The Use of Networked Computers in Real Time Process Simulation

Eion Morrison Johnston

Master of Philosophy

University of Edinburgh

1988

This work is dedicated to Sheila and Richard

Acknowledgements

Thanks are due to the following in enabling and helping during the course of this project.

-- the management of the former Leith Nautical College
-- the management of Napier College
-- the library staff at Leith Nautical College and Napier College
-- the technicians at the Electronics Department of Napier who had to move and set up the simulation equipment a total of four times in one year
-- those inside the Chemical Engineering Department and outside at Edinburgh University who helped with advice on various points
-- especially my supervisor Jack Ponton

This thesis was written by me and all the work described in it, unless attributed to others, was done by me,

ABSTRACT

This work details an approach to the simulation of chemical process plant for training purposes using networked mass produced microcomputers. The work was prompted by the need to provide training simulation of the cargo systems on Liquefied Petroleum Gas Tankers. The rationale is based on a reduction in hardware costs by creating a powerful computing unit from a number of connected low cost computers. Previous work on multi-microcomputer use are described but no reports of the current approach have been noted.

A ring network is implemented using the RS423 interfaces on BBC Model B machines but is abandoned in favour of the extremely low cost network system (Econet) which can be used with these machines. Three different methods of communicating values between the machines via Econet are described, culminating in a "user friendly" packet system which hides the machine details.

Description is given both of the modelling of the individual plant items of the LPG ship and of the simulation of the whole cargo system. Two other simulations (pressure swing adsorption and compressor modelling) are reported which use the approach and attendant difficulties are highlighted.

The concluding chapter gives appropriate methods of decomposing a plant simulation for use on the multi-microcomputer system and also gives suggestions for future hardware approaches.

CONTENTS

APPENDICES

INTRODUCTION

1.1 Background

The origin of this research project lay in an attempt to provide low cost computer simulation of chemical processing systems. It arose initially from the desire to provide a training simulator for liquefied gas cargo ships. At the start of the project, I was a lecturer at Leith Nautical College and worked with the Hazardous Cargo Handling Unit which ran training courses for LPG ship personnel. Much of the work described in this thesis refers to this gas simulator.

There is a need for computerised simulation of all types of process plant whether on land or situated on a ship. Typically it is a requirement during the commissioning period of the plant when personnel are trained in the use of the plant.

This may be contrasted with the use of computerised flight deck simulation in the aircraft industry. Here there is a requirement for the training of many operators over the long life of a given model of aeroplane. Each plane of a given type will be identical to the rest. It is cost effective to use a simulator priced at a million pounds when it will be used intensively for several years. The simulator hardware usually includes a powerful mini computer with purpose written software and a convincing hardware man-machine interface representing the flight deck found on the aeroplane.

The computing power required to simulate process plant is certainly no less than that to simulate the flight deck. To give extremely accurate results and allow for prediction, optimisation and development of control strategy requires both expensive hardware and software. However each plant is unique and different from the next to be built. The full time use of a simulator is likely to be short lived.

To make the computer simulator worthwhile it must either be cheap or, if expensive, somehow it must be very flexible so that the capital investment can be justified.

1.2 Approach

It was decided to approach the problem of providing a practical simulator with a novel hardware configuration which would reduce drastically the hardware costs. A network of low cost, mass produced personal computers was assembled as a multiprocessor.

This approach had several advantantages over the minicomputer simulator and evidence of its viability could be demonstrated.

The basic principle was that for any given simulation problem, a certain amount of computing power was required. This could have been provided by a minicomputer (although this type of computer might have provided more than the required amount.) The alternative was to calculate in some way the equivalent power in

terms of units of a smaller computer. A comparison by Ponton et al[1] involving a number of personal microcomputers using a benchmark weighted heavily towards "number crunching" is shown in table 1.2.1.

The conclusion was that taking into account the relative costs (at 1983), computing power in the form of mainframe or minicomputer installations cost more than five times the equivalent power purchased as a number of personal computers. This of course ignored the overhead in making a number of microcomputers interact as one multiprocessor.

Since 1983 the amount of computer power purchased per pound has been increasing greatly. However it was still demonstrable that the proposition held that the mass produced micro provided a cheaper form of computing power compared to the minicomputer.

Other benefits of this system concerned its flexibility, the system was unit incrementable. That is to say, when more computing power was required for the purpose of adding another item of process plant, it could be achieved by the addition of one more microcomputer at a small fraction of the cost of extending a minicomputer system.

There was no dedicated man-machine hardware interface since the colour graphics monitors (one for each machine) provided all the necessary output and a touch screen was used for input. This meant that the same hardware could

Computer Benchmarks

Due to Ponton et al[1]

TEST PROGRAM - BASIC VERSION

```
DIM A(1000)
FOR I=1 TO 1000
A(I)=SIN(I) : B=LOG(I)
NEXT I
```

Machine	Time(s	Approximate Cost (pounds)	Estimated Unit Cost (1000pound/Vax)
ICL2976	0.1	1M	100
VAX 750 (no f.p.a.)	1.0	100K	100
M68000 (10 MHz)	10	2000	20
IBM PC	25	2200	55
(with 8087)	3.0	2500	7.5
BBC B*	35	500	17.5
(with 6502C)	23	700	16.1
(with 32016)	8.0	1300	10.4

Note a) these figures are based on 1983 prices

b) the final column is an attempt to quantify the cost of computing power supplied by different machines in terms of the number of thousands of pounds to provide the power of one Vax 750

Table 1.2.1

be used to provide simulation of many different systems by changing the software.

The system was made up from "off the shelf" general purpose machines. These could be used for other purposes in an organisation when not required for simulation. It meant that downtime of the system was greatly reduced as a single faulty machine could be replaced by a standard item found elsewhere in the organisation.

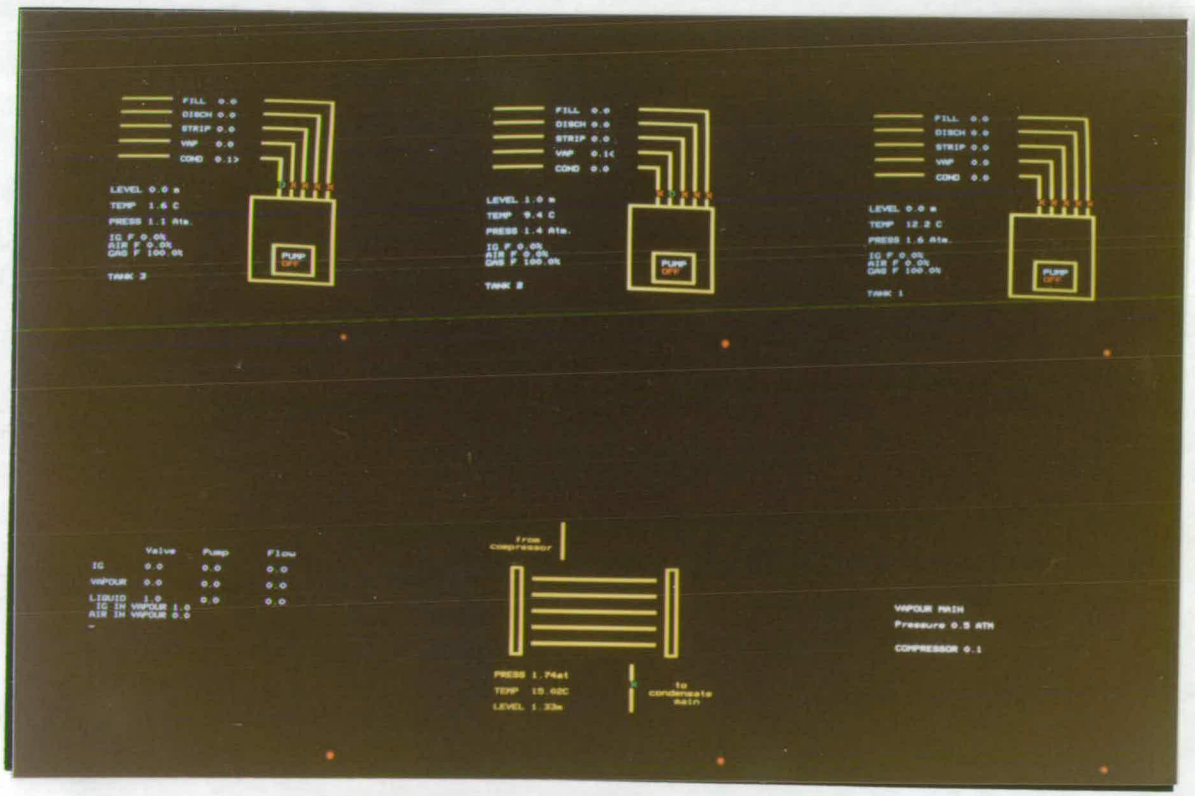
1.3 Conclusions

A system representing an LPG ship with three cargo tanks and a liquefaction system was developed. It was found that the various LPG cargo system procedures could be simulated successfully, for training purposes, using the multi-microcomputer approach. Three different approaches were implemented using the Econet local area network to connect a number of Acorn machines. This resulted in a "user friendly" interface for the modeller who could use "pipes" to connect the various parts of the plant located in different machines and ignore the details of programming the Econet network. The communications system allowed the connection of different Acorn models, such as the BBC Model B along with the Acorn 32016 Co-Processor, programmed in different languages - the BBC Model B in BBC BASIC and the 32016 in ISO Pascal.

The approach was applied to two other process systems -

a pressure swing adsorption plant and a complex compressor system. Here the simple problem decomposition method used in the LPG case had to be re-examined due to the "stiff" nature of the equations modelling these other processes and the effect of time delays in the communications on closely coupled processes in different machines. An alternative "two level" approach was applied which met with some success. Finally it is suggested that with the falling cost of computing power the multi-microcomputer configuration using a local area network and mass produced machines is a practical method of achieving low cost simulation, especially for training purposes.

A photograph illustrating the screen displays for the LPG simulator developed in this work is shown below.



SURVEY OF PREVIOUS WORK

2.1 Introduction

There has been little work published on practical multiprocessor systems and virtually none on the topic of "off the shelf" personal computers being used as a multiprocessor. Pimentel[3] commented that although intensive research has occured in the area of multiple microcomputers, most of it has been theoretical and actual multi-microcomputer implementations were few. The use of such systems in process simulation has not been widely reported.

2.2 Multiple Processor Projects

Korn[2] in 1981 considered the possiblity of using multiple 16bit microprocessors as a replacement for the then current 32bit super-minicomputers for continuous system simulation. He likened the speed gain due to parallel microprocessors with that of the parallelism in analogue computer simulation. He used an informal benchmark of flight simulation. His conclusions were that in 1981 the supermini was clearly superior for floating point simulations, however, for fixed point calculations, the 16bit multiprocessor had the advantage. He predicted that by 1985 the development of 32bit microprocessors would swing the argument in favour of the multiprocessor and he suggested the development of

suitable software for parallel computation for use on standard microprocessors connected together without special hardware development.

Pimentel[3] in 1983 described a multiprocessor system used for real time simulation of a spark ignition internal combustion engine. Five microprocessors of the 6800 family were used. Four were used to carry out the simulation and the fifth was used as a central coordinator for the system. For example it handled requests for data transfer within a shared memory. The system was successful in providing real time simulation.

It was admitted that the same result or even better performance could have been obtained using a minicomputer but the cost would have been prohibitive in this application. The hardware cost was estimated at 1000 U.S. dollars which was clearly very low. Although the authors claimed a flexibility in terms of addition of extra processors and the ablility to simulate other systems, the software for each processor was stored in EPROM and changes to the configuration would have been accompanied by considerable software effort.

Sundararajan[4] in 1984 pointed out that 8bit microprocessors were too slow to use in real time digital filtering of audio frequencies. A system involving a number of INTEL 8085 microprocessors with shared memory was proposed. An actual implementation was built using two 8085 processors, hardware multiplication and shared

memory. A sampling frequency of 30kHz was achieved but it was not made clear how much of the improvement over a single 8085 was due to the hardware multiplication.

This signal processing application was an example where the multiprocessor approach was highly applicable in the absence of more powerful microprocessors at reasonable cost. The competition was not with minicomputers, which would have been prohibitively expensive, but with analogue devices. Thus the multiprocessor was the only feasible digital approach.

Rogers[5] in 1985 examined the effects of employing parallel computation using four Apple microcomputers with a shared hard disk. A finite element analysis problem on a single microcomputer took a total of 57 minutes to run. The problem was sub-structured to allow parallel computation on three microcomputers, with a fourth controlling the system. Not all of the problem was susceptible to parallelism and the result gave a run time of 27 minutes. The aim of the experiment was to minimise the rewriting of code when transferring from serial to parallel environments. The authors claimed success in the reduction in run time and minimal code changes, although the example was one which lent itself to parallelism. Interestingly the hardware consisted of standard personal microcomputers using a shared hard disk as the communications link.

Another system described by Wilton[6] in 1985 also used

a single hard disk as the database for a number of IBM personal computers to deal with planning and monitoring of the radar in the Shuttle Imaging Radar experiment (SIR-B). The machines were networked using the Ethernet local area network.

2.3 Summary

The literature concerning the use of grouped microcomputers as multiprocessors showed a polarisation.

One trend was to build up systems involving a number of microprocessors and their supporting chips with a shared memory for communications. This required hardware development, but gave a "tight" configuration allowing very strict control and fast commun ications. The shared memory was effectively a database for the whole simulation problem and was accessed as required by the individual processors almost as though it had been their own RAM. The negative feature was the inflexibility of the approach, demanding EPROM resident code with a resultant difficulty in software development.

The alternative approach was to use standard machines linked by a local area network and a hard disk for the sharing of information. Control was created by causing each of the personal microcomputers to read certain "flag" files on the hard disk. This "high level" approach clearly had advantages in flexibility and ease of development but speed of communications between the

processors was rather poor.

The multiprocessor system which was developed in this current research lay in between the two extremes described above. The aim was to use standard personal microcomputers to simplify software development along with a local area network to allow direct communication between the machines instead of via a hard disk file server.

CHOICE OF MULTI-MICROCOMPUTER HARDWARE

3.1 Introduction

Having decided to approach the problem of inexpensive computerised simulation by the use of a multi-microcomputer system, it was necessary to make choices about the hardware to be used and how the processors were to be connected to allow the necessary intercommunication.

A variety of methods have been proposed for data transfer within a multiprocessor system. The tightly coupled type uses a single shared memory for all the processors. A variation on this is for each processor to have its own local memory plus access to shared memory (via an additional processor). A loosely coupled system uses no shared memory. The processors have local memory and communicate via an input/output network. The two types are illustrated in figure 3.1.1. A tightly coupled system requires considerable hardware development before implementation. Thus it was decided to use a local area network (LAN) for the interprocessor communications to give a loosely coupled communications method.

3.2 RS423 Ring Network

The two network topologies which most commonly have been used with microcomputers are illustrated in figure

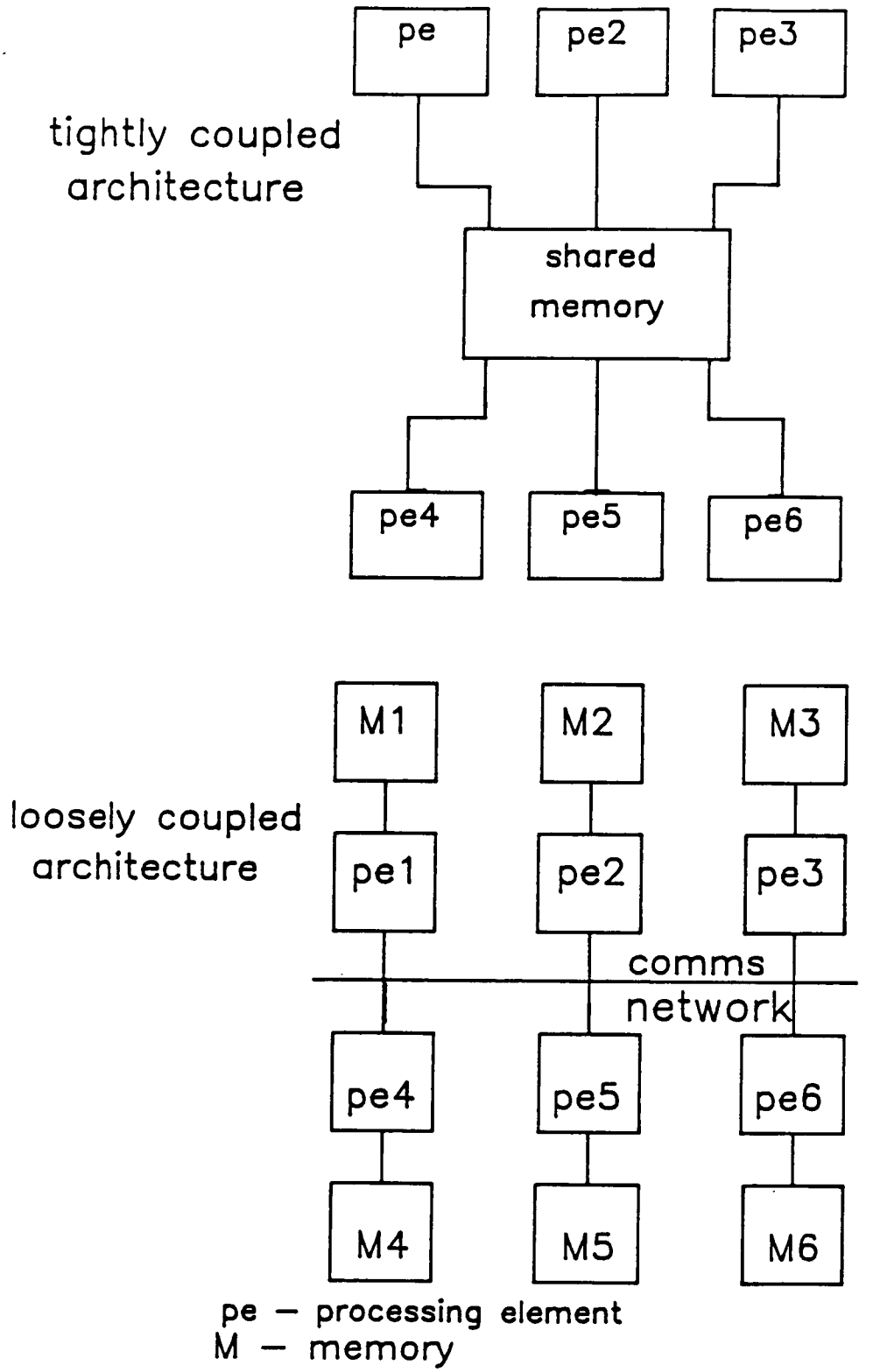


figure 3.1.1 multiprocessor architecture

3.2.1. ETHERNET[7] represents the bus network and CAMBRIDGE RING[8] exemplifies the ring network. In each case there is considerable hardware between the cabling and microcomputer, so these two implementations are rather expensive at up to £1000 per station for the aim of low cost simulation. Alternatives had to be considered.

Local area networks (LANs) have been commonly used for the transfer of large pieces of information such as swapping files between microcomputers and the sharing of hardware such as printers and hard disk storage. The requirement for the simulator was much more modest - the passing of a set of real numbers between the processors every second or so. As a result an experimental ring network was implemented using four Acorn BBC model B microcomputers (fig 3.2.2.) This involved the use of the RS423 interface provided on the machine. There was a 256 byte input buffer which could be used to hold messages until the processor was ready to deal with them.

The message format consisted of a header with the destination station code along with an ASCII string comprising the data. The message was passed around from machine to machine until it reached its destination. Each machine interrogated its RS423 buffer at the end of a processing cycle and sent on any messages not destined for itself. It also removed messages destined for itself and send out messages of its own.

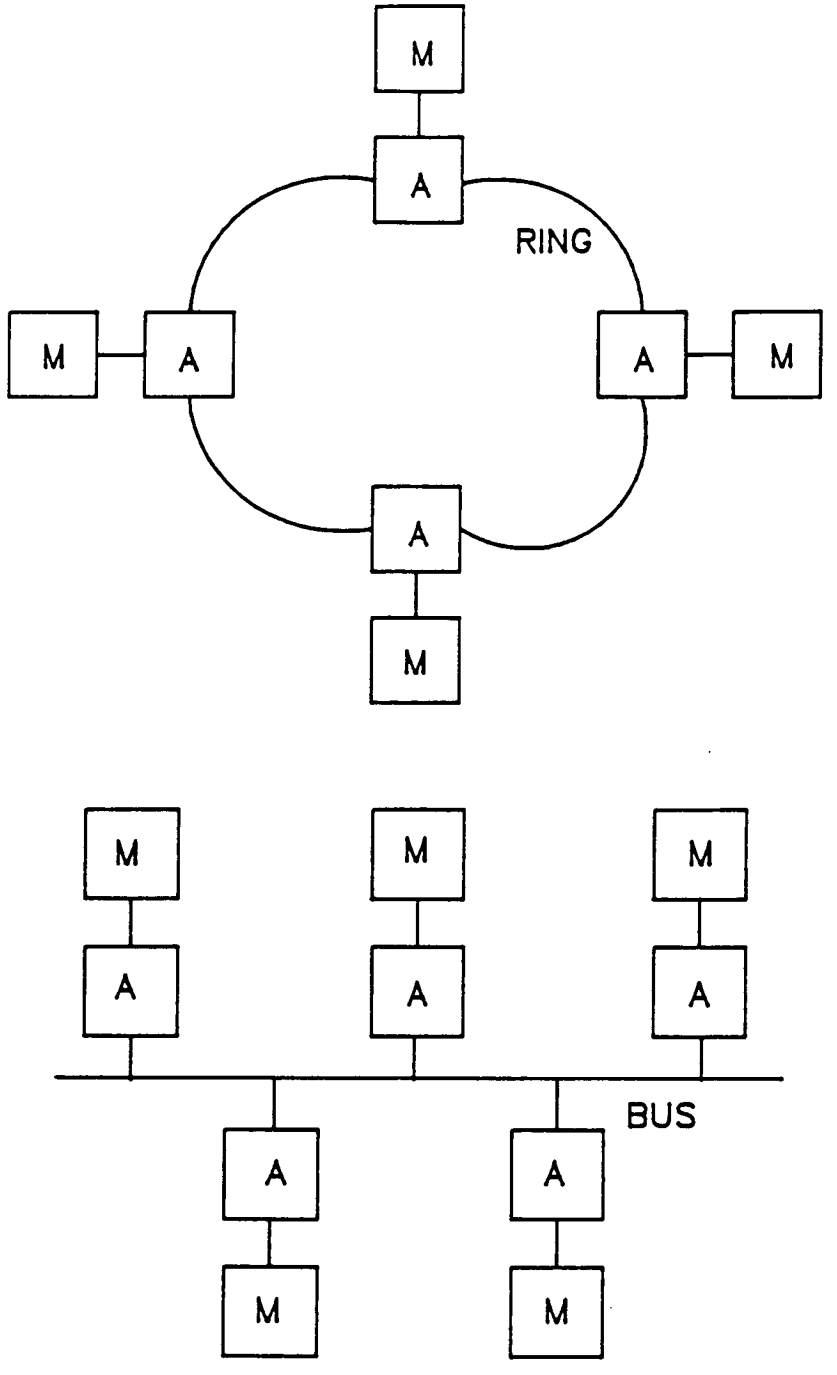
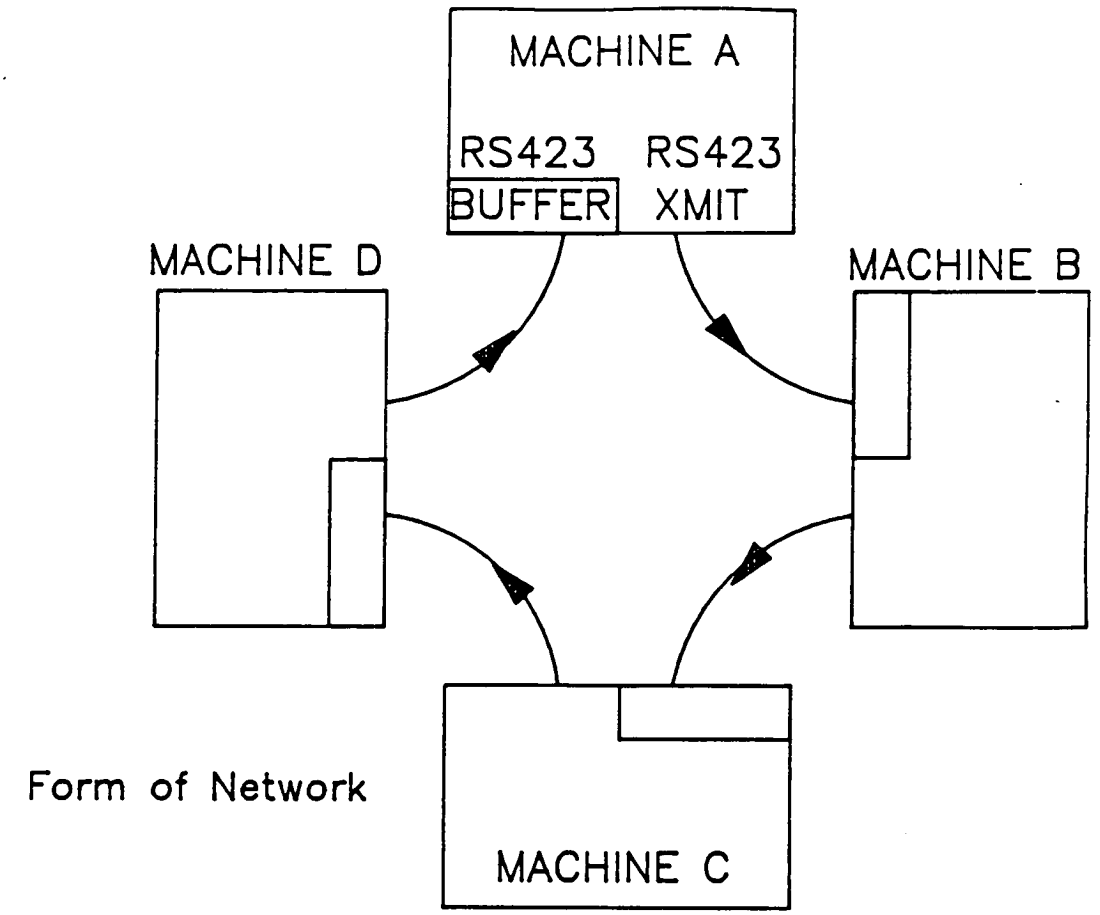
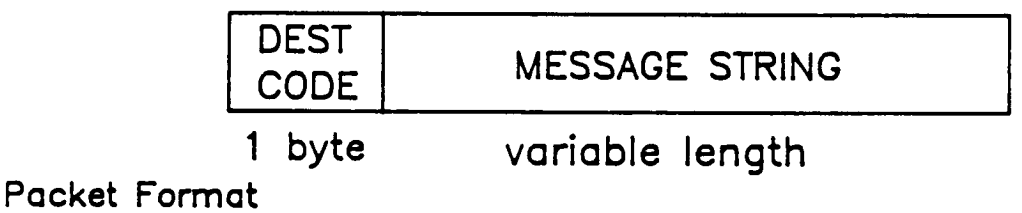


