take place when a program is applied to inputs to produce outputs.

Process (2) - in learning, process is the action of doing things (eg designing & making), rather than the product of doing those things (eg a working model). Process skills enable learners to become aware of, and responsible for, their own learning and actions.

Product - the result of doing something. Too often assessment of classroom activities focuses on the end product, rather than on the way in which this was achieved (ie the processes).

Program - a sequence of instructions, written to implement an algorithm.

Programming - the act of writing a program, usually taken to include the designing, writing, testing, debugging, and documenting a program.

Pulley - a wheel (usually with a groove around the edge), used with an elastic band, string or belt, to transmit rotational motion. Often used with other pulleys (with the same or different diameters) to make mechanisms, instead of using gears.

Resistance - electrical property of materials that impede the flow of electricity (current). Measured in ohms.

Robot - machine that can be programmed to perform tasks, usually those previously carried out by humans.

Sensor - device that detects changes in the physical world and convert them to electrical ones. Sensors can be attached to SEQ's inputs, providing feedback. Typical sensors include those that detect light (opto-sensors) and mechanical movement (switches). Some sensors use transducers.

Sequence - one thing after another. A sequence of instructions will be obeyed by SEQ serially, in the same order they were entered, unless a Jump or Repeat instruction is encountered.

Sequential - method of computation when a computer obeys instructions one at a time (SEQ is sequential). Contrast with parallel.

Serial - method of connecting electrical devices. One wire from each device is connected to the next, daisy chain fashion. Each device experiences the same current, but the overall voltage is shared between each device in proportion to its resistance. Contrast with parallel.

Signal - a physical change or changes that is taken to represent information. Sensors send electrical signals in response to physical events. Computers (eg SEQ) sends electric signals to output devices.

Switch - device that converts energy (usually mechanical, but other energy forms can also be converted) into electricity. Switches usually have two states: an off, high resistance state and an on, low resistance one. Commonly made by moving metal parts together to complete a circuit.

System - in control technology this describes an arrangement of input (sensing) & output (action) devices inter-connected by decision making (processing) components, usually a computer (eg SEQ).

Technology - widely used and misunderstood term, emotionally loaded, denoting things to do with using tools, science and materials to achieve various ends. Used in Craft Design Technology taken narrowly to mean the application of the principles of physics (energy, especially mechanical and electrical), chemistry (materials), biology (form & function) or computers (control) to the problem solving process (designing).

Transducer - device converting energy or energy changes into changes in an electrical property, for example resistance or voltage. Attached to an input some transducers are used as sensors.

Turtle - an accurate, computer controlled vehicle which can be programmed to move forwards and backwards, or turn to the left or right, and leave a trail of its path using a pen, which can be raised and lowered. Invented by Seymour Papert as part of the Logo environment.

Understanding - a way of thinking about facts using ideas or concepts, together with the ability to do problem solving.

Unidirectional - literally 'moving one way'. SEQ's Pulse and Out outputs are unidirectional, because the output voltage cannot be reversed. Thus a motor connected to these outputs can only be driven in one direction, either clockwise or anti-clockwise, depending on which way around it is connected. The motor outputs on SEQ are Bidirectional.

Value - parameter given to an instruction. A number from 1 to 99. Motor output instructions use the value alter the duration of the output. The Pulse instruction uses the value as the number of pulses to produce.

Vehicle -mechanism, usually with wheels, that can move along the ground and transport things. Cars, bicycles, buggies and turtles, are all vehicles.

Voltage - a potential for producing a flow of electricity (current), measured in volts. A voltage is produced by a power source, by converting other forms of energy into electricity.

15. Appendices

15.1. Appendix 1: SEQ specifications

15.1.1. Physical

Size:	62 by 112 by 28 mm
Weight:	under 150 grams
Sockets:	9 pairs of 2.5 mm diameter sockets, with centre hole, LEGO plug compatible
Keyboard:	25 keys under polycarbonate membrane
Indicators:	1 green led for battery inputs, 2 red/green leds for motor outputs, 2 red leds for auxiliary outputs, 4 yellow leds for input sockets
Beeper:	built in, giving audible indication of each accepted instruction & command

15.1.2. Software

Commands:

Go, Stop, Test, Clear Memory, Clear Error

Instructions:

