The University of Strathclyde

Machine Tool – Computer Interfacing

J. Shaw

Department of Production Management and Manufacturing Technology

MSc in Computer Integrated Manufacture - 1986

We can't return,
We can only look behind from where we came,
And go round and round in the circle game.
J. Mitchell

ACKNOWLEDGEMENTS

I would like to acknowledge the help and guidance given to me by my project supervisor, Dr. C.N. Larsson throughout the project, and during the preparation of this thesis.

I would also like to thank the departmental laboratory staff for their willing assistance with the practical aspects of the project.

The assistance of Mr. C.S. Wilson in producing the program listings was much appreciated.

Finally, I would like to thank the Manpower Services Commission for enabling me to read for this degree.

CONTENTS

Appendices

Abstract

The communication problem of interconnecting machine tools to the factory computer system is discussed, existing and future techniques being outlined. A practical example of applying computer control to an indexing conveyor is fully described.

Introduction

This thesis is presented in two parts; the first part discusses the problems involved in interfacing machine tools to computers and computer-to-computer. This later aspect is of major importance as the computer in the form of the microprocessor finds new applications within the factory and there is a move to integrate the whole in the guise of Computer Integrated Manufacturing. The existing communication techniques are examined, as are those likely to be introduced in the near future.

The second part describes the conversion of a conveyor from a self-contained assembly machine into a computer controlled flexible conveyance system capable of being integrated into a machine cell. The possible roles of the machine are discussed and hence the controller requirement determined. A suitable controller was selected as a result of a survey of those available and details are given of the circuit arrangements needed to interface with the existing control components. The operation of the new system is described along with the control program required to implement that operation. A software tool used to support the development of the control program is listed and its function briefly described.

PART 1

CHAPTER 1

The Machine Tool - Computer Interface Problem

In recent years the concept of using a computer to direct the operations of a machine tool has broadened to the extent that whole manufacturing systems can now be conceived of as being controlled by a unitary computer system. The term Computer Integrated Manufacturing has been introduced to embrace this concept. Computer Integrated Manufacture (CIM) implies that all facets of the manufacturing operation will be integrated using computer systems as the control mechanism. Not only will the older established computer based functions of Accounting, Payroll and Purchasing be brought together but also the newer applications of Computer Aided Design (CAD) and Computer Aided Manufacture (CAM). CAM can be thought of as the natural extension of using a computer to control the operation of a single machine, which merely replaces the old manual controls, to the control of several machines in a co-ordinated fashion. In effect this creates a new, large complex machine, which in turn can become part of a total manufacturing system with common purpose and direction.

If the ideal of a CIM system is to be realised then several problem areas must be tackled and the whole integrated together. The first problem is convert information into physical action, and also the reciprocal action of physical action into information. In the case of a machine tool the machine's manufacturer will have already solved this problem; however in any automation scheme there will be many bespoke devices requiring integration. These could range from simple positioning mechanisms to position or temperature sensors for process control. Because each device is relatively primitive, the inputs and outputs will relate directly to the operation of the mechanism, a pick-and-place device may require a substantial electric current to operate a control solenoid whilst a temperature sensor may produce an output of fractions of a volt. Somehow all of these various needs must be interfaced to the rest of the system. Once all of these physical elements have been interfaced to a suitable controller the next problem to be solved is one of communication. Having produced a sub-system which can respond to instructions and the physical environment, this sub-system must be put in communication with the other sub-systems with which it will form the total CIM system. The communication problem can be broadly split into two sections. The first is physical and concerns the choice of transmission media, the transmission technique and the mechanics of the connection to the transmission system. The second part of the problem is really one of semantics, the language that is used to describe manufacturing operations, the rules to determine that messages are routed correctly and have not been corrupted during transmission. Having solved all of these problems we have now got a 'manufacturing vehicle' capable of performing a multitude of tasks asked of it. Just as a motor vehicle is not designed to drive to a particular destination but will respond to the requests of its driver within the range of its capabilities, then so the 'manufacturing vehicle' can go in many directions under the instruction of its driver. The driver in this case will be a suite of application programs defining the operating procedures of the factory. Many of these problems are common to other data systems, wherever several computers have to be interlinked there is a communication problem to be solved. This problem is usually exacerbated if a different manufacturer, each with its own mechanical, electrical and software standards, supplies each computer. The International Organization for Standardization (ISO) in an attempt to resolve these difficulties has produced the Open Systems Interconnection reference model (OSI). This model splits the communication problem into seven levels ranging down from level seven, the Application Layer, to level one, the Physical Layer. Whilst mainly intended as a model for

telecommunication and computer networks this model provides a feasible description of the CIM system problem. The OSI is at present a reference framework only, the details need to be filled in at each level before a practical system would exist; indeed, many different systems could be created that conform to the OSI model. Different representations of the model may however be interconnected if suitable interfaces are created at any of the seven layers. One implementation of the OSI model has been initiated by General Motors under the name Manufacturing Automation Protocol (MAP) and is directly applicable to the CIM solution as it is the avowed intention of the company to only procure automation systems conforming to this specification

CHAPTER 2

The Open Systems Interconnection Model

The International Organization for Standardization (ISO) has proposed a model for communications systems under specification ISO 7498 called the Open Systems Interconnection Reference Model (OSI). The model is formed of seven layers, the first three being primarily concerned with providing reliable interconnection between terminals, the remainder handling higher level protocols. Each layer of the model provides a service to the next higher layer, building upon the facilities provided by the one below to make more sophisticated facilities available. The intention of ISO is to issue a set of standards that will define the services provided by a layer and specifies the protocols to be used.

Brief Description of the Model.

Layer 7:	Application	Provides the interface to user applications and common services such as file transfer and terminal support.
Layer 6:	Presentation	Provides independence to application processes from differences in data representation through syntax transformations.
Layer 5:	Session	Controls dialogues between users and supports synchronisation of their activity.
Layer 4:	Transport	Provides user-to-user services, including multiplexing, to make the most effective use of the network facilities.
Layer 3:	Network	Provides destination switching, routing and relaying functions, independent of actual network use.
Layer 2:	Data Link	Provides transfer and control of data over communication lines with error correction.
Layer 1:	Physical	Deals with the physical attachment to communication lines.

Because the model is split into layers it is possible to interconnect to other systems provided an interface is made at the appropriate layer. The degree of communication that will be available will depend on the compatibility of other layers in the system. For example it might be desired to use a different form of physical layer in a non-OSI system, in this case an interface could be made at layer 2 on the OSI model with connection to the non-OSI system made via the new physical layer. Any higher layer in the OSI model that also exists in the non-OSI system can communicate on a peer-to-peer basis. In the case of a terminal device such as a machine tool designed to be connected to an OSI system a transformation must take place between the physical link to the communication network and the physical link to the machine's mechanical functions. The first three OSI layers are required to connect to the network and a 'bridge' to the machine functions must be made at OSI layer 3 or above; the

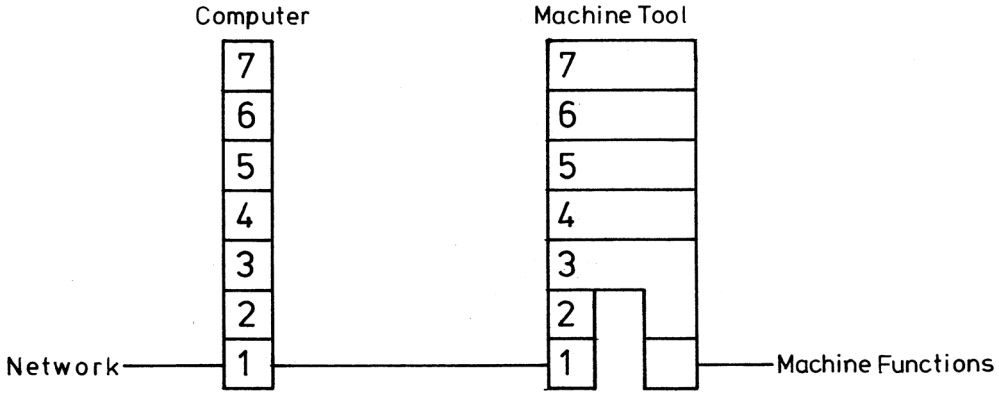
higher the bridge is made the more 'intelligent' the machine will appear, as it will be possible to carry out a dialogue using a higher level language or protocol. Figure 1 shows in diagrammatic form the OSI model and its application to a machine tool and the interconnection to the control system network

Figure 1

OSI Model

7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data Link
1	Physical

OSI Machine Tool Interface



CHAPTER 3

The Manufacturing Automation Protocol

General Motors of America, in common with all large vehicle manufacturers, has a keen interest in automating the manufacturing process, and as a consequence is well aware of the associated problems. Because it has bought equipment from many suppliers it has had to solve the problem of interconnecting this plant to form a coherent system. While it can be expected that individual vendors would produce families of equipment capable of interworking, it is impractical and undesirable for a large manufacturing company to be dependent on a single supplier. The General Motors approach was to set out to create its own standard, which it would impose on any potential supplier, rather than be forced to accept what was available. Obviously such a strategy is only possible if the specifier represents a significant market to the suppliers. Rather than set out on a completely new path it was decided at an early stage to base the General Motors system on the International Organization for Standardization 's OSI model, adopting the ISO protocols where applicable. The resulting system has been called the Manufacturing Automation Protocol (MAP). The vendors of automation equipment have also suffered from the communication problem, their best strategy is not clear, they can either develop their own superior standards or attempt to be 'plug-in' compatible with the market leader. This later strategy is fraught with difficulties, the standard might not be published, it might be protected by patent or new facilities might be added. It is therefore an attractive option to liase with a large user of automation in producing and developing an industry standard. As a result many prospective suppliers and users in the USA have joined a MAP users group to provide expertise and backup for the project. A European users group has also been formed.

As the OSI model is not yet fully defined, and, as some of its facilities are not applicable to a manufacturing scheme MAP will not be exactly the same as OSI. However, because of the layered architecture of the OSI model it is possible to establish links between other systems, which while not being true OSI representations, nevertheless conform to the same architecture. An example of the flexibility that the OSI model allows is given by the National Engineering Laboratory's (NEL) demonstration of MAP. Because the MAP is still under development, and because the final physical link will require specialised semiconductor integrated circuits to implement it which were not available, NEL has used a proprietary local-area network, 'Ethernet', for the physical link. The fact that the physical link is non-MAP will be transparent to the higher layers of the architecture. This feature will be of great benefit in the early stages of introducing MAP to a factory, allowing existing 'islands of automation' to be tied in to the MAP network.

The current specification (MAP 2.1) uses a broadband token bus operating over cable television type co-axial cable as defined by the IEEE 802.4 standard. The data link protocol is defined by IEEE 802.2, this standard allows two different techniques to be employed: connectionless (type 1) and connection orientated (type 2) operation. MAP uses type 1, which can detect and discard messages with bad parity, leaving the detection of missing or duplicated messages to the higher layers. A third system is to be added to the standard based on IEC PROWAY type C, and it is intended to adopt this for MAP for small sub-nets within a larger scheme. This type 3 technique will offer a lower cost real-time service, allowing instant acknowledgement. It can be used for networks of up to 1000 metres and with a maximum of 24 terminal tap points with a 5 M bit/s data rate. Layer 3 has adopted the ISO CLNS option (connectionless network service), while Layer 4 uses ISO Class 4 transport protocol. These four levels guarantee delivery, message sequencing and full error detection. The ISO BCS Layer 5 session protocol is currently being used, but it is anticipated that an expanded variant

will be required. Layer 6 is not yet specified but it is planned to adopt the ISO Presentation protocol. At the application layer three main branches of services are supported: ISO CASE (common application support elements, file transfer and MAP messaging. To handle file-transfer and file-access, the ISO FTAM standard has been adopted, however as no standard exists for handling interactions between small programmable controllers and NC machines or Robots General Motors has developed its Manufacturing Message Format Standard (MMFS). GM hopes that this will be adopted as a standard within the USA.

Various suppliers are engaged in providing sub-systems for MAP such as chip-sets and hardware and software interfaces to existing computers. Other suppliers are providing 'gateway' systems, which act as an interface between their proprietary networks and MAP, performing both a hardware interface and protocol conversion function.

CHAPTER 4

The Physical Layer

The physical layer in an automation scheme will be in two forms; the first is the interface to the manufacturing environment and the second the interface to the communications environment. It is to be hoped that a standard communications system will be employed throughout the factory and the choice will merely be a matter of selection from those available from equipment suppliers. The interface to the manufacturing environment however is unlikely to be standardised, each primitive component needing separate consideration. In many cases some efforts will have been made by the makers of these components to simplify these requirements. An actuator may contain its own power source and may only require a switch closure to cause it to operate, the switch current being buffered by an amplifying device internal to the actuator. In other cases, particularly in the case of sensing devices a substantial amount of signalling conditioning may be required as the output relates directly to the properties of the sensing mechanism. With the spread of microelectronics some of this conditioning circuitry may be integrated into the sensor itself so aiding the interface task. Any interface must be electrically robust; the manufacturing environment contains many high power devices liable to radiate energy into control circuits. There are advantages in using as high a voltage as possible for signalling, as it is easier to discriminate between interference and the true signal. However there are also disadvantages with high voltages too, principally that of reducing hazards to personnel and the expense and size of high-voltage semiconductors. As a compromise 24 volts D.C. is often used in industrial systems, a large number of sensors and standard control interfaces being available.

For connection to the communications environment there are many standards in use. Some of these were originally devised to connect computers to remote terminals or for data communications (Telex). As computers have become smaller and cheaper there has been a trend towards having many small computers dedicated to a single task or user. Following on from this is the realisation that greater utility would be obtained if all of these computers could be inter-linked for the exchange of information files or the sharing of specialist peripherals. This has resulted in the development of communication networks called Local Area Networks (LAN) primarily intended for linking many devices within a relatively short distance of each other, such as within a factory or office building. The earlier standards are really intended for point-to-point communication whilst the later network standards allow for the interchange of data between any two terminals connected to the network. Obviously the operating procedures for a point-to-point link are a lot simpler than those for a network, simply involving the two parties involved. In the CIM situation a network would appear to be the more appropriate as the manufacturing data flow would be expected to mimic the flow of the manufacturing product, which in the case of flexible manufacture would change with the product type. Fixed data links would appear better suited to fixed product sequences.