figure 3.2.1 network topologies





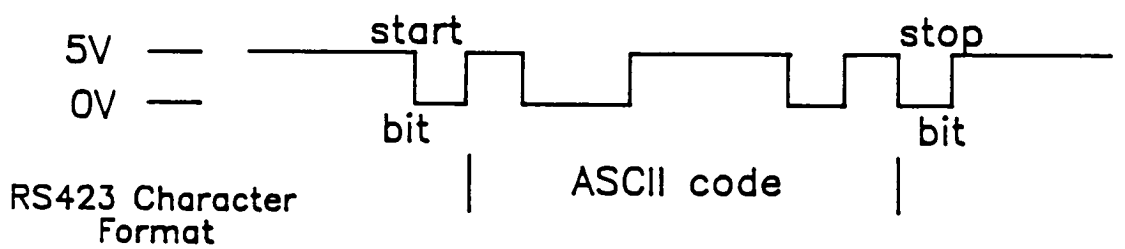


figure 3.2.2 RS423 ring network

This was a very cheap network system, with a hardware cost on top of the computer of about £1 per station. However the following difficulties were realised. There was no simple way of increasing the buffer size or interrogating the buffer other than at a set point in the program (BBC BASIC was used as the programming language so that interrupts were not really available). The transmission rate of 19200 baud was rather slow, but more importantly, the buffer was interrogated at each machine at the end of its processing cycle. If this was around two seconds in duration, it would have taken a message around six seconds to go from machine one to machine four. This delay would have become even more unacceptable with an increase in the number of machines on the network. It would have been possible to speed up the system by using external hardware but this would have been equivalent to redeveloping a LAN like CAMBRIDGE RING where the passing on and accepting of messages was performed by the LAN hardware with no overhead on the microcomputer. Costs could be kept low only if "off the shelf" mass-produced items were used. Hardware development would have been an expensive business and led to non-standard boards which would have been costly to produce and replace in case of failure.

3.3 Choice of Microcomputer

The choice of processor hardware must be introduced at

this point. Various constraints were imposed on the choice. The most obvious being that of cost, in view of the aim of inexpensive simulation. Secondly the ease and cost of networking was to play a vital part in the choice. Other factors were ease of use in terms of software development, past experience of particular types of microcomputer and availability of machines from suppliers.

The machine chosen was the Acorn BBC model B. It satisfied the criteria above and had the particular advantage of the promise of very cheap and effective networking.

3.4 Econet LAN

The Acorn Econet local area network (see appendix[1]) was similar to the "carrier-sense multiple-access with collision detection" system of the Ethernet LAN. It ran at a slower clock speed of around 200kHz instead of 10MHz but was cheap at about £50 per machine. Although developed for file transfer operations it could be used at the operating system and machine code levels since there was a set of documented operating system calls allowing the transfer of data between machines. Essentially this network system allowed a block of memory in one machine to be copied into another using a direct memory access. This could be carried out with or without the receiving machine's "knowledge".

"Cooperative transfers" required a control block containing destination codes and a buffer containing the data to be set up in the transmitting machine. The receiving machine also had to have a control block indicating source and destination codes and a memory buffer at least as large as the transmitted data. This type of communication was very secure in that a positive indication of the receipt of the data was passed back to the transmitting station. Figure 3.4.1 shows the nature of the transfer in terms of the contents of the memory of the two machines. The transmitting machine's control block contained the destination port, the destination machine's Econet station number and the machine addresses of the start and end of the buffer of data to be transmitted. The receiving machine's control block contained the source port and the source Econet station number of the intended transfer. It also held the start address of the buffer for the incoming data. The end address was given also for the buffer area but this was to prevent overflow and did not prescribe the size of the data to be transferred other than by setting a maximum length.

The Econet station number was set by a hardware switch in the microcomputer and ranged from 1 to 254. Each machine on the network had to have a unique station number. The port number in the control block was an identifier ranging from 1 to 255 which allowed a machine

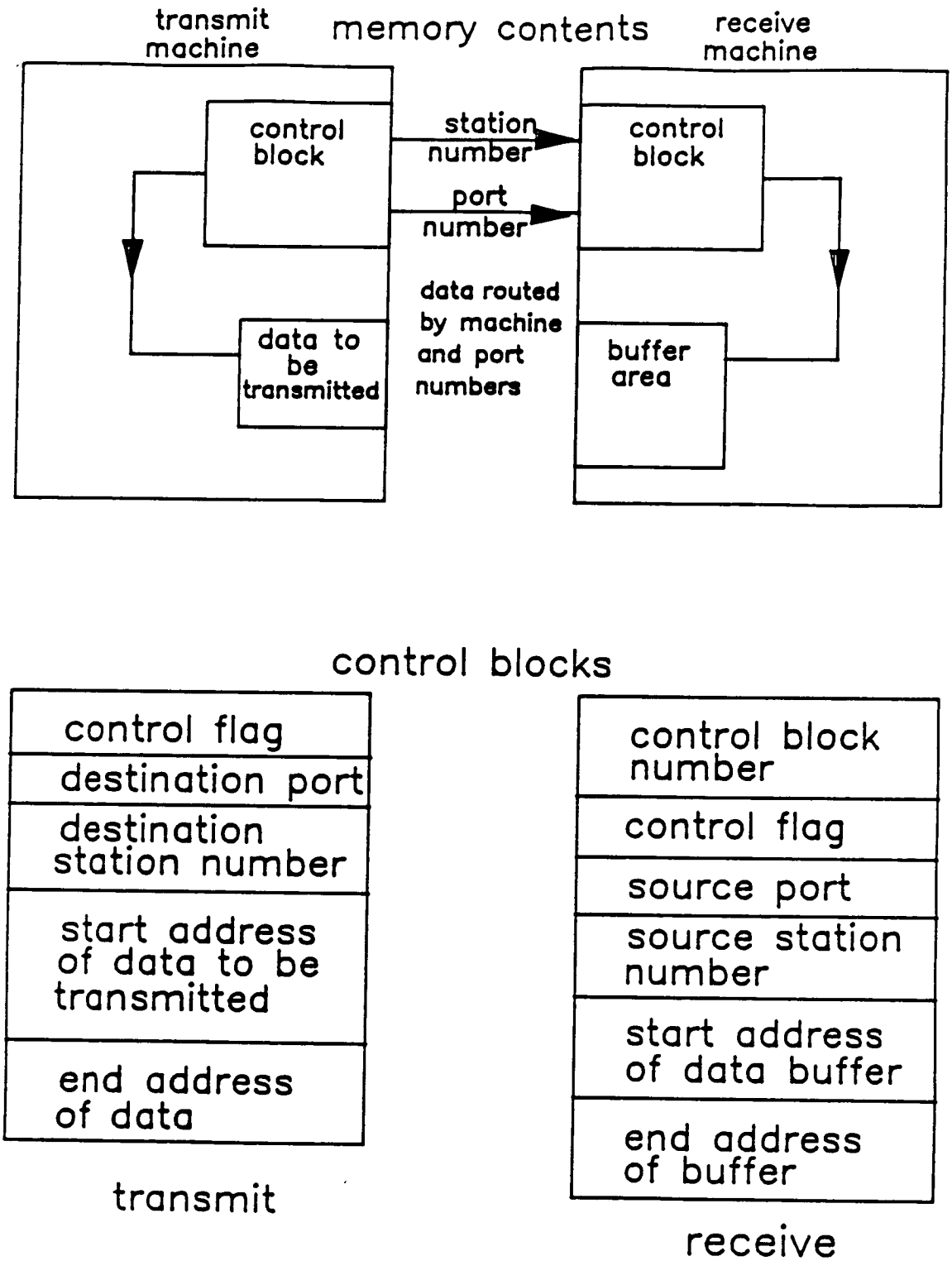


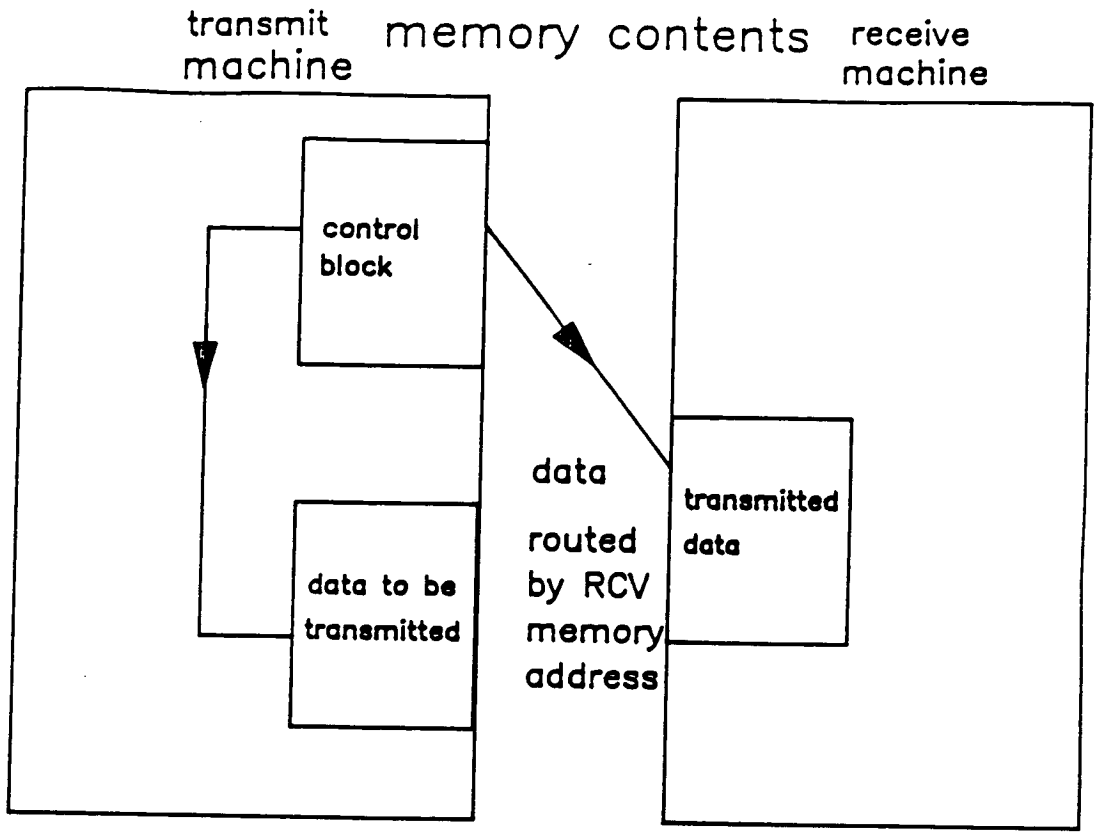
figure 3.4.1 cooperative transfer

to be set up to receive messages from a number of "sources" within another machine i.e. all the messages from one machine did not need to go to the same buffer in the receiving machine. For a transmission to be successful, the machine and port values in the receive and transmit control blocks had to match up.

The transmission was carried out by opening a receive control block in the receiving machine and then setting up a transmit control block in the transmitting machine. The first byte of the transmit control block was a flag which indicated when a transmission failed to start. This was tested and the block reopened if necessary. At the receiving machine, reception was polled and, when successful, the receive control block could be read to determine the source Econet station number and port number, as well as the size of the data buffer transmitted. (It was possible to open a receive control block without specifying a particular source station number or port). At the transmitting machine the success of the transmission could be polled and error codes returned if unsuccessful.

With "immediate transfers" there was no requirement for a control block to be set up at the receiving station.

The transmitting station had to have the machine address for the destination of the transmitted data in the receiver in its control block. This is illustrated in figure 3.4.2. This method had clear limitations since it



&82
&00
destination station number
start address of data to be transmitted
end address of data
address of destination in RCV machine

transmit control block for a "poke"

figure 3.4.2 immediate transfer

depended on intercommunicating machines possessing considerable information about each others' memory maps.

When an immediate transfer was carried out, although no receive control block was required, it was necessary to decide beforehand to which address in the receiving machine the message was to be sent. A control block was set up in the transmitting machine and a "poke" carried out. It was possible to poll the transmission and return an error code if unsuccessful. There was no operating system call for use at the receiver to indicate that a transfer had occurred. This had to be detected by the user software.

3.5 Summary

The philosophy adopted with the hardware design was to obtain as much as possible off the shelf. This led away from the development of a custom built networking system. Commercial general purpose LANs were considered and the extemely low cost of Econet confirmed that the BBC model B, a machine with which considerable experience had been built up, was the microcomputer which would be used.

THE INITIAL SIMULATION

4.1 Introduction

The initial work undertaken in this project dealt with the modelling of the LPG tanks and pipe network of a liquefied gas tanker and hence the solution of the associated algebraic and differential equations. It was only towards the end of the first year that any progress was achieved with the LAN due to the lack of proper documentation of the Econet operating system calls. This chapter describes the first model which used the Econet system. The requirements of the communications system are also discussed. Details of the modelling of networks and tanks are given in chapter 6 and details of the communication methods in chapter 5. The main purpose of this chapter then is to give an appreciation of the type of problem first attempted on the multi- microcomputer and the nature of the data transmission requirements for an LPG system simulator.

4.2 Parallelism and Disposition

Having made the choice of hardware to implement the system, the first consideration in assessing the feasibility of the method was to answer the question "How can the model be successfully distributed over a number of processors?". Clearly the simulation problem had to involve parallelism. There had to be elements within the problem which could be solved simultaneously without

requiring the results from any of the the other parallel processes. An example would be the system of two tanks connected by a single valve. The calculations for the pressure in each of the two tanks had to be able to be carried out simultaneously in two separate machines without any loss in accuracy.

There was a spectrum of problem partitioning methods available, ranging from a "geographical" disposition where each physical item of plant was modelled by a separate processor to the functional approach where each processor handled only one class of mathematical problem. It was felt that a geographical approach to partitioning the simulation of the cargo system of an LPG ship would lead quickly to a first estimate of the suitability of the overall approach and minimise the communications between the microcomputers. This topic is taken further in chapter 8.

4.3 Overall View

The system being modelled consisted of two LPG tanks and a liquid flow network including a pump and kickback loop. This is shown in figure 4.3.1.

Three machines were used for the model, one for the liquid network and one each for the tanks. This three machine system was the simplest that could be used to prove the value of the communication method since a system of only two machines could have been connected

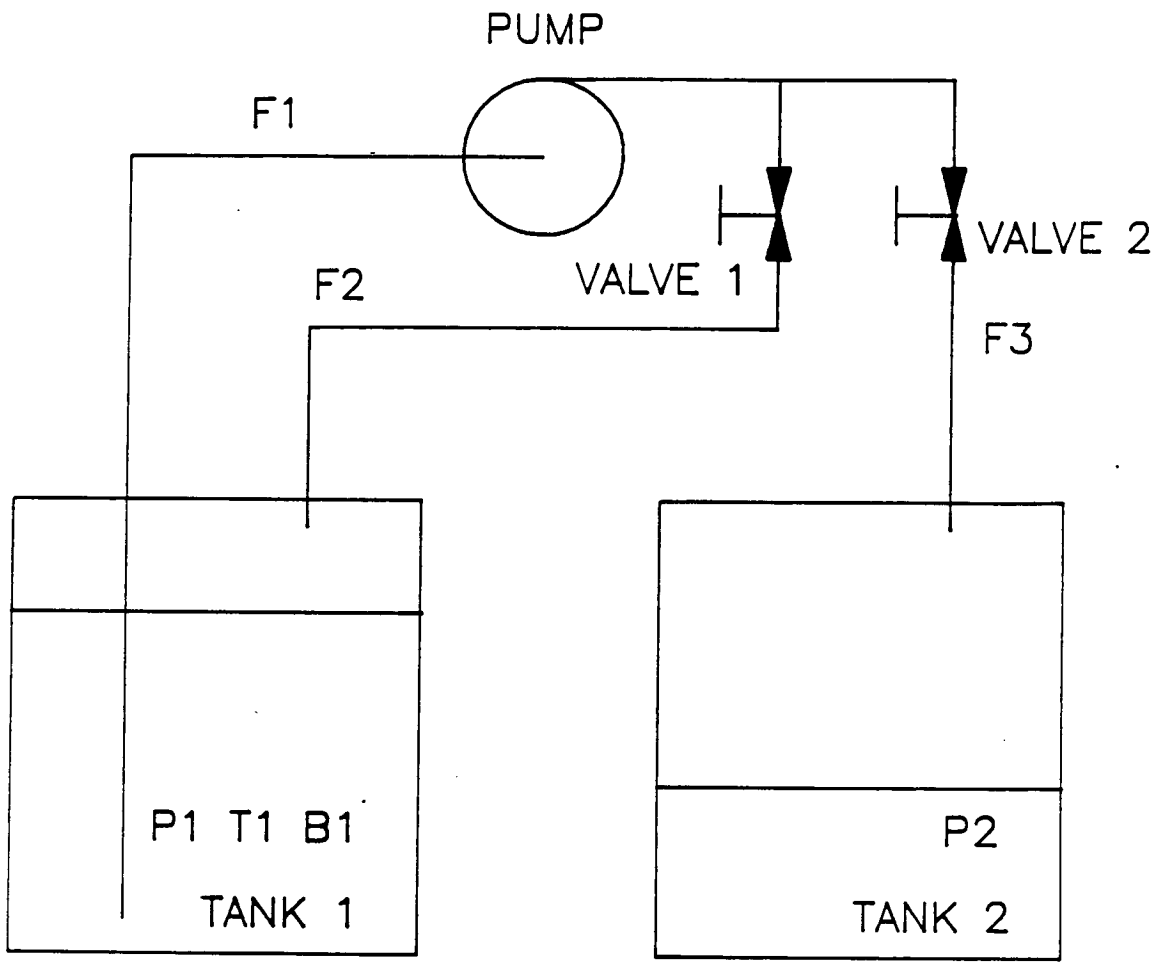


figure 4.3.1 two tank model

with a direct link via the RS423 serial communication ports provided as standard on the BBC B machine.

Only liquid flow was modelled in the network, although the tank processors modelled both liquid and vapour and carried out the vapour liquid equilibrium calculations. The network consisted of a pump and three pipes with two pipes going to tank 1 and one to tank 2. It was assumed that the flow would always be from tank 1 to tank 2 (i.e. non return valves had been fitted in the pipes). The liquid network solver also synchronised the operation of the system by sending control bytes to the other two machines and checking when they had finished their time steps. The valve and pump controls too were handled by this machine via analog inputs from sliding potentiometers. The processor layout and information flows are shown in figure 4.3.2.

A pictorial representation was displayed on the graphics monitors associated with each of the three machines. This displayed a mimic of the plant and the flow rates in the pipes.

The main task of the tank processors was to calculate the tank pressure at the end of each time step. This involved the solution of the differential equations for mass and thermal balance.

$$dM/dt = \Sigma Fin - \Sigma Fout$$

$$dH/dt = \Sigma Hin - \Sigma Hout$$

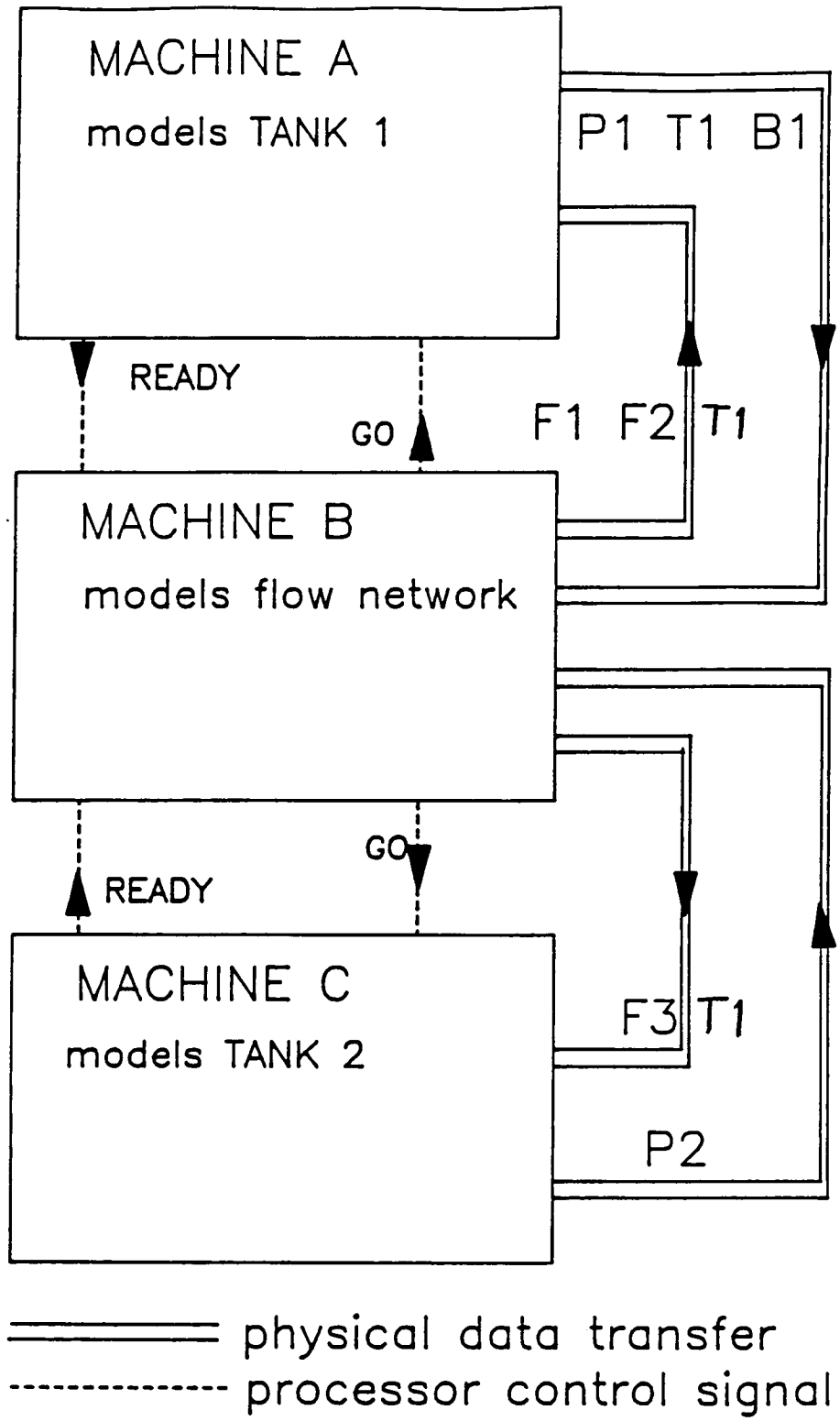


figure 4.3.2 implementation of two tank model

The third processor was used to solve the algebraic equations for the pipe flows. This involved an iterative binary chop using linearised models for the flow through the valves. Square root models had been used but were abandoned in favour of linear ones to speed up the solution. Although this meant that an analytic solution could have been used, the equations were still solved numerically since it was realised that this would be required in more complex models.

