CONTROL SYSTEM DESIGN FOR A ROBOTIC BACKHOE

R.H.Bracewell, D.A.Bradley, R.V.Chaplin, D.W.Seward

Department of Engineering
Lancaster University
Bailrigg, Lancaster
LA1 4YR
England

SUMMARY

This paper describes the main requirements for, implementation details of and philosophy behind, the controller currently being implemented on a 1/5th scale model "JCB style" backhoe arm at Lancaster University. It is somewhat unconventional in its use of the Harris RTX2000 microprocessor and FORTH language, which offers great potential for real-time "intelligent" control.

KEYWORDS

FORTH, Harris RTX2000, STE bus, real-time, production system, Inmos Transputer, FORPS

1 INTRODUCTION

At Lancaster University Engineering Department, the inter-disciplinary Mechatronics Research Group has for several years been interested in the introduction of robotics into the civil engineering and construction industries. One area which we have been investigating is the automation of earth-moving equipment. A SERC-funded series of detailed on-site studies of current excavation practice is now in progress, looking at the whole operation from the economics and planning down to the instinctive and tactical skills of backhoe operators. We hope to use the data gathered during these site studies in two ways:-

(a) Identify the tasks into which the introduction of robotic technology would be most economically favourable and attempt to quantify the potential benefits.

(b) Develop a way of expressing coding and enhancing the experienced excavator driver's skills to produce a microprocessor-based real-time "skilled artisan" controller [BRA89], capable of at first assisting and ultimately replacing the operator.

Given the high cost, not to mention the destructive potential of a full-scale, experimental, computer-controlled excavator, we constructed a 1/5th scale model steel backhoe arm, hydraulically powered and with a soil box into which it can dig. This paper explains the ideas behind the control system we have assembled for the arm, with which to attempt to develop our "skilled artisan".

2 CONTROL SYSTEM SPECIFICATION

In designing the control system, there were a number of different requirements to be considered:-

- We need an asynchronous multiple input/multiple output real-time system, responding by means of prioritised, vectored interrupts to both external attention requests and internal countdown timers.

- It should support multitasking and multiple processors. This allows functionally separate parts of the software to either be run concurrently on the same processor, or later be easily moved to separate processors if time constraints become critical.

- It must be computationally fast and efficient, not just at number crunching but also at making logical inferences, with sufficient spare capacity to suffice for the next few years' development.

- The system is currently being put together as a group project for 4th year Mechatronics students and will probably be used by students in future. It must therefore be as simple in use and concept as possible, so they can learn to use it effectively within a short period of time.

- While the site studies work is presently officially funded, the hardware implementation is not and funds are limited. Hence it must be economical, use as many plug-in, reusable components as possible and be sufficiently rugged to survive a mechanically and electrically hostile environment. It must also be as flexible as possible in scope for future expansion, to reduce the risk of having to redesign from scratch if perceived requirements change.

- It is desirable that all the software should be writable in a high level language, with no need of recourse to assembler, easing the production of clear, well structured, well documented, maintainable code.

3 THE SYSTEM HARDWARE

A schematic diagram of the system is shown in fig. 1, the main features of which are as follows: -

- The controller is based on the **STE (IEEE 1000) bus**, housed in an industry standard **19" Single Height Eurocard Rack**.

- Boom, dipper and bucket are driven by **bidirectional hydraulic cylinders**, slew by a **hydraulic semi-rotary vane actuator**.

- Each of these 4 services is controlled by a **Bosch proportional, bidirectional solenoid valve**, mounted in parallel on a common manifold block.

- Each valve has its own proprietary drive card, which implements analogue closed loop PID control of the spool position, based on feedback from an LVDT.

- The four valve cards are housed in the 19" rack, but shielded from the other electronics to avoid electromagnetic interference from their power switching circuitry.

- Feedback of the arm joint angles is provided by 4 **servo potentiometers**, one on each bearing pin, protected by rugged metal cans.

