Progression within Computer Control in the National Curriculum

Mike Bostock

The author of this article has had a long interest in the development of computer control in schools. He was the originator of the BBC Buggy and the project team leader for the computer control teaching resource 'LEGO Programmable Systems'. He was co-ordinator for the project which gave rise to the graphical control language 'Logicator'.

Systems and the National Curriculum

One of the key features of the original orders for National Curriculum Technology was that it described Technology as 'designing and making artefacts, systems and environments'.

In practice schools have found few problems with developing schemes of work around *artefacts*, as these have been the mainstay of the subject for many years. *Environments* have caused some difficulty because it was difficult to define what they were exactly, but *systems* have been a real problem. HMI have reported that nationally they found relatively few examples of technology work involving computer control and systems. With the revision of the Technology orders we find even stronger emphasis on systems work in Design and Technology, and specific mention of computer control within Information Systems syllabi. It is an area ripe for development.

What does systems work look like?

Systems may be broadly defined as entities whose functions are *self-regulating*.

In the information technology sense an example of work with systems would include building and testing logic circuits whose outputs (motors, lights, buzzers etc.) are arranged to respond to particular input states (switches, heat and light sensors, potentiometer etc.) in such a way as to solve a technological problem.

An example of a systems design brief would be to '*design a doorbell which will work during the day but will not wake you up at night*'.

The brief itself tells us that the output will be a door bell, that one input will be a bell push, and that a light sensor will be needed as the other input. The circuit will need to provide the logic form:

> IF [switch is true] AND [light is true] THEN [bell is true] ELSE [bell is false]

The system sentence tells us that an AND gate will be needed.

This problem can be solved quite quickly using MFA or System Omega materials, or within a longer timescale form the basis for a design-and-make project.

One basic principle of teaching about systems is that when one moves from hard-wired logic systems (e.g. electronic circuits) into software-programmable systems (computer or micro-processor control) it is possible to design systems with more complex or more subtle functions, and to adjust or modify the logic that governs those functions with greater ease.

Computer control provides us with a medium for understanding about logic structures and how these can cause a system to perform a self-regulating task. Such systems will range in sophistication from a control program that can satisfy the previous doorbell design to one that can respond in a number of subtle ways to the information provided by a range of sensors, e.g. an environmental control system, an automatic door, or a sensing robot.

Where computer control really comes into its own as a medium for promoting progressive information technology experiences is when we are dealing with one, two, or many sensors in conjunction with actuators (motors, lights etc.) and held together within some sort of structure. For this set of parts is capable of simulating most examples of systems that can generally be found in industry or in society generally, or can provide a test bed for designing and testing new ideas about systems that can present a solution to a technological need or problem.

Problems with progression

Computer control is one of the last component IT capability strands in the National Curriculum where detailed practical guidance on what constitutes a progression of experiences is still rare.

It is currently the experience of many that when computer control is observed in schools, whether it is in the primary school, at Key Stage 3, or within examination courses, the level of achievement will typically constitute 5 or 6 LOGO style procedures. In the simplistic sense this may satisfy one or two basic IT attainment targets, but in terms of pupil experiences it represents a minimal experience of what is a rich area of study.

To move forward from this position will require some important decisions to be made by schools. Becoming clear about the educational goals for this medium will be the starting point. Choosing appropriate resources that can provide pupils with accessible and

progressive experiences will be the next step, and some good in-service training and support will help make it happen in practice.

System algorithms

It is important to consider what the outcome or product to a computer control activity will be, for it is not as obvious as when one creates an artefact. What is created is not easily visualised, and when it can be seen, is not always comprehended. The product of such activity, in addition to any physical system which may be built, is a *system algorithm.* As a product it is as important as any other outcome within Technology work.

A *system algorithm* is a description of the logic paths that govern the function of the system. In information technology terms what is produced is a structure built from information. This is the same as when a pupil creates a spreadsheet, a database, or in fact writes an essay using a word processor. However, it is the case that an essay can be easily understood, a database can be explored and comprehended as a set of items, a spreadsheet, though less understood as a concept, will look like an accounting sheet with columns and totals and be understood by most. But a computer control algorithm has real communication problems. What is it? What does it look like? How do we judge it?

Programming or designing?

At the heart of the problem about valuing system algorithms as products is the historical confusion created by computing in schools in the 60s and 70s which originally emphasised the importance of the skills of programming over and above the functionality of what was programmed. Programming was once a very low level activity based on putting together number sequences that made a microprocessor work. The activity was one of '*coding*' — interpreting a function understood by a human into one understood by a machine.