The flow expressions were

KICK * KVkick (Ptank1 - Px) = kickback flow

TRANS * KVtrans * (Px - Ptank2) = flow into tank 2

Kpump * (PUMP * PS + (Ptank1 - Px))= flow from tank1

where Px was the unknown pressure, Ptank1 and Ptank2 were the tank pressures, KVkick and KVtrans were the valve constants, Kpump was the pump flow constant, PS was the pump shut off pressure and KICK, TRANS and PUMP represented how far opened the valves were and the rate that the pump was running.

These were rearranged into a suitable form for a "binary chop" to enable the unknown pressure, and hence the flows, to be found.

The tank equations were solved to produce a value of pressure (the tanks could be designated as pressure setters, see Ponton[9]) and the network equations were

solved for flows (and the network model could be designated as a flow setter.) Strict synchronism was employed so that the pressure setters carried out their calculations, then the flow setter did its calculation and so on. Note that this meant that the flow setter was idle when the pressure setters were operating and vice versa. The idle periods were utilised for graphics screen updating.

4.4 The Data requirements

Throughout the project the intention was to have actual system cycle times in the order of one second. This produced a satisfactory refresh rate when viewing the colour graphics output. In practice the times were somewhat longer. However on an LPG ship the system was extremely slow moving and in reality variations would be noted over a period of minutes rather than seconds. The simulation had to run much faster than real time to be useful due to the long times taken to fill tanks on an LPG ship.

The data transfer was arranged as the sending of individual variables to named addresses by the immediate Econet transfer. For the pipe from tank 1 the temperature, pressure and vapour fraction were transmitted. Tank 2 was required to send only the tank pressure. The information from the network solver consisted of the flow and temperature for the pipe to

tank 2, the flow and temperature for the kickback pipe and only the flow for the other pipe to tank 1.

For a complete cycle, the tanks sent four real variables and the network solver sent five. These were five byte reals so the data transfer amounted to 45 bytes. The Econet LAN could operate at a maximum of 20 kilobytes per second. So, even allowing for the transmission overheads, the full bandwidth of the communications network was barely utilised.

4.5 Summary

This simple example served to show that it was feasible to use a LAN for interprocessor communications in a multi-microcomputer. It was apparent that more complex models would be required, and for a full system, more machines would be added for the additional plant. However the bandwidth of the LAN was shown to be more than adequate. It would allow hundreds of times the present traffic. The basic principle had been established.

5.1 Introduction

This chapter describes the development of the software for the multi-microcomputer communications. It evolved through three stages. The first two were directly concerned with getting the immediate and cooperative Econet commands to work. The third stage was concerned with producing a packet system to minimise machine dependency and to provide a "user friendly" environment for the modeller. This third stage represented a large part of the total effort in the project.

The code for each of the communications methods is given in appendix[2]

5.2 Communication Using Immediate Commands

The first system developed using Econet for the interprocessor communications was very simple. Three machines were used, as described in chapter 4, two modelling tanks and the third modelling the kickback loop flow network involving valves and a pump. The communications between the processors used the immediate calls. This meant that the data was transferred (poked) into the receiving machine at memory addresses which had to be "known" by the sending machine. Also the software in the receiving machine needed to take no action to initiate reception. The data just appeared in the memory

locations without warning. A problem existed here in that the processor could have been carrying out memory reads during which time the data might have changed in a variable. A corrupt value would have resulted. This problem and that of synchronising the multiprocessor were tackled by using the flow network processor as a master machine. This ensured that the three machines kept in step by receiving and sending single byte flags. These were transmitted to indicate the end of processing in a given machine and that the data which had been transmitted was valid. The flow of processing for the machines is shown in figure 5.2.1.

A difficulty in the data transfer was that while BASIC dealt with variable names, the Econet calls could deal only with machine addresses. A machine code routine was developed which allowed the machine address of a BASIC variable to be found. A number of variables, equal to the number of communicated process variables, were declared and had their addresses found. The "communicate" procedures involved transmitting the predeclared variables. These were poked into the corresponding addresses in the receiving machine's memory. There was no retrieve process since the values simply appeared in the predeclared variables at the receiver.

The communications scheme was rudimentary with each process variable being poked into an individually allocated memory location. For the flow network

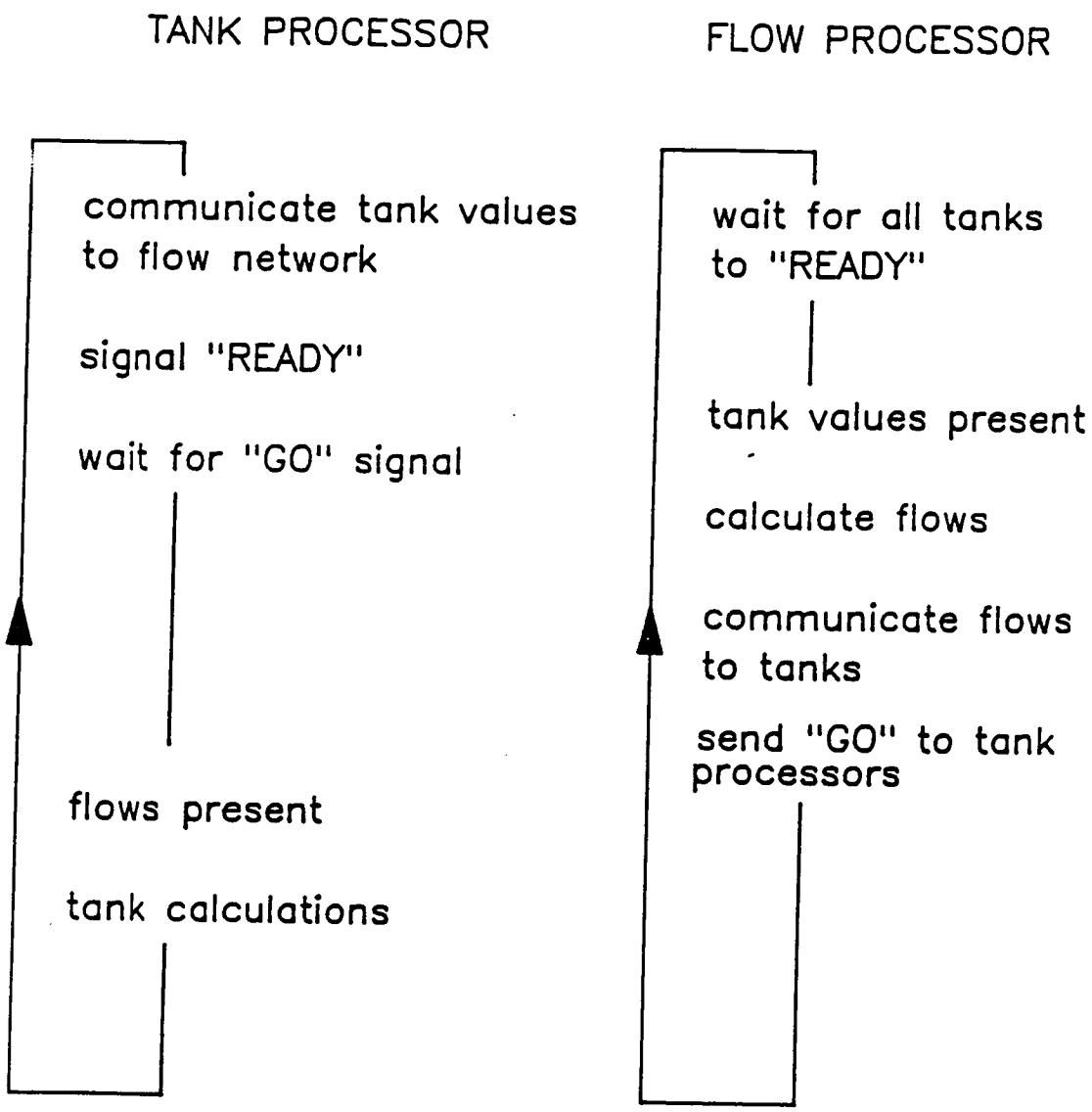


figure 5.2.1 processing flow for immediate implementation

processor this involved four locations of five bytes each for the pressure, temperature and vapour fraction for the two tanks and two single bytes for the the tank ready codes. It was clear that, although this was a very fast and direct method of communication, if many more processors were involved, it would have become too clumsy. The problem was that, for each additional pipe connection to a processor, a memory address had to be held in the transmitting machine for each of the process variables associated with the pipe. For this small system the addresses were passed between the machines automatically during initiallisation. However this requirement would have become more involved with additional processors and would have hindered development.

5.3 Communication using Cooperative Commands

It was decided to use cooperative calls to remove the need for a given machine to have to use machine addresses located in remote machines in order to communicate with these remote machines. Instead the receiving machine set up a receive control block which contained the address of the area of memory which acted as input buffer, the Econet machine number of the source of the transmission, and a port number which allowed a distinction between different data flows from the same machine. The sending machine also set up a control block with the memory

address of its output buffer, the Econet number of the receiving machine and the port number used at that machine. There was no need for any machine to have addresses for another machine. A further advantage was that after transmission a test could be made to ensure that reception had occurred.

Using this more sophisticated method a larger system was built up. Initially three tanks were modelled along with a liquid network. At a later stage a compressor condenser liquefaction system was added resulting in six machines being used for the modelling with a further machine for operator control of the valves and pumps.

For a given processor, each of the pipelines entering that part of the model, and therefore requiring data from another processor, was assigned a communications port. This meant that an individual buffer of data corresponding to that pipeline could be sent. Figure 5.3.1 shows the port connections for a three tank model and liquid network with fill, discharge and strip lines. The data transferred from a tank to a port in the liquid network solver consisted of tank pressure, temperature, vapour fraction and the valve constant associated with the pipeline. The data returning to the tank processor port consisted of liquid flow, temperature, and vapour fraction.

As in the immediate transfers, the Econet primitives referenced the data by machine addresses whereas BASIC

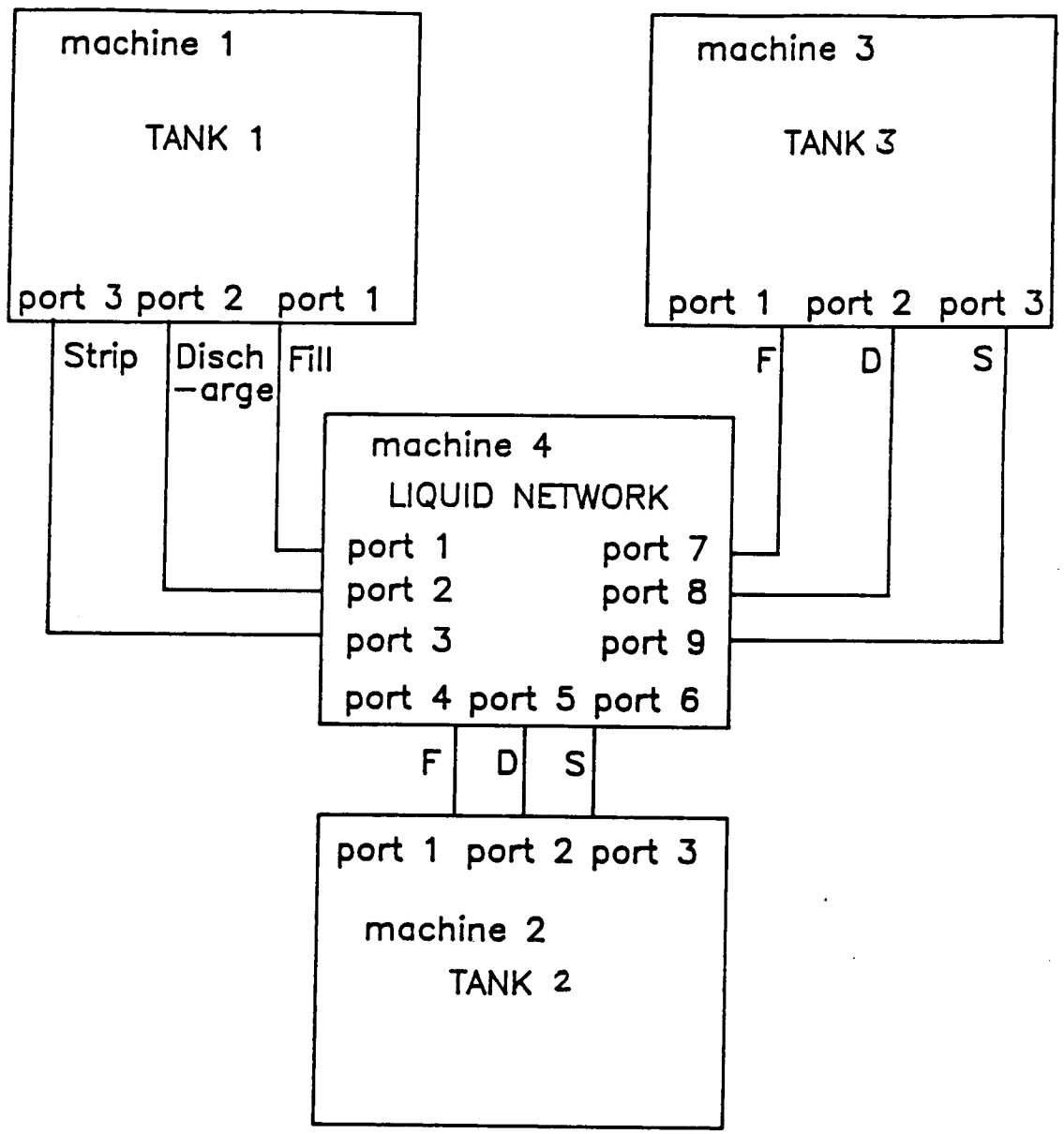


figure 5.3.1 port allocation for three tank model

used named variables. To simplify the transfer, in both the receiving and transmitting processor, the buffer consisted of an array of memory whose machine address corresponded to the start of a real array. This allowed the data for a given pipe to be assigned to an array before and after transmission. The form of the data transfer between two machines is illustrated in figure 5.3.2.

Although cooperative transfers were used for the process values, the synchronisation and valve control signals from the operator console machine were transferred using the immediate calls.

It was appreciated that the communications software developed so far had a number of deficiencies. Firstly, it had to be tailored especially to the particular model in the processor. Secondly, in order to prevent the use of additional machine address dependent transfers, the items such as valve constants and pump shut off pressures were tranferred as part of the pipeline information along with the process values. Thirdly, the valve controls and synchronisation data used the machine address dependent immediate call. Finally, there was an operating system limit of sixteen to the number of ports which could be used with any one processor. This strictly limited the number of pipes connected to the model in a single machine and hence a more complex flow network requiring more than sixteen pipe connections could not be

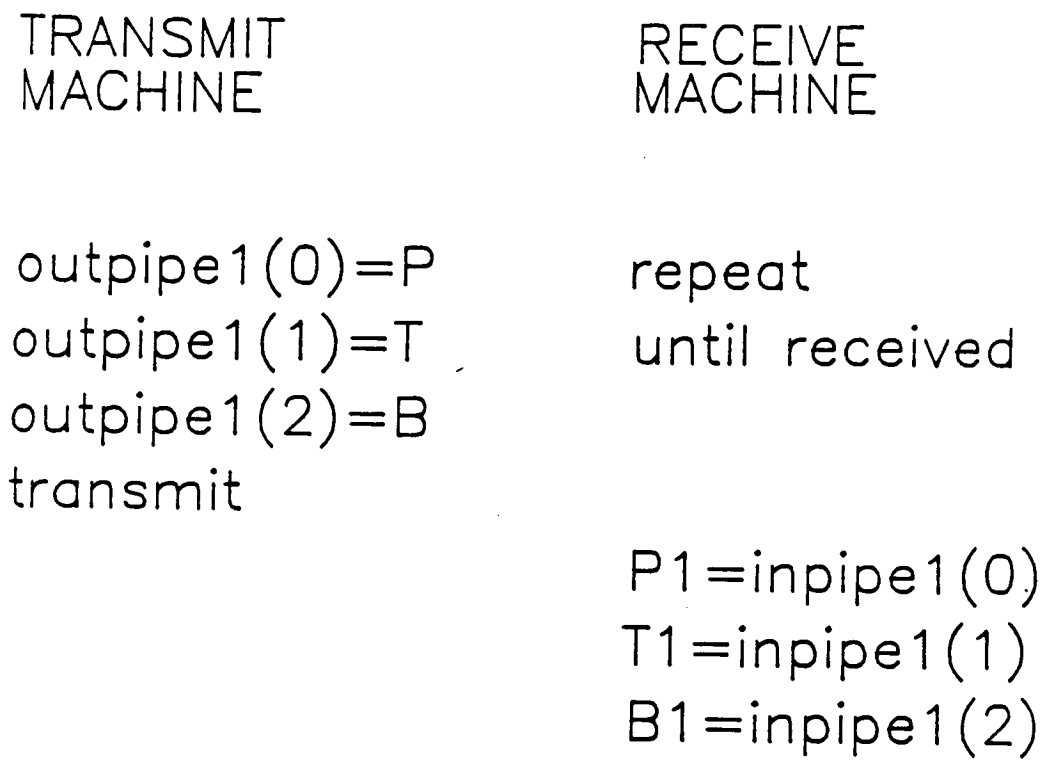


figure 5.3.2 example of cooperative communication coding

implemented.

5.4 Packet Switching Software

At this stage it was decided to design a communications system which would have a high degree of transparency. The intention was that the modeller need not be aware of the details of memory areas and packet structures. The overall approach would be to have only one input port to each of the modelling processors and to remove as far as possible the need for processors to "know" in which machines different parts of the model resided. This approach would also remove the restriction imposed by the maximum of sixteen receive ports being active simultaneously on a single machine.

To allow this, the simulation model was divided into a number of "areas", each to be run on one machine. These areas were numbered for reference. A notation analagous to that used in flowsheeting programs was chosen. All the pipelines entering an area were numbered sequentially so that the variable values associated with fluids flowing in them could be assigned. Similarly the valves and pumps were numbered in sequence so that the control actions relating to them could be assigned. An example of this is given in figure 5.4.1. The area number was multiplied by 100 before being added to the item value to give the reference number for the item.

eg. pipe 402 was the second pipe in area 4

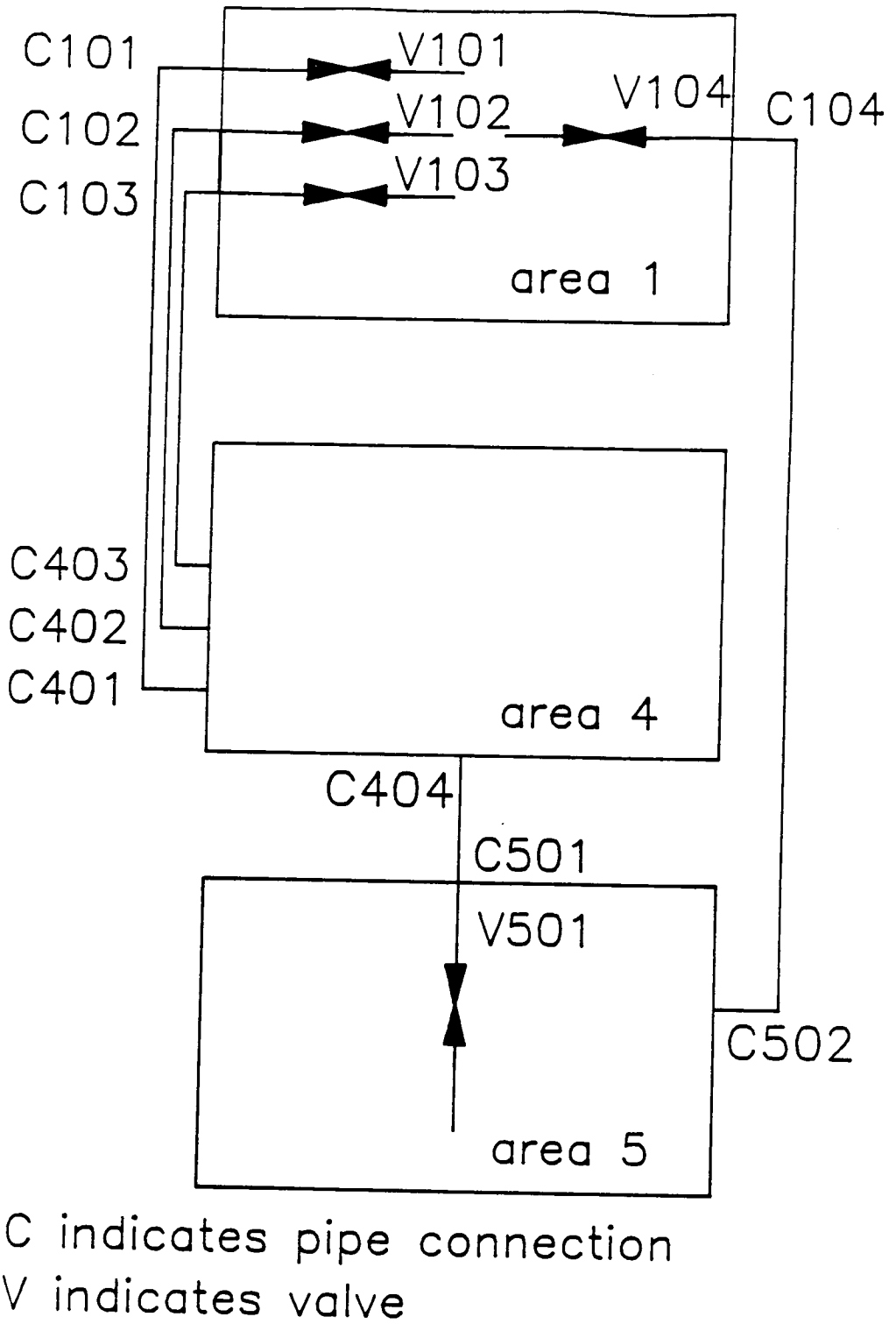


figure 5.4.1 example of packet switching number scheme

The unit of communication was the packet. Different types of packet were implemented which were distinguished by the nature of the data carried. All packets had the following common fields: packet length, destination area, source area, packet type, destination item and a terminator. This is illustrated in figure 5.4.2. The physical data packet had fields for pressure, temperature, vapour fraction, concentration1, concentration2 and flow. Control packets carried a single data field as did "shut off" packets which were used to carry emergency shut down signals.