Motor control:	Forward, Backward, Left, Right
Auxiliary output:	Pulse, Out
Control:	Pause Repeat, Jump, Jump on Switch

Program memory:

40 instructions maximum

Instruction parameters:

every instruction takes one parameter, value 0-99

15.1.3. Electrical

Power requirements:

Voltage: 4.5 to 12 volts (reverse polarity protected)
Current: 100 mA quiescent (no sensors attached),
50 mA (max.) per sensor,
1A (max.) per output

Motor outputs:

Two bidirectional outputs
Both pins driven, in a diode bridge between
power supply positive and ground.
Maximum current: 1A (2A peak), short circuit protected
Maximum voltage: 1.5 volts below power supply

Other outputs:

Two unidirectional auxiliary outputs
Top pin connected to power supply positive
Bottom pin driven
Maximum current: 1A (2A peak), short circuit protected
Maximum voltage: 0.9 volts below power supply

Inputs:

Four switch or sensor inputs
Top pin connected to 5 volts positive
Bottom pin input, tied to ground with 100 ohms
Maximum current: 50 mA (short circuit)
Typical on resistance: 2000 ohms
Typical off resistance: 2400 ohms
Typical on voltage: 240 mV
Typical off voltage: 200 mV

Motor feedback inputs:

Minimum pulse rate: 10 Hz (approx.)
75 rpm with 8 segment shaft encoder
Maximum pulse rate: 100 Hz (approx.)
1500 rpm with 4 segment shaft encoder

15.2. Appendix 2: SEQ features

Indicators: All inputs and outputs have indicators to show exactly what is going on. If an input sensor doesn't work, the light can be used to test it. If a motor fails to move, the power and direction can still be seen on the appropriate light.

Sound: each key press that SEQ accepts is acknowledged by an audible beep. The pitch of the beep is different for each key. Additionally, the Pulse output is accompanied by a distinctive beep, allowing rhythmic sequences to be composed.

Emergency stop: if something goes wrong, the STOP button can always be used to immediately stop all motors and outputs. A switch can also be attached to the Stop input, so the model can stop itself.

Overlays: children can write on their own overlays so that the buttons are labelled with exactly what they want to call the action each one controls. This also helps them develop a sense of the importance of a meaningful name or symbol for each action. Both completely blank, partial and complete overlays are available.

Consistency: every instruction takes a number as input (parameter), with an intuitively clear meaning. For example:

Forward 50
Pulse 7
Out 15
Jump 1
Pause 2

All numbers are from 1 to 99. Every SEQ command is immediately obeyed, and therefore have no inputs:

Go
Stop
Test
Clear error
Clear memory

Familiarity: it is possible to use SEQ as a Bigtrak substitute by building a buggy and using an appropriate overlay. Existing Bigtrak workcards and activities can be used.

Safe & reliable: Safe low voltages only: no mains electricity is needed. SEQ is internally protected against accidental short circuits and reverse polarity connection, so incorrect wiring will not do any harm.

Compatibility: LEGO plugs allow models to be quickly connected and used. If you are not using LEGO, suitable plugs can be easily attached to leads, allowing any construction system to be used with SEQ. A wide range of motors, output devices and sensors can be attached to SEQ without any additional electronics.

Small & lightweight: Working models often move; SEQ is small and light enough to be attached to the model, avoiding trailing wires that become tangled & disconnected. If the batteries are also carried (or towed) by the model, it can be completely self-contained.

External batteries: Gives additional flexibility, allowing you to connect small, large or rechargeable batteries as required. Any voltage from 4.5 to 12 volts is ok, so existing battery packs and holders can be used.

Large program memory: Up to 40 instructions can be stored, allowing complex sequences to be built up.

Accurate: Precision electronics gives exact control and excellent repeatability. If feedback from driven shafts is used SEQ automatically ensures that both shafts rotate by the same amount, regardless of mechanical resistance. This allows accurate movements to be programmed, just like a floor turtle.

Positive keyboard: Each key has a click action switch beneath a durable plastic sheet, giving positive tactile feedback each time a key is pressed.

15.3. Appendix 3: SEQ and Bigtrak differences

Some of the more important differences between SEQ and Bigtrak are given below, in order that existing Bigtrak activities can be more easily adapted for use with SEQ.