Things improved a little with the use of BASIC though observers of computer studies courses which used to rely heavily on this medium often wondered about what was actually being learnt when pupils would spend long periods of time in what appeared to be a 'trial and error' activity. The word 'hacking' that has entered our general vocabulary perhaps unkindly describes this activity. If one asked about what had been created, the printout of the program would rarely serve as a medium for explanation, though in the best examples one could determine a main algorithm at the top and a set of subroutines below.

However the main difficulty caused by early computing ideas was that it tended to encourage low-level skills of coding, rather than high level skills of *designing* algorithms.

Developments in computing in schools in recent years have recognised that one of the most important skills in this context is the ability to design the algorithm that will be encoded onto the computer, and to communicate it as a solution to a problem. The advice given within Information Systems courses is to use high level software approaches. If the medium used for designing and communicating the solution was the same as the medium for the computer program then no translation up or down, to or from lower level code would be needed. This was one objective for the development of the computer control program 'Logicator' — a design environment that didn't require users to translate their ideas into computer code.

Open and closed-loop systems

The description of a system as something that is *self-regulating* is a helpful description in beginning to decide what will constitute a set of progressive experiences for pupils in this medium.

As a simple idea it will be fair to say that *closed-loop* control systems will tend to offer more complexity than an *open-loop* one. In a closed-loop system the inputs affect the outputs. In an open-loop system there is no feedback of this sort. An example of a closed-loop system would be a thermostat. An example of an open-loop system would be a light switch working a light.

Some people would argue that an open-loop system is not actually about control since nothing is actually being controlled; the system will carry on regardless, like a driverless car. Work with a floor turtle would fall into this category as it traditionally doesn't use sensors. Whether it is control in the strictest sense or not is unimportant as such early programming tasks as using a floor turtle are important in developing concepts of *sequencing* which are at the heart of any control algorithm. It is clear however that progressive systems work will make good use of sensors.

The place of LOGO

LOGO, and the powerful ideas expressed by Seymour Papert about 'real maths', have been one of the major marketing achievements of educational computing in the 80s. Many of the control languages available currently have adapted the characteristics of the LOGO language successfully into relatively friendly programming environments. But what the many who rightly proclaim the educational value of LOGO often forget is that LOGO is much more to do with learning in Mathematics than it is about understanding Technology. However, LOGO-style programs are very suitable for young learners exploring their first ideas of computer control, particularly if they are using the medium also for other work — but we do need to question the assumption that the medium will be suitable for more advanced work in this medium.

The problem with linear programming languages

Any language chosen for use in the development of computer control algorithms must be judged on the basis of fitness for purpose. There does however seem to be a fundamental problem with all languages that are linear in structure, i.e like a list. A linear language like BASIC begins at the top and ends at the bottom. They are excellent at representing sequences. The ability to jump to procedures encourages the expression of an algorithm in hierarchical form. They are, in essence, *command-driven* languages.

However, in the medium of computer control where sensors are used, a different functional style dominates, and the requirement is for a *decision-making* language or medium.

There is a very real problem caused by trying to enact a decision in a linear language. This will take the form:

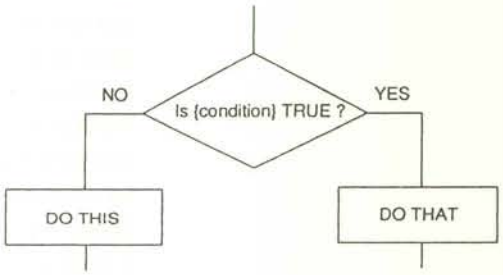
IF [switch is on] THEN do [condition true] tasks
do next tasks
etc.

Decision-making within such constructs is therefore limited to conditional procedures.

In the above example, if the switch closes then the program will go to a procedure, but then carry on down the list. To a pupil designing a control algorithm this causes a problem. This is because having returned to the list, the normal state, or [condition false], tasks are then undertaken — and this may not be what is required.

A skilled programmer can get used to thinking of a program as a repeating list which branches to procedures on conditional statements in the list which generate the [true] state. RISC assembly languages in fact work this way — but it is a contrived way of thinking. The problem for the young learner faced with having to think in this way compounds itself if there are two or more sensors each causing [condition true] at different times.