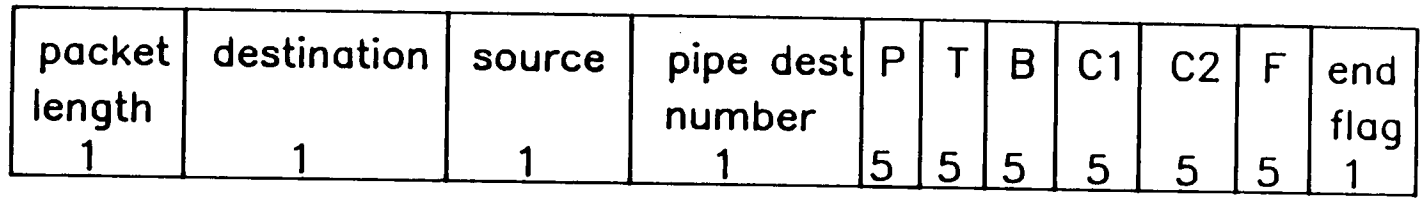
Clearly other types of packets could have been added to the system. There was a constraint of a total packet length of 255 bytes due to the packet length field consisting of one byte.

The packets to be sent from a given processor were assembled in a buffer area in its memory and a cooperative call was used to transmit the buffer to a processor whose sole purpose was organising the communications. This "sorter" unpacked the buffer, placed the individual packets into outgoing buffers for the destination processors, and returned a buffer containing any packets waiting for the transmitting processor. This is outlined in figure 5.4.3.

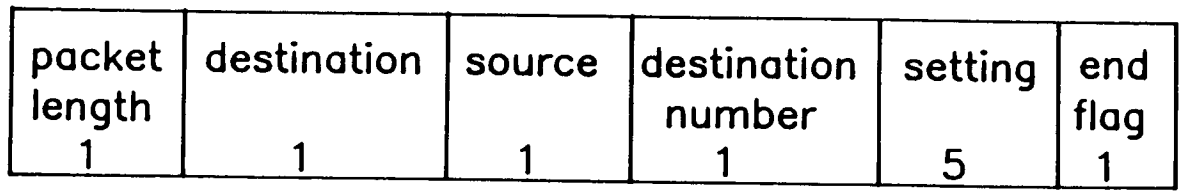
Since the viability of this approach depended on the speed of the sorting processor, its software was written in assembly language.

figure 5.4.2 packet switching packet structure

PHYSICAL DATA PACKET



CONTROL PACKET



number of bytes included in each field

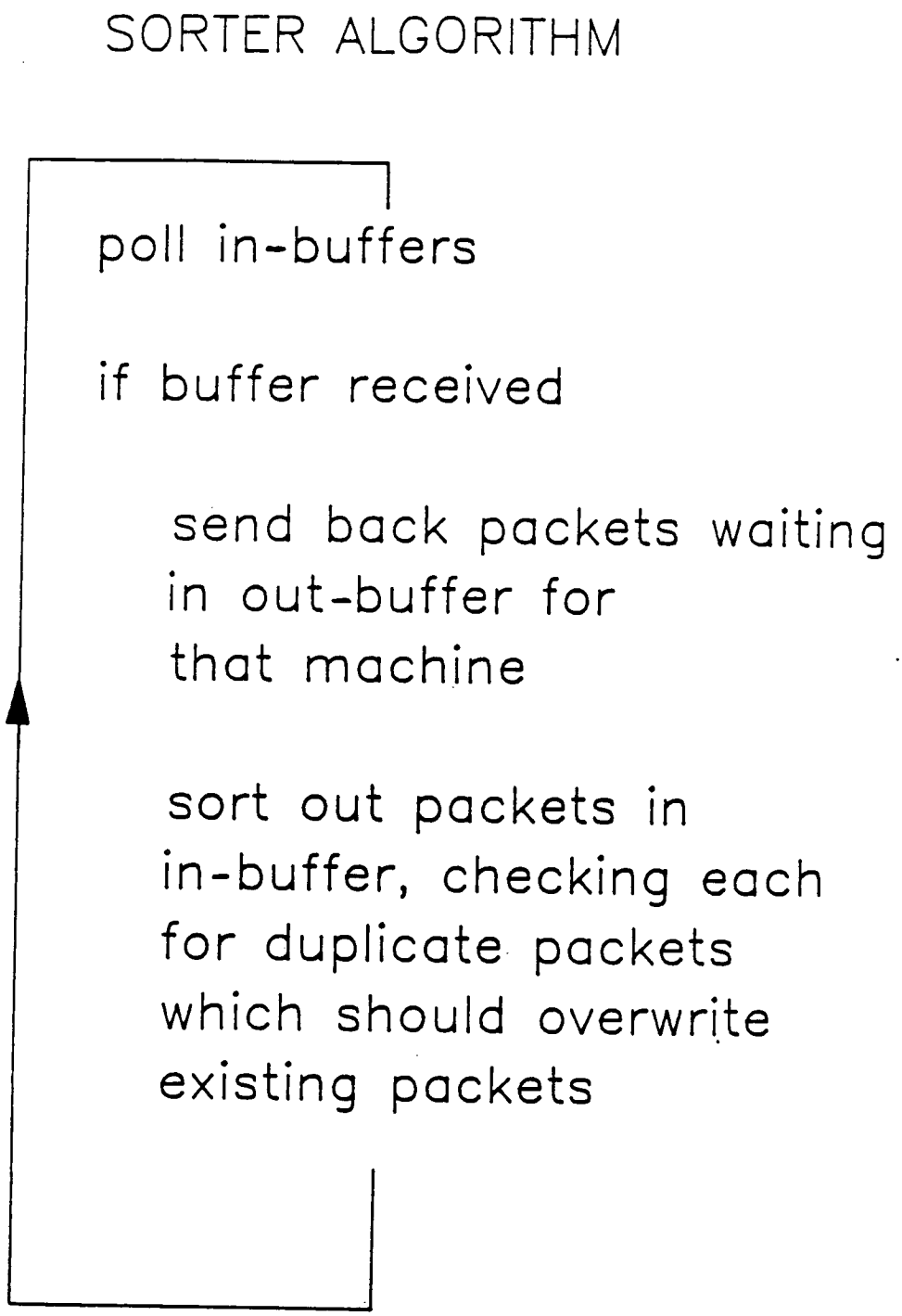


figure 5.4.3 processing flow of packet sorter

A restriction existed concerning the size of the buffers used. This was imposed by the size of the memory of the sorter processor. Having ascertained that the memory was large enough to take each possible packet once, there was a possibility of overflow if a particular processor were slower than average and duplicate packets built up waiting for transmission in the sorter buffer. This difficulty was overcome by a check during buffer unpacking to identify any packet with identical features to one already stored in the appropriate out buffer. If this were the case, then the most recent packet was written over the former. Two packets with identical features had the same length, type, destination and source but could possess different data. Although this check took time it was vital to avoid buffer corruption.

The aim here was to make the sorter as general as possible. It would have been simpler if a buffer slot had been assigned to each expected packet. Then the overwriting of duplicates would have occurred automatically and the sorter would have been faster. However this would have imposed the restriction of the sorter requiring details of the packets which it would receive.

To make the communications more user "friendly", a set of routines was written to allow software development to be carried out with very little knowledge of the Econet machine calls. These involved a number of variables

named appropriately "out_pres", "out_temp" etc. in which the values to be transmitted were placed, and a number of functions such as "get_pres" and "get_temp" which allowed the retrieval of received process values. These are illustrated in figure 5.4.4.

This packet switching communications system was successfully applied to the largest simulation problem in this project. Nine machines were used to model the LPG ship cargo system. One of these machines was a 32016 32bit processor programmed in Pascal which modelled the liquid, vapour and condensate flow networks. The floating point representation in BBC BASIC was different from that of the Pascal implementation. It was decided to use the BBC BASIC format for the communications and do all the conversion in the 32016 co-processor, since it was extremely powerful. To have carried out this conversion in the BBC B machines would have caused a large processing overhead.

The LAN was used for other types of transfer as well as for passing process values between the processors. In particular, each of the machines had to be loaded with a program before the start of the simulation. This was done quickly and automatically over the network by using immediate communications. The machines could also be started running by an operating system call which entered text into the keyboard buffer of the desired machine.

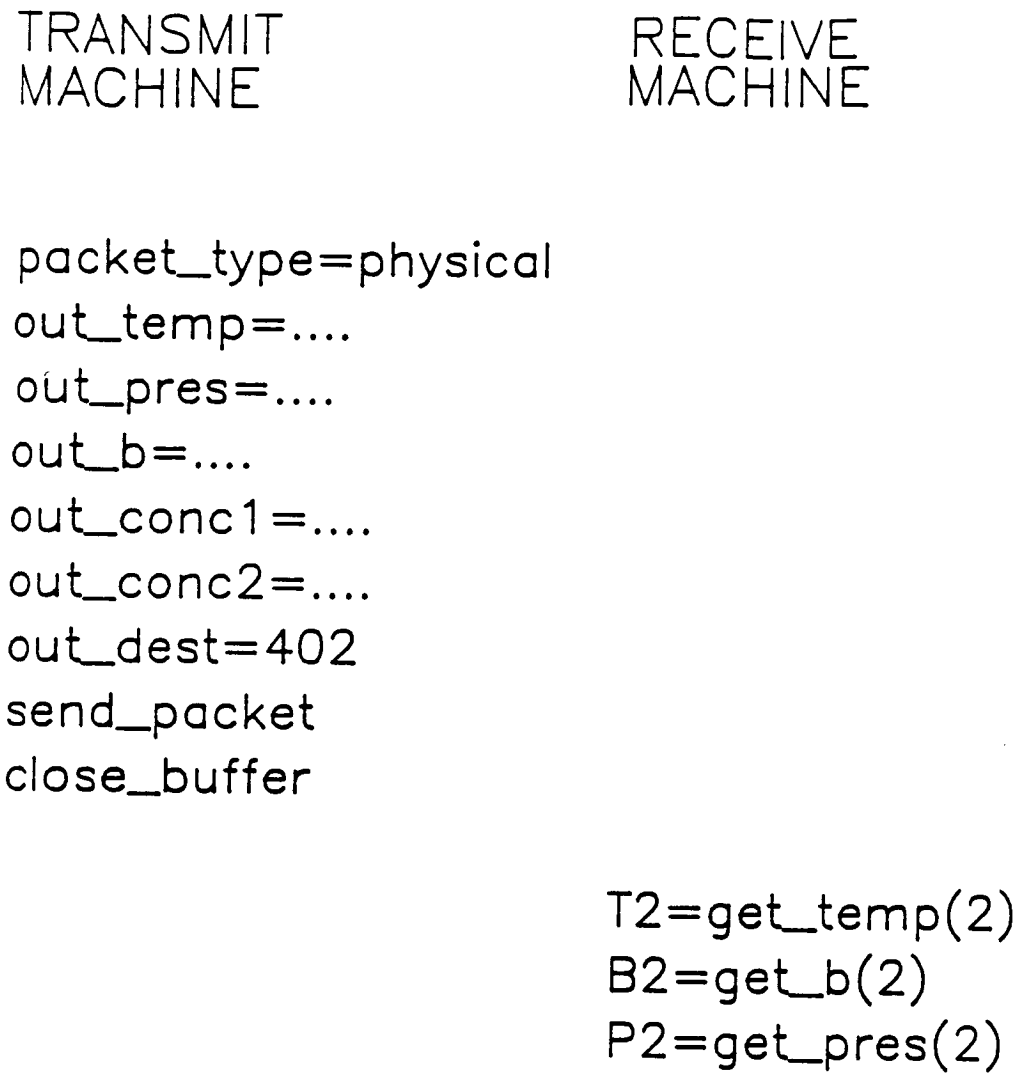


figure 5.4.4 example of packet switching code

5.5 Software Development

Comparing development of software for the processors of a multimicrocomputer with that for a single computer, the modelling techniques for the simulation were the same and modular development was required. Once a piece of software was developed for an item of plant, it could be duplicated for use in an additional processor if a second item of the same type were required. However control when testing the system as a whole was clearly not so great as with a single machine.

It was important to be aware of the effects on the multi-microcomputer caused by the communications network. Safeguards would be required in the software to take account of these following.

a) Each processor was working on its own version of the overall problem database in its local memory. Depending on the effectiveness of the network implementation, this information may have been significantly different from values held elsewhere. An example would be the flow between two tanks existing in separate processors. The simulation would become less convincing if the flow rates at each end of the connecting pipe were different.

b) The speed at which data could be passed between the processors. Clearly this was a factor contributing to a). There were two components in assessing the speed. One was the transport time due to the physical data rate

on the network and the other was the processing time, i.e. the overhead in preparing and decoding the data before transmission and after receiving. Since the LAN was fairly fast (200 kilobits per second), the latter predominated. In the more severe cases this time delay due to communications would cause instability in the simulation.

This effect is illustrated in the operation of the other systems modelled in chapter 8.

c) There was a possibility that an individual transfer of data would fail. At best, it would cause a hiatus until retransmission occured and at worst, it would cause a "catastrophic" failure of the simulation.

5.6 Summary

The use of a geographical distribution of the simulation model over the processors allowed the communications to be organised as groups of data associated with individual pipelines. This was easy to conceptualise and allowed a simple transfer from the flowsheet stage to the software.

The immediate Econet calls were found to be very fast but cumbersome as they required the use of machine addresses which were only obtainable at run time and had to be passed to the other processors.

The use of the cooperative calls removed the need for remote machine addresses but allowed only a maximum of

sixteen ports per machine. This meant that data had to be grouped together to minimise the number of ports used.

The packet switching system was the most sophisticated interprocessor communications method developed in the project, allowing the transfer of a large number of different packets to a given machine and transparency for the writer of the modelling software. Only the sorter processor needed to "know" which machine was modelling each area of the model. The disadvantage of this method was two fold. Each message was transmitted twice over the network, to the sorter and then to its destination. Also time was taken to pack and unpack the buffers at the sending and receiving processors. The resulting reduction of speed in the data transfer had to be balanced against its ease of use. There would be simulations involving short time constants which could not be modelled using this approach.

6.1 Introduction

This chapter describes the development of the simulation models. The items of plant are dealt with here separately before considering the complete ship model in the next chapter.

The modelling of LPG in tank, condenser, compressor and flow network is described. The modelling of LPG in a tank results in the evolution of a continuous simulation language applicable to BBC BASIC and the initial single node flow network model is developed into a reconfigurable network solver.

6.2 The Tank Model

The first element to be modelled was a closed tank with inlet and outlet flows with characteristics shown in fig 6.2.1. The tank was assumed to contain a single component LPG with liquid and vapour in equilibrium.

There were essentially two types of solution required for this. The solution of the differential equations associated with the thermal and mass balances:

$$dM/dt = \Sigma Fin - \Sigma Fout$$

$$dH/dt = \Sigma Hin - \Sigma Hout$$

where M=mass H=enthalpy F=flow

and the solution of the algebraic equations associated

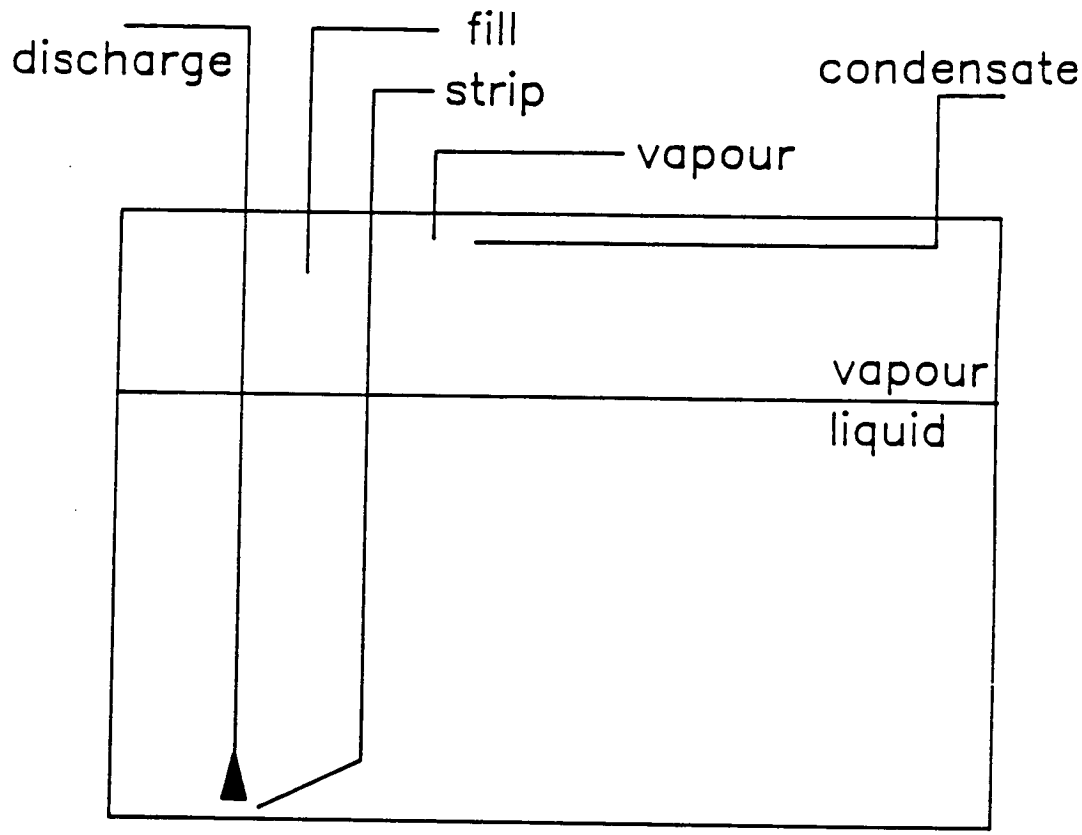


figure 6.2.1 LPG tank

with the liquid vapour equilibrium:

$P = \exp[a + b/(c+T)]$

$H = M.[B.(SLH + Cvap . T) + (1-B) . Cliq .T]$

$V = M.[(1-B)/Dliq + B/Dvap]$

where P=pressure, T=temperature, H=enthalpy

a,b and c =Antoine coefficients

SLH=specific latent heat of vaporisation

Cvap=specific heat capacity of vapour

Cliq=specific heat capacity of liquid

M=mass of tank contents

B=vapour fraction

V=volume of tank

Dliq=density of liquid

Dvap=density of vapour

The solution of these equations reduced to the problem of calculating the value for "B" the mass vapour fraction. It was found that the simple binary chop and secant methods were too slow, due to the number of iterations to give the necessary accuracy. The method decided upon was Wegstein which gave a quick convergence, speeding up the tank solution by a factor of two. The solution started with a check that the vapour fraction lay between one and zero, i.e. the tank contained liquid. Then the previous value of vapour fraction was used as a first point for the Wegstein iteration which is a direct substitution extrapolation method, see Franks[10]. If the tank contained only vapour then the solution was

greatly simplified. The temperature and pressure could be found from the two equations:

$H = M . (SLH + Cvap . T)$

$P = M . R . T / (V.MW)$

where P=pressure, T=temperature, H=enthalpy

SLH=specific latent heat of vaporisation

Cvap=specific heat capacity of vapour

M=mass of tank contents

V=volume of tank

MW=molecular weight of vapour

R=universal gas constant

The solution of the differential equations was approached by using each of first, second and fourth order Runge Kutta methods. A set of "user friendly" routines was developed which allowed the use of any one of these integration methods by a choice of parameter. The approach was similar to that of the continuous simulation languages such as ACSL and CSMP. The routines extended BBC BASIC to allow the solution of differential equations with first, second or fourth order method. It was only necessary at the start of the program to indicate the order of the solution.

The solution of tank problem was carried out as follows:

```
REPEAT
  {SOLVE THE ALGEBRAIC EQUATIONS}
```

```
        {EVALUATE THE DERIVATIVES}
        PROC_DERIVATIVE("M",DER_M)   }
                                     }--INTEGRATE
        PROC_DERIVATIVE("H",DER_H)   }--VARIABLES
    UNTIL FN_END
```

Each PROC_DERIVATIVE placed the variable being integrated and the associated derivative value into a table. When FN_END was encountered, these variables and values were used to carry out the solution of the differential equation, and the previously stated solution method order governed the number of times the whole loop was traversed for each time step. An example of use and the code for these language extensions is given in appendix[3].

When the LPG simulator was developed further, it was found that the simple Euler method gave satisfactory results. So to maximise the speed of the tank processors the simulation language was omitted.

After successfully modelling the tank containing liquid and vapour, it was decided to include a multicomponent vapour phase. This was to allow the possibility of air and inert gas being introduced to the tanks. The pressure in the tank now consisted of the sum of the partial pressures due to the vapour pressure of the butane, the air and the inert gas. The concept of the vapour fraction was maintained as the relation of the mass of liquid to the total mass in the vapour space of the tank.

Two additional fractions were introduced to represent the mass of inerts and mass of air as a fraction of the mass in the vapour space.

6.3 Liquefaction

A simple compressor condenser seawater liquefaction cycle (called reliquefaction on an LPG ship) and a cascade refrigerator were both modelled. In each case positive displacement pumps were used with the flow rate being set by the pump speed. As this was assumed to be an adiabatic process, the necessary temperature rise was included:

$$T_{out} = T_{in} \cdot \left[(P_{out}/P_{in})^{c} \right]$$

where $c = \frac{\gamma - 1}{\gamma \eta}$, T=temperature and P=pressure
γ = ratio of specific heats c_p/c_v
η = isentropic efficiency

The condenser model was rather more complicated. Due to the possibility of there being little or no liquid present at start up, catastrophic instability could have occurred since the time constant for the thermal balance in the condenser in this state reduced to far less than the time step of the model. It was not feasible to reduce the time step for the simulation so the method of solution of the differential equations had to be adjusted. An implicit integration method was used for the thermal balance. This resulted in stability under all conditions at the expense of accuracy. Given a tank with contents at temperature T and tank wall at

temperature Tw with area A and heat transfer coefficient U, the heat lost (Hw) from the tank during one time step could be written as

$$Hw = \int_{t}^{t+dt} U.A.(Tw-T)\ dt$$

approximately

$$Hw = U.A.(Tw-T(t+dt))\ dt$$

where T(t+dt) was the tank temperature after the time step.

This was now an implicit or backwards integration and stable for all values of time step. However since T(t+dt) would only be known afterwards, a compromise had to be made which could limit the accuracy i.e. T(t+dt) was taken to be equal to T(t), the temperature of the tank at the start of the time step. The thermal balance without the tank wall cooling was:

$$H = M.[B.(SLH + Cvap\ .\ T) + (1-B).Cliq.T]$$

where

H=enthalpy

T=tank contents temperature

M=mass of LPG in tank

B=vapour fraction of LPG

SLH=specific latent heat of vaporisation of LPG

Cliq=specific heat capacity of liquid

Cvap=specific heat capacity of vapour

with the inclusion of the tank wall and the implicit integration the relationship became

$$H + U.A.(Tw-T).dt = M.[B(SLH + Cvap.T) + (1-B).Cliq.T]$$

where U=thermal transfer coefficient

A=tank contact area

Tw=tank wall temperature

dt=integration time step

hence

$$T= \frac{[H/M + U.A.dt.Tw/M - B.SLH]}{[B.Cvap + (1-B).Cliq + U.A.dt/M]}$$

The algebraic part of the solution, to evaluate B, was carried out using the above expression for the temperature of the LPG, then the value of the temperature obtained was used to calculate the updated value of H.

The model of the seawater reliquefaction system is shown in figure 6.3.1.

The condenser was modelled as a tank in contact with seawater.

A more sophisticated model was developed using a cascade system. This is illustrated in figure 6.3.2. R22 was used as the refrigerant on the seawater side and butane as the cargo.