Changes:

Bigtrak: Test key which runs a test program
SEQ: no test program

Bigtrak: 16 instructions maximum
SEQ: 40 instructions maximum

Bigtrak: repeat instruction (x2) can only be used once
SEQ: repeat can be used as often as required

Bigtrak: OUT instruction has a fixed duration (3-4 secs)
SEQ: OUT takes a parameter (0.1 to 9.9 secs)

Additions:

Bigtrak:
SEQ: two extra keys and instructions, JMP and JSW

Bigtrak:
SEQ: STOP key added to abort an instruction sequence

Name changes:

Bigtrak: Fire key & instruction
SEQ: Pulse key & instruction

Bigtrak: tick key to try the last instruction called Check
SEQ: tick key called Test

15.4. Appendix 4: SEQ & program structure

15.4.1. Sequences

SEQ sequences should be regarded as basic blocks of instructions, each to be obeyed serially, one after the other. This is exactly the same as traditional programming languages, but not applicable to non-deterministic, rule based (such as Prolog) or concurrent, parallel processing languages (for example, Nimbus Logo with multiple turtles).

One obvious shortcoming of SEQ is the lack of procedures, due partly to the difficulty of naming and examining them without a full keyboard and screen. The CALL and RETURN structure does not exist. Two jump instructions could be used, although the sub-procedure can be used only once:

```
1       Forward 1
2       JMP 10              Call sub-procedure
3       Backward 1
4       JMP 1

10      Pulse 3
11      Out 2
12      JMP 3       Return to next instruction
```

The effect is more like threaded code (as used in Forth). This does not seem clearer than:

```
        Forward 1
        Pulse 3
        Out 2
        Backward 1
        JMP 1
```

JMP and JSW can be regarded as GOTO's in SEQ programming (and therefore 'bad'), but a little discipline or self-discovery can enable simple, understandable control structures to be used, while avoiding the worse excesses of spaghetti programming. These control structures are all characterized by one entry and one exit point, and familiarity with their use should prove an advantage when progressing to other programming languages.

15.4.2. Conditionals

IF condition THEN block

SEQ has only one condition, the switch input, which simplifies conditional and loop statements. The IF..THEN is easily implemented with a conditional forward jump around the block. Note that the condition is the opposite to that in the IF..THEN statement, as SEQ jumps when the switch is in a TRUE state, and therefore doesn't execute the block:

```
IF not switch THEN
        BEGIN
        Pulse 3
        Out 2
        END
```

```
1       JSW    4
2       Pulse 3
3       Out 2
```

IF condition THEN block1 ELSE block2

Implemented with a combination of two forward jump instructions, one conditional on the switch, and the other unconditional. Note again that the condition is opposite to the statement, and the blocks are swapped:

```
IF switch THEN
        BEGIN
        Pulse 3
        Out 2
        END
ELSE
        BEGIN
        Pulse 2
        Out 3
        END
```

```
1       JSW 5
2       Pulse 2
3       Out 3
4       JMP 7
5       Pulse 3
6       Out 2
```

15.4.3. Loops

REPEAT block FOREVER

One of the simplest loops, with a jump back to the start at the end of the program. Like tail recursion without parameters:

```
REPEAT
        Pulse 3
        Out 2
FOREVER
```

```
1       Pulse 3
2       Out 2
3       JMP 1
```

REPEAT block UNTIL condition:

Obeys a sequence (block) of instruction first, then tests at the end of the sequence, and either jumps back to the beginning, or exits the loop completely. The test is at the end of the sequence:

```
REPEAT
        Pulse 3
        Out2
```

UNTIL not switch

```
1       Pulse 3
2       Out 2
3       JSW 1
```

Unfortunately the Logo repeat loop is actually a for...next structure, as the number of loop iterations is known before the loop starts.

WHILE condition DO block

Tests the condition first, then either obeys the sequence (block) of instructions and loops back to the test, or exists the loop completely. The test is at the beginning of the sequence:

```
WHILE not switch DO
        BEGIN
        Pulse 3
        Out 2
        END

1       JSW 5
2       Pulse 3
3       Out 2
4       JMP 1
```

LOOP ..... EXIT IF condition ..... ENDLOOP

This most general form of loop has (any number of) exit conditions. Instructions in the loop are obeyed sequentially, including any exit conditions. If any of these conditions are met, the loop terminates, otherwise it continues:

```
LOOP
        Pulse 3
        EXIT IF switch
        Out 2
ENDLOOP

1       Pulse 3
2       JSW 5
3       Out 2
4       JMP 1
```

FOR Start TO Finish DO ....