A more logical way of thinking is to regard a decision as something which causes a branch to either one or other of two equally weighted set of tasks:



List-based languages cannot do this as they can only express themselves in one dimension. However, they have tried hard to get round this problem. They have devised at least three formats for making decisions:

IF..... THEN [procedure],
REPEAT....UNTIL,
WHILE...ENDWHILE

All of them are ways of moving up and down the list in response to testing a condition. However, there is only really one sort of decision, it is 'the decision'. At its heart a decision will make a test and give rise to two different outcomes. Having re-routed the algorithm in this way new decisions can be encountered and handled in the same way. The result will be a two-dimensional branching logic map where program flow can easily be followed and the overall algorithm visualised.

Design criteria for a good programming medium

The criteria for choosing a productive programming medium for use in schools would probably include the following:

- it should promote designing rather than coding.
- it should make use of skills that students already have from using drawing programs and spreadsheets.
- it should avoid the need to translate ideas to a lower level so that a computer can carry out the operations.
- it should use a medium that can communicate the algorithm as a form that is readily understood by others

These ideas were used to guide a development in this area that recognised that a flow chart could also be a computer language. The choice of a flow chart as the medium was governed by the fact that flow charts are used fairly widely in situations unconnected with computing. A maintenance man recently used a technical flowchart in our office to help him to mend the photocopier. Here there is scope for people who are not computer specialists to understand a programming medium.

The control language 'Logicator' uses stylised flow charts to create computer control algorithms. The similarity of the medium to the cellular structure of a spreadsheet has given rise to the description 'flowsheet' for what is created. The pupil will work at the surface level, moving commands around and drawing routes just as one would do in using drawing software. The lower level functions are taken care of by the program.

'Logicator' provides one choice for a medium to work in. There are, however, other approaches that one could adopt, which will also follow the previous criteria. 'Structured flow diagrams' provide another graphical medium where one could assemble boxes containing commands on top of boxes, and boxes within boxes to represent sequences, subroutines and the different levels of a system.

Hypercard, and similar programs have also been explored successfully as a medium for computer control, providing a 'card index' approach to sequencing and branching. This particular metaphor works well, but for the same reasons as expressed above, we really need to spread the cards out in a way that shows the links between them, rather than keep them together as a stack as they are currently represented.

Some very interesting research into the representation of complex systems was undertaken in 1984 by researchers of The Weizmann Institute of Science, Israel. They developed the concept of 'statecharts' as a technique for providing visual descriptions to dynamic systems, by defining the beginning, intermediate and end states of a system, and then defining the parameters which described the likelihood of movement between the states. Within this research lies another powerful concept for developing as a medium for computer control with strong links to concepts of 'modelling'.

Current practice in computer control work

Finding the right medium for developing and testing system algorithms will be the first task for promoting this subject in schools. However, there are other decisions to be taken if systems work within school technology is to experience true growth.

One major issue to be considered is 'What am I going to control?'

The answer should be 'A system which I have built.' In the best practice pupils will follow the stages of designing and making from beginning to end in developing a system that will solve a given problem. But what is the current practice in those schools that are trying hard to do computer control work at Key Stages 3 and 4?

What we might see is a six-week module. A theme may be given. Buggies might be in there somewhere. The Buggy will have two motors and if one is fortunate it will have a sensor to detect bumps. The basis is there for some worthy closed-loop control work.

However, the exploring of the problem, the drawing out of various design options, the learning of some new skills of construction, have begun to eat into that six weeks. Making the system has proved troublesome and two more weeks have passed. Despite having three computers in the room all the pupils want to use them at the same time. One interface is a different sort from the rest and requires different software. However, success follows, because eventually everyone has made their

buggy work, and then it is on to the next module.

This may sound familiar but it is clearly a caricature. However, if we take it as real, what has actually gone on in that six weeks?

What has gone on is that there were five weeks of designing and making, and one week scratching at the basics of building control algorithms. This hardly constitutes an appropriate experience of systems. The real test comes however when we assess the pupils' work. What we find is that we can credit pupil attainment against designing and making targets and we can also put the tick in the IT Capability box which says 'pupils can produce a series of commands. . .'.

However, we do note in passing that there are many ticks already in the boxes for designing and making. So what was the real value of the project if it simply duplicated previously covered construction activities?

A systems approach to systems work

The real problem with the above example lies in the common insistence that everything used must be made. They must make the system that they control. Yes, this is important. But the real secret to getting good, progressive experiences within systems work is in making the right decisions about *appropriate starting points*. Pupils are not asked to go and chop down trees to get the wood to use in a construction project, and they don't have to smelt the iron ore to make metal sheets.