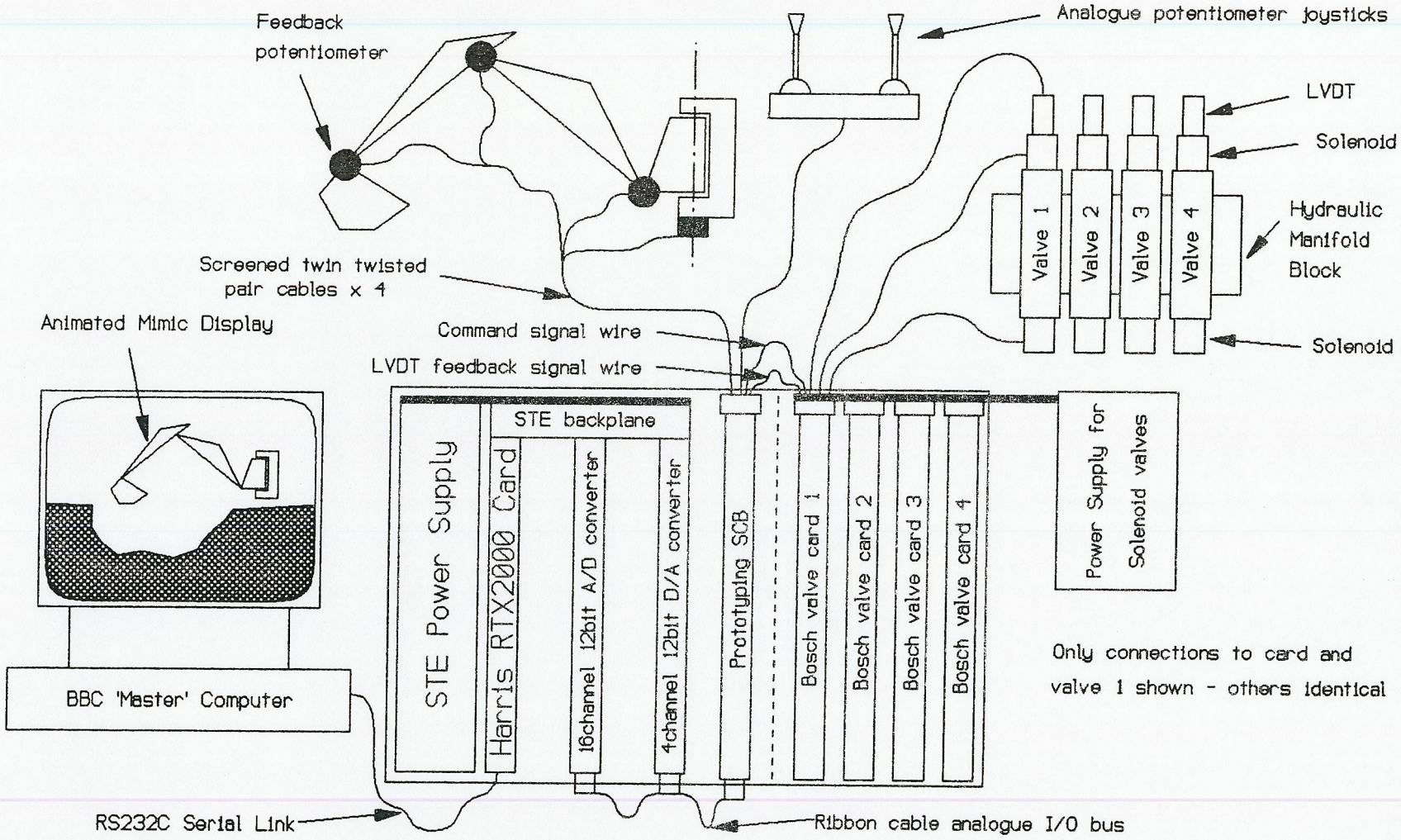

Fig1. ROBOTIC BACKHOE CONTROL SYSTEM

- User input is via two **biaxial potentiometer joy-sticks**, which may be used either to emulate conventional backhoe control, or for some enhanced scheme such as cartesian coordinate control, as implemented by Deane et al [DEA86].