The butane and R22 condensers were modelled using

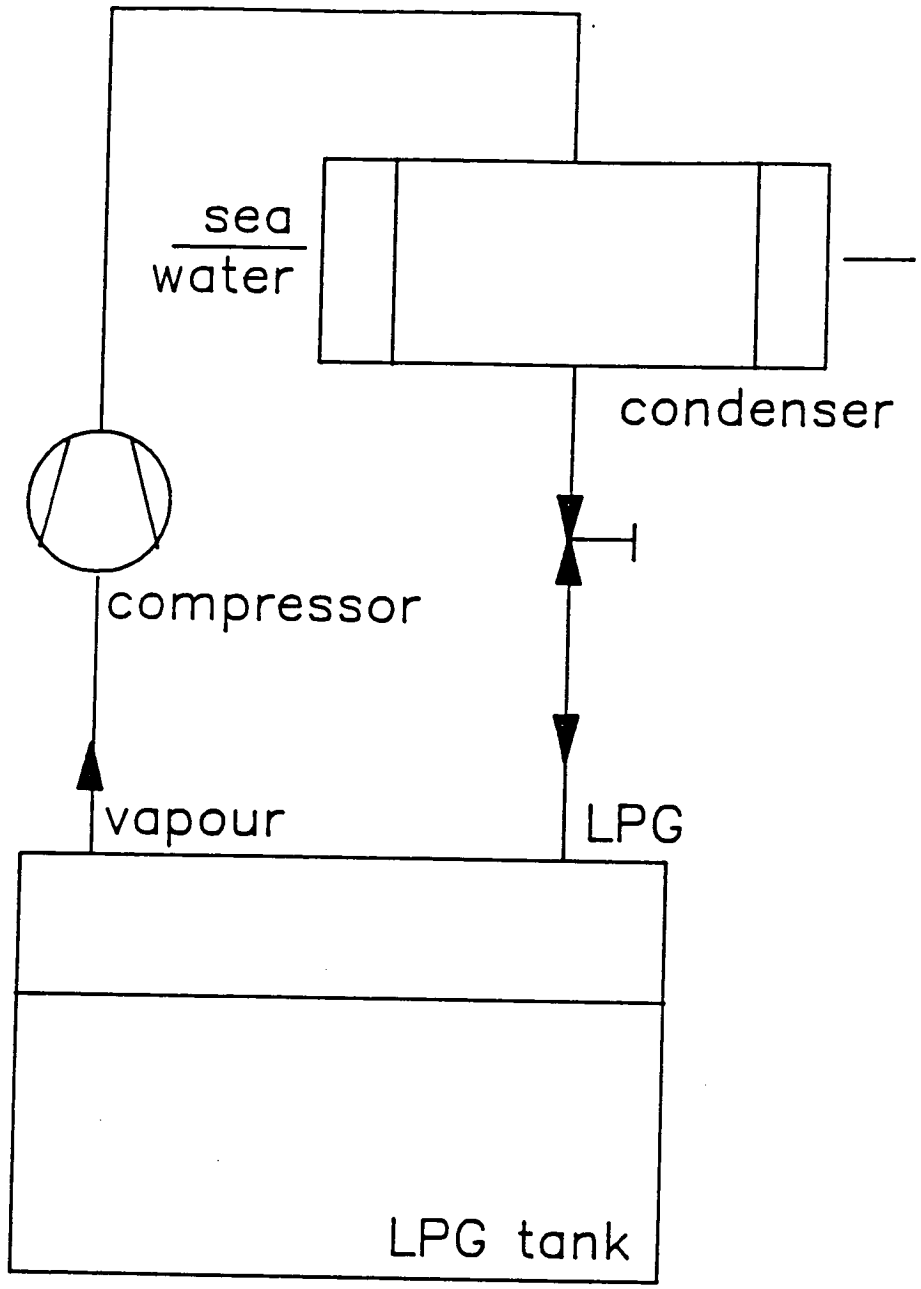


figure 6.3.1 seawater liquefaction plant

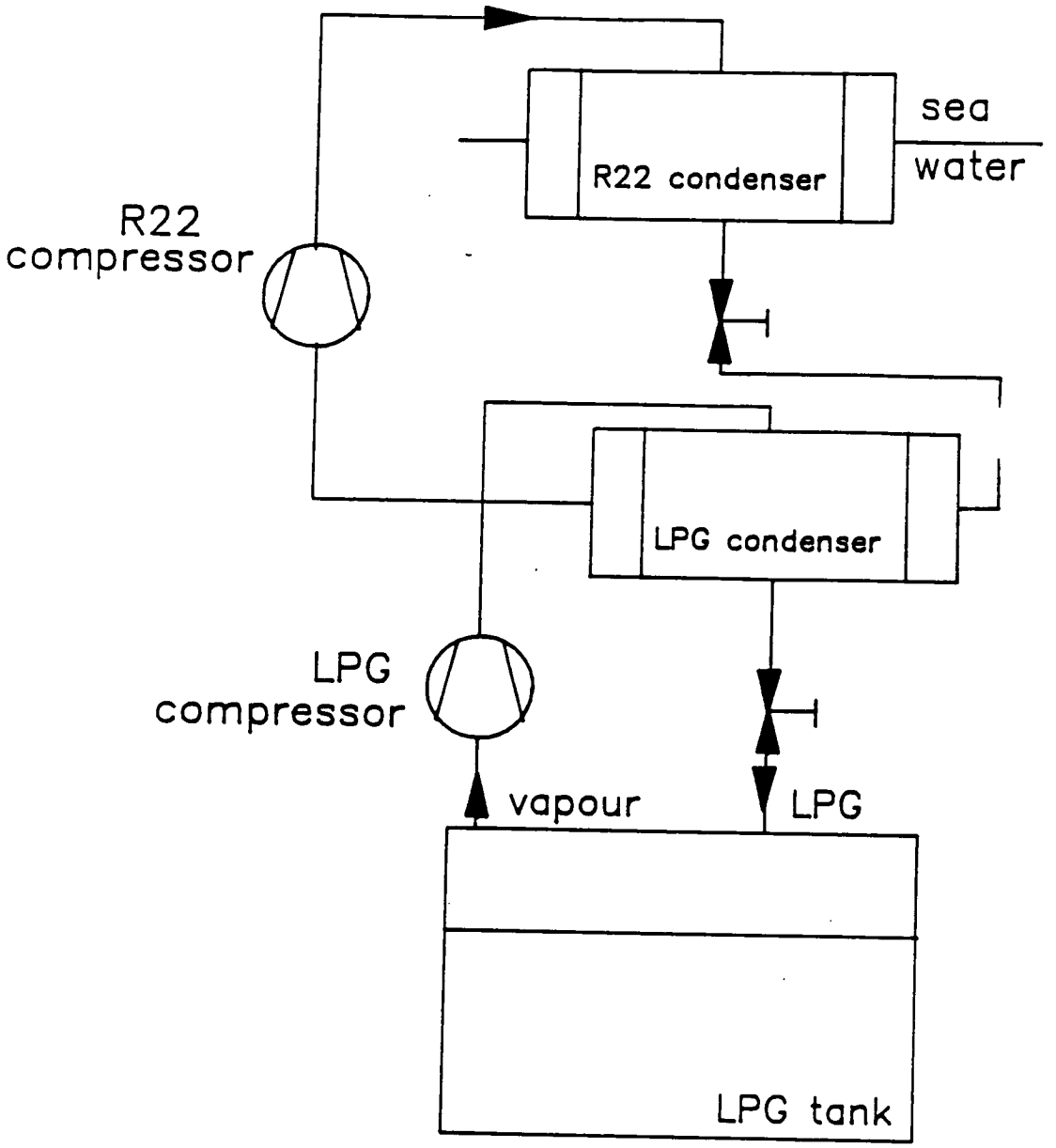


figure 6.3.2 cascade liquefaction

implicit integration for the thermal balance equation. This was required due to the possibility of the system starting up with only vapour present in the condensers. The assumption was made that there would always be liquid available in the R22 vapouriser so that an explicit integration method could be used there.

6.4 Flow Networks

The flow networks posed two problems. Firstly, if the valves were modelled with flow proportional to the square root of the pressure difference and the network was complex then, the solution was complex and took too long. Secondly, the flow network processor communicated with most of the machines in the network. This meant that any communications-related effects would be maximised here.

The liquid network was modelled initially alone in a single machine. In the three tank system using simple valve models it was possible to solve for the network flows analytically. Since the simple layout shown in figure 6.4.1 resulted in a single unknown node, this gave a fast calculation and sufficient realism for an extrememly simple network.

Hence for the simplest case using linear valve models and a single unknown node the flow though a valve was

$$F = kv \cdot (Pin - Pout)$$

where F=flow, P=pressure and kv=valve constant

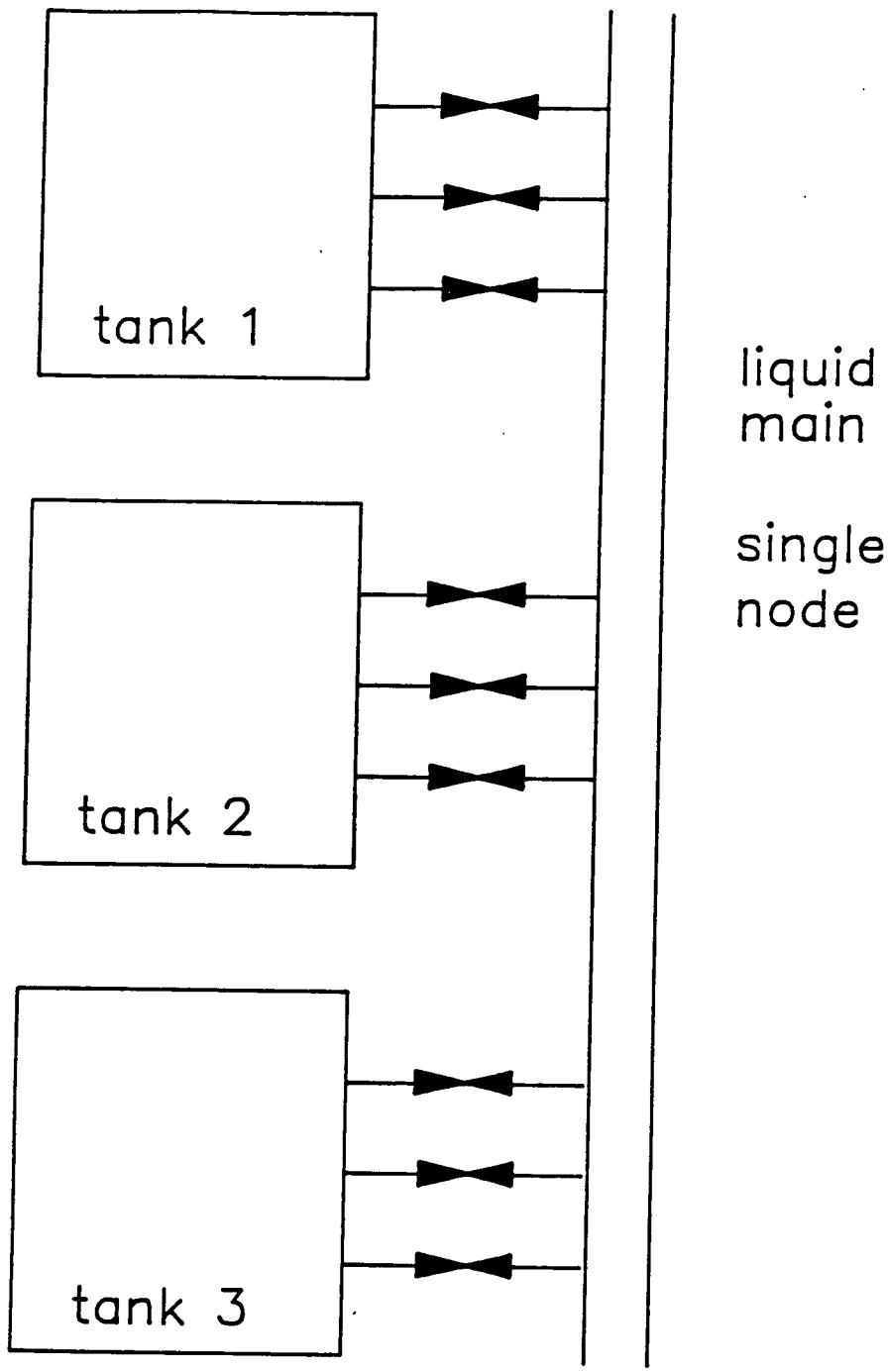


figure 6.4.1 liquid flow network

the solution for the liquid main pressure became

$$Pmain = \sum_{i=1}^{n} kv_i . P_i \; / \; \sum_{i=1}^{n} kv_i$$

where there were n pipes connected to the single node.

The vapour and condensate return networks were modelled as lumped tanks in the compressor and condenser processors respectively. The tank capacities were large enough to avoid instability due to stiffness in the equations.

If a more detailed non-linear valve model had been used, an iterative solution of algebraic equations would have been required and the time for a single solution pass would have become greatly increased. This option was not explored.

6.5 Combined Flow Network

A more complex network involving several unknown nodes but still with linear valve models was developed. A general solution method which could easily adapt itself for different network configurations was implemented on the 32016 co-processor machine. As can be seen from the benchmarks in appendix[4], this was a very fast addition to the BBC Model B and had one megabyte of memory. Gaussian elimination was used to solve over the complete pipe network, which could then allow crossovers between the vapour, condensate and liquid networks. To test the capacity of this powerful processor all the known nodes

(essentially the tanks) as well as the unknown nodes were treated as unknowns in that they had an equation in the system. This resulted in a matrix of fourteen equations to solve for the pressures. Also a matrix filling method based on a stored branch table was used for each pass through the program. This could have been speeded up since the structure of the network would not change so the matrix contents were for the most part fixed.

The matrix filling method consisted of generating a table containing associated source node, destination node and valve constant for each of the branches in the network. For each solution all the coefficients were set to zero, then the matrix was set up by adding the valve constant to the [upstream,upstream] and [downstream,downstream] matrix coefficients and subtracting from the [upstream,downstream] and [downstream,upstream] coefficients. After this for each known node, its row in the matrix was set to zero, its diagonal coefficient set to one and its value in the constant vector set to the node pressure. This is shown in figure 6.5.1

The three unknown nodes - liquid, vapour and condensate - were all treated in the same way. If a node contained no liquid then the pipework was modelled as a vapour tank. An implicit integration method was desirable in this case because of the relatively small volume of vapour (typically around one cubic metre) contained

for a six node matrix

a branch connecting node 2
to node 5 with coefficient kv

	1	2	3	4	5	6
1						
2		+kv				--kv
3						
4						
5		−kv				+kv
6						

when node 5 is known
and has a pressure of100kPa

$$\begin{bmatrix} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & & & & & & \\ 2 & & & & & & \\ 3 & & & & & & \\ 4 & & & & & & \\ 5 & 0 & 0 & 0 & 0 & 1 & 0 \\ 6 & & & & & & \end{bmatrix} \times \begin{bmatrix} \\ P1 \\ P2 \\ P3 \\ P4 \\ P5 \\ P6 \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \\ \\ 10^5 \\ \\ \end{bmatrix}$$

figure 6.5.1 examples of matrix filling

within the pipework making up the node. Both the thermal and mass equations were subject to very small time constants compared with the rest of the system and so both were also integrated implicitly. This was implemented as follows.

For a node represented as a tank with volume V, pressure P, inlet flow F1 through valve of constant kv1 from pressure P1 and outlet flow F2 through a valve of constant kv2 to pressure P2:

the flows were

$F1=kv1.(P1-P)$

$F2=kv2.(P-P2)$

the mass balance was

$dM/dt = F1 - F2$

since $M=(P.V.Mw)/(R.T)$

where Mw=molecular weight of product

M=mass

R=universal gas constant

T=temperature

then $dM/dt = d/dt((P.V.Mw)/(R.T)) = kv1.(P1-P) - kv2.(P-P2)$

so $V.Mw/(R.T).dP/dt = kv1.P1 + kv2.P2 - (kv1+kv2).P$

For one time step t the old pressure in the node was P' and the new value P.

$V.Mw/(R.T).(P - P')/t = kv1.P1 + kv2.P2 - (kv1+kv2).P$

let $E = V.Mw/(R.T.t)$

then $E.P - E.P' = kv1.P1 + kv2.P2 - (kv1+kv2).P$

and $-E.P' = kv1.P1 + kv2.P2 - (kv1+kv2+E).P$

This could be compared with the equation for an incompressible node used in the gaussian solution

$0 = kv1.P1 + kv2.P2 - (kv1+kv2).P$

The changes to accommodate this implicit integration were dealt with directly in the matrix by adding to each vapour diagonal in the matrix the old pressure multiplied by E. Also E had to be added to the matrix coefficents along with the valve constant terms.

The temperature and concentrations had also to be subject to implicit integration methods. These were formulated in a similar manner to the implicitly integrated thermal balance of the condenser in an earlier section (6.3).

The concentration case is illustrated:

For a node represented as a tank of vapour of mass M with several inflows and outflows Fin and Fout, the rate of change of concentration of a particular component x was:

$$d(x.M)/dt = \sum_{i=1}^{n} Fin_i . x_i - \sum_{i=1}^{n} Fout_i . x$$

the product rule

$d(x.M)/dt = M.dx/dt + x.dM/dt$

and $dM/dt = \Sigma Fin - \Sigma Fout$

hence

$$M.dx/dt + x.\sum_i Fin_i - x.\sum_i Fout_i = \sum_i Fin_i.x_i - x.\sum_i Fout_i$$

and $$M.dx/dt + x.\sum_i Fin_i = \sum_i Fin_i . x_i$$

for one time step δt with old concentration x' and new value x

$$M.(x-x')/\delta t = -x\sum_i Fin_i + \sum_i x_i.Fin_i$$

$$x.(M/\delta t + \sum_i Fin_i) = M.x'/\delta t + \sum_i x_i.Fin_i$$

$$x = [(M.x'/\delta t + \sum_i x_i.Fin_i)/(M/\delta t + \sum_i Fin_i)]$$

This expression had to be applied separately for each component in the vapour.

A similar expression was derived for the temperature:

$$T = [(M.T'/\delta t + \sum_i T_i.Fin_i)/(M/\delta t + \sum_i Fin_i)]$$

When a node contained liquid it was assumed to be wholly liquid and modelled more simply. A compressibility factor was chosen to eliminate the problem of closed valves isolating part of the network and producing an indeterminate pressure. This was similar to the E term above in the implicit integration and was implemented by adding a very small value to the node diagonal if the coefficient were zero.

The performance of the 32016 co-processor was outstanding in comparison to the model B machine in that

the combined processing of the complete network using the above methods for one time step could be accomplished in less than one second.

6.6 Summary

During the period of this research project many models were developed, most of them described in this chapter. However in the current simulation some have been modified.

The tank model ran quite acceptably using first order integration methods so the "user friendly" integration routines allowing second and fourth order were jettisoned in favour of code to give elementary colour graphics. In the BBC Model B there was competition for memory use between code, data and graphics. Mode seven graphics, which gave merely block characters and no drawing facilities, had to be used to give the maximum space for code. Routines were written to facilitate the representation of the plant being modelled in each processor and this took up some precious bytes.

The seawater reliquefaction plant was originally run on a single machine but it was rather slow and was split up into a compressor with vapour network and a condenser with condensate network in two machines. Finally all three fluid networks were combined in a machine with a 32016 coprocessor.

Clearly, here was a difference between the single

machine simulator and the present approach. The availability of more advanced microcomputers could alter the modelling scheme. However this ablility to alter the scheme was also the flexibility of the approach. Additionally, as each new machine has appeared on the market it has shown an increase in power, allowing more complex models in one processor. It could be predicted that no unexpected retrograde move is likely to be required of the modeller in the future.

THE LPG SHIP MODEL

7.1 Introduction - Evolution of the LPG Ship Model

The LPG ship model has been used throughout the project as the subject for simulation. As such it has had several incarnations. Firstly there was a three tank system with only the liquid network modelled. Then the simulation was expanded to include a reliquefaction plant resulting in vapour and condensate networks being added. The next step was to add the ability to deal with a multicomponent vapour phase so that inert gas and air could be introduced to the tanks. Finally a shore plant was added. The model decomposition was always geographical with the attendant tidyness and minimisation in data transfer. The significant feature of the system being modelled was that in real time it was very slow moving. The time constants were large. The first attempts used a tightly coupled two phase processing cycle. The pressure setters (tanks) were solved first, then the flow setters (valves and pumps). Clearly there was considerable parallelism here since all the flow setters were being solved simultaneously. However it was decided that the solution would not be seriously compromised if the flow and pressure setters were run simultaneously. This resulted in calculations on data from the previous cycle but the process values were changing very slowly and local updates would occur twice

as quickly, assuming an equal load on all the processors.

The problem of deciding how to relate the simulated process to real time was not tackled at this point. This was instituted in an arbitrary way, since the simulation would always be run considerably faster than real time. Each time step was taken to be one second even though the cycle time was between one and two seconds. A true relationship could have been used if the network machine had passed a control byte round the processors to indicate the time period, measured on it real time clock, represented by a single time step.

7.2 The Present Model

The system described so far was tightly controlled with one processor, designated the "master", which made sure that all the machines had completed a time step before issuing a flag for the next. It was realised that some processors would complete their task quicker than others and they would have to wait until the others caught up. A decision was taken to step further away from serialism by allowing the processors to operate freely as fast as they could. Synchronisation was abandoned. This meant that each machine would update its values locally as fast as possible, transmit these values and accept any new incoming data, if available, in each cycle. Each processor ran effectively in real time with its time step calculated from its own internal clock† This meant that

† the time measured for a given cycle of processing being used as a predictor for the time step of the next cycle

fast processes took small time steps and slow processes took correspondingly longer steps. This development coincided with the development of a packet switching technique, using the sorter processor. This allowed the maximum of flexibility in the communications. In particular a slow processor would always receive the latest packet from any given source. This was particularly important with control packets, such as those shutting off a process due to an error.

The layout of the LPG cargo system is illustrated in figure 7.2.1. It consisted of three cargo tanks, a compressor, a seawater condenser, an inert gas plant and a shore plant.

The LPG ship simulator was implemented on nine processors, shown in figure 7.2.2. It consisted of three processors for the cargo tanks, one for the compressor, one for the condenser, one for the shore and inert gas generator, one for the packet sorter, one with a touch screen for operator control and a 32016 coprocessor machine which handled all the flow networks.