Factory machines can be classified into three categories, the first being those with no provision built in for communication with other computers, the second those having a single communication port and the third those with a network port. At the present time nearly all equipment will be of the first two categories, equipment in the third category should present no problems of interfacing to the physical layer of the communications environment if made to conform to the same specification. In the first category will be found older machines that use paper tape for inputting machining data and some of the small Programmable Logic Controllers (PLC). The second category will include newer DNC machine tools and Robots. This category will usually have adopted one of the point-to-point communication standards for its data port. The implication is that this equipment is expected to be connected directly to

a supervisory computer, the design making no allowance for any sort of integration or networking. In effect the problem is pushed out of the communications port and passed on to the supervisory computer. Given the situation where there is no industry wide standard for integrating machinery together this is probably the best that can be expected of any machine tool design. If machinery of this class is to work together with others in a group or cell then some sort of supervisory computer will be required to provide the communication interface between the different machines and initiate the machine sequences. This supervisory computer or cell controller may also by means of specialised interfaces communicate with the class 1 machinery, either directly or via a machine operator (also class 1). The result is a tree-structure, with the cell controller at the node. In many existing automated factories this principle is extended with several cell controllers being connected directly to a shop computer, which in turn may be one of several connected to a main-frame computer serving the whole factory. This type of structure is directly analogous to the traditional organisation of a factory with operators supervised by a foreman, who reports to a shop superintendent and thence to a production manager. While such a structure might be appropriate to a people system with its need for position and status these factors do not apply to the CIM system with its emphasis on flexibility or the ability to dynamically restructure itself. Once class 3 machines become available for all manufacturing functions the tree-structure will be obsolete, confined solely to the extreme branches where the cost of a network connection per function has to be balanced against the cost of sharing one between several functions via a local computer.

CHAPTER 5

Point-to-Point Communication Standards

There are many standard systems available for point-to-point communication. Originally required for specifying telegraphic systems they have been adopted and adapted for communication between computers and some peripheral equipment. With the advent of computer control of machine tools these standards have found their way into the factory. All of these systems employ serial transmission techniques and can be used over distances of at least 15 metres, which is adequate for many factory applications. Several authorities issue standards, the most well known being the Electronic Industries Association (of America) (EIA) and the International Telephone and Telegraph Consultative Committee (CCITT). The EIA issue 'RS' and the CCITT 'V. 'specifications.

Current Loop

This system switches a current of 20mA (or 60mA) on or off to send a code element. Originally used for the operation of teleprinters it has the advantage of being usable over long distances (several miles), the constant current drive being unaffected by different line resistances, a prerequisite for a telegraph system. It is now obsolete.

RS-232C and V.24

This system employs a bi-polar voltage to signal each element, a positive voltage being a 'space' and a negative voltage a 'mark'. Both standards provide for separate transmit and receive channels as well as duplicate secondary channels. The signals are carried on the same pins for each standard but a different nomenclature is used. RS-232C specifies the electrical performance and the pin numbers but not the connector style. V.24 does not specify the electrical performance, this being part of V.28. Maximum cable lengths of 50 feet and capacitance of 2n5F are specified. Most applications only employ a sub-set of the standard, the full standard being applicable only to Modem control.

RS-422A

This standard covers the electrical performance of an improved system. Balanced transmission is used for transmit and receive channels, eliminating the need for a common signal return as employed in RS-232C with its attendant problems of crosswalk and earth currents. Lower signalling voltages are used allowing the use of +-5 volt supplies. Higher data rates are possible than with RS-232C.

RS-423A

This is also an electrical specification. It supports lower data rates than RS-422A. It also employs two wires for each channel, but employs unbalanced transmission. While RS-232C and RS-422A are incompatible with each other they can both interwork with RS-423A.

S5/8 (BSI DD 153)

A proposed serial interface standard that specifies the electrical standards, (unipolar 5 volt), and the mechanical standard, (8 pin connector to DIN 45326). The word structure is also defined as one start bit, eight data bits (no parity) and one stop bit. Data rate is set at 9600 bits per second. It is intended to be used for most of the applications to which RS-232C has been misapplied, e.g. computer-to-computer links, computer to peripherals etc. It has the advantage that the interface is better specified than most, reducing the tendency for variations to be produced for special needs.

The electrical requirements for three of the above standards are represented in figure 2. The acceptance 'window' for input signalling voltages are compared, as are the permissible output voltages. It is not really meaningful to relate these levels to those of the balanced circuit RS-422A specification. The ability of systems conforming to the electrical requirements of these serial transmission standards to inter-work is illustrated in figure 3. It is anticipated that early implementations of the S5/8 standard might have to provide a negative going signal level rather than a small positive level in order to interface with the older standards. Those entries shown as conditional in figure 3 are based on the premise that this drive ability has been provided.

Figure 2

Serial Interface Signal Levels

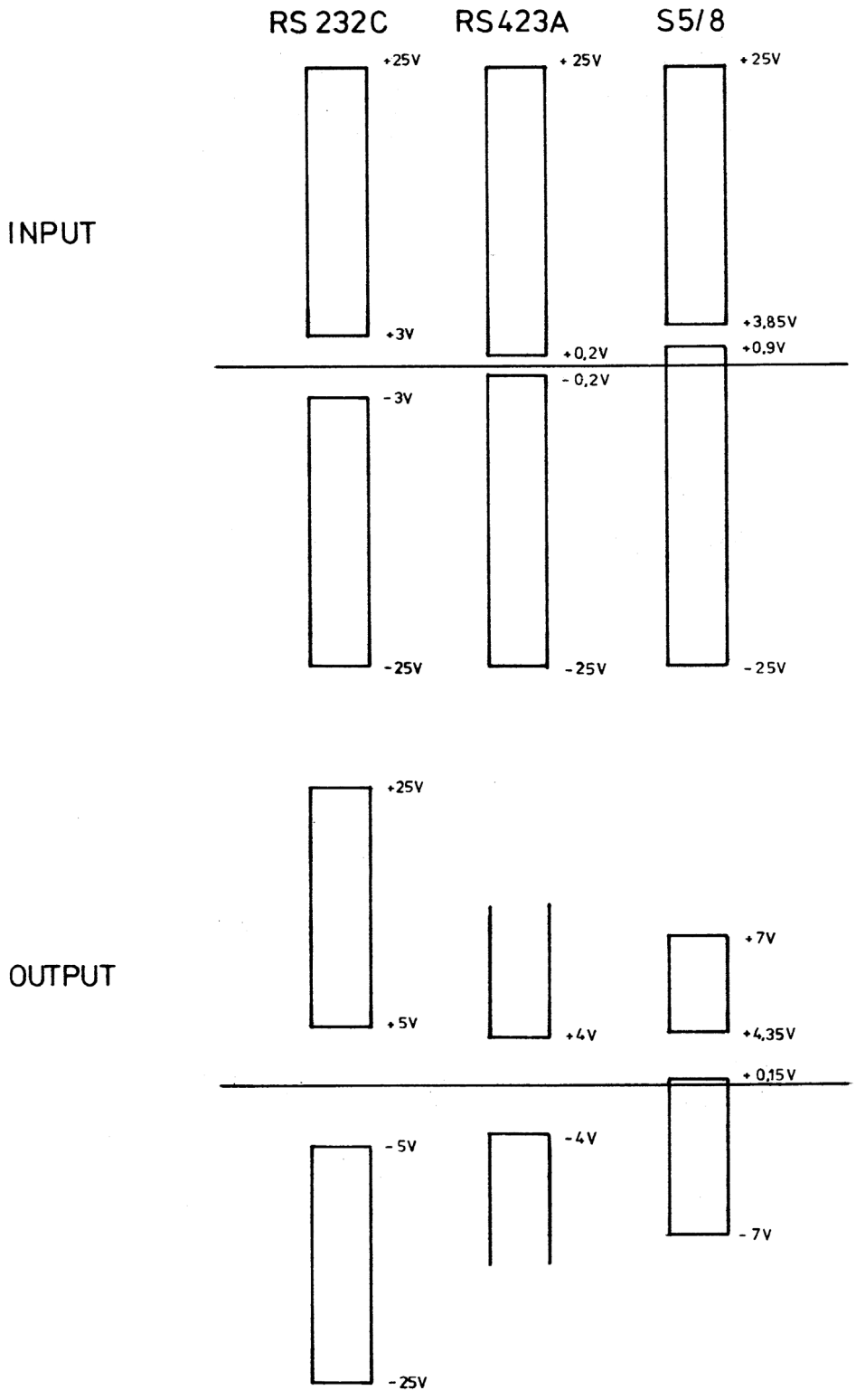


Figure 3

Electrical Compatibility

		Receive			
		RS 232C	RS 422A	RS 423A	S5/8
Transmit	RS 232C	Y	N	Y	Y
	RS 422A	N	Y	Y	N
	RS 423A	Y	N	Y	Y
	S5/8	C	N	C	Y

Y = Compatible

N = Not compatible

C = Conditional

CHAPTER 6

Networks

A network allows for communication between all of the terminals connected to it. There are three types of network, the star, bus and ring. These terms refer to the topography of the network. The star employs a central switching node connected by radial links to each of the other terminals, thus all communication is via the central node. The bus is linear in form, all terminals sharing a common physical link. Because the link is common to all terminals any terminal can communicate with any other directly. The ring, as its name implies, is a closed bus. All data flows in the same direction around the ring, which simplifies transmission problems. In a bus system transmission and reception may have separate buses with opposite direction data flows, all transmitted data being returned back down the receive bus at the 'head end' of the bus. Because the star system is essentially point-to-point (terminal to node, node to terminal) any of the established point-to-point data systems may be employed. It is evident that the topology of the network has an effect on the operating protocol, where several users share the same physical link some means must be available to ensure the correct routing of data. As with point-to-point links technology used in other areas has been adopted as a basis for some networks. The cable television industry has long had a need to send many wide bandwidth signals to a large number of users distributed around a relatively small area (compared to broadcast television). Apart from the fact that most 'users' in an industrial network would also be data providers too, there are similarities in the requirements. Indeed the newer cable television systems make provision for return signals from the user, partly as a means of signalling programme selection and audience reaction, but also to provide for future needs such as 'shopping by wire'. As a consequence cable television techniques (CATV) have been employed in some industrial network systems.

Another approach which has been adopted based on that employed by the telecommunication authorities for their new generation of data systems uses the so-called 'packet' technique in which data is grouped into blocks of defined length with the 'address' of the sender and addressee prefixed as a 'header'. Unlike a telephone conversation where a connection is made between two parties for the duration of the call the packet system routes each individual packet as and when required. This means that other users can use the network during 'pauses' and the network is free to route packets around any congestion points. The full embodiment of packet switching is unlikely to be needed for single-site factories, however some of its principles are applicable. In a bus or ring system some means must be found of selecting the data that is relevant to a particular terminal and once it has been correctly received determining who sent it so that receipt can be acknowledged. The data packet with its header serves this purpose. A second problem is sharing the common link between many users, and here again the packet is of use. Because each message is split into blocks of defined maximum length (the packet) the network protocol can determine that other users have opportunities to send their packets when the current user has sent its packet. This ensures that all users can get service from the network and are not locked out by one user sending a lot of data, in many cases the longest messages may be those of lowest priority e.g. the printing of monthly exception reports.

While the use of the data packet allows for the solution of the routing problem in a network it does not of itself control access to the network. One technique that has been employed is that of Carrier Sense Multiple Access/Collision Detection (CSMA/CD). This method is based on the fact that when no messages are being sent the network is 'quiet' but a 'carrier signal' will always be present when a message is transmitted. All terminals have access to the network at all times (Multiple Access) but before transmitting they first check

that the network is quiet (Carrier Sense). Due to the inherent delays in any extended network it is possible that several users may attempt to signal at the same time. The resulting corruption of the message is detected by each user, (Collision Detect), and both will cease transmission, waiting a random length of time before checking again for a quiet network. Because the wait periods are random, the same two users will not collide again and over a period of time their priorities will be equalised. As long as the network does not become congested it should give good service. The data packet itself can be used to give network access, as well as its use as a transport mechanism. A special packet, the token, is transmitted around the network to control access. In the case of a ring each station passes the token to the next station around the ring. If this station has data awaiting transmission it is sent, if not the token is sent on to the next station and so on around the ring. The bus network uses a more complex system, each station is provided with an access table which the idle station currently holding the token refers to determine which station the token should be sent to next. The access tables can be biased in favour of high priority users if required by giving these stations more entries in them. Some of the available and proposed network standards are briefly described below:-

IEEE 802.3

A revised form of the commercial 'Ethernet' system as supplied by DEC, Xerox and Intel. A baseband system using CSMA/CD access protocol. Compatible fibre optic systems are available.

IEEE 802.4

A token bus network. Has options for two media, baseband co-ax and broadband co-ax. Compatible with IEC 955 PROWAY (PROcess industry data highWAY) (originally designated PROWAY C.) at levels 1 and 2 in the OSI model. Used for the baseband version of MAP.

IEEE 802.5

A token ring network.

IEEE 802.8

Will be a fibre optic implementation of IEEE 802.5. IEC propose that the physical requirements of this specification be combined with the protocol of IEEE 802.4 to produce a new variant of PROWAY.

CHAPTER 7

The Cell Controller

In a factory employing Flexible Manufacturing Systems (FMS) it is likely that groups of machines will be formed, each group performing as a composite machine. This group would perform all of the operations forming a logical stage in the production of an item. It is unlikely that all machines in a large factory would be formed into one group, as it would be preferable to have duplicated groups to minimise the disruption caused by breakdowns. These groups or cells might presently be supervised by a single operator and would be physically compact so as to aid that supervision and also minimise material handling. Associated with each cell would be a cell control computer, or cell controller. Where an 'island of automation' approach has been adopted this controller would co-ordinate the activities of the cell machines under the direction of the operator via a local video display unit.

With the introduction of integrated automation the cell computer is the logical connection node between the cell and higher-level computers. The cell computer will then have the following tasks:

1) Act as communication node to factory computer.
2) Provide interface to operator.
3) Act as storage device (or virtual storage via factory computer) for machine programs.
4) Sequence machine operations, co-ordinate cell activity.
5) Supervise machines, monitor operations and fault detection.