The traditional FOR..NEXT loop (and Logo's REPEAT loop) cannot be emulated by SEQ, because it has no variables or counters. It would have been nice to let users press a repeat key, then type the number of times to repeat, but this would also involve some way of specifying the end of the block to repeat. Logo uses square brackets, Dart uses END, etc. Better to allow naming of blocks (as sub-procedures), and then to repeat them.

However, there is a fixed, repeat twice loop. Although SEQ doesn't impose any constraints on how you use this loop, you are advised against jumping out of it (unless you are going to jump back again). SEQ

implements x2 by maintaining a repeat flag for each x2 instruction. If you jump out of a x2 loop this flag will not be properly reset, and so the next time the instruction is obeyed it will not repeat. The structure is:

```
REPEAT 2
        BEGIN
        Pulse 3
        Out 2
        END

1       Pulse 3
2       Out 2
3       x2 2
```

All loops (and conditionals) can be nested to any depth, until you run out of room for instructions, but can be hard to follow unless you plan them carefully. Some may like to use flow charts, others may prefer pseudo-code describing the intended flow of control. For example, do the SEQ instructions below correspond to the pseudo-code?

Pseudo-code:

```
REPEAT
        REPEAT
                Forward 1
                WHILE switch DO
                        BEGIN
                        REPEAT 2
                                REPEAT 2
                                        BEGIN
                                        Pulse 1
                                        Pause 5
                                        END
                        Out 2
                        END
                Backward 1
                Pause 20
        UNTIL switch
        Right 15
FOREVER
```

SEQ equivalent program:

```
1                       Forward 1
2                       JSW 9
3                                       Pulse 1
4                                       Pause 5
5                               x2 2
6                       x2 3
7                       Out 2
8                       JMP 2
9                       Backward 1
10                      Pause 20
11              JSW 1
```

```
12              Right 15
13      JMP 1
```

Try it (it uses inverse switch mode). Does SEQ's behaviour match your expectations? Notice the use of indentation to represent loop levels. Perhaps this is an example of the limitations of SEQ control. Would it be more readable and easier to understand in Logo?

Busy waiting

Degenerate forms of the REPEAT..UNTIL and WHILE..DO loops can be used to form busy waiting loops (sometimes referred to as polling loops). These are useful when you want the program to wait until a condition exists, or ceases to exist. The repeat loop waits until the switch goes off:

```
REPEAT
        busy wait
UNTIL not switch
```

```
1       JSW 1
```

and the while loop waits until the switch goes on:

```
WHILE not switch DO busy wait
```

```
1       JSW 3
2       JMP 1
```

15.5. Appendix 5: Teaching material masters

The following pages are masters that you may photocopy and use to make additional copies of the teaching materials supplied with the SEQ Pack. They consist of:

Instruction & command summary posters. Six A4 display posters, showing selected SEQ instructions or commands:

1: SEQ Commands
Shows all six commands (Go, Stop, Test, CM, CE).

2: SEQ Instructions: actions (normal motor control)
Examples of each of the six output instructions
(Forward, Backward, Left, Right, Pulse & Out)

3: SEQ Instructions: control
Example of the four control instructions
(Pause, x2, JMP and JSW)

4: SEQ Instructions
Examples of all ten instructions

5: SEQ Commands & Modes
All six commands, & Stop mode commands

6: SEQ Instructions: actions (independent motor control)
Examples of the six output instructions

Domino cards

There are 90 printed dominoes on each A4 sheet. Cut the cards up to make dominoes. Each domino is a complete instruction (an instruction with a number). They can be arranged to form complete programs.

Key cards

There are 24 printed keys on each A4 sheet. Cut the card up to make key cards. Each key is an instruction, a number, or a command. They can be arranged to represent your SEQ program, so you can plan, test and modify it.

Overlays

There are six different overlays on each A4 sheet. Cut them out to make your own overlay. Blank and partially complete overlays are included, allowing you to write or draw symbols representing each key's function.