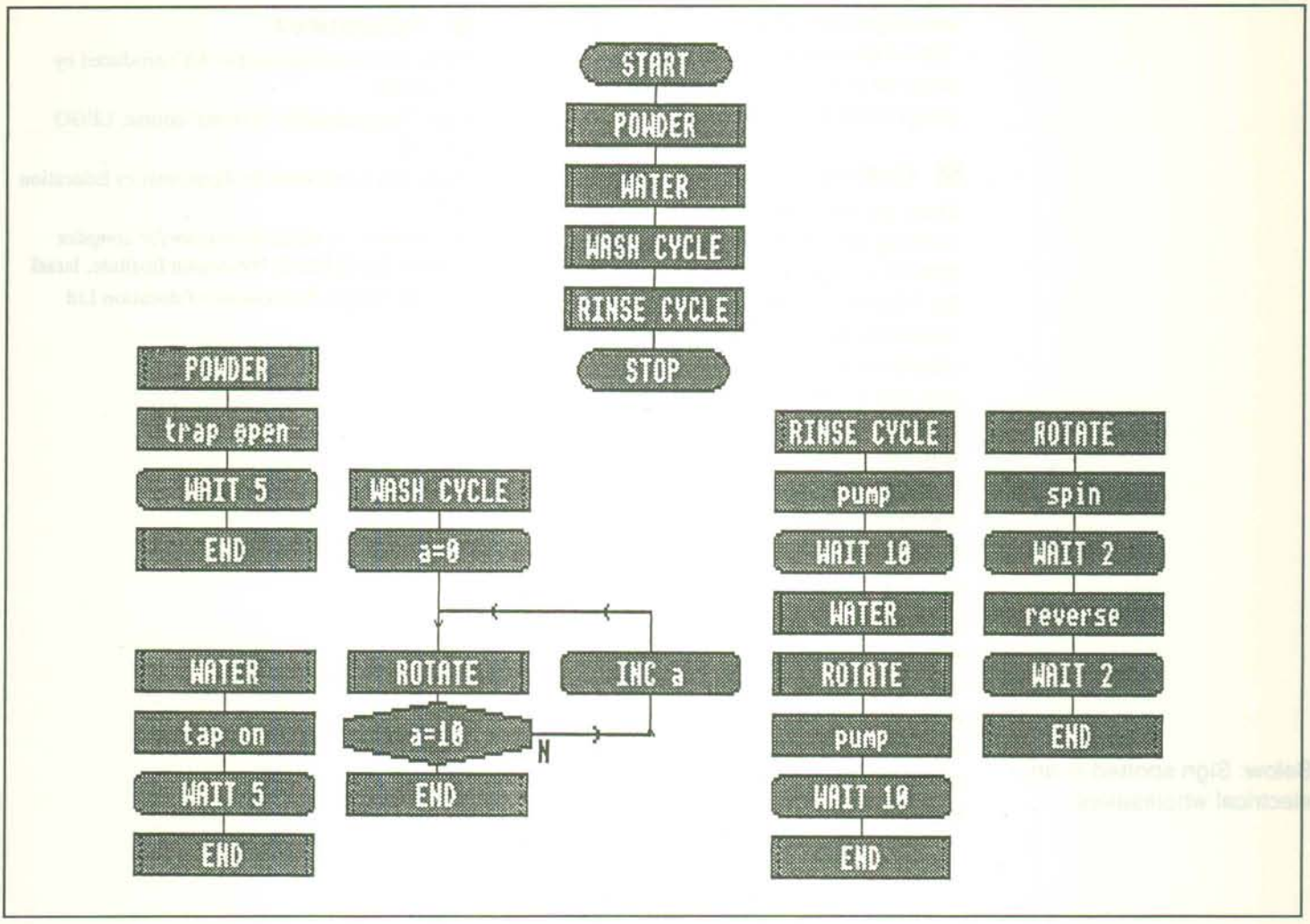
So what is absolute about the starting point for a project being a sheet of plywood one foot square? If a pupil understands what a wheel is and they have already covered basic making skills, what value is there in insisting that they make a wheel? They should be able to choose to make a wheel, or choose a wheel that is already made. The same applies to a gearbox or other basic building blocks used to make systems. Some will defend this 'start from scratch' approach by citing cost as the reason, and see great virtue in using low cost materials, even when these are plainly inappropriate. Making a gearbox from cotton reels and elastic bands may be appropriate for primary age children but this is hardly sensible for Key Stage 4.