Where machines form a fixed sequence flow-line there is less need for a cell controller as such. Any programmable device along the line could act as the controller, with the completion of each operation triggering the next. However where the cell is set up for flexible manufacturing the flow of parts around the cell is variable, entailing different routings and machine operations. In this case it would seem to be essential to have a distinct supervisory cell controller. Such a controller could make dynamic changes to the cell operation without reference to higher authority, recognising the type of part entering the line and directing the cell accordingly. This control could be extended to handling priorities within the cell to optimise part flow-rate or in response to priorities passed down from the factory computer. Thus parts might be set aside to allow higher priority work to proceed, then brought back on line as machine time allows, in other words emulating the actions of a good foreman, except that the cell controller would have better information on which to base its decisions. At the moment there are many programmable devices available that might be employed as cell controllers. These range from the Programmable Logic Controllers (PLC) through to small general-purpose microcomputers. The PLC's, particularly the smaller ones, have been designed to emulate relay logic controllers allowing factory electricians to programme them without having to learn a formal programming language. With this type of programming it is a reasonably straightforward task to control a sequential process, or perhaps a few optional alternatives in a flexible cell. The more sophisticated functions of a cell controller would be difficult to programme, the decisions probably having to be made by the factory system and passed down in the form of a new cell control program. The task of operator interfacing would probably have to be handled by a terminal connected independently to the factory system, unless simple go/no go messages are considered adequate. This would have the result of reducing the operator to being merely a simple part of the machine, whereas a more sophisticated interface could allow the operator an overview of the whole manufacturing

operation. Going to the other extreme the small personal computer offers a very flexible tool for solving the cell control problem at the expense of requiring specialist programming skills and protection from the workshop environment. The need for programming skills will reduce as off-the-shelf program packages become available, offering both suitable readymade solutions and high-level programming aids which will assist program generation. There will also be a greater degree of computer literacy amongst the work force in the future. With the introduction of factory wide automation schemes using machinery conforming to a standard such as MAP, it may be thought that a cell controller might no longer be required. Whether this is so will depend on the information handling capacity of the network, the processing power available and the sophistication of the terminal equipment. For example if a machine tool has sufficient internal storage for many part programs and the ability to handle the higher level protocols the demands on the network will be quite low. If all programs have to be sent along the network in a low-level form for each part produced then it is likely that the network will be overloaded, which in turn will reduce available machining time. As the real price of electronic controls falls it is likely that there will be pools of under-utilised electronic intelligence throughout the system, indeed this is happening now with identical microprocessors in some instances replacing simple relay controls and in others doing the work that needed a main-frame computer in years past.

Even when MAP-type machines become commonplace there will still be a need to interface simple mechanisms such as position detectors to the system, this might be via spare inputs/outputs provided on machine tools or robots, or the cell controller. If a single type of cell controller were to be used throughout a factory then it would be an advantage to use them to handle these bespoke devices as the design staff within the company, who usually specify these items, would gain in experience rather than have to redesign the sub-system to be compatible with plant that might have been specified by contractors.

Probably the best reason for retaining cell controllers would be the security that they would give to the total manufacturing process. Faults in part of the system would be confined to a single cell, leading to a degradation of throughput rather than total stoppage. Given the ability to transfer information rapidly anywhere within the factory and that distributed control is possible and economic, it would seem to be best to give the cell as much autonomy over its own operations as possible. The network would then be restricted to handling the exceptions and process monitoring rather than for direct control. Consider the case where a cell has become defective, the network would be used to set up different routings, minimising the disruption, real-time control still being maintained by the individual cells. When the system is running fault free, network traffic would be relatively small and of low priority. Should the network fail the cells would continue to function, leaving throughput unchanged.

The analogy could be drawn with a manually controlled factory, the absence of the production manager does not result in zero production, the foremen continue to work to previously established schedules, however if it became possible for the production manager to direct all operations then the loss of the manager or any of the links in the chain of command would be disastrous. The computer makes this option possible but no more secure. The disadvantage of the manual system is that instructions are misinterpreted and the manager misinformed. The use of autonomous local control coupled to a reliable network retains the established advantages of the manual system while adding the sought-after benefits of direct control.

PART 2

CHAPTER 1

The Indexing Conveyor

As a practical example of machine control and computer interfacing it was decided to convert an existing Indexing Conveyor to computer control such that it could be integrated into a work cell with other machines and an assembly robot.

The conveyor itself consists of 48 small pallets attached to a chain driven by a Geneva mechanism to give intermittent motion. A plan view of the machine is given in figure 4. The original control system was provided by a hard-wired sequence controller fitted in a control cubicle which also housed a transformer and motor control gear. The operator interface to the conveyor was provided by the switch panel, shown in figure 5, attached to the control cubicle. Several pick-and-place mechanisms were also located around the track, each with their own sequence controller. An air and vacuum system was mounted below the track to serve these devices. The conveyor, in conjunction with the pick-and-place mechanisms, formed a keyboard switch assembly machine. Because the conveyor control system was undocumented and in any case related to the needs of its previous owner it seemed appropriate to replace the old controller with a small industrial computer which could be programmable to suit future, as yet unknown, requirements.

There are two possible main control modes for a conveyor; the first would use direct manual control via push buttons. This mode would be useful for maintenance and commissioning. The second permits the conveyor to operate under some automatic sequence. In reality the conveyor's own computer would control both types of operation, in the manual mode the state of the manual control panel would be monitored and used to direct the control program. The conveyor itself could be used either as part of an assembly machine (its original use) or as a materials handling system. The first application would require the conveyor to index to the next position, trigger assembly operations, and on completion of these operations proceed to the next position. The second application would use the conveyor primarily as a transport mechanism, moving parts loaded on at one end to an unloading point further along the track. This application may require the conveyor to move past many positions without stopping, the move may be initiated by the need of the load device for an empty pallet, the need of the unloading device for a full pallet or the need to transfer parts down the line as fast as possible or a combination of any of these. A more sophisticated use would be to permit re-sequencing of loads, thus the next free pallet would be presented to the loading point but the loaded pallets would be stopped at the unloading point in a different sequence. It is possible that the loading and unloading points in this case might be one and the same, the conveyor acting as a storage magazine for a machine tool or assembly robot. This last application would provide flexibility in a fully integrated factory, allowing for changes in priority without stopping the whole system as would happen if there were no re-sequencing capability in the materials handling system.

To carry out these more sophisticated tasks the conveyor controller would need to keep track of the status of each pallet so that it could identify which were free for use, and where specific loads were positioned. However this type of control program would make heavy demands on computer memory and there would be the danger that as needs changed this controller would become overloaded. A more practical approach would be to confine the conveyor controller to the control of simple conveyor movement, with for provision of reporting conveyor status as required to a supervisory computer. Because the more complex moves would be dependent on the needs of other machines within the cell, or indeed of the

needs of other cells it would be more appropriate to pass the scheduling task to a computer higher in the control hierarchy, as this would have better access to other machines within the cell and the factory control system. This approach also has the advantage that the conveyor controller program would be fixed and yet flexible enough, with external support, to handle future needs.

Having considered some of the possible uses for the conveyor it is possible to make some broad statement of requirements for a replacement controller as follows:-

1) Able to communicate to other machines or cell controller.

2) Be programmable so that it could be reconfigured to suit future needs.

3) Be readily interfaced with the existing sensors and auxiliary motor control gear.

4) Fit inside the existing control cubicle.

5) Low cost.

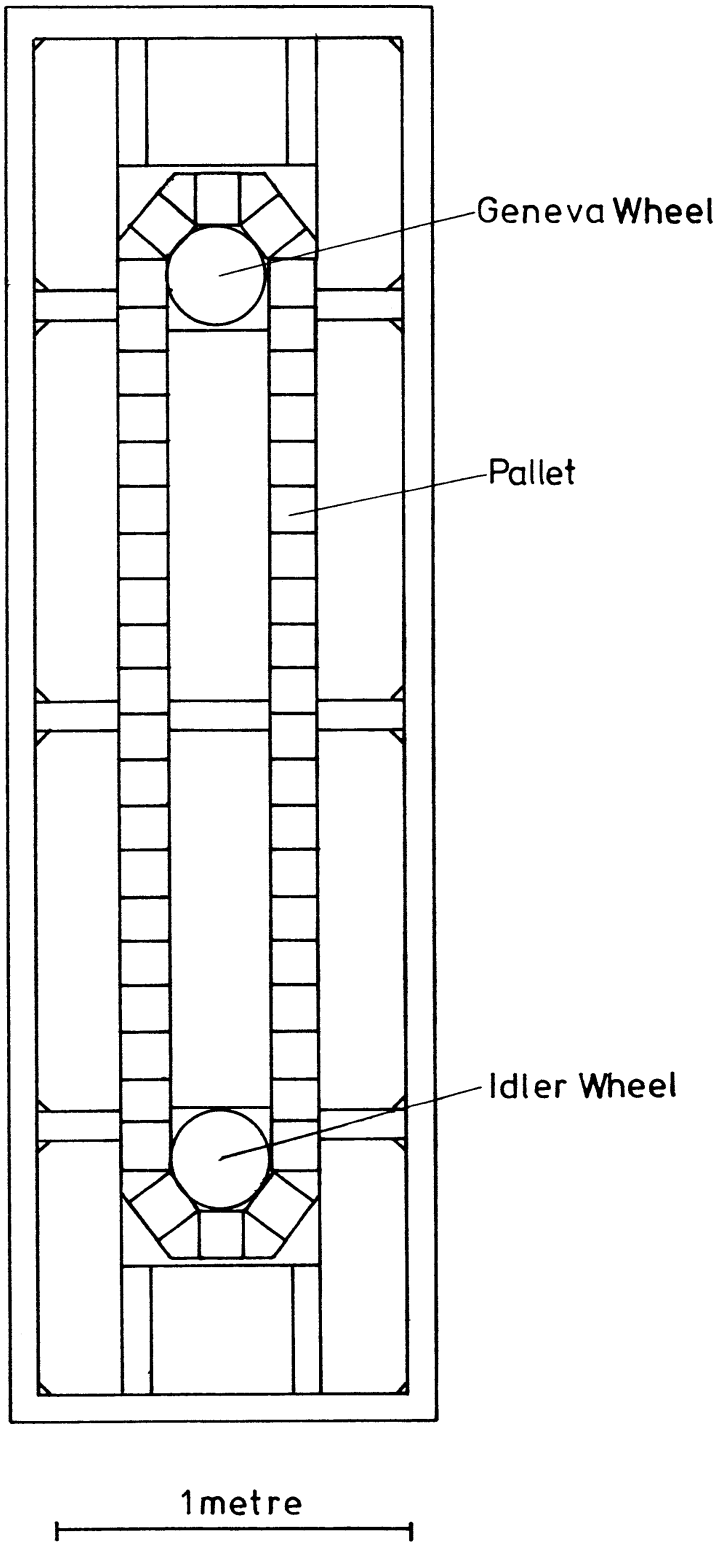
The original controller was completely self-contained and could only operate to a fixed program, or respond to simple manual controls, it was unsuited to integration with other machines. The controller utilised 24 volt logic and its outputs were interfaced to the conveyor three-phase power components by means of contactors mounted in the lower part of the cubicle. Most of the contactors use alternating current coils and solid-state relays were provided to operate these. The control signals came either from manual push buttons or buffered proximity detectors. If the prospective new controller could handle 24 volt d.c. levels at its inputs and outputs then interface problems should be minimal. While in the future there is no doubt that machinery will be provided with a data-link interface conforming to some standard such as MAP the existing de facto standard is clearly an ill-defined sub-set of RS-232C, and if the conveyor is to interwork with existing machines or cell control computers then a nominal RS-232C interface would be essential. In any case, during the transition phase the interface with a MAP system would be best made at cell controller level, with this computer carrying out the protocol conversion for each of the machines in its charge. As the control cubicle is quite large there should be little difficulty in accommodating most controllers, and the mechanical construction of the cubicle allows some reorganisation of the internal layout.

A survey of available controllers was carried out to determine which would be the most suitable for the task in hand, a summary of this survey forming appendix 3. Most of the smaller Industrial Programmable Logic Controllers utilised 'ladder diagram' programming, presumably because the personnel employed in works engineering departments are familiar with relay control systems, while formal programming languages would appear arcane. These controllers also tended to be intended for stand-alone applications with no provision for data links to other controllers. Inter-machine communication would seem to be confined to some sort of handshaking making use of input and output ports as required. In the ideal situation it should be possible to download control programs from the integrated factory communication system to any machine in the factory to provide maximum flexibility. These 'ladder diagram' programmed controllers are obviously intended to be programmed infrequently, perhaps only once in their life, rather like the hard wired-controllers that they replace. Such an approach requires that the task be very well defined. It would be expected that this would be true of any real factory automation exercise, the same care being taken in specifying the control program as any other part of the scheme. In the case of a University laboratory different factors prevail, the needs may change frequently and unpredictably; they are not geared to some production cycle or product plan. When the programming aspect is considered the situation is the reverse

presumed to be prevalent in Industry, it is relay logic that is alien to students brought up on general-purpose computer languages. As a consequence it was thought that one of the several controllers supporting various forms of BASIC might be the more suitable choice.

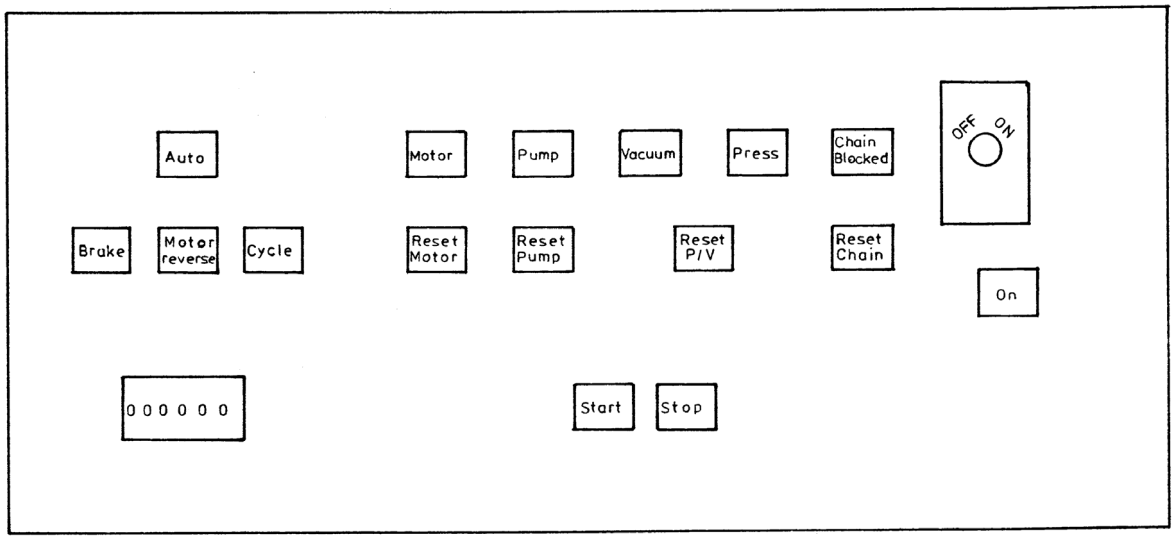
All of the pick-and-place devices had been removed before this project commenced so there was no need to consider them in the new control system, but if possible the air and vacuum systems were to be retained. A detailed study of the minimum needs in terms of inputs and outputs was made, and as a result of that study the required number of inputs was fixed at 16, and the outputs at 8.