Sequence planning sheets

An A4 sheet for planning and recording simple control sequences. Each sheet shows the instructions and commands. There is space for you to record the program, how SEQ was connected, and to draw a plan.

Program record sheets

An A4 sheet for planning and recording more advanced control programs. Instructions are numbered, so more complicated programs can be planned.

15.5.1. SEQ Commands

SEQ Commands

Go - obey instructions

Stop - immediately

Test - try last instruction

Clear memory - start again

Clear error - remove last instruction

15.5.2. SEQ Instructions: actions (normal motor control)

SEQ Instructions: actions

(Normal motor control)

Forward 1

Both motors forward 1 unit

Backward 2

Both motors backward 2 units

Left 5

Left motor backward 5 units,
right motor forward 5 units

Right 10

Left motor forward 10 units,
right motor backward 10

Pulse 5

Pulse 5 times

Out 7

Output for 7 tenths of a second

SEQ Instructions: control

Pause 20

Delay for 2 seconds (20 tenths of a second) before doing next instruction

Repeat last 3 instructions

Do the last three instruction once again

Jump to instruction 4

Don't do the next instruction, jump to instruction 4

Jump if switch closed to 1

If the switch is closed don't do the next instruction, jump to instruction 1 (the first one).

15.5.4. SEQ Instructions

SEQ Instructions

Forward 1

Backward 2

Left 5

Right 10

Pulse 5

Out 7

Pause 20

Repeat last 3 instructions

Jump to instruction 4

Jump if switch closed to 1

15.5.5. SEQ Commands & Modes

SEQ Commands & Modes

Go - obey instructions

Stop - immediately

Test - try last instruction

Clear memory - start again

Clear error - remove last instruction

Stop mode commands:

Independent motor control

Normal motor control

Count 2 pulses per unit

Inverse or normal switch mode

Beeper off or on

15.5.6. SEQ Instructions: actions (independent motor control)

SEQ Instructions: actions

(Independent motor control)