- Analogue signals from the valve spool LVDTs and the arm and joy-stick potentiometers are conditioned by custom circuitry on the **Arcom Prototyping Signal Conditioning Board** and routed to the **Arcom 16 channel, 12 bit, 25 microsecond A/D Converter Board.**

- The valve driver cards each require an analogue demand signal for spool position. These are provided by an **Arcom 4 channel 12 bit D/A converter board**, via an amplifier on the signal conditioning board.

- The processor card is supplied by MPE Ltd. and is based on a **Harris RTX2000 microprocessor**, running at 8MHz. It communicates by RS232 with an **Acorn BBC Master microcomputer**, from which the Harris is programmed, and which itself runs a real-time animated display for the backhoe operator. Both are programmed in the FORTH language.

At first glance, this system has many similarities to that installed on a McConnel tractor-mounted digger by a team at Liverpool University and described in some detail in [DEA86]. However, in using an 8 bit 8085 with floating point co-processor and entirely programmed in assembler, they were at the limits of their processing capacity and would have found it difficult to expand to more ambitious control schemes.

Our system, through the use of the RTX2000 and STE bus, has up to two orders of magnitude greater processing speed as standard, scope for far more powerful software, particularly in the field of real-time "intelligent" control and has many routes for upgrading and expansion. Equally importantly, it is still simple and inexpensive.

3.1 STE Bus

The obvious choice for flexibility and a wide range of plug-in components seemed to be one of the various standard backplane bus systems, of which common ones are VME, AT, MULTIBUS II, STD, G64 and STE. Of these STE was chosen because:-

- Most memory will probably be on the same card as the processor addressing it, so the backplane will largely be used for intermittent I/O work. In this case the throughput of an 8 bit bus is easily sufficient and the additional cost of a 16 or 32 bit bus, such as VME or MULTIBUS II cannot be justified.

- STE is the most modern 8 bit bus available and having been adopted as an IEEE standard, has gained rapid acceptance in the last few years. It has two big advantages over older designs such as STD and G64 (see [MIT89]). Firstly, it is completely asynchronous, all data transfers being coordinated by handshake, so peripherals of widely differing speeds can be used without difficulty. Secondly, it is designed to allow the use of multiple processors, each one first requesting the bus from an arbiter, before performing a transfer operation. It has a 20 bit address bus, allowing up to 1 Mbyte to be addressed directly.

3.2 Proportional Valves

The type of proportional solenoid valve which we are using was designed for analog PID type controllers of fairly linear systems, in situations where the pressure drop across the valve is reasonably constant. In these cases the actuator velocity will be roughly proportional to the control signal. The situation here is very different however. The joint angles are related to piston displacement in a highly non-linear manner. The effective inertia seen by each joint varies greatly with arm position and the mass of earth in the bucket. The pressure drop across a valve varies between zero - when its joint is stalled - and the full system pressure. The best performance will be obtained by the use of advanced, robust closed loop digital control algorithms for which the linearity or otherwise of the valve is irrelevant - it is even possible that bang-bang valves could be the best choice. We do not think that the ideal hydraulic valve for this sort of application has yet been produced.

Having said that, we chose to use conventional proportional valves for two reasons. Firstly, the Liverpool/McConnel team demonstrated that despite all the problems mentioned above, a simple proportional feedback servo system can be made to work using these valves. It would make a good starting point to get the model working using this sort of conventional control, as a yardstick against which more advanced systems can be compared. Secondly, more advanced control can then be implemented purely in software, with no hardware changes required. This is because the valve spools may be controlled directly by the processor, as the spool feedback signals are available via the A/D converter and the PID action of the valve card can be overrided by always driving it to saturation in one direction or the other, so that it acts simply as a solenoid driver.

3.3 The microprocessor

The one really unconventional aspect of this system is its use of the newly developed Harris RTX2000 microprocessor, with which many people may be unfamiliar. This is quite unlike both complex instruction set (CISC) processors, such as the well established 80x86 and 680x0 families and the newer, reduced instruction set (RISC) devices such as SUN's SPARC or Motorola's 88000. It has however, some unique capabilities, which are particularly well suited to real time control applications such as this.