Figure 4



Conveyor Plan View

Figure 5



Conveyor Control Panel

CHAPTER 2

The CHUM Controller

The selected controller is the CHUM 1, made by Warwick Design. It is housed in a small plastics box measuring 150 x 70 x 112 mm, and is designed to be mounted by its rear face either on a standard terminal mounting rail (DIN 46277) or moulded-in mounting lugs. All of the connections are made to screw-terminals on the front face of the module. The controller module is a Z-80 based system with 8 digital inputs and 8 outputs, a RS-232 type port, 4 analogue inputs, 1 analogue output and a real time clock. A 4K battery-backed memory is available for application programs, which may be transferred to an EPROM plugged into a front panel socket. The controller can blast the EPROM directly if an external 25-volt supply is made available.

CHUM 1 is available with several monitor programs, this particular model has the Terminal Monitor fitted. This means that a dumb terminal can be used to program the CHUM. The controller uses a form of Integer BASIC for its programs with special commands for reading input ports, writing to output ports, and communication via the serial port. It will also allow Z-80 machine code routines to be directly inserted within a BASIC program. Line numbers within BASIC are restricted to the range 0 to 999 and so the incremental interval between adjacent lines must be chosen with care if the maximum line number is not to be exceeded.

The inputs and outputs of the controller can be expanded by the addition of Input and Output modules. Each module is of the same mechanical form as the controller module. The Input module has 16 opto-isolated inputs, while the Output module has 8 independent relay contacts. The CHUM controller uses 3 address lines to control the port modules, and so the maximum system is 64 inputs (4 Modules) and 64 outputs (8 modules). The first input port on the first module is special in that it determines the mode of operation of CHUM when reset. On power up, or as a result of an external reset CHUM tests the state of this line and if the input current is below the threshold CHUM will jump to the user program in EPROM, if fitted, or RAM if not. In the event that the current is above the threshold CHUM will revert to Terminal Mode and await direct programming via the serial port. Once a suitable program has been entered it can be started by sending a RUN command.

The CHUM controller family require 9 volts d.c. to supply their internal power supply regulators, while the controller itself needs a source of 25 volt d.c. if it is to be able to blast an EPROM. All of these needs can be met by the CHUM Power Supply module. This module is of similar construction to the others in the range, but is half the size.

The CHUM controller was selected because:-

1) It has a serial communication port.
2) It can handle 24 volt inputs and outputs.
3) It is available with a small number of inputs and outputs, but is expandable beyond any likely future need.
4) It can blast its own EPROM (allowing permanent program storage).

5) It has analogue inputs and an output which might be of use with non-digital sensors.

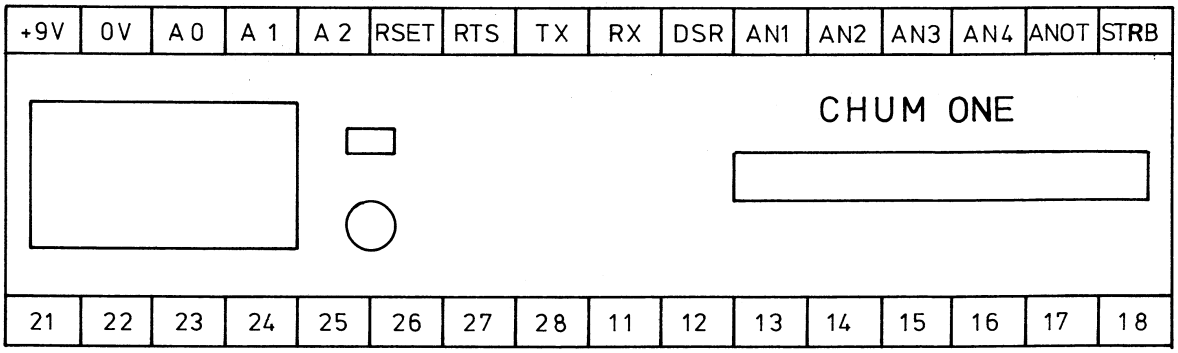
6) It is of small size and would easily fit in the available space.

7) It was available "off the shelf".

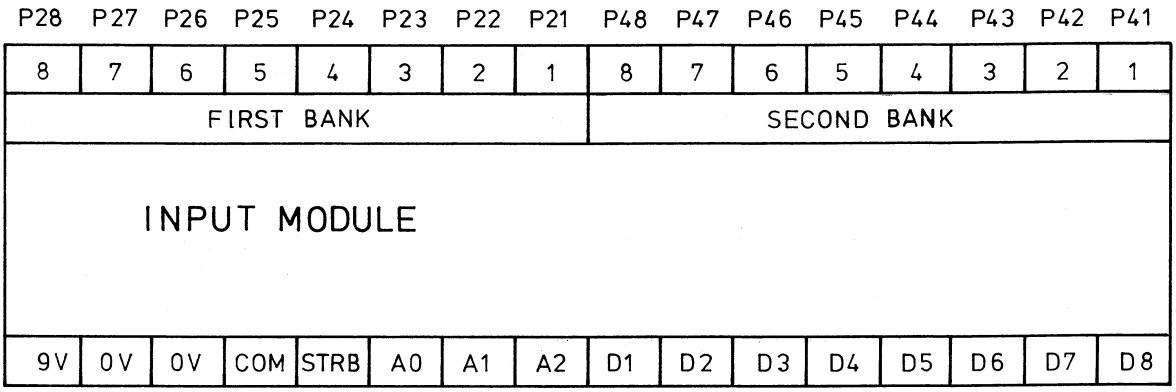
8) It was the lowest priced system surveyed.

Appendix 4 gives outline data for the CHUM computer while figures 6 and 7 detail the arrangements of the screw-terminals on each of the four modules used to make up the controller system.

Figure 6

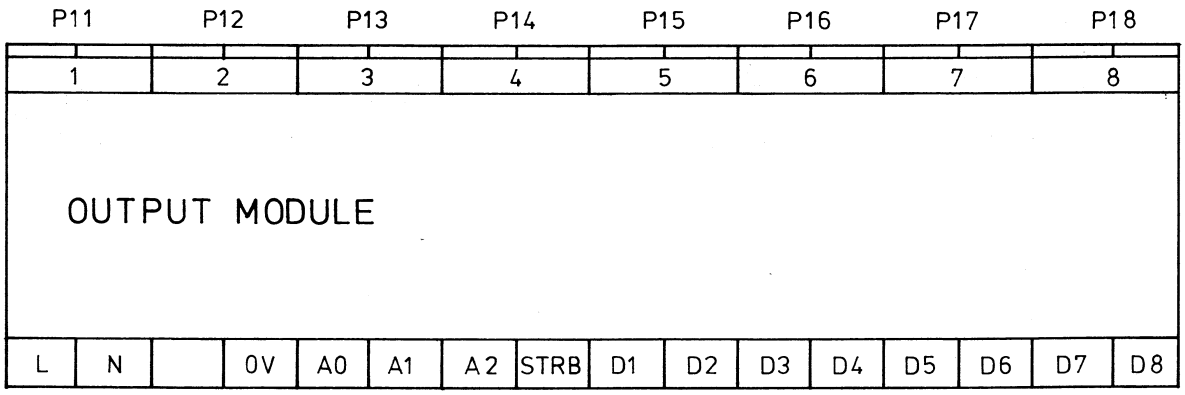


CONTROLLER PIM 100

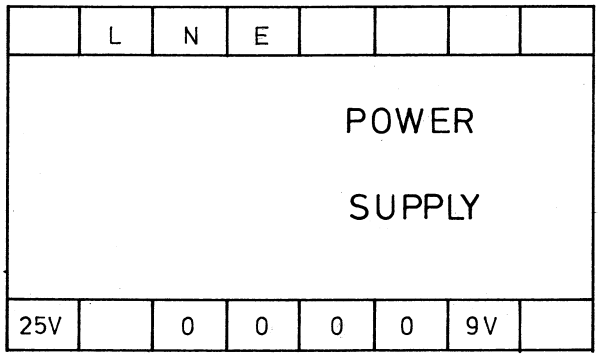


INPUT MODULE PIM160

Figure 7



OUTPUT MODULE PIM150



POWER SUPPLY PIM140

CHAPTER 3

Circuit Description

Remote Control Port (Figure 8)

One of the main reasons for fitting the new controller has been to provide the Conveyor with a control port that can be used to communicate with other computer controlled devices. The design of the port takes account of the need for a serial interface, and also provides for some form of handshaking plus the facility to take direct control of the Conveyor Controller. It has been envisaged that the conveyor would be slaved to a Rediffusion Reflex Robot and so while the port should be useable by most RS-232 type devices it has been configured specifically with the Robot in mind. A remote Control Port has been devised that is composed of two parts, a serial port and three digital circuits. All of these circuits are carried by a 25-way, D-type socket mounted on the upper rear valance panel of the cubicle. Pins 2, 3 and 7 represent the serial port allowing two-way communication with the controller at RS232-C voltage levels, using 9600 baud, eight bits, no parity and two stop bits. Pins 13 and 25 allow the controller to be reset over the interface when they are linked. A local reset switch has been provided inside the cubicle for test purposes, Normally the CHUM controller will reset on power-up. The circuit via pins 13 and 24 has a dual purpose. The CHUM controller will, on power-up, jump to the resident program if this circuit is open, or revert to Terminal mode if closed. Because connections to Port P21 and Reset are available to the interface an external device can reprogram CHUM as well as triggering programmed events. Direct commands can also be directed to the conveyor controller once it has been forced into the terminal mode. The resident program uses this input as Robot Request Service (RRS) to flag a call to the communication routine, Pins 11 and 23 allow CHUM to switch 24 volts (via a current limiting resistor) as a signal to the external device. The resident program can use this to signal that the track is stopped at the station, External At Station (EAS).

Power System (Figure 9) The conveyor was originally manufactured in Switzerland, but intended for use in the U.K. and so the drive motor, vacuum pump and peripheral mains powered devices are rated for the Swiss power system, and supplied via a large transformer, Thus the incoming 415 volt three-phase supply is fed via Circuit Breaker 9 to the primary of the transformer whose secondary produces a 380 volt output. The main circuit is switched by Contactor C1. In order to operate this contactor the key switch must be turned to the right, providing power to transformer T2. The secondary of this transformer provides a 25-volt a.c. output, which is used to provide power for all of the contactors in the cubicle. Once operated C1 maintains the primary circuit to T2 over the normally closed contacts of the key switch. These contacts can be opened by turning the key to the left. A rectifier is also connected to this secondary winding, and supplies 25-volt d.c. via circuit breakers CB5 and CB6 to service the track-mounted devices. A 220-volt single-phase supply is available for each side of the track via circuit breakers CB3 and CB4. The three-phase supply for the Vacuum Pump also originates from the output of contactor C1, being switched by contactor C4. The main drive motor, however, is connected by circuit breaker CB2 and switched by three-phase solid-state relay SSR3 over the normally closed contacts of contactor C3. If C3 is operated SSR3 is isolated and the motor is fed with reversed phase order over C3 normally open contacts. A single-phase supply for the brake circuit is also obtained from CB2 output. Contactors C3 and C4 are fitted with over-current trips whose normally closed contacts are included in the control circuit of the related contactor. Because the CHUM controller requires a 240 volt single-phase supply this has to be obtained up-stream of the main transformer, a connection being made after CB9 over its own fuse-link.

Input Circuits (Figure 10)

The CHUM controller is connected to an input module, which has 16 opto-isolated inputs. An input voltage of between 5 and 30 volts is seen as a logic low, voltages below 5 are seen as logic high, as is an open-circuit. The module has two banks, each of eight inputs. The first bank of 8 inputs, designated as P21 to P28 by the software, is used to read the state of the external request line, RRS, and the various status inputs such as the contactor trip contacts, pressure sensors and proximity detectors. There are five proximity detectors available to indicate that the track is at rest, but it was decided that only one is needed for the present control scheme, so PS5, the one that is pulsed first as the track stops, has been used. In the event that the track drive chain is jammed the upper and lower parts of the Geneva drive wheel move relative to each other and cause a rod to be pushed down through the axis. The position of this rod is detected by proximity switch PS1.

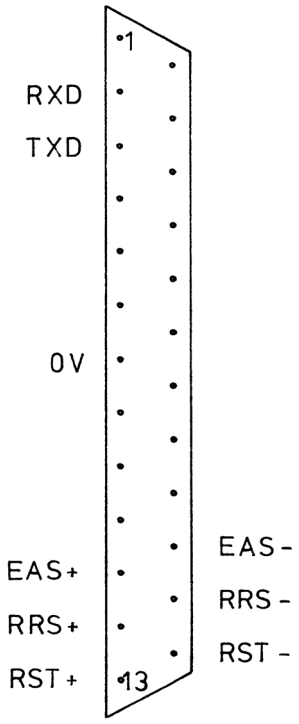
The second bank of eight inputs, P41 to P48, is devoted to reading the states of the control panel switches. The control panel has been slightly rewired to remove any direct acting circuits, e.g. the Brake switch was wired in parallel with the brake lift relay contacts. The effect of the wiring changes has been to produce a "soft" control panel, the connection between inputs and outputs being a function of the controller program. The existing fault warning lamps have been included in the input circuits where practical. Originally there would have been 5 fault circuits each with its own latch. These latches would have been reset by one of the 4 fault reset switches, Reset Motor, Reset Pump, Reset Chain and Reset Air/Vacuum. As there are insufficient inputs to monitor all of the fault reset switches, and in any case it seems unnecessary to have so many, they are now all connected in parallel. The active state of all inputs, e.g. track at proximity switch or panel switch pressed, is seen as logic 0, or OFF by the controller, which has the advantage of consistency if nothing else. The choice of this arrangement was partly dictated by the fact that the proximity switches are polarised and shared a common negative return and the fact that the switches could also be used to operate their associated lamp indicators directly. If the opposite sense had been adopted the proximity detectors would have had to be rewired and reliance would have been placed on the lamp filaments remaining intact to drive current into the input module. The AUTO and BRAKE switches operate their indicators over a second set of contacts, this is simply because the existing wiring was of this form. A 5K6 ohm resistor is used to provide bias current for each of the selected proximity switches, being chosen to give adequate voltage discrimination between their two operating states. The selection of these bias resistors was based on the NAMUR specification figures given for this type of device (Appendix 5.) and the need to exceed 5-volts at the input module, (while it draws a current of 0.8mA) for a logic low, and to be well below 5-volts when the detector sinks its lower limit specified current of 4mA. In practice the input swing is from about 3.0 volt to 10.0 volt, which should give an adequate safety margin. The Internal Action Complete (IAC) input is intended to be driven by controllers mounted along the track; these controllers will implement the drive circuit for this input.