Forward 1

Left motor forward 1 unit

Backward 2

Left motor backward 2 units

Left 5

Right motor forward 5 units

Right 10

Right motor backward 10 units

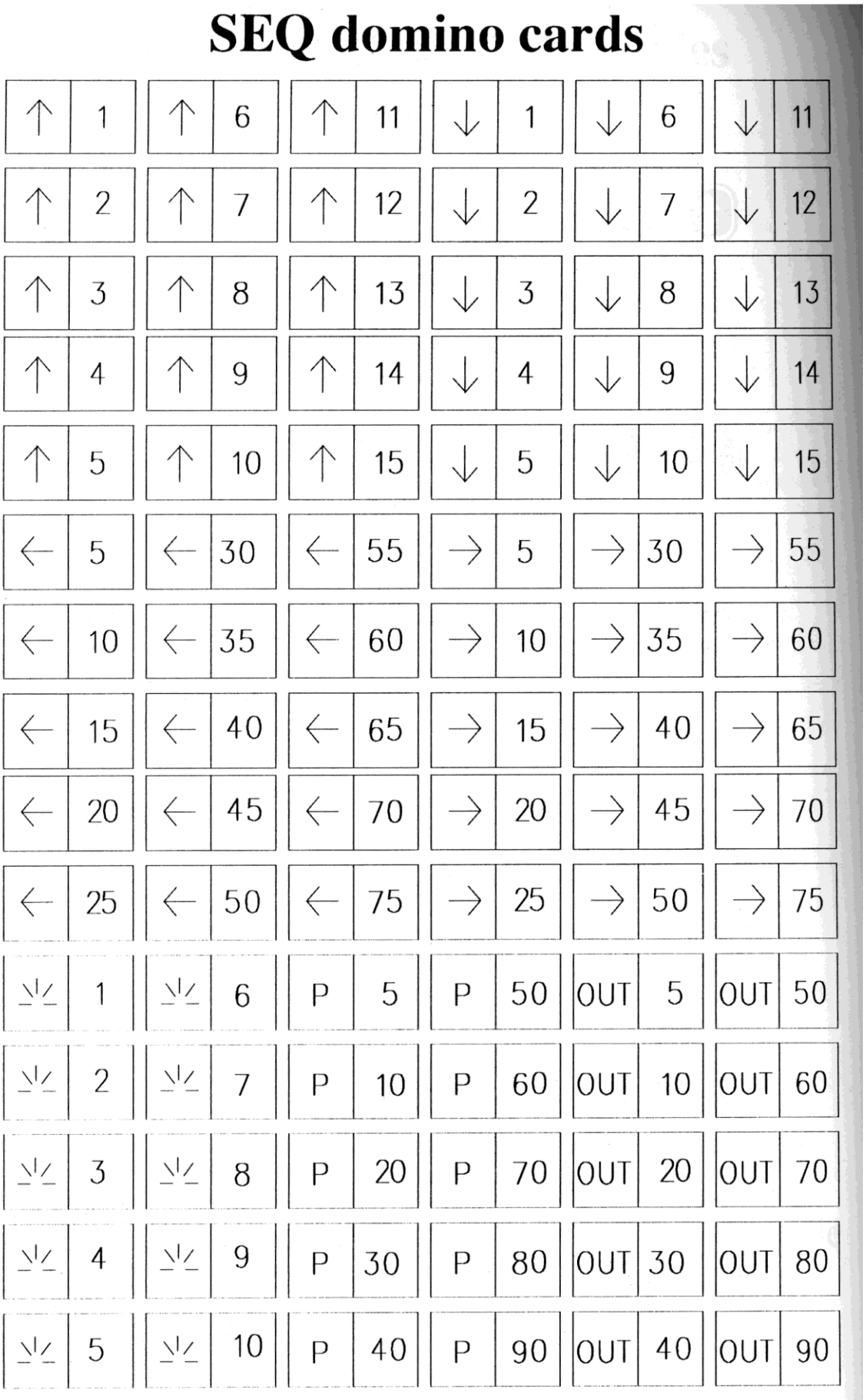
Pulse 5

Pulse 5 times

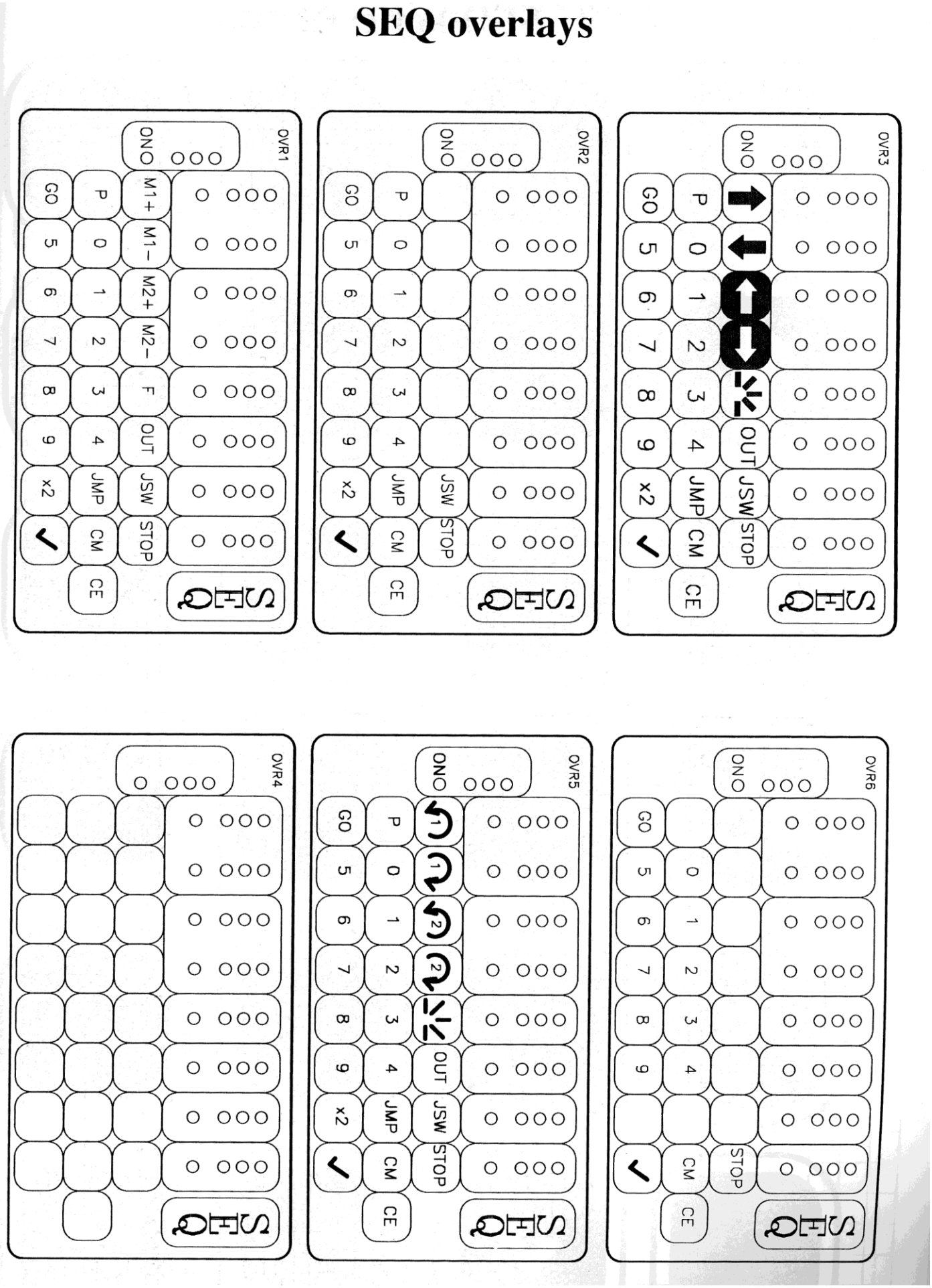
Out 7

Output for 7 tenths of a second

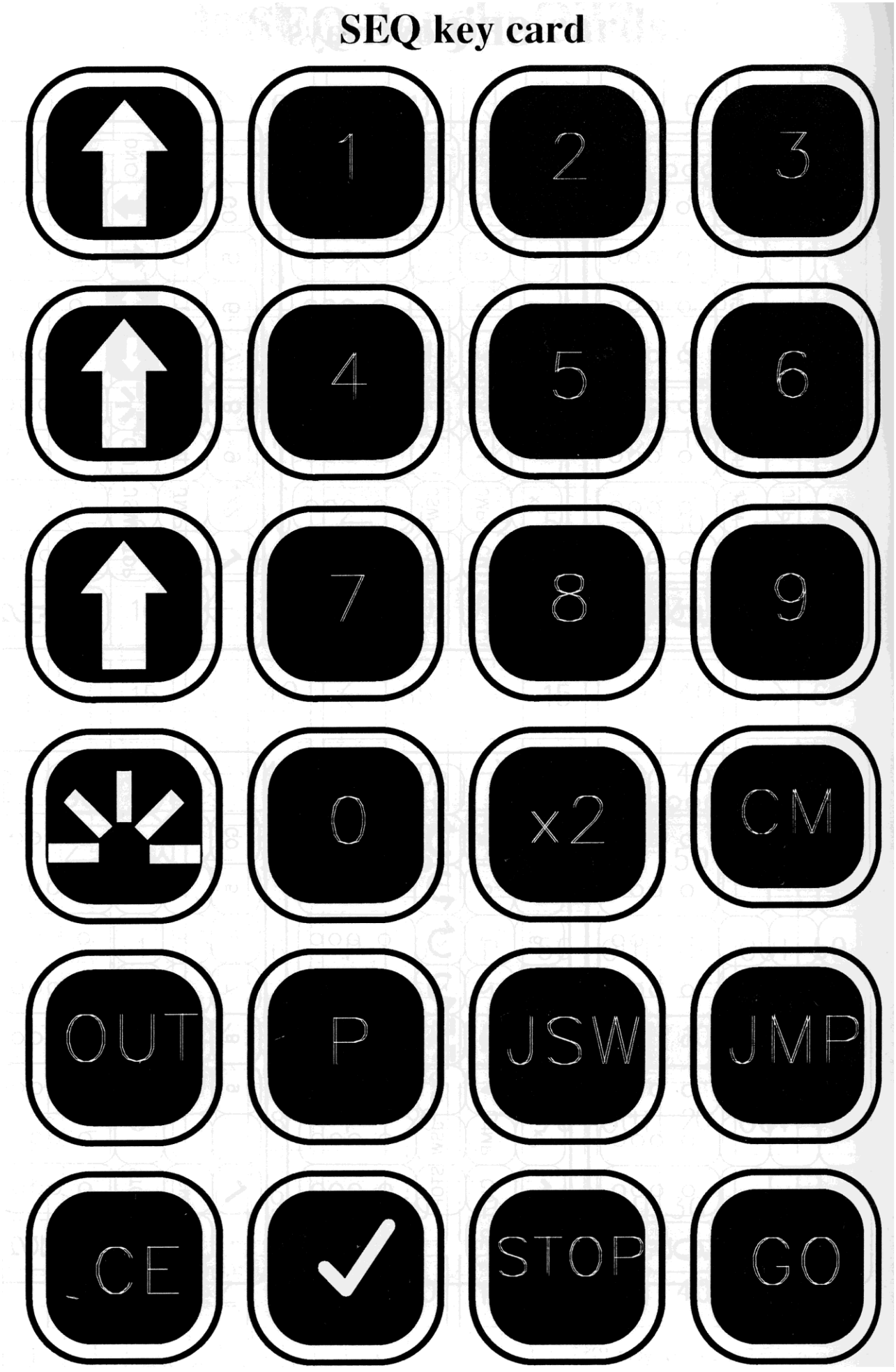
15.5.7. SEQ domino cards



15.5.8. SEQ overlays



15.5.9. SEQ key card