The main principle which should be applied to determining systems experiences for pupils should be to choose starting points that promote pupil achievement and avoid presenting tasks which duplicate previous achievement; for this only has the effect of taking time away from new areas of learning. This 'sub-systems' approach has already been developed at primary school level through those ingenious constructional solutions devised by David Jinks. We need to apply the same techniques higher up.

In order to make real progress with computer control, pupils will need to be able to assemble a functionally effective system that has sensors and a function sophisticated enough to offer a good algorithmic design problem. If they can choose pre-formed building blocks and sub-systems to do this they are probably more likely to achieve a worthwhile outcome. In fact they should choose to work at the highest level in regard to the choice of constructional materials, to mechanisms, and to the computer control medium. In fact good systems work will arise from encouraging work at the highest, or *systems level.*

The choice of materials to use for this work will be a consequence of this consideration, but it is likely that a variety of solutions will be employed in practice. In Key Stage 3 computer control work within design and technology, the emphasis will be on using consumables, with some judicious re-cycling. The task will be to prepare a range of appropriate pre-formed parts which provide solutions to some of the known problems of constructing structures and mechanisms. The indicator of success here will be racks of components and pre-formed solutions in the classroom. Of course pupils will make some parts, but the emphasis will be on the use of sub-systems.

Within an IT Capability lesson in a computer room pupils won't as easily be able to make models as they could in a design and technology room. However, there is scope for assembly in some appropriate form. Pre-built or partially built models offer one way forward here, although their use is controversial. If they are to be used they may be made from whatever material is suitable, but there must be scope for developing the function of the model, or adding sensors as the problem is explored. People may argue with this, wishing to see the system built within a complete design cycle. The answer to this must be IF this can be



Above: A stylised flowsheet describing the cycle of a washing machine developed using the computer control program 'Logicator'. The flowsheet can be run as though it were a computer program.

accomplished, AND that good experiences of developing a control algorithm are provided, AND there is good time to develop, test and refine the overall system, THEN clearly that will be an effective approach.

The importance of prototyping

At Key Stage 4 the same principles will apply, but the emphasis will be on providing greater rigour and progression to the work. There is a strong case here for the technique of *prototyping*, and indeed this does offer a half-way house to the problem of constructing a system. If a system can be put together rapidly, perhaps only in skeletal or representational form, then the system could be developed further as a result of an initial evaluation of function. Having gone through a prototyping stage, the pupil will probably understand much more about the parameters of operation of their proposed system. Subsequent work where their experiences are transferred to the final system will be influenced by this insight and the final outcome is likely to be of higher quality as a result.

Whether prototyping can be built into a design-and-make task will depend, in part, on the time available, but I put it forward as a hypothesis that to have developed and explored a prototype without then going on to build the final product will develop a greater understanding of systems than to have worked towards a product without having explored the prototype.

Prefabricated components and kits have their place here, and LEGO, Meccano and FisherTechnik have a role to play which emphasises their non-consumable nature, and the solutions they provide in the construction of prototypes. The final system will be built from whatever is the appropriate material. Although there has been some traditional opposition to using kits, whatever approach is taken will need to be judged by the quality of the learning outcomes experienced by pupils, and in this case, the extent to which the systems experiences of pupils were effective

and progressive. The real question to address is 'Have I chosen the most appropriate starting points so as to have maximised the potential achievement for this activity?'

Getting IT into the system

These are some thoughts on the important and evolving subject of computer control and systems work in schools as they impinge upon the 'Measurement and Control' strand of IT Capability. Unlike work in other IT strands, schools encounter fewer practitioners in this area able to offer advice on appropriate approaches to control work in the context of systems. However, as a nation with a National Curriculum we have considered this subject as important enough to insist that all pupils should experience it.

The task will now be to find opportunities for those interested to debate the issues more widely, and to find ways to provide schools with the guidance that they will need if they are to develop progressive and cost-effective approaches to providing pupils with key experiences of microelectronic and computer controlled systems.

References

MFA: 'Microelectronics for All': produced by Unilab Ltd

Lego 'Programmable Systems' course, LEGO UK Ltd

'Logicator': published by Economatics Education Ltd

'Statecharts': A visual formalism for complex systems, David Harel: Weizmann Institute, Israel

Systems Omega, Economatics Education Ltd

Below: Sign spotted in an electrical wholesalers