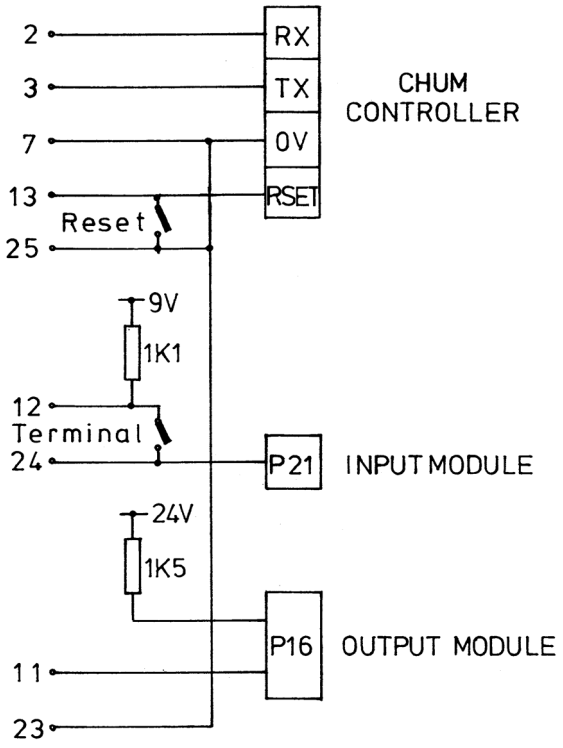
Output Circuits (Figure 11)

The CHUM controller uses an output module to control external circuits. This module has 8 independent relay contacts, designated P11 to P18 by the software. Space has been left within the cubicle for a second module if needed. Two of the outputs are used to signal to peripheral devices when the track is stopped, Internal At Station (IAS) and External At

Station (EAS). IAS is expected to signal to local controllers using TTL circuits so is simply a contact closure to 0 volts. EAS has been designed to work to a Robot fitted with opto-isolated input circuits, so is a current source. Three of the outputs are used to switch on solid state relays, which in turn operate the power contactors C3 and C4, or in the case of SSR3, the drive motor directly. The remaining outputs are used to switch on an air control valve, a control panel fault indicator and the Brake lift relay. The control software ensures that this relay is operated before the track motor is switched on, and released after the motor has been switched off.

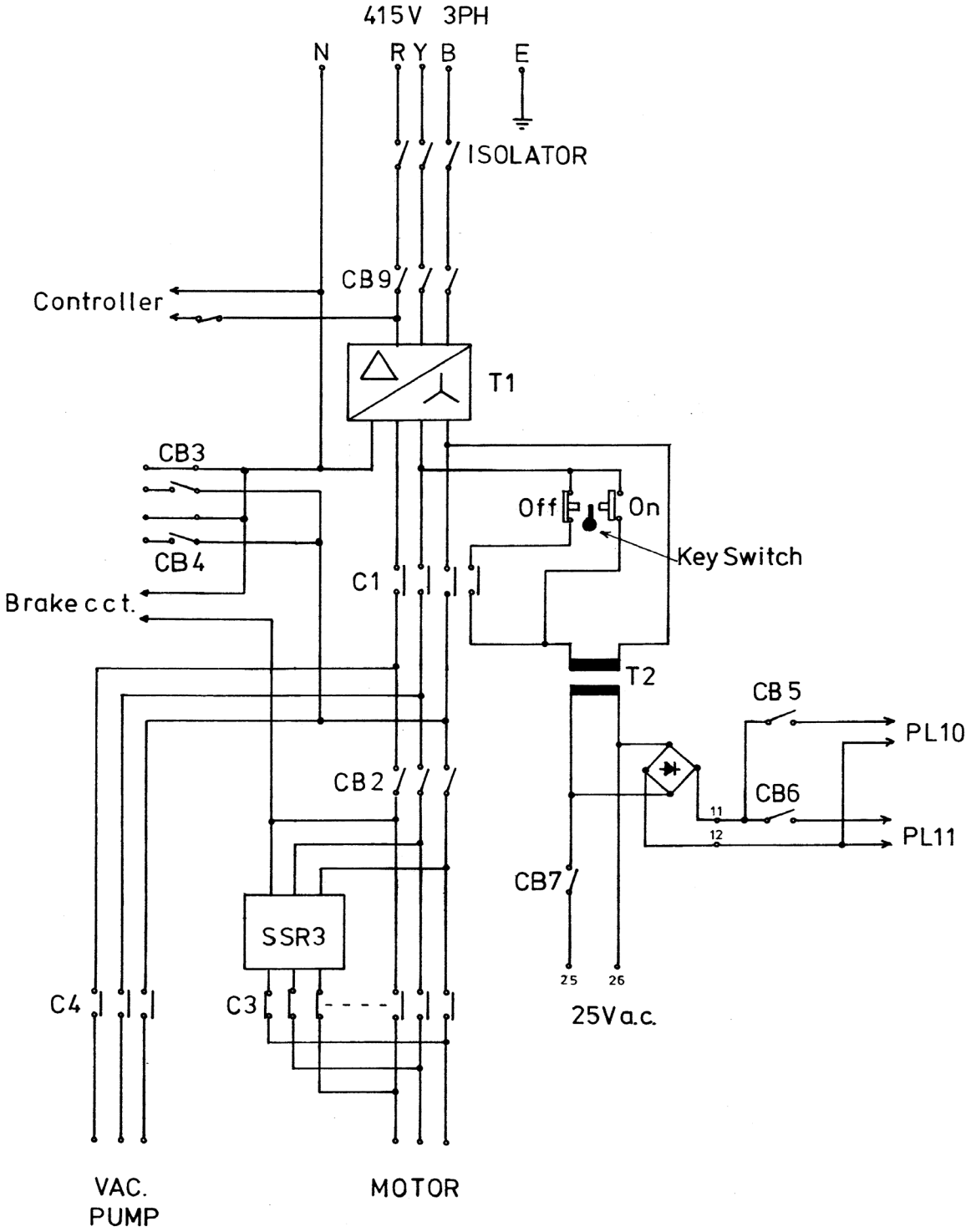
Figure 8





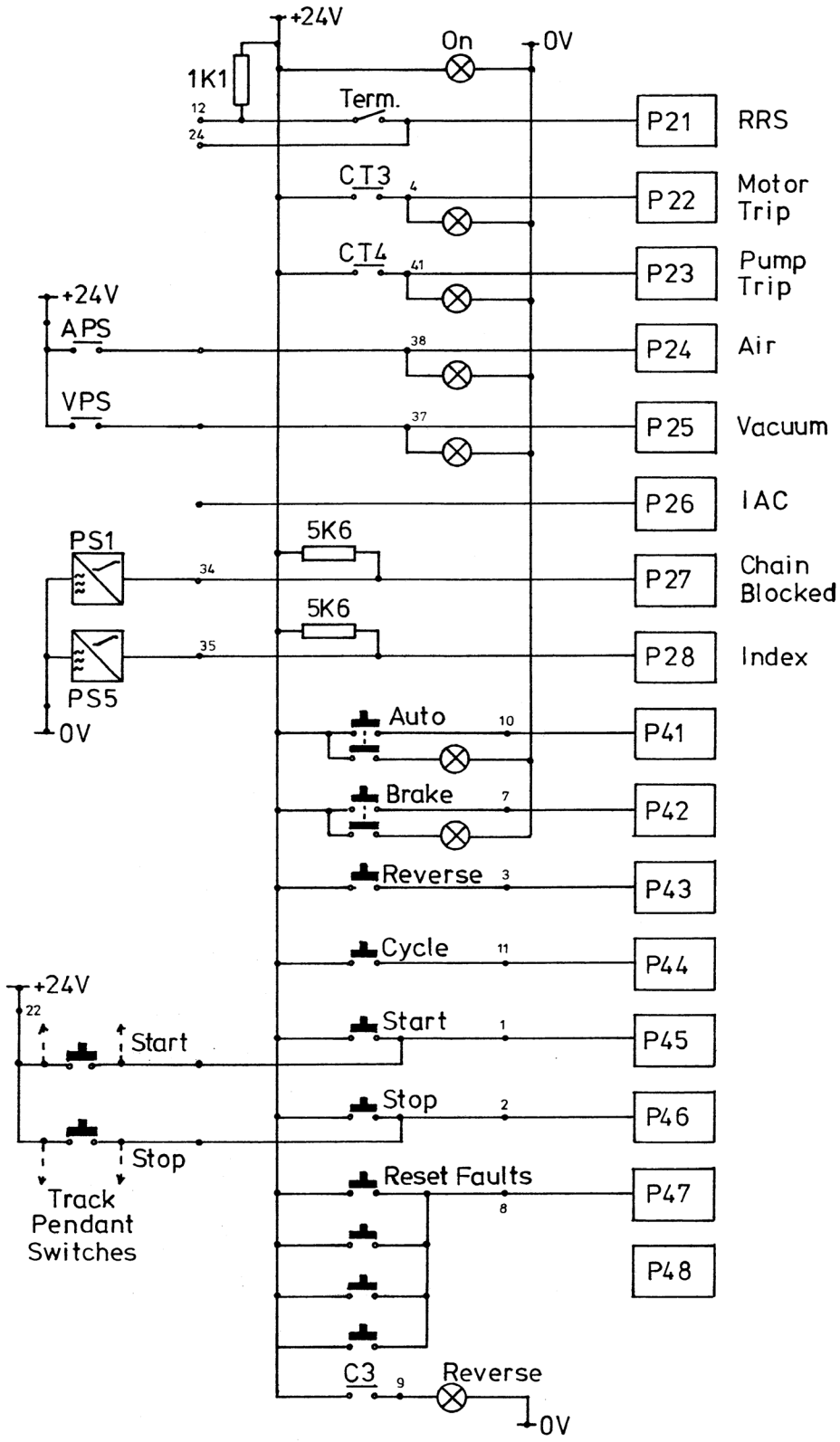
Remote Control Port

Figure 9



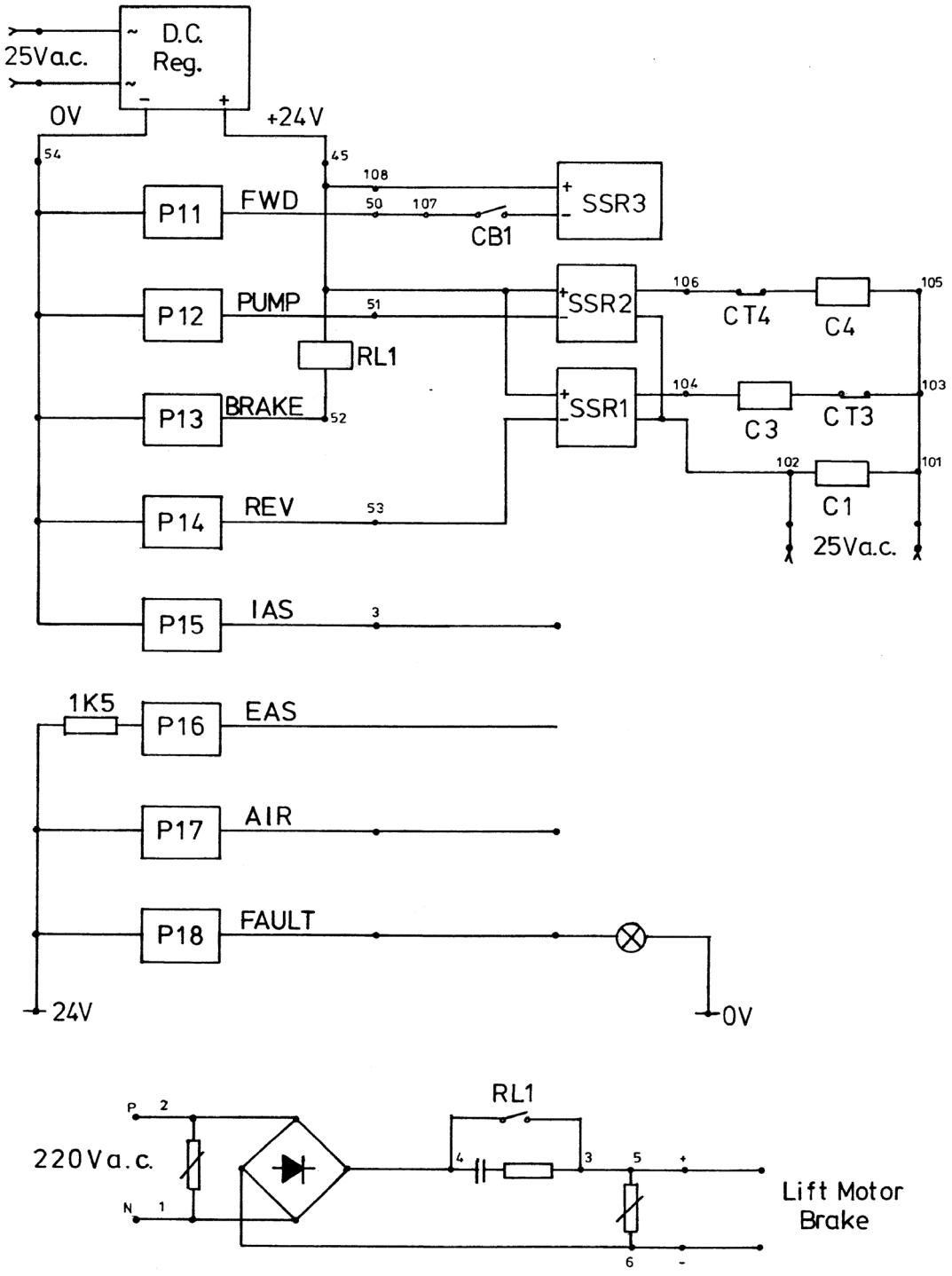
Power Circuit

Figure 10



Input Circuits

Figure 11



Output Circuits

CHAPTER 4

Conveyor Operation

In its present form the conveyor retains the ability to move forward, reverse, brake, control an external air supply and produce a vacuum. The old control panel has been retained, with some simplifications. New facilities are the provision of a remote control interface and a two-wire handshake system for devices local to the track. A control program, described in chapter 5, permits the following modes of operation.

When power is applied to the control computer, (by applying power to the cubicle and turning the key switch to the right) the control program will start automatically, one of its first tasks is to read the state of the AUTO push button. If the latching push button is out (indicator OFF) the Manual Mode is selected. While in this mode the brake can be lifted by pressing in the latching BRAKE push button (indicator ON), and applied by releasing the button with a second push. Pressing in and holding the REVERSE button will cause the conveyor to reverse continuously while the button is held in. A single press of the CYCLE button will cause the conveyor to move forward the number of positions stored in the increment register, at switch-on this will default to a single step. The CYCLE button must be pressed for each cycle, holding it in will result in a single cycle only. The STOP button may be used to suspend track forward motion. START can be pressed to resume the cycle. There is no provision to control the Air and Vacuum systems while in Manual mode.