15.5.10. SEQ planning sheet

SEQ planning sheet

Name..... Date.....

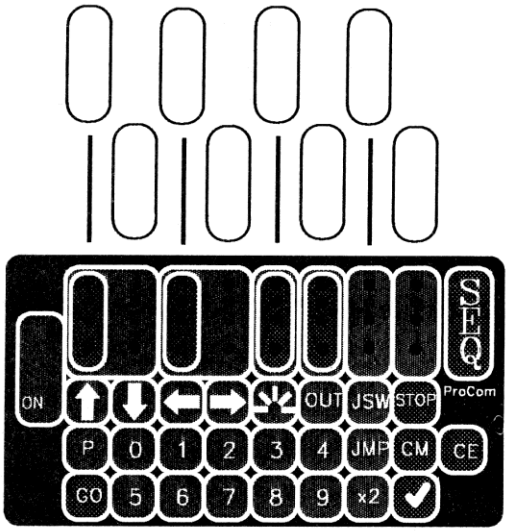
CM Clear Memory

GO Begin the program

STOP Stop the program

Test last instruction

CE Clear last instruction



Pulse output

OUT Out output

P Pause

x2 Repeat instructions

JMP Jump to instruction

JSW Jump on switch

My program:

15.5.11. SEQ program record sheet

SEQ program record sheet

Name..... Date.....