3.3.1 FORTH and the Harris RTX2000

The RTX2000 owes its existence to the FORTH programming language and its creator, Charles Moore. This was invented to solve real time computer control problems, its original use being the control of astronomical telescopes. One thing that sets it apart from other ways of programming computers is the remarkable simplicity and elegance of its internal mechanism, or "virtual machine". This is completely accessible by and just as importantly, understandable to the average programmer, which confers great flexibility in its use. It is based around two last-in-first-out stacks, one for parameter passing, the other for storing return addresses and makes little use of registers. This facilitates fast, efficient multitasking and interrupt programming, since whenever context is switched, very little need be saved to be restored later.

FORTH's biggest problem in time critical situations, was that it always had to be run on conventional processors - either CISC or RISC designs - neither of which are particularly good at it, both types being generally optimised to run C under UNIX, as the greater part of the market demands. They gain high performance through the use of features such as instruction pipelines and fast memory caches, which are all very well

for programs which consist largely of tight loops of in-line code, but are quite useless for FORTH, where the sequence of instructions follows a highly convoluted "thread" throughout the memory space, by way of a large number of subroutine jumps and returns. This means caches and pipelines are continually having to be refilled, so the theoretical performance advantage is lost.

The problem was solved by Moore himself, who decided he would design his own 16 bit microprocessor, not merely optimised for FORTH, but actually creating the FORTH "virtual machine" in silicon, so its instruction set consisted effectively of high level FORTH primitives. Moore collaborated with Bob Murphy, an expert on gate arrays, and together they designed the processor as a large set of logic equations, which were then compiled, implemented on a 4000 gate semi-custom array and released for production as the NOVIX NC4016. It is worth noting that this device, designed by just two men in a single year, containing less than a quarter the number of transistors of a Motorola 68000, out-performs the latter for processing speed by more than an order of magnitude [TIN87]. The rights to the design were then sold to Harris Semiconductor, who cleared up some bugs, moved both stacks onto the chip, added a few enhancements of their own and produced it in custom silicon as the RTX2000.

The key to the RTX2000's performance lies in the amount it gets done per clock cycle. It has no microcode, no pipeline, no memory cache - the instruction is fetched, decoded, executed and the address of the next instruction placed on the bus during the high part of a **single** clock cycle. This is still the case **even for a jump or conditional branch instruction.** And that is not all. The internal architecture consists of 4 independent buses, addressing external memory, I/O and the two stacks respectively. These are controlled by separate bit-fields in the instruction, and can all be in use concurrently. This means that using an optimising compiler, **up to 5 consecutive FORTH words can be packed into a single 16 bit RTX instruction !** This extreme degree of optimisation is admittedly not often possible, but it does make an interesting contrast to most language/processor combinations, in which you expect a single high level word to be compiled into many machine instructions.

3.3.2 Real-time enhancements

Number-crunching speed is boosted by a single cycle 16 bit multiplier and very fast I/O is provided by the G-bus, which allows data transfer at up to 16 Mbytes/s to or from any of 8x16 bit external locations. In order to increase its usefulness as a microcontroller, it possesses 3 on-chip 16 bit timer/counters and a 14 priority-level, vectored interrupt controller. Interrupt latency is just 0.5µs.

4 DECOMPOSITION OF CONTROL PROBLEM

In a previous paper ([BRA89] fig.1) we have argued that the overall control of an item of automated construction plant may be represented by a 3 layer hierarchy, namely instinctive, reflective and consultative modes of operation. It is our immediate intention to implement the first two of these on our model.

The lowest level of instinctive operation, making the bucket tip go where you want in undemanding situations, is best fulfilled by robust, advanced closed-loop positional servo algorithms. For the highest level of the reflective mode, the "thinking time" allowed is quite long and the complexity is clearly in the realm artificial intelligence, requiring perhaps a rule-based production system such as OPS5, the use of which is well described in [BRO85]. In between is a "grey area", in which, for ease of coding knowledge elicited

from expert drivers you might like to use a production system, but be forced into algorithmic programming by its lack of speed - in general they are notoriously slow. So the use of fastest practicable production system is an important requirement.