If the AUTO button has been latched in, (indicator ON), Auto mode will be selected. Six main Auto modes have been implemented in the current control program, as follows:-

1) Wait for Robot, Digital Handshake Off

2) Wait for Robot and Track, Digital Handshake Off

3) Wait for Robot, Digital Handshake On

4) Wait for Track, Digital Handshake On

5) Wait for Both, Digital Handshake On

6) No Wait, Digital Handshake On

The control program currently defaults to mode 1. No Handshake implies that the message 'OK<CR><LF>' is sent over the serial link to the Robot when the track is stopped and ready to be commanded. The conveyor thus 'Waits on Robot' for a new move command, after each command has been sent and executed the controller will send the same response.

Mode 2 is similar except that the track may also be held until local devices have cleared the IAC (Internal Action Complete) line. As before the message string will only be sent once the issued command has been fully executed.

The first two modes have been provided so that the Robot need use the serial port only, to instruct the conveyor. However it will be appreciated that until the conveyor chooses to respond no further commands can be sent, nor can the status of the conveyor be ascertained before receipt of the 'OK' message. Because the conveyor controller, while executing a RECEIVE instruction, will wait indefinitely until it receives a character from the Robot, the control program will only allow this to happen while the conveyor is at standstill. At this time overload condition inputs can be safely ignored. As a further check a Digital Handshake facility is available over the remote interface which allows calls for service to be tested for without running the risk of holding the control program while waiting for serial input. This

handshake line has been designated Robot Request for Service (RRS). All of the remaining modes make use of this signal.

Mode 3, as for mode 1, requires the Robot to send a command before a movement is made, and produces the same response as mode 1 on completion. The difference is that the Robot can assert RRS at any time; the track will stop if it is in motion, and sent the 'OK' message. It will then await the removal of RRS before accepting the command string. This guards against RRS being left active.

Mode 4 prevents the track from moving until the Internal Action Complete, (IAC,) signal is received from devices mounted locally to the track. As before the Robot can force communication by using RRS.

Mode 5 is a combination of Modes 3 and 4, while Mode 6 causes the track to cycle continuously, pausing at each station due to the mechanical action of the drive, but waiting at the end of each cycle for a variable delay period. The modes described are the basic motion modes; within each mode there is a choice of output flagging over External At Station (EAS) and Internal At Station (IAS). Any combination of these signals can be set up. When the track has stopped at the end of each cycle the selected output lines are enabled allowing other events to be triggered. If the EAS signal has been selected the Robot will be able to determine that the track has stopped and it would normally wait on this signal before asserting RRS to avoid stopping the track within the cycle.

When the conveyor controller has sent the 'OK' message it can accept a command string of three characters, which will determine the next action of the controller. These are the set of valid commands:-

AOF Turns the air supply OFF.

AON Turns the air supply ON.

CON CONtinue to cycle.

DXX Delay for XX centi-seconds at station.

FLX Set mode flags. (X=Hex. code, see below)

HOF Turns OFF Handshake mode.

HON Turns ON Handshake mode.

IXX Sets number of steps per cycle.

JMP Jumps to second program.

POS Returns current track position.

REV Reverses conveyor.

STA Returns fault flag status.

VOF Turns the vacuum pump OFF.

VON Turns the vacuum pump ON.

ZER Sets current position as first.

For the FLX command, X is a hex. number where:-

8=Wait on Track

4=Wait on RRS

2=Send IAS

1=Send EAS

For the DXX and IXX commands, XX represents two decimal numbers, in the range 00 to 99 for DXX, and 01 to 48 for IXX.

STA returns two hex. numbers XY where:-

bit X4= Held by IAC

bit X3= Held by STOP

bit X2= Pump O/L

bit X1= Motor O/L

bit Y4= Chain Blocked

bit Y3= Air Pressure Low

bit Y2= Vacuum Pressure Low

bit Y1= Fault Flag

POS returns two characters giving the decimal position of the track in the range 1 to 48. JMP allows command to pass to a second user program. CON causes forward motion to continue while REV produces a short reverse step, as this is intended for clearing conveyor jams continuous motion is not allowed.

As an example, to produce forward motion of 5 steps, with vacuum on, Handshake On, EAS active, Wait on Robot and the current position set as the first, the following command dialogue would be seen:-

Conveyor	Robot
OK<CR><LF>	ZER
,, ,,	VON
,, ,,	I05
,, ,,	FL5
,, ,,	HON
,, ,,	CON

Each character sent by the Robot will be echoed by the Conveyor Controller as it is received. After CON has been sent the track will move forward 5 positions and EAS will be active. The command POS will be met with the response:

06<CR><LF>OK<CR><LF>

- indicating that the all fault flags are clear and the current position is 6.

It should be noted that if a command to either of the Air or Vacuum sub-systems is sent there will be a 10 second delay before further commands can be sent, to allow for the pressure levels to readjust.

Further motion can be produced just by sending CON, all selected options remaining active until changed. Parameters such as Vacuum On and Cycle Increment also remain as set if control reverts to Manual mode. In the above example, once Manual mode has been entered pressing CYCLE will cause a forward move of 5 positions to be executed, with the Vacuum On and EAS sent to the Robot.

The STOP and START buttons on the panel and track pendant boxes can be used to suspend and resume motion even when the track is in AUTO mode, if Handshake is ON the Robot can interrogate the status flags to determine if motion has been suspended.

Should a fault occur, such as a conveyor chain jam or motor contactor trip, the relevant flag will be set along with the fault flag. The 'Chain Blocked' indicator on the panel will illuminate, (this indicator now serves as a common fault lamp), track motion will cease and the response message will be 'ER<CR><LF>'. On receipt of this message the Robot will be aware that a fault has occurred and is not available for normal use, the STA command can then be used to determine the exact fault condition and consequent action. Any of the white Fault Reset buttons can be pressed to clear the fault flags as they are all commoned. There is no provision to reset fault flags over the Remote Control Interface, short of a full controller reset. In any case the exact cause of the fault condition can only be ascertained by manual examination.

In normal use the conveyor can be fully controlled by means of the front panel, pendant switches, and the serial interface, however the conveyor controller can be forced into Terminal mode, either by the digital interface or a reset switch mounted on an internal sub-panel adjacent to the controller. Once in Terminal mode the controller will respond to direct commands (i.e. commands without a program line number) or be reprogrammed. To force terminal mode the internal RESET switch is held down, then the adjacent TERMINAL switch also depressed and held, followed by the release of RESET then TERMINAL. The TERMINAL switch is in parallel with RRS so the same sequence can be carried out over the Remote Control Interface. Once in terminal mode the CHUM controller will not echo any characters sent to it, but will respond to <CR> with a <LF> if it has accepted the command, else it will send ERROR<CR><LF>.

CHAPTER 5

The Conveyor Control Program "CHUM"

The object of the control program is to produce a conveyor which will have several operating modes without the need to re-write the control program. The required modes are:-

1) Respond to commands given from the cubicle panel and pendant switch boxes.

2) Operate under the direction of a Robot.

3) Operate as part of an assembly system, triggering devices mounted along the track.

4) A combination of the other modes.

The controller circuits have been designed so that on power up the control program will run from its start. This means that the fact that the conveyor is controlled by a computer is transparent to the user, no action being required by an operator to initiate the program.

The control program consists of four routines, the first being entered from switch-on via a short initialising stage where the program default values are set up, then forming a loop which monitors the system inputs and sets or clears flags as appropriate. Because the CHUM waits for an input from the serial port when it encounters a RECEIVE command it was important to write the control program so that commands over the serial port are only read when the conveyor is at standstill, otherwise fault inputs such as 'chain blocked' would be in danger of being ignored. To ensure that this does not happen all serial communication is carried out in the COMMS routine which can only be accessed if the conveyor drive motor is off. Two other routines are provided, CYCLE, which controls the forward motion, and REVSE used to allow the conveyor to be backed off in the event of a blockage. The general relationship of these routines is shown by the flow chart in figure 12.

The main routine, after it has set the default conditions, checks the state of the fluid systems and sets fault flags if any of the selected services are not available. The AUTO panel switch is then read to determine if the conveyor is to respond to manual control or not. If the manual mode has been selected then the conveyor can be moved forward by pressing the CYCLE button, reversed by pressing and holding in the REVERSE button, or the brake lifted by pressing the BRAKE button. The STOP and START buttons can be used in both auto and manual modes to suspend or resume motion. If no service flags have been set the program flow continues around the main loop. Should there be a request for serial communication the COMMS routine is accessed. This routine sends the message 'OK' if no fault flags are set or 'ER' otherwise. If the conveyor is operating in the Handshake mode the program will wait on the Robot removing its Request for Service (RRS) before continuing. The COMMS routine then expects to receive three characters, which determine which flags are to be set.

The routine REVSE allows the conveyor to be backed up, continuously while the REVERSE button is held in while in Manual mode, or for a short period if in Auto mode.

The routine CYCLE once entered will cause the conveyor to move forward. The number of positions that it will move can be varied by using the IXX command, thus if I48 has been sent the conveyor will perform a complete rotation before returning control to the main routine. During the CYCLE routine the status inputs are checked to protect the conveyor from damage and prevent mal-operation. Should a fault occur the appropriate fault flag is set and the drive motor switched off. Control then returns to the main routine. By using the FXX command the operating mode can be set. The conveyor can be set to wait on signals from the

Robot and/or devices on the track before a cycle commences. This command also determines whether the Robot or track devices are flagged when the conveyor has reached the target position. When the Handshake mode is on it is possible for the Robot to Request Service while the track is in motion. If this occurs the track will stop, to prevent overload inputs being ignored, and the COMMS routine entered. Once RRS has been removed and the new command received the main routine will determine if control is to be passed back to CYCLE. In the event that it does the track will continue to drive forward to the demand position. In normal use the Robot would, of course, wait on receiving the External At Station signal before attempting communication. With the Handshake mode off the track cannot be interrupted from the external device, but will be sent 'OK' at the completion of each cycle and will await a 'CON' command before resuming motion. There is no means provided for uniquely identifying a particular pallet, but if the ZER command is sent the current position is taken to be the first position to which all moves are referenced. Once the track has moved forward the number of positions determined by the IXX command it will, if no other 'wait on' modes are active, wait for a time determined by the DXX command before terminating the cycle.

The full control program "CHUM" is listed in Appendix 7, but in order to fully understand it the allocation of ports to signals and variables used must be known, these are given in Appendix 6. The CHUM controller has a very restricted selection of variable names, simply the letters A through F, plus these letters followed by a single digit. Certain logic constructions are not valid in CHUM BASIC so the variables prefixed by 'A' have been used as temporary stores within a set of program lines within a routine. Inter-routine communication uses input flags 'E' and output flags 'F'. Delays are set up using the TIMER command and 'D' variables.

Main Routine (Line 2 to Line 112)

Line Number	Action
2	Auto-start from here.
2-16	Set up default timers and flags.
18	Re-entrant point for MAIN.
18-20	Set up fluids to Input Flags
22-34	If last command for fluids wait 10s.
36	Sub-routine call to check fault inputs.
38	Set Fault Indicator on Error.
40-54	Clear Fault Flags if RESET FAULT pressed.
56	Set Auto Flag to AUTO button.
58-60	Goto COMMS if Auto, H/S On and RRS.
62-64	Goto COMMS if Auto, H/S Off and No Motion.
66-70	Update STOP flag, goto MAIN if set.
72-74	If Manual and BRAKE lift brake.
76-78	If Manual and REVERSE goto REVSE.
80	Goto MAIN if Error.
82	By-pass Manual routine if Auto

84-100	Go to CYCLE if motion had been suspended else check CYCLE button and flag to enforce single-shot control, goto CYCLE if valid else MAIN.
102-112	Auto routine, goto REVSE if flag else check for motion conditions met, if so goto CYCLE else MAIN.

CYCLE (lines 112-200)

114	If new cycle reload increment counter
. 116-122	Lift brake, delay, start motor.
124-126	Determine current state of index input.
128	Check state of fault inputs, set flags.
130	If fault. goto forced exit.
132	Set stop flag if STOP.
134-138	If Auto, H/S On and RRS goto forced exit
. 140-144	Check index detector and flag for valid operation, goto 128 if increment not complete.
146-152	Increment position counter, decrement step counter, goto 126 if cycle not complete.
154-160	Stop motor, apply brake, delay to ensure motion stopped.
162-164	Send At Station signals to enabled outputs
. 166-172	Wait at station for variable delay, if Auto and H/S On check for RRS to terminate wait.
174-180	Remove At Station signals, clear cycle flag, goto MAIN.
182-186	Forced exit, stop drive goto MAIN.
188-200	Sub-routine read faults and set flags.

COMMS (lines 202-372)

202	Goto error message if fault flag set.
204-210	Send 'OK',goto 218.
212-216	Send 'ER'
218-220	If H/S On wait for removal of RRS.
222-226	Get three characters from port.
228-250	Test for valid first character and goto to command routine, else MAIN.
254-260	AIR routine, set flag if AON.
262-264	CONtinue routine, set flag.

266-270	Delay routine, load station delay.
272-284	FLag in routine, set mode flags.
286-290	Handshake routine, set flag.
292-300	Increment routine, load increment.
302	Jump routine, goto second program.
304-316	Position routine, send position.
318-320	Reverse routine, set flag.
322-334	STAtus routine, send status.
336-342	Sub-routine, form ASCII character.
344-348	Vacuum routine, set flag.
350-354	ZERo routine, resets cycle counter and position counter.
356-360	Sub-routine, convert 2 ASCII characters to binary byte.
362-366	Sub-routine to send <CR><LF>.
368-372	Sub-routine to delay sending characters.