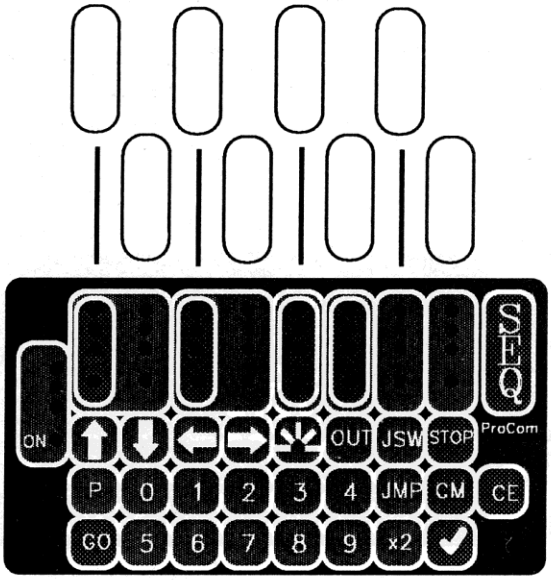
CM Clear Memory

GO Begin the program

STOP Stop the program

Test last instruction

CE Clear last instruction



Pulse output

OUT Out output

P Pause

x2 Repeat instructions

JMP Jump to instruction

JSW Jump on switch

	Instruction	Comment		Instruction	Comment
1			16		
2			17		
3			18		
4			19		
5			20		
6			21		
7			22		
8			23		
9			24		
10			25		
11			26		
12			27		
13			28		
14			29		
15			30		