The three cargo tanks ran on BBC model B machines as did the condenser, compressor and shore/inert gas plant. Model B's with 6502 second processors were used for the sorter and the operator control. This allowed additional speed for the sorter and the use of high resolution colour graphics for the operator's touch screen. The simulation operated reasonably realistically. It allowed

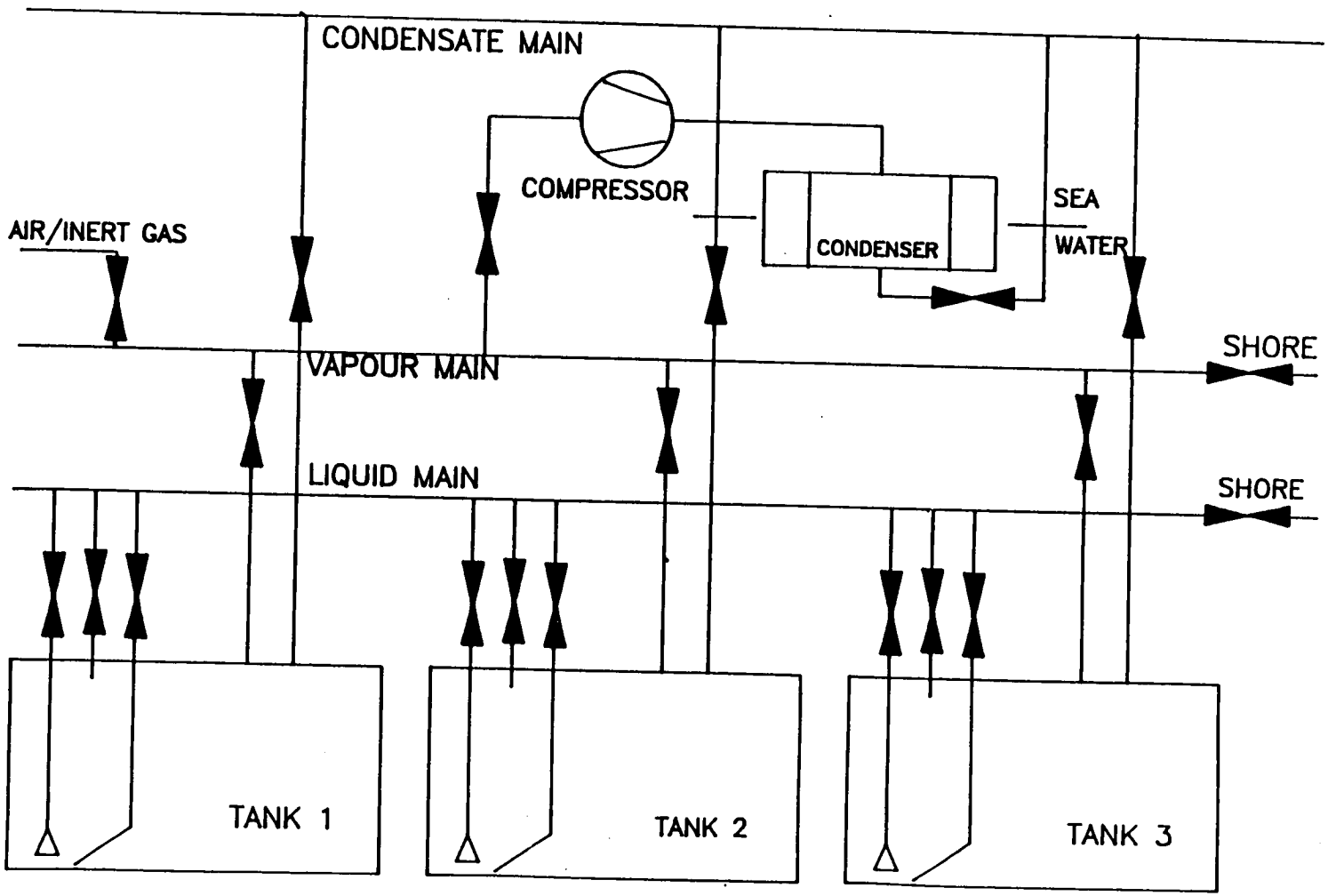


figure 7.2.1 LPG ship cargo plant

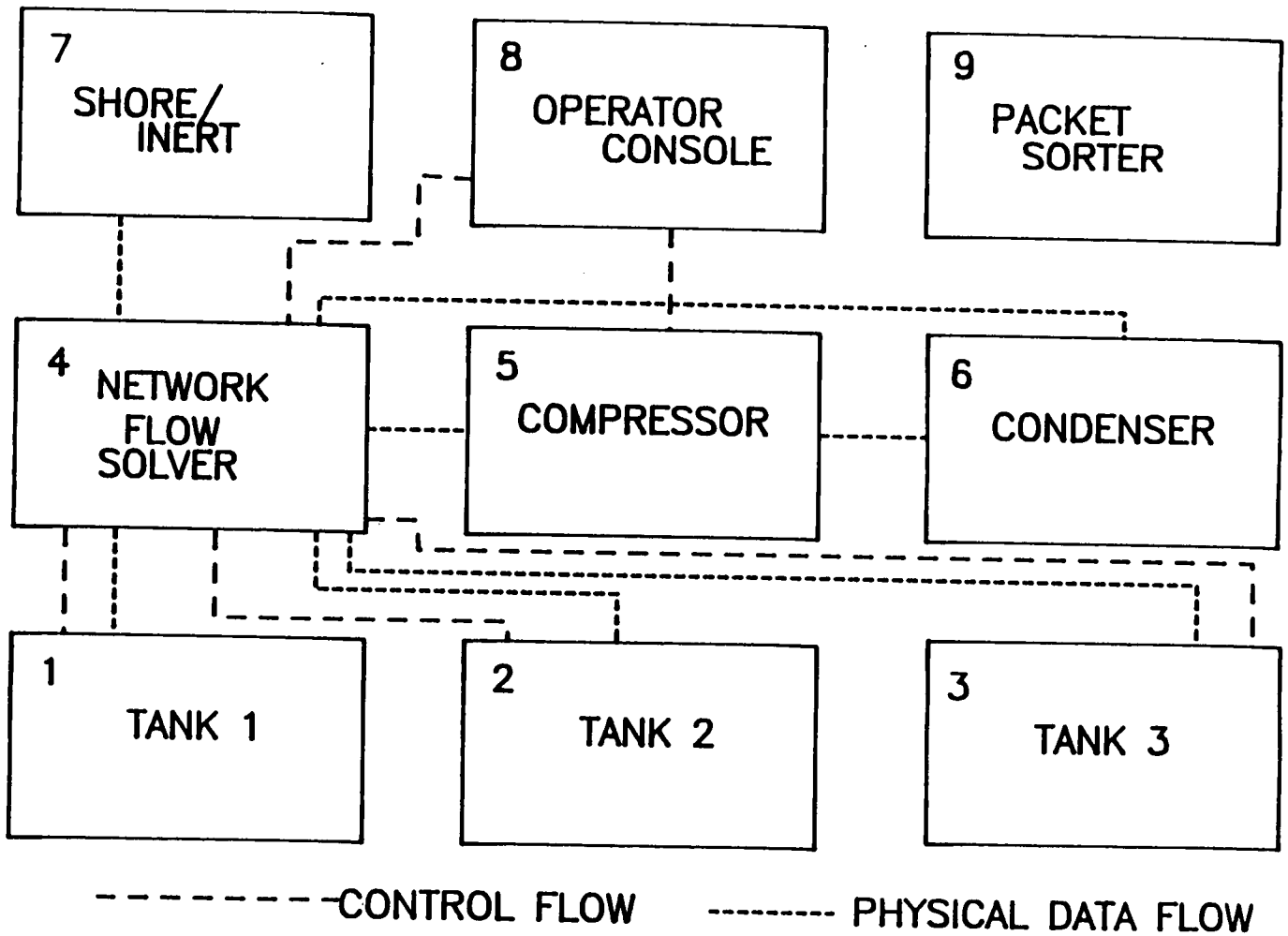


figure 7.2.2 implementation of LPG plant

the demonstration of normal cargo operations such as inerting a tank, purging, filling, cooldown, discharge and gas freeing.

In use, the appropriate flows occurred dependent on the valve and pump settings selected at the touch screen. It was possible to verify, taking the tank dimensions into account, that the liquid flow rates indicated on the tank screens corresponded to the level rise and fall in real time.

The following LPG ship characteristics were observed from the simulation:

a) the fall in tank temperature and pressure during discharge

b) the rise in tank temperature and pressure during filling

c) the cooling effect when LPG was introduced to a vapour filled tank whose pressure was less than the saturated vapour pressure.

In the present model, only the case of complete vapour-liquid equilibrium in the tanks has been explored. In reality the above mentioned effects are more noticeable on the ship where the liquid and vapour are not always in equilibrium due to the size of the cargo tanks.

7.3 Summary

A weak point was that the Model B machines modelling the cargo tanks were rather slow, taking on average just

over two seconds for each time step. Investigation showed that a considerable time, just less than 50%, was spent on the packing and unpacking of data transmitted by the packet switching system. The actual sorter processor ran extremely quickly and the data transmission on Econet was very fast. However the process plant machines used BASIC for handling the packets unlike the sorter which used machine code.

Use of the BBC Master machines with Turbo boards in the system was investigated. This resulted in processing times around three times faster than the Model B. This performance gave more than adequate cycle times of around 0.7 second. However, although compatibility between different machines was one of Acorn's strong suits (the 32016 programmed in Pascal was fairly easy to integrate into the system), there was a slight difference in the specification of the Econet interface of the Master compared to that of the Model B. This resulted in data corruption on the LAN due to a more stringent cable and connection specification being required compared to that which had been used. Thus it has not been possible to explore fully the use of the potentially excellent Master machines in the network.

THE SIMULATION OF OTHER SYSTEMS

8.1 Introduction

As an example of a chemical process the LPG ship model had the fairly unusual characteristic of consistently large time constants. Two other examples of chemical plant were simulated using the present multi-microcomputer approach. The author was involved in the planning of the simulation and this work was carried out using the communications software developed in the present work, but the author was not responsible for the modelling.

A pressure swing adsorption plant was modelled by Matheson and Rutherford[11]. This was a complex system involving unsteady state processes and the solution of partial differential equations in the distributed processors. The simulation could be used to increase operator understanding of the process and develop optimal sequencing strategies.

A distributed compressor system was modelled by Forsyth[12]. This was similarly complex and was fast changing. The simulation could be used for operator and control system training on the classic problem of surging during the startup of the compressor.

This chapter describes the work on these systems, highlighting the difficulties encountered and the solutions applicable with particular emphasis on the

problems caused by the multi-microcomputer approach.

8.2 PSA System

A pressure swing adsorption plant was chosen to be simulated using the distributed simulation techniques outlined above to provide operator training. The process consisted of four adsorption beds separating hydrogen from hydrocarbons. It is illustrated in figure 8.2.1.

The beds contained activated carbon or zeolite molecular sieves. The cycle for a single bed is shown below.

1 Feed consisting of a mixture of hydrogen and hydrocarbons was introduced at high pressure at the bottom of the bed.

2 Pure hydrogen product was available at the top of the bed until it become saturated.

3 The pressure in the bed was then reduced, removing the feed supply and using the product to pressurise and purge other beds.

4 The bed was now "blown down" when the impurities were exhausted at the feed end and the vessel pressure fell.

5 The bed was now purged, and repressurised before the cycle started again.

Complex switching between the beds was required to minimise loss of product and pressure by using the out

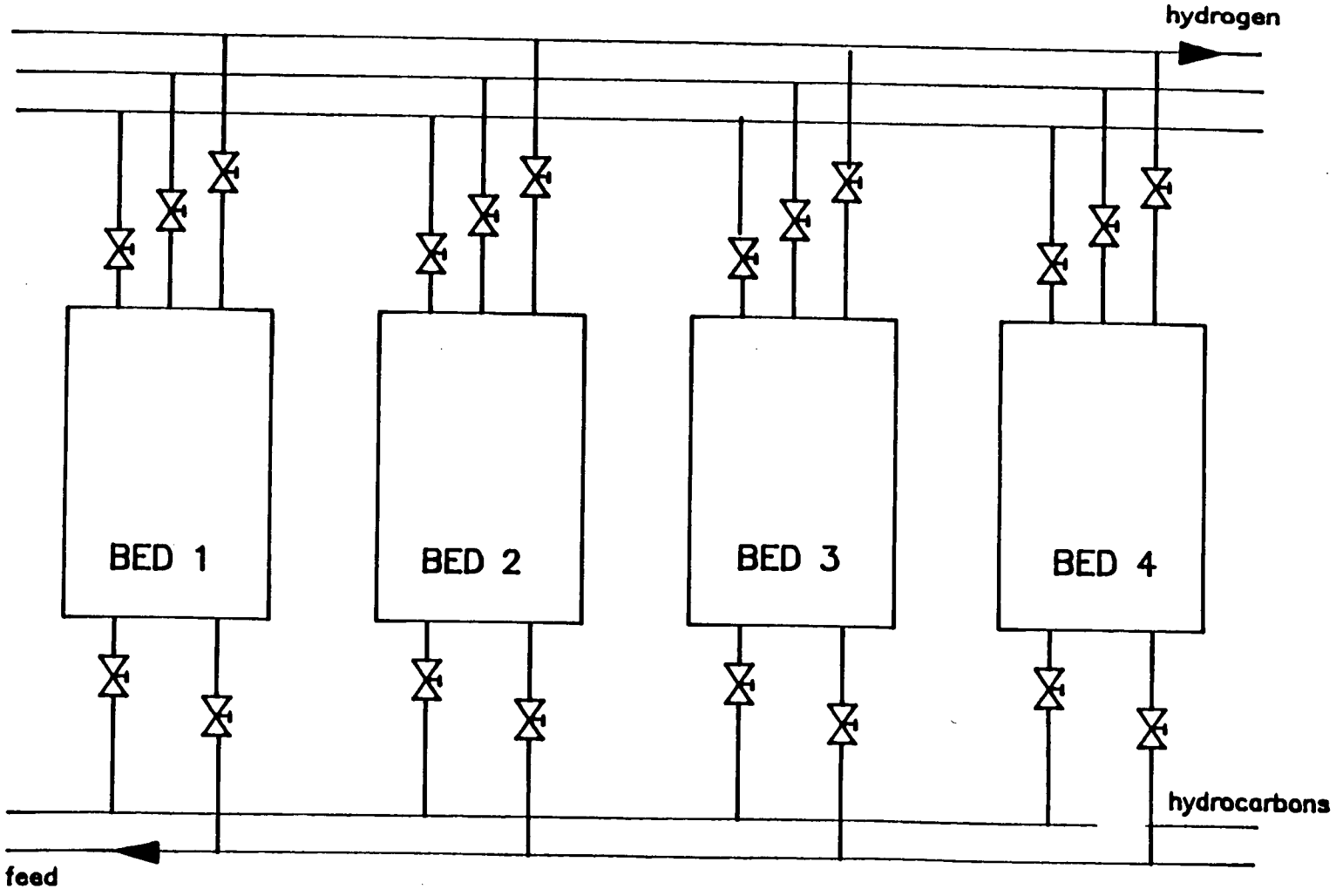


figure 8.2.1 PSA model

flow of one bed as its pressure reduced to pressurise up another bed.

Figure 8.2.2 shows the distribution of the model over the processors.

One machine was used for each bed. Additional machines were used for flow network solution and valve control sequencing. This was a purely geographical decomposition and the results were poor. Although the time constants of the composition in the beds were large, around the order of one minute, those associated with the pressure transients when flashing off could fall as low as one second, the time step value. An alternative approach was required to overcome the gross inaccuracy and instability which occurred. The problem was exacerbated by the delay at the flow solver in receiving updated values from the beds via the communications software.

A move away from the purely geographical decomposition suggested itself. If the equations with small time constants could be grouped together in one machine these closely coupled calculations would be solved in a stable manner. The approach was to have a two level model (see Ponton[9]).

This is illustrated in figure 8.2.3.

The adsorption models which used partial differential equations were distributed as one bed per machine. This coped with the accurate modelling of the composition of the bed which changed slowly. An approximate, but still

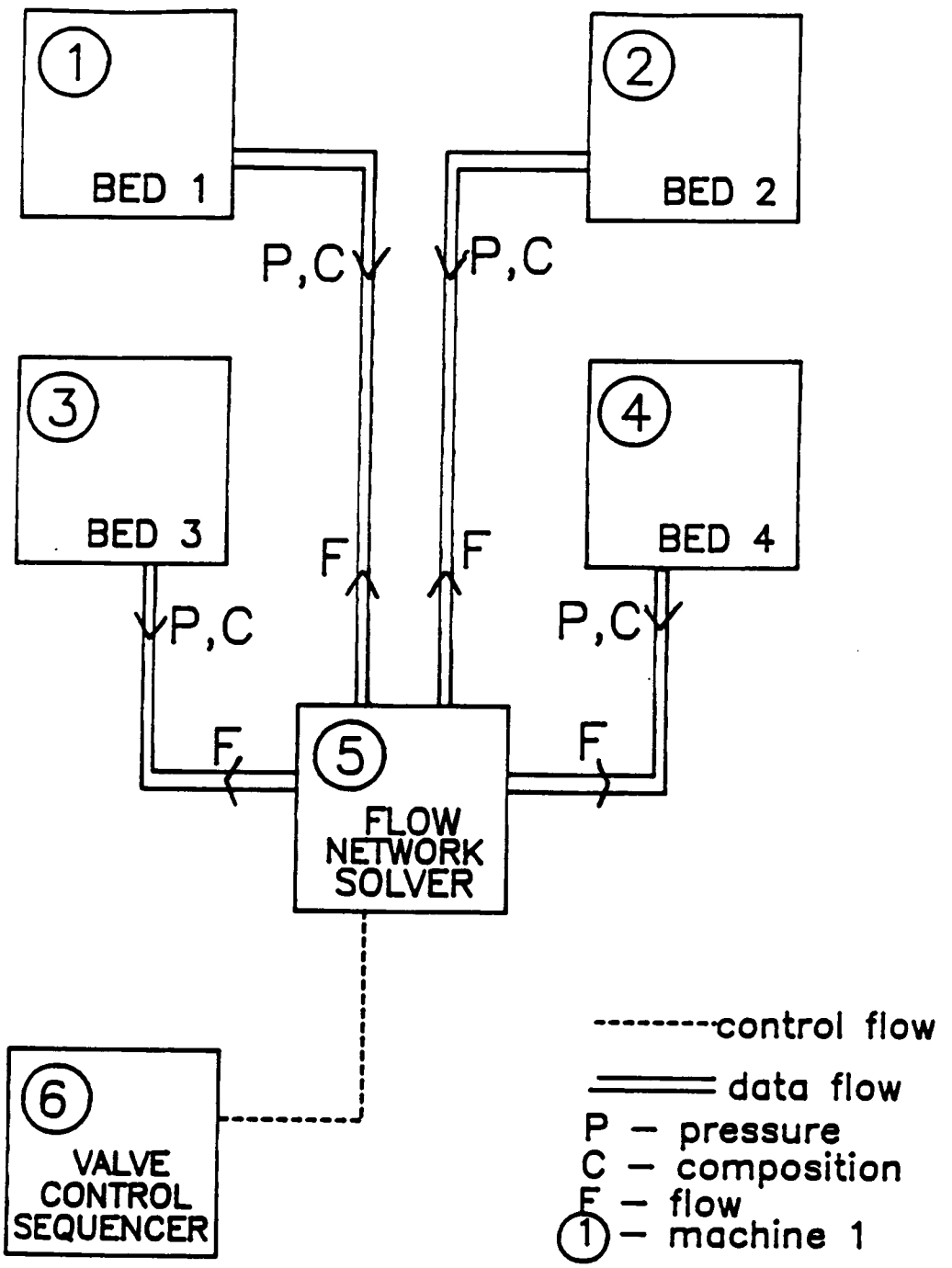


figure 8.2.2 initial implementation of PSA model

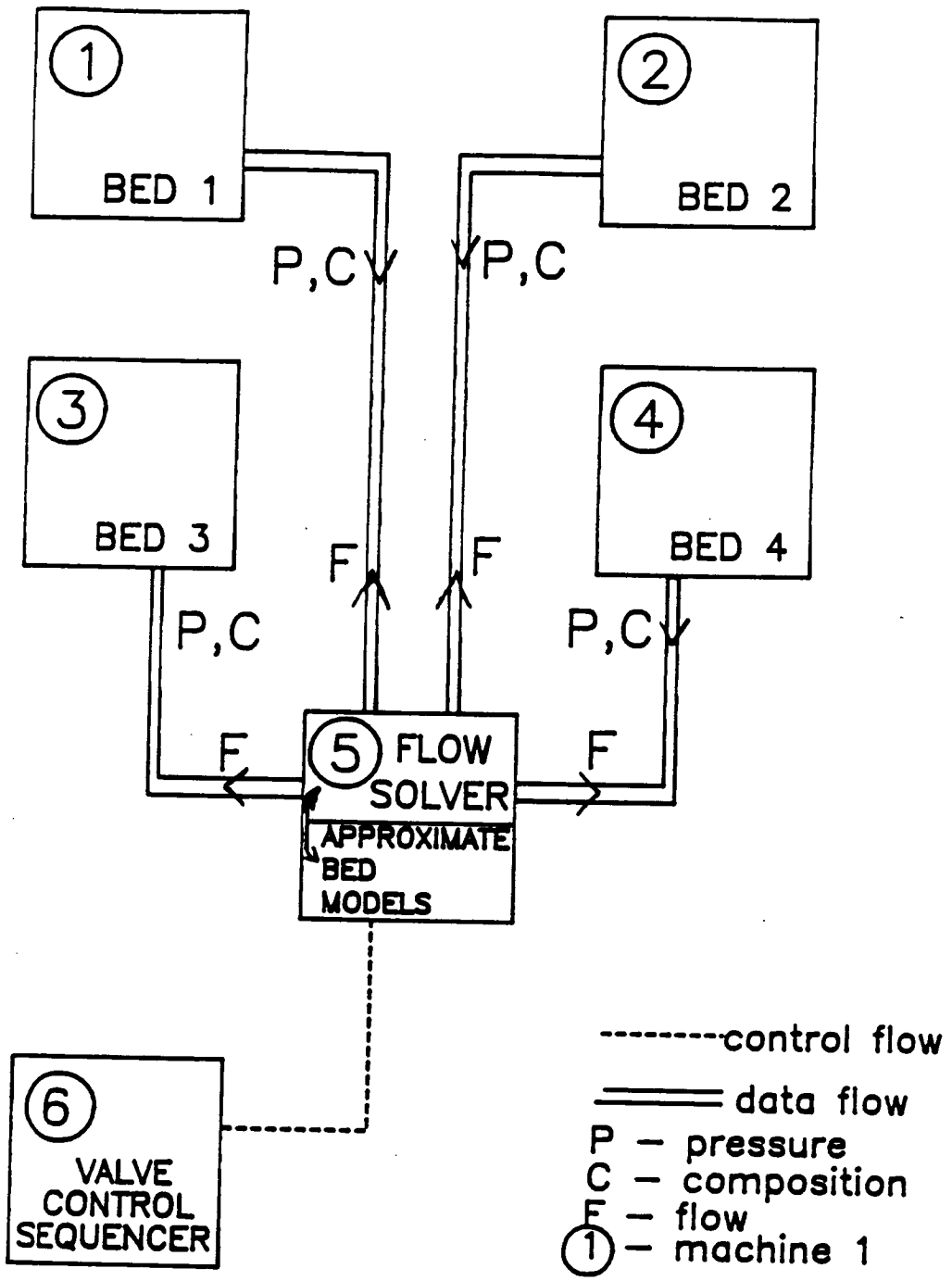


figure 8.2.3 two level implementation of PSA model

non-linear, model for the calculation of all the bed pressures resided in the single machine which also carried out the pressure flow calculations. Parameters were passed between the bed machines and the approximate model when available. This gave an effective simulation without instability. ~~and with adequate results.~~

8.3 Distributed Compressor Model

Forsyth[12] looked at the modelling of compressors in detail. He investigated a multi-microcomputer approach because it was not possible to run the complex compressor program in real time on a single microcomputer.

A number of systems were modelled culmimating in a simulation of part of a chlorine liquefaction plant. The plant is illustrated in figure 8.3.1.

Four processors were used as shown in figure 8.3.2. The "complex valves" machine modelled all the valves in the system and the "complex compressors" machine modelled both compressors. The "complex" processors had the task of modelling accurately the valve and compressor behaviours. The global model carried out approximate calculations and was updated with fresh parameters from the complex models. An estimate of the required refresh rate was less than half a second.