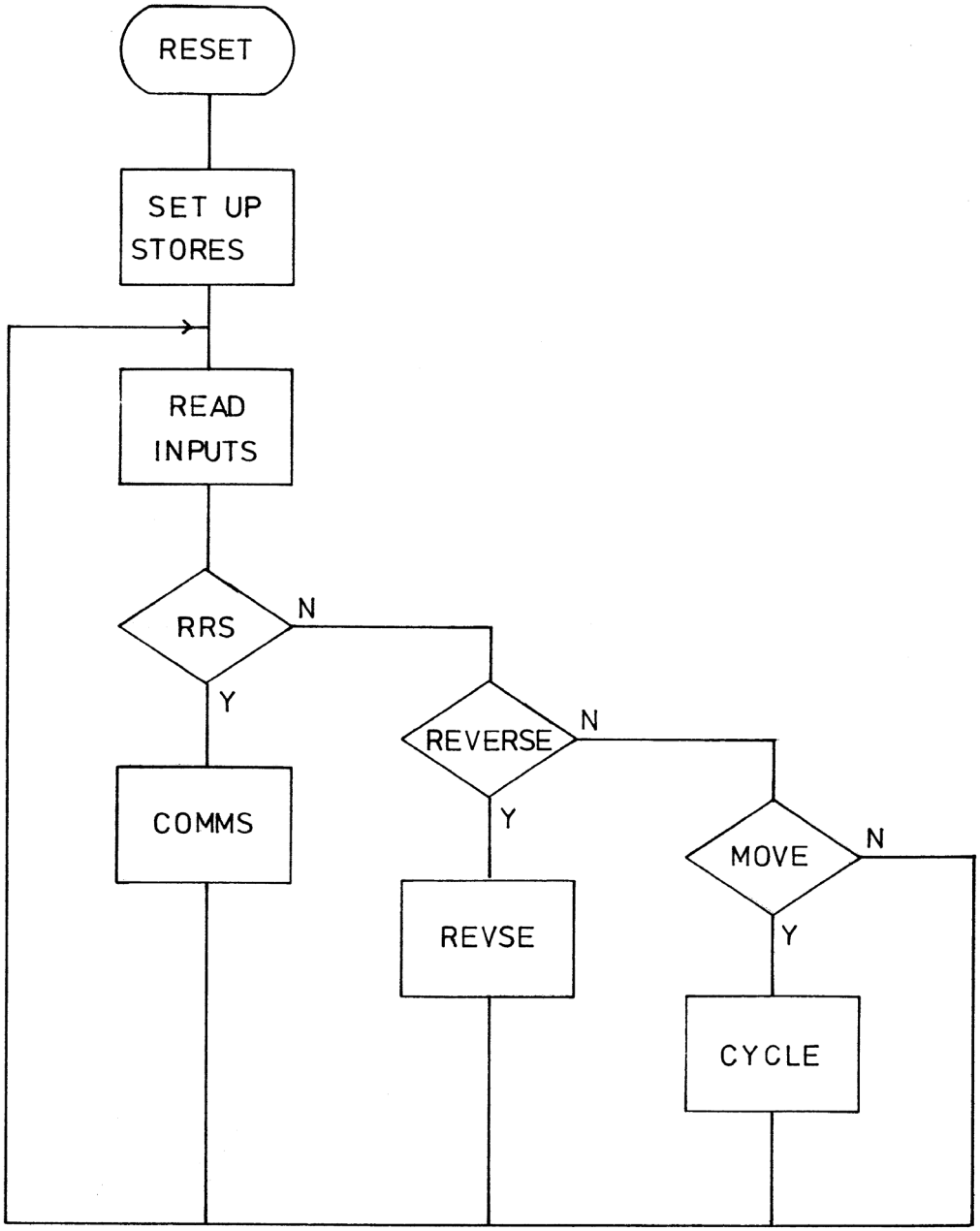
REVSE (lines 374-396)

374	Clear Reverse Flag.
376-378	Brake off, motor on (reverse).
380	If Auto goto 386.
382	Loop if REVERSE button in.
384	Goto exit at 390.
386-388	Loop while delay on.
390-394	Motor off, brake on, goto MAIN.

Program 2

396	Goto MAIN.

Figure 12



CONVEYOR PROGRAM

CHAPTER 6

Cell Control File System - CELFIL

It had been decided to use a BBC B computer to simulate a cell controller, as it was readily available. However the ideal Cell Control Computer would have separate ports for each of the machines under its control, whereas the BBC has only one serial port. The Rediffusion Reflex Robot has four spare serial ports and so could in theory act as a cell controller, but in practice would be unable to do so as the cell control software would restrict the maximum size of Robot control program that could be handled and there would be difficulties with integrating the two roles. As the communication needs of the conveyor are small an alternative architecture has been employed, where the conveyor is interfaced to the Robot and the Robot to the BBC, acting as Cell Controller Simulator. This configuration could also be used in a practical situation as well because there would be little need for upstream computers to communicate with the conveyor, most transactions would be between the Robot and the conveyor, the conveyor would then act as a slave to the Robot.

Because the BBC computer was to act as the Cell Controller there was a need for some sort of Cell Control Program. Time did not allow the development of such a full-blown program, this would be a major task in itself requiring the development of interfaces to higher level computers, other machines within the cell and the connecting control software. The selected conveyor controller CHUM 1 is rather limited in its purchased form, there is no way of programming it without some sort of terminal. As a result of this need and mindful of the ultimate needs of a full Cell Controller the Cell Control File System program CELFIL was written, the full program listing being presented in Appendix 8.

CELFIL presents the user with a menu allowing the user to select:-

1) The File Editor.

2) Transmit File.

3) Terminal.

4) Exit.

Option 3 was needed in order to program CHUM as it has no program entry facilities as purchased, consequently it was the first part of CELFIL to be written. Using this option it is possible to program CHUM, however there is no means of editing any program stored in CHUM, apart from deleting, overwriting or inserting lines. A further difficulty is that the maximum line number is restricted to 999, so that using conventional practice of incrementing line numbers by 10 only one hundred lines would be available. Consequently if an attempt was made to write the control program in situ it would be full of 'patches' as the available spaces were filled and control had to branch to free spaces. Once a working control program has been produced CHUM has the facility to copy it to an EPROM for more permanent storage, should this EPROM fail there would be no back-up unless duplicate EPROMS were made. As a result of these observations it was felt prudent to produce an editing facility so that the control program could be produced off-line, then down-loaded to the target CHUM. This type of facility would, in any case, be required for the Computer Integrated Factory as it would be unacceptable to develop programs for each piece of plant on the machinery itself. Thus option 1 was added along with option 2 so that the program could be transferred to CHUM. Option 4, of course allows an orderly exit from CELFIL to BBC BASIC.

Because the file editor was written on the BBC B computer it was decided to emulate the program editing facility of the host system, one of the curses of the computer age being that every system has its own protocol which has to be learnt before the user can be fluent,

which makes it difficult to transfer between similar systems. To avoid confusion the editor generates the prompt '>>' whenever it expects input, rather than the '>' of the BASIC command interpreter. The user can type against the prompt a program line beginning with a number, or any of the commands AUTO, LIST, RENUMBER, LOAD, SAVE, NEW, OLD or MENU.

Typing AUTO causes line numbers incrementing by 10 to be generated against the prompt, while typing LIST will list the whole program or can be set to start or end at given line numbers as for the BBC BASIC command. The command RENUMBER will renumber the program with a default increment of 10, or by a number entered as a suffix to the command. It will handle GOTO and GOSUB referenced numbers correctly as long as they are the last part of a statement (the target computer CHUM does not support multiple statement lines so all program jumps must be of this form). Entering NEW clears the editor, (the editor may be left, say to use Terminal, then re-entered), while OLD permits recovery of the edit file if NEW has been used and no new lines entered. LOAD and SAVE allow files to be transferred from and to the disc drive. The special command MENU returns control to the main menu. Most of these commands are available via the function keys or can be typed to the prompt. The in-built screen editor or COPY key facility can also be used to edit lines or parts of lines, and if Control B is entered all output appearing in the edit window can be sent to the printer. Thus if LIST is typed a printed listing of the edit file can be obtained.

The Transmit option was the last to be added and at the moment is purely tailored to the needs of CHUM. It will send a file from disc to CHUM, or if the Edit file is open, offer that as an option. Further development would allow the use of different protocols for different target machines, possibly automatically selected by target address so that the operation is transparent. At present CHUM has to be manually prepared to receive the file, a future task might be to automatically seize CHUM, place it in Terminal mode, send NEW to clear the memory then down-load the file, merely by specifying the Conveyor as the file destination.

The Terminal option allows communication over the serial port at 9600 baud. When CHUM is being used in its Terminal mode it does not echo back characters sent back to it so CELFIL defaults to display all outgoing characters. However when CHUM is running a program and executes a RECEIVE command it will echo back the received character, CELFIL therefore allows the use of a function key to toggle off the local echo. Another function key allows escape back to the main menu. Because the command LIST was often sent to CHUM during program commissioning one of the function keys was programmed to give this command string.

It will be realised that CELFIL is a valuable tool for developing programs for the CHUM controller as not only does it perform the basic but essential task of emulating a 'dumb terminal', but also allows programs to written off-line, expanded for insertion of lines, compacted for use by CHUM and saved on disc as a back-up to the EPROM. The use of the AUTO command alone probably reduces program entry time by some 20%, and in the event of an error being made while typing the line the delete, copy and cursor keys allow corrections to be made readily. If direct programming of CHUM is attempted not only must a valid program line number be typed but should any mistake be made the whole line must be re-typed again. The use of the editor permits dummy lines to be entered such as '100IF A=1 THEN GOTO somewhere', when the target line has been identified the editor can be used to copy over the valid part of the dummy line. This is impossible in direct programming as each line is checked for syntax errors on receipt of the terminating carriage return

CONCLUSION

The feasibility of using a microprocessor based Industrial Controller to upgrade the control system of an Indexing Conveyor so that it ceases to be part of a dedicated assembly machine and becomes a flexible, multi-mode conveying system able to be integrated into a machine cell has been demonstrated.

It is also evident though, that while the hardware interfacing problems are minimal with this type of controller, due to the lack of in-built software aids, programming them is not as straightforward as is often suggested. However if a standard controller was to be used throughout an automation scheme the development of software tools on a host computer would be justified and highly desirable, and would in turn lead to greatly improved ease of application as befits such flexible devices.

BIBLIOGRAPHY

1. Friend, F.E., Fike, J.L., Baker, H.C., and Bellamy, J.C. "Understanding Data Communications". Texas Instruments, Dallas, U.S.A., 1984.
2. Bennett, S. and Linkens, D.A. "Computer Control of Industrial Processes". Institution of Electrical Engineers, London, 1984.
3. Maxwell, M. "Distributed versus Centralized Control"(Distributed Computer Control Systems 1983 editor M.G. Rodd. International Federation of Automatic Control, Pergamon Press, Oxford, 1984)
4. Harrison, T.J. "I.E.E.E. Project 802: Local Metropolitan Network Standards". *ibid*.
5. Brown, I. and Bosch, E.F. "The Synergism of Microcomputers and PLC's in a Network". *ibid*.
6. "Worldwide Standardisation Activities of Open Systems Interconnection and Local Networks". Institution of Electrical Engineers, London. 1986.
7. Millar, W.G. "What is MAP?". National Engineering Laboratory, East Kilbride. 1986.
8. Moore, G. "Manufacturing Automation Protocol". Electronics and Power, 32, p269-272. 1986.
9. Holmes, L. "Computers in data communication". Electronics and Power, 29, p390-393. 1983.
10. Weston, R.H., Hanlon, P.D., Salihi, A. "The use of a commercial local area network in distributed machine and process control systems*".ibid*.
11. Moore, G. "Local area networks. Running rings round computers". Electronics and Power, 30, p775-779. 1984.
12. Hardie, A. "S5/8 -the technical details". Electronics and Wireless World, 92, p51-53. 1986.
13. . Moore, G. "Programmable controllers. Driving the wheels of industry". Electronics and Power, 31, p833-836. 1985.
14. Coll, J. and Allen, D. "The BBC Microcomputer User Guide". British Broadcasting Corporation, London, 1983.
15. "Reflex Robot operators and users guide". Rediffusion Robot Systems, Crawley, Sussex, 1903.
16. Omron Sysmac S6 Programming Manual" IMO Precision Controls Ltd, London, 1985.
17. Chum 1 Programming Manual" Warwick Design Ltd, Georges Rd., Leamington Spa, Warwickshire, 1986.
18. "Opto 22 Power I/O Systems Guide" Systems Devices Ltd, Letchworth, Hertfordshire, 1985.
19. "Meto-fer A.G. Automation Products Catalogue" Meto-fer A.G., Grenchen, Switzerland, 1984.

Appendix 1 RS-232C Pin Designations

Pin No.	Signal Description
1	Protective Ground
2	Transmitted Data
3	Received Data
4	Request to Send
5	Clear to Send
6	Data Set Ready
7	Signal Ground
8	Received Line Signal Detector
9	Reserved
10	Reserved
11	Spare
12	Secondary Received Line Signal Detector
13	Secondary Clear to Send
14	Secondary Transmitted Data
15	Transmitter Signal Element Timing (DCE)
16	Secondary Received Data
17	Receiver Signal Element Timing
18	Spare
19	Secondary Request to Send
20	Data Terminal Ready
21	Signal Quality Detector
22	Ring Indicator
23	Data Signal Rate Selector (DTE)
24	Transmitter Signal Element Timing (DTE)
25	Spare

Appendix 2

S5/8 Pin Designations

Pin No.	Signal Description
1	Data In (DINP)
2	Return
3	Data Out (DOUT)
4	Handshake In (HINP)
5	Handshake Out (HOUT)
6	Secondary In (SINP)
7	Secondary Out (SOUT)
8	+5 Volts
Screen	Screen

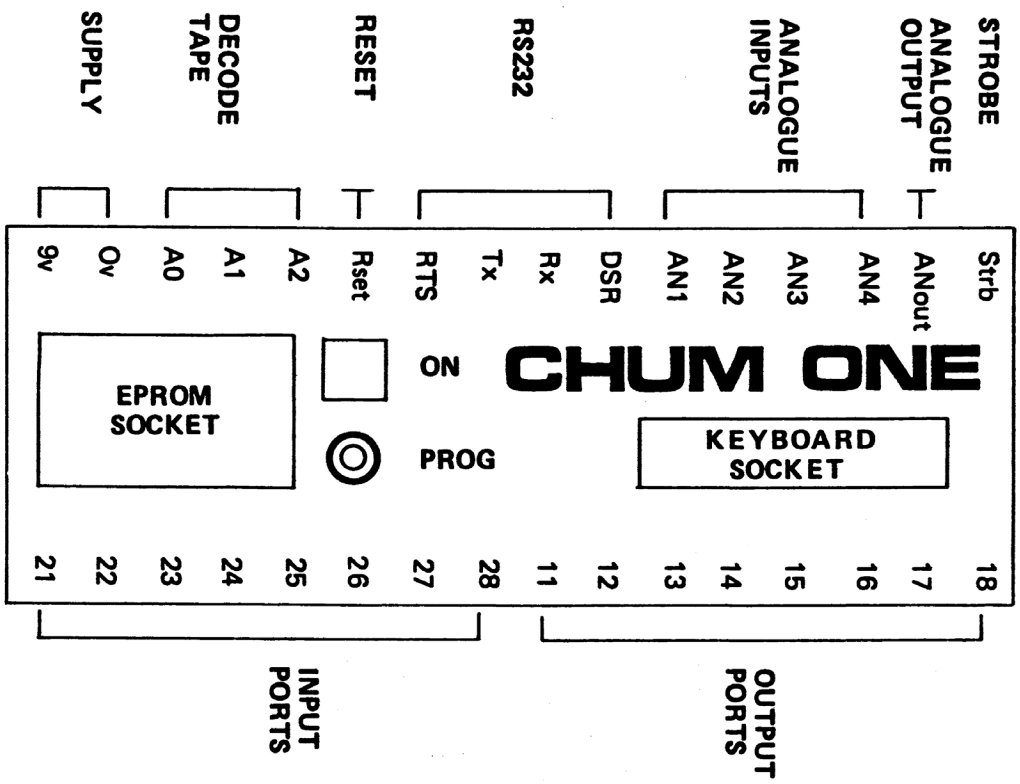
Appendix 3 Controllers

Industrial Programmable Controllers

Make / Model	Memory Size	Input/Outputs	Programming Methods
Control Universal CUBE-65	16K	16 inputs/outputs	BASIC/FORTH
Eberle PLS 509	512	16 inputs 16 outputs 4 input/output	ladder
Electromatic	768 steps	16 inputs 8 outputs	ladder
Gould Micro 84	512	112 inputs/outputs	ladder
IMO Omron Sysmac 86	512 words	12 inputs 8 outputs	ladder
JB Microsystems JB 3200	16K	16 inputs 16 outputs	BASIC
Klockner-Moeller PS 31-1	8K	16 inputs 12 outputs	ladder
Mitsubishi F-12R	320 steps	12 inputs 12 outputs	ladder
MTE PC 100	320 steps	18 inputs 12 outputs	ladder/bool
Opto 22 LC2	32K	16 inputs/outputs	BASIC
SattControl SattCon 05-20	2K	16 inputs 12 outputs	ladder/bool
Siemens S5-101U		10 inputs 6 outputs	ladder/other
Texas Instruments TI 510	256 steps	24 inputs 16 outputs	ladder
Toshiba EX20	512 steps	12 inputs 8 outputs	ladder
Warwick Design CHUM 1	4K	16 inputs 8 outputs	BASIC