4.1 Positional servo control of the bucket

Lancaster Engineering Department possesses considerable expertise in this field, using advanced algorithms programmed in Occam on transputer networks. The RTX2000 STE Powerboard from MPE Ltd. possesses a 10/20 Mbit/s Inmos serial link, connected to the processor's G-bus and two highest priority interrupt lines. This will allow a transputer to be connected directly to the existing system, using a simple 3-wire cable, so that the RTX2000 can do the "fetching and carrying" of data and commands for the servocontroller program running on one or more transputers. Only a small proportion of the RTX2000's processing time would be required to do this, leaving the rest for the use of a production system to provide the artificial intelligence for "reflective" operation.

4.2 Fast rule-based production system

The last few years have seen some interesting developments in the use of FORTH for artificial intelligence particularly in the area of real-time control. For example there have been several PROLOG systems (e.g. [PAL87]) and an implementation of OPS5 [DRE86], all written in FORTH.

Of more immediate interest to us is a system called FORPS, a very simple, efficient production system intended specifically for real-time control - it was first used with a servo-manipulator for intelligent obstacle avoidance. It is in the public domain and is described in intimate detail with full source code (just over one page !) in [MAT86a]. Apart from use by its developers, it has also been employed by NASA to rewrite a small expert system originally coded in a fast, commercial OPS5, both running on an IBM AT clone [RAS87]. They were impressed by its greater speed, flexibility and readability, and concluded that FORPS is "a serious alternative to OPS5 for smaller expert system applications". But this was using a conventional microprocessor (Intel 80286); the RTX2000 will put the speed of FORPS into a different league altogether. It is reported in [MAT86b] that porting FORPS from their original 10MHz Motorola 68000 system to a 6MHz NOVIX NC4016 (the precursor of the RTX2000) **increased its speed by a factor of nearly 30:1!** (this would be nearly 40:1 on our 8Mhz RTX2000). He concludes that it should be possible to run at the remarkable speed of 10,000 rules/s, which completely transforms the usefulness of this sort of system for machine control problems.

5 REFERENCES

BRA89 Bradley, D A, Seward, D W, Bracewell, R H & Anson, M, 1989, 'Real-Time Expert Systems for the Civil & Construction Industries', Workshop on Real-Time Systems Theory & Applications, University of York, U.K. 28th September 1989.

BRO85 Brownston, L et al, 1985, 'Programming Expert Systems in OPS5', published by Addison-Wesley, Reading, Massachusetts, U.S.A.

DEA86 Deane, E R I et al, 1986, 'A Microprocessor-based Agricultural Digger Control System', Computing & Electronics in Agriculture.

DRE86 Dress, W B, 1986, 'REAL-OPS. A real-time engineering applications language for writing expert systems', Journal of Forth Application and Research, v4 n2 pp113-24.

MAT86a Matheus, C J, 1986, 'The Internals of FORPS: a FORTH-based production system.', Journal of Forth Application and Research, v4 n1 pp7-27.

MAT86b Matheus, C J, 1986, 'An Implementation of FORPS on a NOVIX Beta Board', Journal of Forth Application and Research, v4 n2 pp185-8.

MIT89 Mitchell, R J, 1989, 'Microcomputer Systems Using the STE Bus', Macmillan Computer Science Series, ISBN 0-333-49649-3.

PAL87 Paloski, W H, et al, 1987, 'Use of a FORTH-based PROLOG for real-time expert systems', Journal of Forth Application and Research, v4 n4 pp463-86.

RAS87 Rash, J L, 1987, 'An OPS5 Expert System Converted to FORPS', Journal of Forth Application and Research, v5 n1 pp209-12.

TIN87 Ting, C H, 1987, 'Footsteps in an Empty Valley', published by Offete Enterprises, U.S.A.