Note that this simulation was an example of a two level approach and a somewhat functional distribution of tasks over the processors. There was not necessarily a

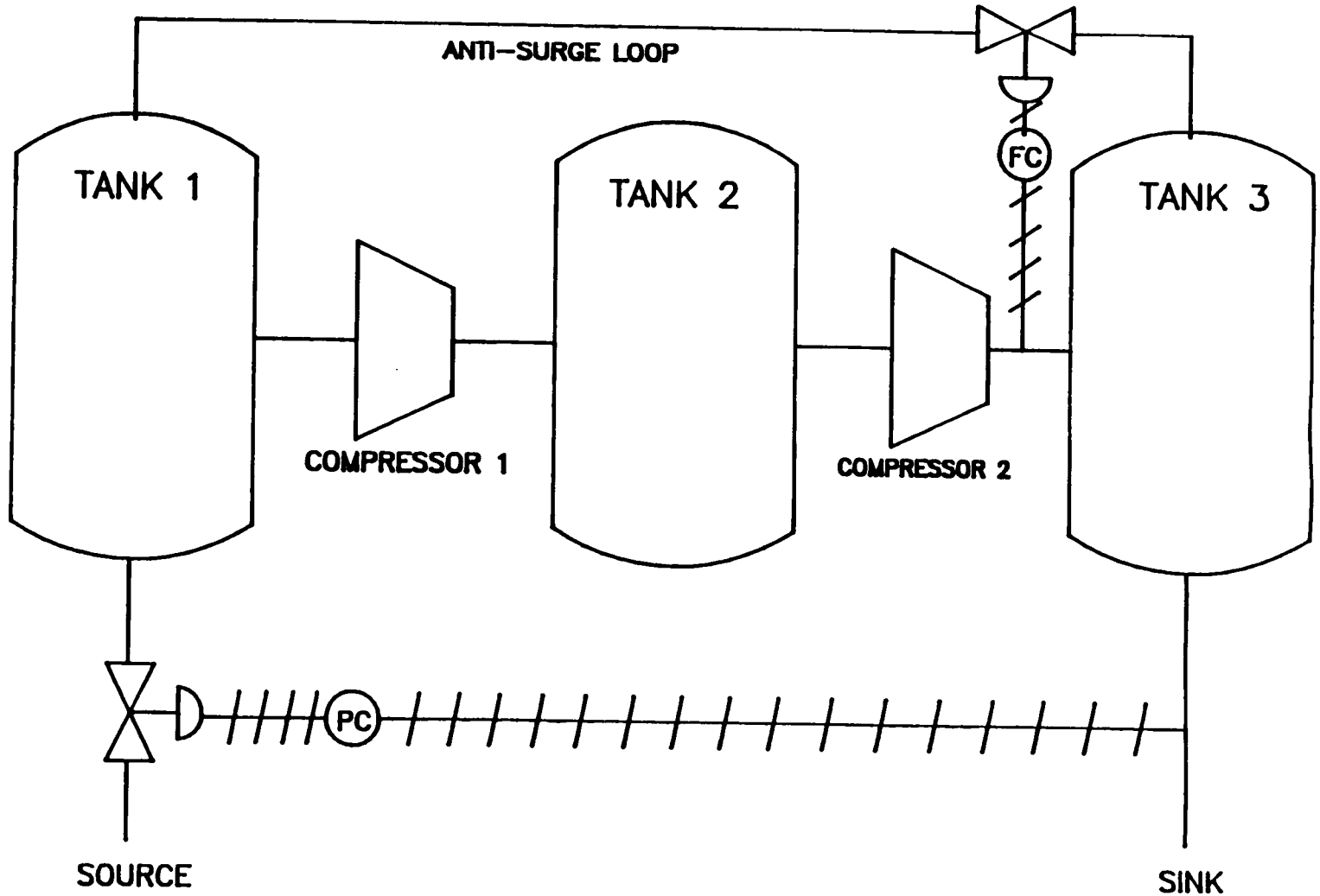


figure 8.3.1 chlorine plant model

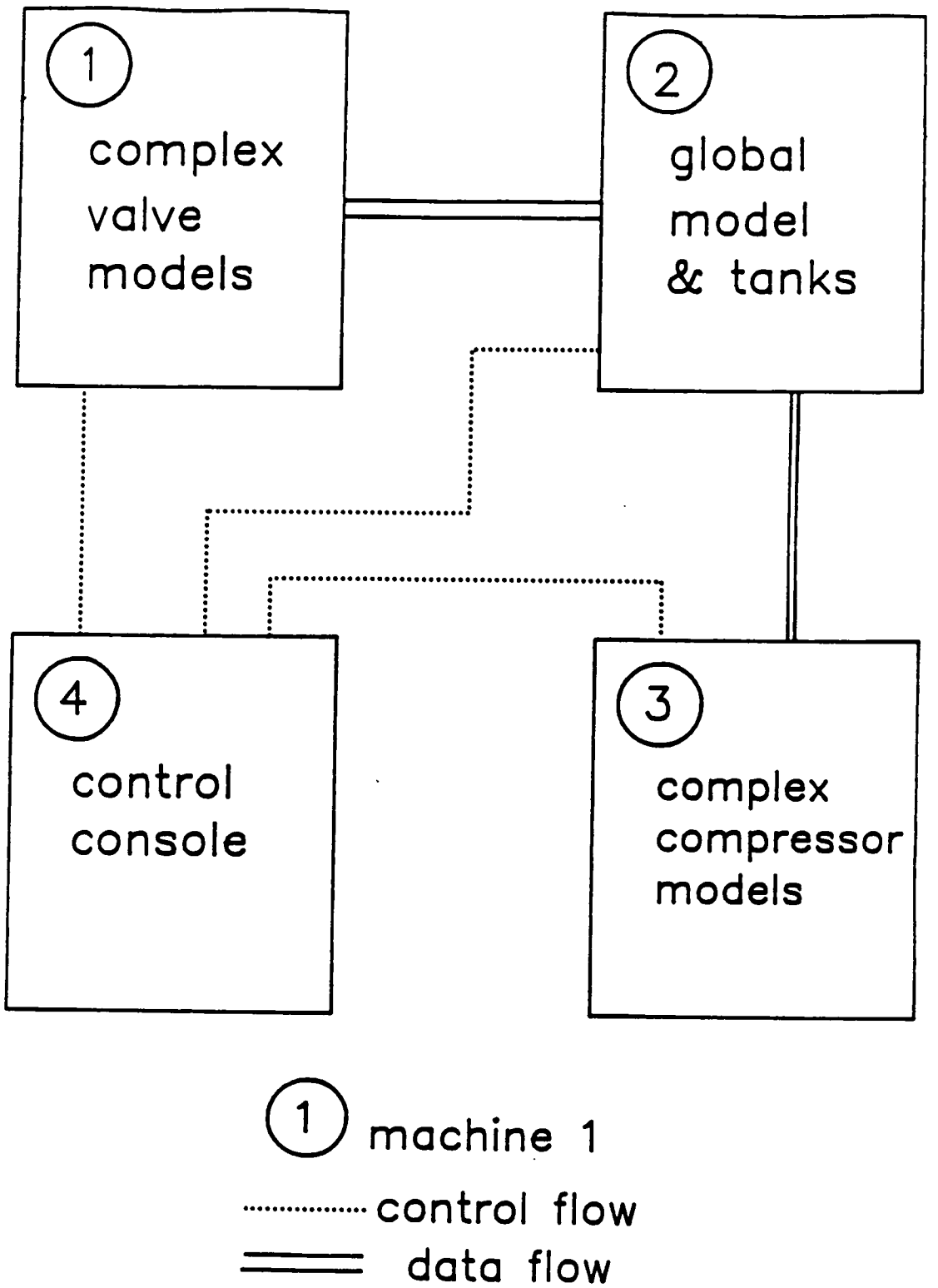


figure 8.3.2 implementation of chlorine plant

geographical correspondence between plant and machines.

Although the system worked it was not entirely satisfactory. The system could not operate in real time. When the step length for the computational cycle was increased to allow near to real time performance, instability resulted. Implicit methods were used for the solution of the differential equations, and so the instability resulted from the extrapolation becoming too large in the linearised compressor model. When the single machine implementation was compared to the distributed system, the single machine was quoted as allowing the model to perform three times faster. Two points were responsible for the poor performance compared to the other systems modelled. Firstly the communications overhead was relatively exagerated, the rate of updating was rather high at twice a second. A more complex compressor model would have run much more slowly and the high communications overhead would have been less significant. Secondly the system was operated synchronously, so that at certain points in the cycle machines were waiting for a flag and doing no processing.

8.4 Summary

Simulation problems can be placed in a number of classes with respect to modelling on a multi-microcomputer. When uniformly large time constants are involved, straightforward geographical disposition can be

used. In the case of systems with some tightly coupled short time constant equations, it may be possible to use a two level approach which results in a partly functional disposition. The LPG cargo ship system was not modelled using the two level approach. The vapour equilibrium calculations in the LPG tanks are much more complex to model than the pressure and flow characteristics of the beds in the PSA plant. Further work would have to be done to produce simple enough approximate models for the tanks to allow several to be run in one machine which carried out the pressure flow calculations. The partly functional approach is illustrated in the compressor example, but does not work satisfactorily. The difficulty in the compressor approach seems to be related to the frequency of update of variables between the processors and the slow rate of packet processing in the BASIC code.

It is necessary to choose an appropriate model on an individual basis with due attention to the performance of the computer system and the data transmission delays between the processors.

CONCLUSION

9.1 The LPG System

The original aim of low cost simulation has been explored using mass produced hardware. For the LPG cargo simulator some success can be claimed. The system developed consisted of three pressurised cargo tanks, liquefaction plant, inert plant and shore station. Touch screen control was provided for the valves and pumps. A large scale video display was provided by the colour graphics monitor for each machine. It was possible to demonstrate gas tanker operations such as inerting, loading, discharging, purging and cooling the cargo (see McGuire[13] and Woolcot[14]).

As part of the project a communications system was developed for use on LAN's especially for process software. This allowed the transfer of data between process plant items using a "pipe" concept with a large amount of machine independence.

9.2 The Two Level Approach

The LPG project was ideally suited to the present multiprocessor approach due to its large time constants. This reduced the effect of delays due to the communications system. However this simple geographical approach did not work with the other two systems modelled. Here tightly coupled processes forming a set

of stiff equations required a more complex solution.

At first sight the PSA system was similar to the LPG system in that a number of beds (similar to tanks) were connected together in a flow network. However, unlike the LPG system, the problem was that fast pressure transients (due to the working fluid being gas and high pressure beds being flashed off) created equations with very small time constants. The solution used a two level approach. A geographical layout was used with the beds being modelled in individual machines and a single machine modelling the flow network. However the flow machine moved slightly towards the functional disposition in having an approximate model for all the beds, so that the small time constant calculations involving the tightly coupled plant items could be carried out in one machine. The accurate bed models updated this approximate global model periodically. This was successful.

The approach in the compressor simulations was more functional. A global flow machine was used with a second machine providing complex valve models and a third for the complex compressor models. The method was similar to the PSA system in that the complex models updated a global model periodically, the global model using these values to correct approximate models. The system operated but attempts to run at real time caused instability. This was due to the linearisation of the

compressor characteristics in the global model. (No linearisation was required in the global flow machine of the PSA simulation.) The extrapolation required was so great that the errors caused instability. The only way to run the system, other than with some faster machine, was much slower than real time when the extrapolations fell within acceptable error.

9.3 Difficulties in the Multi-microcomputer Approach

It is clear that difficulties in the multi-microcomputer approach lie in two areas, - the time delays in communications and the disposition of the system being modelled over the machines.

Given that a geographical split of the model is feasible, then the potential problem is that of speed of communications. This itself has two components, - the transport time between sending and receiving, and the processing time at the sender and the receiver. It has been noted that with the BBC B machines the processing time at the receiver, predominates. Thus it is possible to have an improvement in the communications by either jettisoning the "user friendly" package or, more acceptably, using less BBC BASIC and more machine code in the communications routines along with faster machines.

When the system being modelled has tightly coupled calculations involving small time constants over a number of physical items, then the two level approach could be

fruitful. To achieve this it must be possible to create a sufficiently simple global model to accommodate the critical sections in one machine. However this simple global model must also be accurate enough to prevent the difficulties as highlighted in the compressor study. This requires a neat balance. If the global model becomes too complex in order to give the required accuracy then it may be that the whole simulation would be as well in a single more powerful machine.

The LPG system was developed such that each plant item became freestanding using its own real-time clock to calculate the integration time steps. This lack of centralisation gave a system which was easy to develop since individual processors could be run without requiring control flags from a central co-ordinator. This non-rigorous approach worked with the LPG system but is unlikely to be generally applicable. To cater for a synchronised system of processors, the communications system could be broadened to encompass both the "user friendly" packets and the more direct cooperative transfers. Essential data, such as control flags for synchronisation and other items which require fast transfer could use the cooperative system.

9.4 Future Applications

Processing hardware is becoming continually more powerful. The latest machine from Acorn is the

Archimedes, costing less than £1000 for the base version including monitor and disk drive. The cost is not so different from the cost of the original BBC B in the same configuration. This machine uses a very fast processor operating as a "reduced instruction set chip". The benchmark (see appendix[4]) shows that it is about fourteen times faster than the original BBC Model B. (It will be interesting to try it out with the promised language compilers - it should be considerably faster still.) This tremendous fall in the cost of processing power has not been accompanied by a comparative fall in price or increase in availability in high speed communications. It appears that Econet is still the only contender in the low cost network field. It allows around 200 kilobits per second transfer at a cost of around £50 per station. Although LAN's like Ethernet allow 10 megabits per second, the cost is very great. In order to allow low cost simulation with many machines in the system, it will be necessary even in the future to keep to the data transfer rates provided by Econet. This scenario would allow much more powerful modelling in each machine, allowing more accuracy, but data updates still of around once per second. It was noted that much of the data transfer delay was due to the speed of processing at the transmitting and receiving machines. This factor will reduce with more powerful machines.

Other approaches are possible. The workstation type

microcomputers now available are powerful enough to allow a small number, say four, to give a useful computing unit for simulation. These machines are designed to operate over a LAN which allows very fast interprocessor communications. The workstations are fairly expensive, so that the fast LAN would represent a relatively small fraction of the overall hardware cost. However, these machines do not appear to be many times more powerful than the Archimedes machine, while they cost several times as much.

Another processor which should be considered in the context of multi-processor simulation is the Inmos Transputer[15]. This is a powerful board level microcomputer which has been designed to connect to other transputers to form a multi-processing unit. Each transputer has its own local memory and four links to connect with other Transputers. The programming language OCCAM[16] has been designed for the processor and incorporates the features required for parallel processing and communications between processes. The system has the great advantage of being able to be increased in size by one Transputer at a time until the appropriate performance is obtained. Very little program change is required to adapt code, written for a single Transputer, to run on a number of Transputers. It does appear that the main drawback is expense. The cost is in the order of £1000 per basic processor board which does

not include any power supply or input/output device.

In attempting to provide low cost simulation for training purposes the hardware approach described in this work is extremely attractive. The use of low cost microcomputers, each with a display monitor, produces an excellent graphical interface for instructional purposes. However more work is required in developing modelling techniques which allow multi-processor computation.

Appendix[1]

A1.1 Operation of Econet LAN

Econet is a bus type local area network which has similarities to the more widely known Ethernet system. It also uses the csma/cd (carrier sense multiple access with collision detection) developed with Ethernet. This governs the basic action of transmitting on the bus. When a machine is to transmit, it senses the bus. If there is traffic on it, it waits and tries again. If it is free, the transmission proceeds. As the transmission progresses, the transmitter still senses the bus in case another station has started transmission simultaneously, in which case both stations "back down". A random delay is executed by both stations before a retry to prevent another simultaneous access.

The cooperative transmit primitive in Econet consists of a "four way handshake". Four distinct actions occur as follows:

1 the source station sends out a "scout" message to ensure that the destination station is prepared to receive

2 the destination station returns an acknowledgement indicating that it has a suitable control block set up for the transmission

3 the source station sends the data

4 the destination station sends an acknowledgement indicating that the data has been received.

During the transmit operation, no other station can initiate a data transfer because any pauses during the four way handshake are accompanied with a "flag fill". This continuous pattern on the bus, which causes it to appear busy, inhibits any other transfer.

A1.2 Physical Details of Econet

The Econet topology is that of a single bus as shown in figure A1.2.1. A clock box is situated in the centre of the cable run and terminator boxes at the ends.

The cable consists of two twisted pairs. This carries the clock pulses and the data on balanced lines to reduce electrical interference. The connection to the microcomputer consists of a 180 degree five-pin DIN-plug.

To get the maximum performance from the system care has to be taken to follow the manufacturer's advice on cabling methods. However for the short runs involved in the simulator project it was found that the system performed well without rigour in the cabling or the use of terminators. It appears that the BBC MASTER machine may not tolerate this approach and that more care needs to be taken with systems involving this machine.

Further information on the Econet system can be found in the Acorn Manual[17].

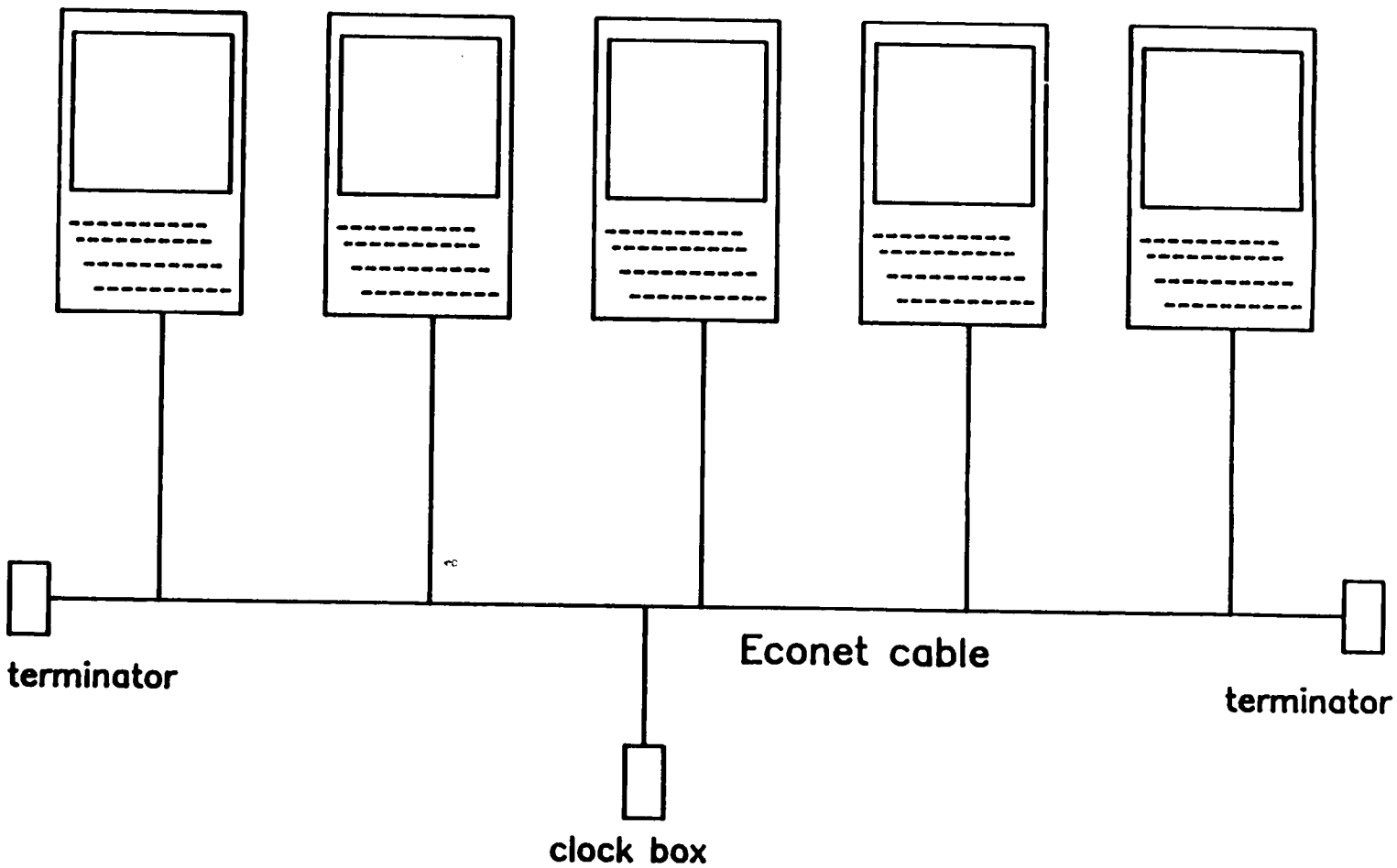


figure A1.2.1 Econet topology

Appendix[2]

Communication Routines

A2.1 Using Immediate Commands

The first interprocessor communications system used the Econet immediate command which "poked" a number of bytes directly into the memory of the receiving machine. No action was required at the receiver for the transfer to occur. As a result there was only one routine, that which was used at the transmitting machine - PROC_POKE.

The routine required the following parameters: the receive machine station number (STATION), the transmitting machine's and receiving machine's buffer addresses (LBUF1 and DBUF) and the number of bytes to be transferred (BYTES).

```
DEF PROC_POKE(STATION,LBUF1,DBUF,BYTES)  :REM**************
REM  **** POKES IN BYTES ********
LBUF2=LBUF1+BYTES
?PSPACE=&82
PSPACE?1=0
PSPACE?2=STATION
PSPACE?3=0
PSPACE?4=LBUF1 MOD 256
PSPACE?5=LBUF1 DIV 256
PSPACE?6=0
PSPACE?7=0
PSPACE?8=LBUF2 MOD 256
PSPACE?9=LBUF2 DIV 256
PSPACE?10=0
PSPACE?11=0
PSPACE?12=DBUF MOD 256
PSPACE?13=DBUF DIV 256
PSPACE?14=0
PSPACE?15=0
A%=&10
X%=PSPACE MOD 256
Y%=PSPACE DIV 256
CALL &FFF1
ENDPROC  :REM_____________________________
```

A2.2 Using Cooperative Commands

The cooperative Econet commands allowed the interprocessor communications to be carried out without the need for machines to have the addresses of buffers in other machines.

The PROC_TRANSMIT routine was used at the transmitter to send out a buffer of data.

```
DEF PROC_TRANSMIT(START,FINISH,OUTBLOCK)   :REM*****************
REM  Sets up a transmit control block, initiates call until accepte
REM  then polls until complete.  Returns an error code if unsuccessf
OUTBLOCK?1=SORT_PORT :REM  sets up control block
OUTBLOCK!2=SORT_MACH
OUTBLOCK!4=START
OUTBLOCK!8=FINISH
A%=&10:X%=OUTBLOCK:Y%=OUTBLOCK DIV 256
REPEAT
  ?OUTBLOCK=&80
  CALL OSWORD :REM repeatedly calls transmit until sent
  UNTIL ?OUTBLOCK<>0
REPEAT: REM  polls until transmitted
  A%=&32:POLL=USR OSBYTE
  UNTIL (POLL AND &8000)=0
RESULT=(POLL AND &FF00) DIV 256
ENDPROC : REM_____
```

FN_OPEN_RECEIVE was used at the receiver to prepare a buffer to accept data from a transmitter. It returned a control block number which was used to monitor progress.

```
DEF FN_OPEN_RECEIVE(BLOCK,ARRAY_ADD,PORT) :REM********
REM  Sets up receive block and returns block number
REM  it deletes the control block
?BLOCK=0  :REM sets up control block
BLOCK?1=&7F
BLOCK?2=PORT
BLOCK!3=0
BLOCK!5=ARRAY_ADD
BLOCK!9=ARRAY_ADD  + ARLEN*5
X%=BLOCK:Y%=BLOCK DIV 256:A%=&11
CALL OSWORD :REM  opens up receive buffer
=?BLOCK
REM end of fn _____________________________________________
```

FN_RECEIVED was used to poll the control block for the arrival of a buffer to a particular port. The control block number (BLOCK_NUM) was passed to it as a parameter.

```
DEF FN_RECEIVED(BLOCK_NUM)  :REM**********
REM  Polls ONCE for a receive and then returns the result
LOCAL POLL,RESULT
A%=&33:X%=BLOCK_NUM
POLL=USR OSBYTE
RESULT=(POLL AND &8000)
=RESULT
```

PROC_DEL_BLOCK was used to delete the control block information reserving the buffer after a transmission had been received. It also was passed the control block number (CTRL) as a parameter.

```
DEF PROC_DEL_BLOCK(CTRL) : REM******
REM deletes receive control block
A%=&34:X%=CTRL
CALL OSBYTE
ENDPROC : REM__________
REM
```

A2.3 Implementing the Packet System

The packet switching system was built upon the Econet cooperative routines described above. In order to send a packet of data associated with a pipe or a control, the data was placed in the reserved words such as OUT_PRES, OUT_TEMP and the PACKET_TYPE and AREA values assigned. The PROC_SEND_PACKET routine was called.

```
DEF PROC_SEND_PACKET : REM*************************************
REM Uses proc make packet to place packet on buffer user friendly
REM  GLOBAL PACKET_DEST,PACKET_TYPE,PACKET_ITEM,OUT_PRES,OUT_TEMP,OUT_CONC
LOCAL PLEN
 IF PACKET_TYPE<>1 AND PACKET_TYPE<>4 AND PACKET_TYPE<>6AND PACKET_TYPE<>8
 THEN ENDPROC:REM non existent type
 IF PACKET_TYPE=1 THEN XX(0)=OUT_DEST : XX(1)=OUT_PRES : XX(2)=OUT_TEMP :
 XX(3)=OUT_B: XX(4)=OUT_CONC1:XX(5)=OUT_CONC2 : XX(6)=OUT_FLOW:PLEN=7
IF PACKET_TYPE=4 THEN XX(0)=OUT_DEST : XX(1)=OUT_SET :PLEN=3
  IF PACKET_TYPE=8 THEN XX(0)=OUT_DEST:XX(1)=OUT_CPRES : XX(2)=OUT_CTEMP
  :XX(3)=OUT_CB:XX(4)=OUT_CCONC1:XX(5)=OUT_CCONC2:XX(6)=OUT_CLEVEL:PLEN=7
IF PACKET_TYPE=6 THEN XX(0)=OUT_DEST : XX(1)=OUT_SHUT:PLEN=3
PROC_MAKE_PACKET(OUT_DEST DIV 100,OWN_AREA,PLEN,PACKET_TYPE)
ENDPROC : REM_____________________________________________________
```

This in turn called the PROC_MAKE_PACKET routine. The parameters of destination area (DEST), source machine (orig), length of packet (N) and type of packet (TYPE) were produced by the SEND_PACKET routine.

```
DEF PROC_MAKE_PACKET(DEST,orig,N,TYPE)
REM Makes a packet of N reals from xx() and places on buffer
REM  GLOBAL XX(),POINTER,ADDXX,FROM,
?POINTER=4+5*N+1 : REM length of packet
POINTER?1=DEST
POINTER?2=orig
POINTER?3=TYPE
POINTER=POINTER+4
PROC_TRANSFER(5*N,ADDXX,POINTER)
POINTER=POINTER+5*N
?POINTER=&0D
POINTER=POINTER+1
ENDPROC : REM_____________________________________________
```

Once all the packets had been stored in the buffer by the use of the SEND_PACKET routine, the PROC_CLOSE_BUFFER routine was called at the end of a processing cycle. This in turn called the PROC_SEND_BUFFER routine which activated the Econet cooperative transmit. Also the CLOSE_BUFFER cleared out any data which had arrived during the last cycle by using the PROC_READ_BUFFER routine.

```
DEF PROC_CLOSE_BUFFER : REM*************************************
 REM Closes up out buffer, sends it and if successful receives and unpac
     in buffer
REM GLOBAL
!BUFF_START=POINTER-BUFF_START : REM  places buffer length at start
PROC_SEND_BUFFER
  IF (RESULT AND &40)=0 THEN PROC_READ_BUFFER
POINTER=BUFF_START+4
ENDPROC :    REM_____________________________________________


DEF PROC_SEND_BUFFER : REM*************************************
REM Uses transmit to send buffer of packets to sorting processor
REM and immediately reopens the receive buffer
PROC_TRANSMIT(BUFF_START,POINTER,EBLOCK)
ENDPROC : REM_____________________________________________
```

The PROC_READ_BUFFER utilised the Econet cooperative commands to open a buffer for expected data and to test for its arrival. When a buffer arrived from the sorter, the individual packets were retrieved and acted upon. In every case the FN_GET_REALS routine was used to transfer the values from the buffer to an array. Then depending on the type of packet one of the following were called:

PROC_ALTER_VARS to update process values from a data packet

PROC_ALTER_CONTROL to change a valve or pump setting

PROC_ALTER_SHUT to update the esd status of a valve or pump

```
DEF PROC_READ_BUFFER : REM************************************
REM Removes all packets from buffer and allocates to varibles
REM GLOBAL CTRL_NO,POINTER,BUFF_START
LOCAL S,BUFF_STOP,T,N,I
IF BOPEN=FALSE THEN CTRL_NO=FN_OPEN_RECEIVE(BLOCKA):BOPEN=TRUE
I=0
REPEAT
  S=FN_RECEIVED(CTRL_NO)
  I=I+1
  UNTIL S<>0 OR I=20
IF S=0 THEN ENDPROC:REM no message yet
PROC_DEL_BLOCK(CTRL_NO):BOPEN=FALSE
BUFF_STOP=!IN_START+IN_START : REM Length of whole buffer at start
POINTER=IN_START+4
REPEAT
  T=POINTER?3
  IF T=1 THEN N=FN_GET_REALS :PROC_ALTER_VARS(N)
   IF T=4 THEN N=FN_GET_REALS :PROC_ALTER_CONTROL
   IF T=8 THEN N=FN_GET_REALS :PROC_ALTER_CONS(N)
   IF T=6 THEN N=FN_GET_REALS :PROC_ALTER_SHUT
  POINTER=POINTER+?POINTER
  UNTIL POINTER>= BUFF_STOP
ENDPROC : REM_____________________________________________


DEF PROC_ALTER_CONTROL : REM*****************************
REM changes the control value for a valve or pump
REM GLOBAL CONT(),XX()
CONT(XX(0)MOD100)=XX(1)
ENDPROC : REM_____________________________________________
REM
DEF PROC_ALTER_SHUT : REM********************************
REM changes the shut off flag for a valve or pump
SHUT_OFF(XX(0) MOD 100)=XX(1)
ENDPROC : REM_____________________________________________
REM
REM
REM
DEF FN_GET_REALS : REM***********************************
REM Transfers a real array from the buffer to XX()
REM GLOOBAL POINTER, FROM
LOCAL SIZE
SIZE=?POINTER
PROC_TRANSFER(SIZE,POINTER+4,ADDXX)
=SIZE/5 -1 : REM number of reals
REM end of FN _____________________________________________
REM
REM
DEF PROC_ALTER_VARS(VARNO) : REM*************************
REM Places values inXX() into appropriate array
 REM GLOBAL XX(), PIPE_VAL()
LOCAL PIPE, K
PIPE=XX(0) MOD 100
FOR K=1 TO VARNO-1
  PIPE_VAL(PIPE,K)=XX(K)
  NEXT K
ENDPROC : REM_____________________________________________
```

Finally the "GET" routines were used to give a "user friendly" method of accessing the passed values. In each case this involved the access of the appropriate elements of an array.

```
DEF FN_GET_FLOW(PIPE): REM**
REM returns flow from data array
=PIPE_VAL(PIPE MOD 100,6)
DEF FN_GET_B(PIPE) : REM********
REM returns B from data array
=PIPE_VAL(PIPE MOD 100,3)
DEF FN_GET_CONC2(PIPE) : REM****
REM returns CONC2 from data array
=PIPE_VAL(PIPE MOD 100,5)
DEF FN_GET_CONC1(PIPE): REM****
REM returns CONC1 from data array
=PIPE_VAL(PIPE MOD 100,4)
DEF FN_GET_TEMP(PIPE): REM**************************************
REM returns TEMP from data array
=PIPE_VAL(PIPE MOD 100,2)
DEF FN_GET_PRES(PIPE): REM**************************************
REM returns PRES from data array
=PIPE_VAL(PIPE MOD 100,1)
REM  end of fn ____________________________________________________
DEF FN_GET_SET(PIPE): REM***************************************
REM returns KV from data array
=CONT(PIPE MOD 100)
DEF FN_GET_SHUT(PIPE): REM**************************************
```

Appendix[3]

Simulation Language Routines

An example of the use of the routines is given in the following tank routine PROC_TANK1.

```
DEF PROC_TANK1  :REM  ****************************************
REM   **  General tank routine  **
REM LOCAL DER_H,DER_M,HFin
REM GLOBAL LEVEL,B,M,Dliq,CSA,Fin,Cliq,Tin
LOCAL DER_M,DER_H,HFin
REM  GLOBAL LEVEL,B,M,Dliq,CSA,Fin,Fout,Cliq,Tin,OLDP,OLDT,T,P
REPEAT :REM  integration contolled loop  **
  REM  algebraic part   **
  PROC_TANK_ALG
  LEVEL=(1-B)*M/Dliq/CSA
  REM  integration part  **
  REM DERIVATIVES
  DER_M=Fin-Fout+Vin-Vout :REM dM/dt
  CALL derivative,M,DER_M
  HFin=Fin*Cliq*Tin
  HFout=Fout*Cliq*T
  HVin=Vin*(SLH+Cvap*T)
  HVout=Vout*(SLH+Cvap*T)
  DER_H=HFin-HFout+HVin-HVout :REM  dH/dt
  CALL derivative,H,DER_H
  UNTIL FN_END
ENDPROC
```

Each "CALL derivative" caused a machine code routine (derivative) to be run which stored the address of the variable being integrated and the present value of its derivative w.r.t. time.

```
.derivative LDA &801
LDY £00\STORES INT VAR ADDRESS
STA (&70),Y
INY
LDA &802
STA (&70),Y
INY
LDA &804\ TEMP STORE OF DER ADDRESS
STA &82
LDA &805
STA &83
LDA &70\SPACE POINT +2
CLC
ADC £2
STA &70
LDA &71
ADC £00
STA &71
LDY £04\XFERS DERIV VALUE
.AGA LDA (&82),Y
 STA (&70),Y
 DEY
 BPL AGA
```

```
LDA &70\SPACE POINT + 5
CLC
ADC £5
STA &70
LDA &71
ADC £00
STA &71
INC NUM\ count of int vars so far
RTS
.FIND LDA &601\  FIND ADDRESS
STA DUM
LDA &602
STA DUM+1
RTS
.SWAPG LDY £4\ GETVAL SWAP
.RESG LDA (&74),Y
STA (&78),Y
DEY
BPL RESG
RTS
.SWAPS LDY £4\ STORVAL SWAP
.RESS LDA (&78),Y
STA (&74),Y
DEY
BPL RESS
RTS
]
NEXT PASS
```

At the end of the integration pass the routine FN_END was run. This retrieved the addresses of the integrated variables and the derivative values. It then chose the appropriate integration routine from PROC_FIRST, PROC_SECOND and PROC_FOURTH. These implemented the different order Runge Kutta methods according the type indicated by the variable IORD. The FN_END routine also controlled the number of passes round the integration code by passing back the value "FALSE" until the correct number of passes had occurred.

```
 DEF FN_END
  REM  does integration and housekeeping
 POINTER=?NUM: REM  number of integrated variables on this pass
 LOCAL I,ADR,DER
 ADR=SPACE
 FOR I=1 TO ?NUM
   VARAD(I)=?ADR+ADR?1*256
   DER=FN_GETVAL(ADR+2)
   DER1(I)=DER
   ADR=ADR+7
   NEXT I
  REM  transfers the SPACE table to arrays for use by integrating routines
 REM   No display from integration routines
 IF IORD<>1AND IORD<>2 AND IORD<>4 THEN PRINT "****ERROR    incorrect
    integration order****":STOP
 IF IORD=1 THEN PROC_FIRST
 IF IORD=2 THEN PROC_SECOND
 IF IORD=4 THEN PROC_FOURTH
 ?&71=INT(SPACE/256) :REM  resets SPACE table pointers
 ?&70=SPACE-?&71*256
 ?NUM=0:IF TROUND=2 THEN =TRUE :REM***DOES ONE 2nd ORDER TIME STEP********
 =FALSE

DEF PROC_FIRST
LOCAL I
FOR I=1 TO POINTER
  ADR=VARAD(I)
  VVAL=FN_GETVAL(ADR)
  VVAL=VVAL+DER1(I)*TSTEP
  PROC_STORVAL(ADR,VVAL)
  NEXT I
Time=Time+TSTEP
ENDPROC

DEF PROC_SECOND
LOCAL J
TROUND=TROUND+1:IF TROUND=3 THEN TROUND=1
IF TROUND=1 THEN FOR J=1 TO POINTER:DER2(J)=DER1(J):NEXT J
FOR J=1 TO POINTER
  ADR=VARAD(J)
  VVAL=FN_GETVAL(ADR)
  IF TROUND=1 THEN VVAL=VVAL+DER1(J)*TSTEP ELSEVVAL=VVAL
                          +(DER1(J)-DER2(J))*TSTEP
  PROC_STORVAL(ADR,VVAL)
   NEXT J
IF TROUND=2 THEN Time=Time+TSTEP
ENDPROC

DEF PROC_FOURTH
LOCAL K
FROUND=FROUND+1:IF FROUND=5 THEN FROUND=1
IF FROUND=1 THEN FOR K= 1 TO POINTER:ADR=VARAD(K):VVAL=FN_GETVAL(ADR):
                    Valu(K)=VVAL:DER2(K)=DER1(K):NEXTK:FTSTEP=TSTEP/2
IF FROUND=3 THEN FTSTEP=DSTEP
IF FROUND=2 OR FROUND=3 THEN FOR K=1 TO POINTER:
           DER2(K)=DER2(K)+2*DER1(K):NEXT K
IF FROUND=4 THEN FOR K=1 TO POINTER:DER1(K)=(DER2(K)+DER1(K))/6:NEXT K
FOR K=1 TO POINTER
  VVAL=Valu(K)+DER1(K)*FTSTEP
  PROC_STORVAL(VARAD(K),VVAL)
  NEXT K
IF FROUND=4 THEN Time=Time+TSTEP
ENDPROC
```

Appendix[4]

Microcomputer Benchmarks

The following benchmark program was developed to test various machines on their likely performance in running the simulation software. The program involves arithmetic operations and the use of array access. The program was used in BBC BASIC form, in TURBO PASCAL on the IBM machines and ISO PASCAL on the 32016 processor. With the faster processors the program was repeated a suitable number of times so that the timing might be carried out by observation of a wristwatch.

```
REM BENCHMARK FOR SIMULATION PROGRAMS
B=1.32
D=17.36
DIM A(100)
FOR I%=1 TO 100
  A(I%)=I%
  NEXT I%
FOR J%=1 TO 100
  FOR I%=1 TO 100
    A(I%)=A(I%)/(A(I%)-0.001)*A(2)+B
    C=-I%/D
    NEXT I%
  B=EXP(C)
  NEXT J%
```

The results are summarised in table A4.1.1

machine	number of cycles	time taken	time for one cycle
BBC model B	1	121	121
" "+6502 2nd proc	1	81	81
BBC MASTER with turbo board	1	47	47
32016 2nd proc	10	36	3.6
" " with no checks	10	17	1.7
ACORN ARCHIMEDES	1	8.7	8.7
IBM XT	1	57	57
IBM AT	1	16	16

notes: 1) all times are in seconds

2) ACORN/BBC machines (apart from 32016) used BBC BASIC

3) IBM machines used compiled TURBO PASCAL

4) 32016 2nd processor used compiled ISO PASCAL

table A4.1.1

Appendix[5]

Use of System as a Simulator

During the development phase the physical items comprising the simulator were arranged as in figure A5.1.1.

This was appropriate for the development of the software, but in normal use it would not be necessary for the processors to be immediately accessible. All control actions can be executed from the touch screen.

The bank of monitors is a feature of the system which would be retained in the training situation. A colour display is provided by each, enabling the status of valves and values of flows to be seen easily and in relation to the associated hardware.

In the training situation the simulator could be used in two roles. Firstly, the gas processes can be demonstrated with the system much as it is in the development configuration. Secondly, the trainee can be put in charge of the system to get some "hands on" experience. The configuration would be adapted in this case to provide a trainee's console and an instructor's console.

The purpose of the instructor's console would be to initialise the system with a particular scenario and then to monitor the trainee's progress. A set of scenarios could be provided and the ability to create "faults" in the system could be included. The layout is shown in

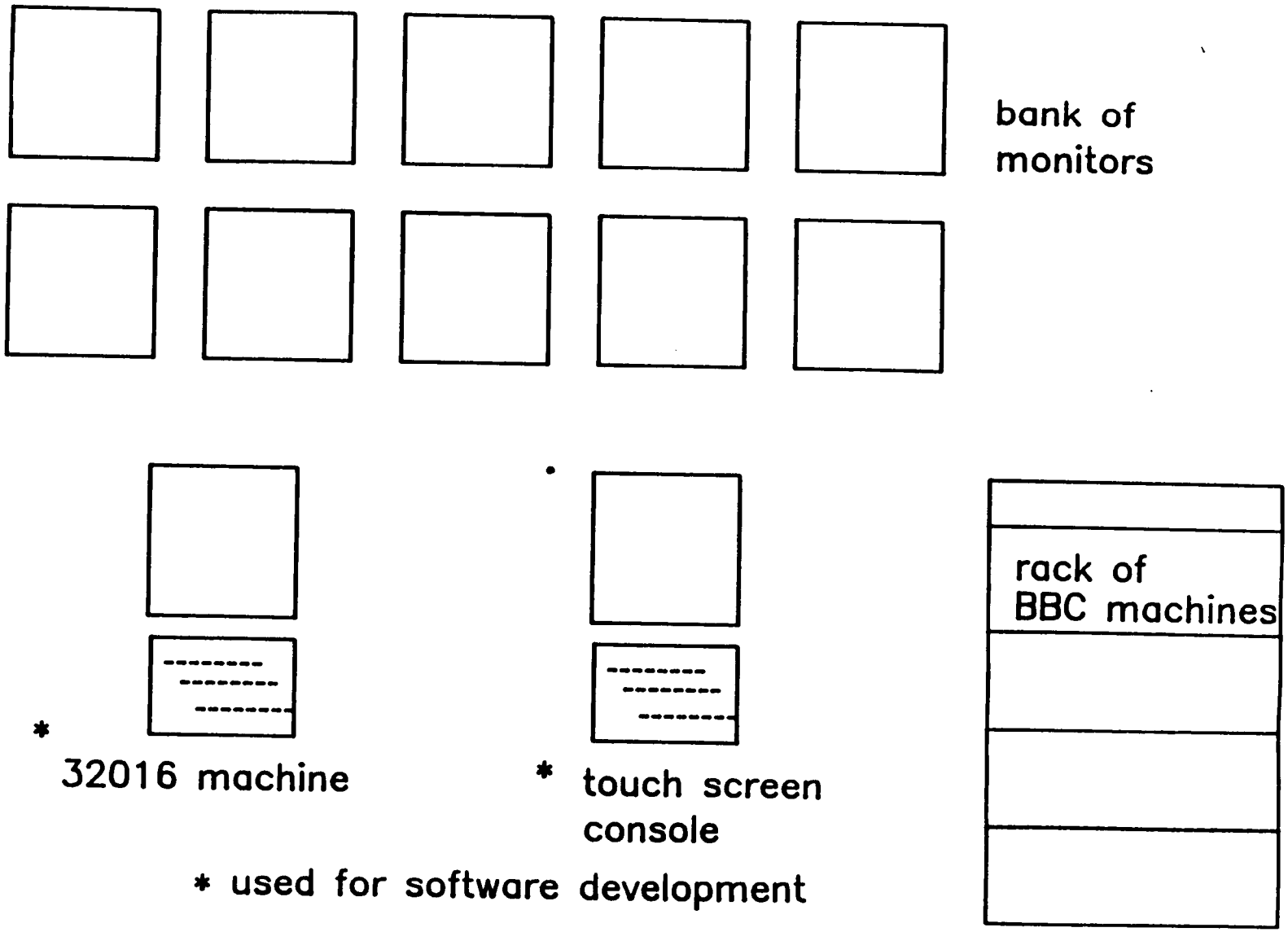


figure A5.1.1 simulator development layout

figure A5.1.2. Clearly the trainee's console would support only a subset of the possible commands to the simulator i.e. the physical controls of the system being modelled.

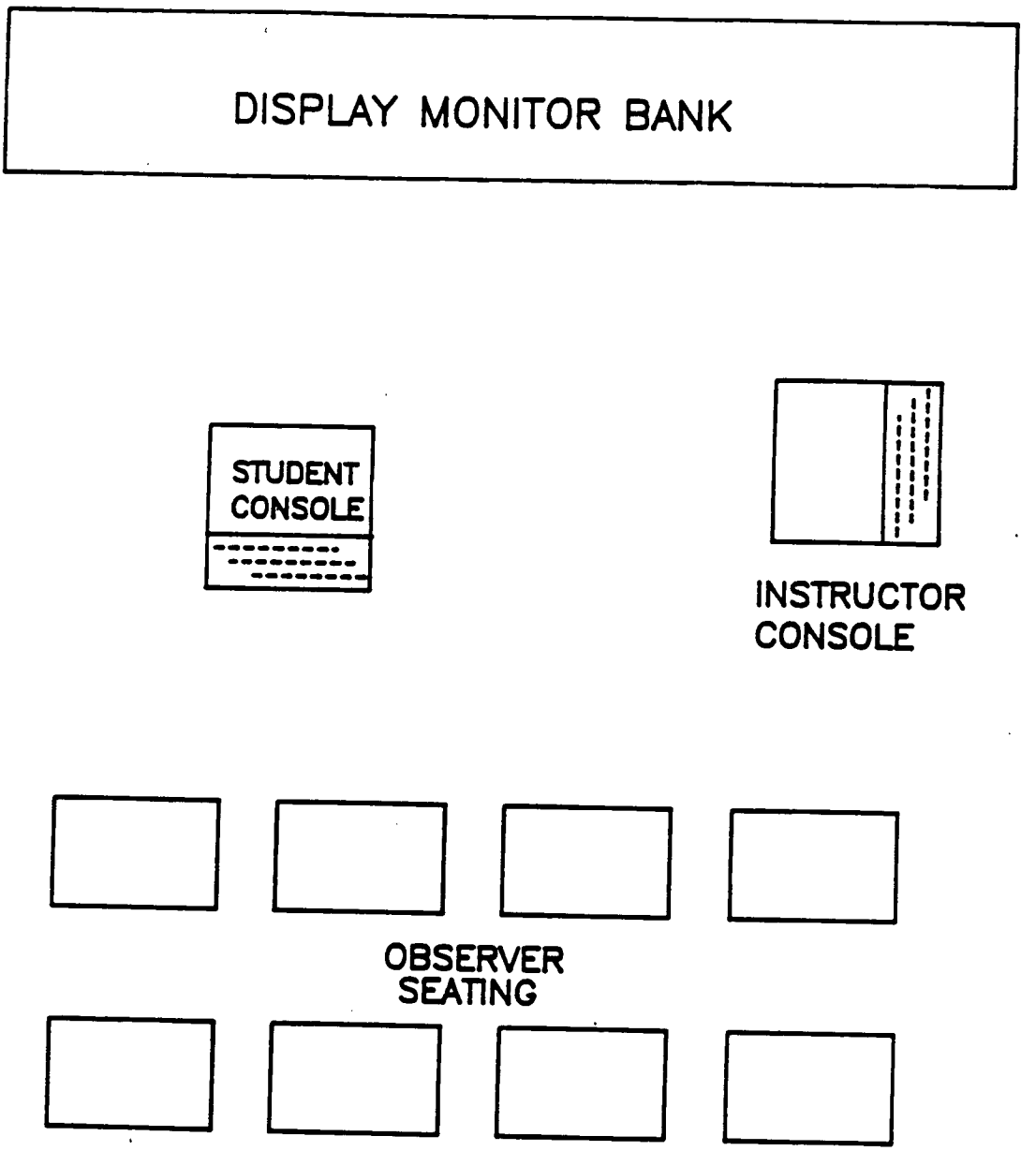


figure A5.1.2 training simulator layout

BIBLIOGRAPHY

1. Ponton, J.W., Johnston, E.M., and McGuire, G.
'Simulators for Liquefied Gas Operations Using Networked Computers'
In
Automation for Safety in Shipping and Offshore Petroleum Operations, vol 22, 37-43, Elsevier, Amsterdam 1986.

2. Korn, G.A.
'Multiprocessor Computers for Continuous System Simulation'
In
Conference on Simulation Methods for Nuclear Power Systems, Tucson, 4.1-4.16, 1981.

3. Pimentel, J.R.
'Real Time Engine Simulator Using Multiple Microcomputers'
IEEE Transactions on Industrial Electronics,
IE-30, 2, 117-126, 1983.

4. Sundatatajan, D. and Ahmad, M.O.
'Implementation of Cascade Digital Filters Using Multi-microprocessors'
In
IEEE International Symposium on Circuits and Systems.
Montreal, 1264-1267, 1984.

5. Rogers, J. L .Jr. and Sobieszczanski-Sobieski, J.
'Exploiting Parallel Computing with Limited Program Changes Using a Network of Microcomputers'
In
Proceedings of the 4th International Conference on Engineering Software,
London, 10.61-10.74, 1985.

6. Wilton, R.
'Microcomputers in NASAs SIR-B'
Byte, pp193-197, July 1985.

7. Schwartz, M.
'Telecommunication Networks: Protocols, Modelling and Analysis'
Addison-Wesley, 1987.

8. Needham, R. M. and Herbert, A. J.
'The Cambridge Distributed Computing System'
Addison-Wesley, 1982.

9. Ponton, J. W. and Vasek V.
'A Two-level Approach to Chemical Plant and Process Simulation'
Computers and Chemical Engineering, vol 10, 3, 277-286, 1986.

10. Franks, R.G.E.
'Modelling and Simulation in Chemical Engineering', Wiley, 1972.

11. Matheson, A. and Rutherford, F.
'Pressure Swing Adsorption Simulation Manual' University of Edinburgh, Department of Chemical Engineering, Project Report, 1986.

12. Forsyth, J.M.
'Simulation of Distributed Compressor Systems', University of Edinburgh, Department of Chemical Engineering, Project Report, 1987.

13. McGuire, G. and White, B.
'Liquefied Gas Handling Principles on Ships and Terminals'
Witherby, 1986.

14. Woolcot T. W. V.
'Liquefied Petroleum Gas Tanker Practice'
Brown, Son and Ferguson, 1977.

15. 'The Transputer Family'
INMOS, 1986,

16. May, D. and Taylor, R.
'OCCAM an Overview'
Microprocessors and Microsystems, pp73-79, March 1984.

17. 'Econet Advanced User Guide',
Acorn , 1983.

List of Figures