Appendix 4

CHUM 1



SOFTWARE SPECIFICATION

LANGUAGE: BASIC
Z80 Machine Code Subroutines

INSTRUCTIONS

IF . . . THEN	GOTO	BANK
FOR . . . NEXT	LIST	LOAD
REPEAT . . . UNTIL	NEW	ASSEMBLER
GOSUB . . . RETURN	RUN	DISPLAY
INKEY	SAVE	TRANSMIT
BLOW (EPROM)	PORT	RECEIVE
CLOCK (R.T.C.)	SINGLE STEP	
TIME DELAY	SET BREAK POINT	

OPERATORS: AND OR ADD, SUBTRACT, MULTIPLY, DIVIDE
EQUALS. ON, OFF, GREATER, LESS, NOT.

HARDWARE SPECIFICATIONS

POWER SUPPLY: VOLTAGE 9 Volts ±1.5 Volts
Current less than 400 mA
Connection +9V OV

Appendix 4

CHUM 1

DIGITAL INPUTS: Number 8 (Expandable to 64)
Voltage 5 Volts
Connections 21, 22, 23, 24, 25, 26, 27, 28

May be connected directly to switches commoned to OV. In addition they can be expanded to 64 by using Input Modules.

ANALOGUE INPUTS: Number 4
Range 0-10 Volts (0-255 reading)
Input Impedance 20 Kohms
Connections AN1, AN2, AN3, AN4

DIGITAL OUTPUTS: Number 8 (Expandable to 64)
Voltage 5 Volts (Emitter Followed)
Current 15 mA
Connections 11, 12, 13, 14, 15, 16, 17, 18

The Outputs can be expanded to 64 Normally Open Relay Contacts

ANALOGUE OUTPUTS: Number 1
Voltage 0—10 Volts (0—255 digital)
Current 10mA
Connection ANout

The output is set by assigning a number to Port O e.g. Setting Port O to 128 will produce 5V on ANout.

DECODE/TAPE READ & WRITE:
Connections AO/TR A1/TW A2.

These connections are used when either Input and Output Modules are used or when programs are saved or loaded to cassette tape.

STROBE: Connections Strb

The output is used to strobe the Input and Output Modules.

RESET: Connection Reset

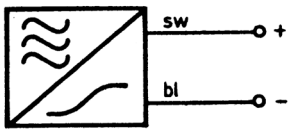
With a switch closed to OVolts the program in the C.P.U. will not run, with the switch open the program will start to run at the first instruction.

RS232: Connections RST Tx Rx DSR

Tx	Transmit Serial Data
Rx	Receive Serial Data
RTS	Request To Send
DSR	Data Set Ready
Baud Rate	9600
Stop Bits	2
Parity	Nil

EPROM: An EPROM socket is mounted on the top of the Unit for storing and running programs permanently. A 2532 EPROM can be programed by connecting 25 Volts to the Prog Jack and pressing the Blow Key on the keyboard.

MEMORY: ROM 4K
RAM 4K
EPROM 4K

Initiators	**M8×1** NAMUR
micro-hetron AG CH-2540 Grenchen / Switzerland	
Connections: sw = black | gr = green bl = blue | ge = yellow br = brown | we = white	
Technical data	
Rated switching distance as EN 50010	1,5 mm
Built in kind	Protected
Switching hystereses	
Reproduceability	< 0,01 mm
Operating voltage	5 V... 24 V DC (with pull up)
Ripple as DIN 41755	10%
Load current	
Open circuit current	
Closed switch current	≤ 1 mA
Open switch current	≥ 4 mA
Overvoltage spike protection	
Polarity protection	
Short circuit protection	
Switching function	Analog[1)]
Output stage	NAMUR as DIN 19234
Status indicator with LED	
Switching rate	2 kHz
Ambient temperature	−20°C ... +70°C
Package material	Metal
Cable cross-section / diameter	2×0,14□ / Ø 3,7 mm
Overall isolation DIN 40050	spilled with cable spilled with socket
Description:	**Order No.**
Initiator switch with 2 m cable moulded in	IR 8 N−K
Initiator switch with 2 m cable connected by screw plug	IR 8 N−SK
Initiator switch and screw plug connected to cable, state length	IR 8 N−SK ...
Initiator switch without cable or plug	IR 8 N−OS
2 m cable and connected plug	ST−J
Cable connected to plug state length required	ST−J ...
Remarks: [1)] As NAMUR by 1,8 mA; U_B = 8,2 V; R_L = 1 k Ohm; t = 20 °C	

Proximity Switch Data

Appendix 6

Variables used by program "CHUM"

A	C Index store	E Out Flags
0 Work.flag	0 Index count	0
1 Work.flag	1 In char. 1	1 Fault Flag
2 Work.flag	2 In char. 2	2 Vac. Fail
3	3 In char. 3	3 Air Fail
4	4	4 Chain Block
5	5	5 Motor O/L
6	6	6 Pump O/L
7	7	7 Held by STOP
8	8	8 Held by IAC
9	9	9 Man. Key

B	D Delay at Station	F Input Flags
0	0 Delay 100ms	0 Auto Mode
1 Out Char.	1 Delay 1s	1 Cycle
2	2 Delay 10s	2 Vac On
3	3	3 Air On
4	4	4 REV
5	5	5 Advise Robot
6	6	6 Advise Periph.
7	7	7 Wait Robot
8	8	8 Wait Periph.
9	9	9 Hand shake On

Output and Input Port Allocation

P11 Motor F.	P21 Robot R. Service	P41 Auto
12 Pump	22 Motor Trip	42 Brake
13 Brake	23 Pump Trip	43 Reverse
14 Reverse	24 Air Mon.	44 Cycle
15 Int. At Stat.	25 Vac. Mon.	45 Start
16 Ext. At Stat.	26 Int. Act. Comp.	46 Stop
17 Air	27 Chain Block	47 Fault Reset
18 Fault Indicator	28 Index	48

Appendix 7

"CHUM" Program

```
2 C=1
4 D=10
6 D0=1
8 D1=20
10 D2=100
12 F5=1
14 F6=1
16 F7=1
18 P12=F2
20 P17=F3
22 A0=OFF
24 IF C1=65 THEN A0=ON
26 IF C1=90 THEN A0=ON
28 C1=0
30 IF A0=OFF THEN GOTO 36
32 TIMER(D2)=ON
34 IF TIMER(D2)=ON THEN GOTO 34
36 GOSUB 188
38 P18=E1
40 IF P47=ON THEN GOTO 56
42 E1=0
44 E2=0
46 E3=0
48 E4=0
50 E5=0
52 E6=0
54 E8=0
56 F0=NOT P41
58 A0=F0 AND NOT P21 AND F9
60 IF A0=ON THEN GOTO 202
62 A0=F0 AND NOT F9 AND NOT F1 AND NOT F4
64 IF A0=ON THEN GOTO 202
66 IF P45=OFF THEN E7=OFF
68 IF P46=OFF THEN E7=ON
70 IF E7=ON THEN GOTO 18
72 A0=(NO P42 AND NOT F0)
74 P13=A0
76 A0=(NOT P43AND NOT F0)
78 IF A0=ON THEN GOTO 374
80 IF E1=ON THEN GOTO 18
82 IF F0=ON THEN GOTO 102
84 A0=F1 AND E9
86 IF A0=ON THEN GOTO 100
88 A0=E9 AND NOT P44
90 IF A0=ON THEN GOTO 18
92 IF P44=ON THEN E9=OFF
94 IF P44=ON THEN GOTO 18
96 E9=ON
98 F1=ON
100 GOTO 114
102 IF F4=ON THEN GOTO 374
104 E8=0
106 E8=F8 AND P26
108 A0=(F1 OR NOT F7) AND (NOT P26 OR NOT F8)
```

```
110 IF A0=ON THEN GOTO 114
112 GOTO 18
114 IF C0=0 THEN CO=C
116 P13=0N
118 TIMER(D0)=ON
120 IF TIMER(D0)=ON THEN GOTO 120
122 P11=ON
124 A1=OFF
126 IF P28=OFF THEN A1=ON
128 GOSUB 188
130 IF E1=ON THEN GOTO 182
132 IF P46=OFF THEN E7=ON
134 A0=0
136 A0=E7 OR (F0 AND NOT P21 AND F9)
138 IF A0=ON THEN GOTO 182
140 IF P28=ON THEN A1=OFF
142 A2=A1 OR P28
144 IF A2=ON THEN GOTO 128
146 C4=C4+1
148 IF C4>47 THEN C4=0
150 C0=C0-1
152 IF C0>0 THEN GOTO 126
154 P11=OFF
156 P13=OFF
158 TIMER(D0)=ON
160 IF TIMER(D0)=ON THEN GOTO 160
162 IF F5=ON THEN P16=ON
164 IF F6=ON THEN P15=ON
166 TIMER(D)=ON
168 A0=F0 AND F9 AND NOT P21
170 IF A0=ON THEN GOTO 174
172 IF TIMER(D)=ON THEN GOTO 168
174 P15=OFF
176 P16=OFF
178 F1=0
180 GOTO 18
182 P11=OFF
184 P13=OFF
186 GOTO 18
188 E2=F2 AND NOT P25
190 E3=F3 AND NOT P24
192 IF P27=OFF THEN E4=ON
194 IF P22=OFF THEN E5=ON
196 IF P23=OFF THEN E6=ON
198 E1=E2 OR E3 OR E4 OR E5 OR E6
200 RETURN
202 IF E1=ON THEN GOTO 212
204 TRANSMIT 79
206 TRANSMIT 75
208 GOSUB 362
210 GOTO 218
212 TRANSMIT 69
214 TRANSMIT 82
216 GOSUB 362
218 IF F9=OFF THEN GOTO 222
220 IF P21=OFF THEN GOTO 220
222 RECEIVE C1
224 RECEIVE C2
226 RECEIVE C3
228 IF C1=65 THEN GOTO 254
230 IF C1=67 THEN GOTO 262
```

```
232 IF C1=68 THEN GOTO 266
234 IF C1=70 THEN GOTO 272
236 IF C1=72 THEN GOTO 286
238 IF C1=73 THEN GOTO 292
240 IF C1=74 THEN GOTO 302
242 IF C1=80 THEN GOTO 304
244 IF C1=82 THEN GOTO 318
246 IF C1=83 THEN GOTO 322
248 IF C1=86 THEN GOTO 344
250 IF C1=90 THEN GOTO 350
252 GOTO 18
254 F3=0
256 IF C3=78 THEN F3=1
258 IF F3=1 THEN F7=1
260 GOTO 18
262 F1=1
264 GOTO 18
268 D=C2
270 GOTO 18
272 C3=C3-48
274 IF C3>9 THEN C3=C3-7
276 F5=C3 AND 1
278 F6=C3 AND 2
280 F7=C3 AND 4
282 F8=C3 AND 8
284 GOTO 18
286 F9=0
288 IF C3=78 THEN F9=1
290 GOTO 18
292 GOSUB 356
294 IF C2>48 THEN C2=48
296 IF C2<1 THEN C2=1
298 C=C2
300 GOTO 18
302 GOTO 396
304 GOSUB 368
306 B1=C4/10
308 GOSUB 336
310 B1=C4-(C4/10*10)
312 GOSUB 336
314 GOSUB 362
316 GOTO 18
318 F4=1
320 GOTO 18
322 GOSUB 368
324 B1=(E8*8)+(E7*4)+(E6*2)+E5
326 GOSUB 336
328 B1=(E4*8)+(E3*4)+(E2*2)+E1
330 GOSUB 336
332 GOSUB 362
334 GOTO 18
336 B1=B1+48
338 IF B1>57 THEN B1=B1+7
340 TRANSMIT B1
342 RETURN
344 F2=0
346 IF C3=78 THEN F2=1
348 GOTO 18
350 C0=0
352 C4=0
354 GOTO 18
```

```
356 C3=C3-48
358 C2=C2-48*10+C3
360 RETURN
362 TRANSMIT 13
364 TRANSMIT 10
366 RETURN
368 TIMER(D0)=ON
370 IF TIMER(D0)=ON THEN GOTO 370
372 RETURN
374 F4=OFF
376 P13=ON
378 P14=ON
380 IF F0=ON THEN GOTO 386
382 IF P43=OFF THEN GOTO 382
384 GOTO 390
386 TIMER(D1)=ON
388 IF TIMER(D1)=ON THEN GOTO 388
390 P14=OFF
392 P13=OFF
394 GOTO 18
396 GOTO 18
```

Appendix 8

CELFIL

```
10 MODE 7
20 DIM SN$(400),SL$(400)
30 AF=0:BL=10:LI=10:PT=0:MX=0:RPT=0:RMX=0
40 FLN$=""
50 PRINT TAB(5);"Cell Controller File System"
60 PROCmenu
70 GOTO 60
80 END
90 DEFPROCmenu
100 PROCcomline
110 PROCheader
120 PRINT TAB(10);"MENU"
130 PROCscreen
140 CLS
150 *KEY 1 "1"
160 *KEY 2 "2"
170 *KEY 3 "3"
180 *KEY 4 "4"
190 PRINT TAB(10,8);"1..Edit File"
200 PRINT TAB(10,10);"2..Transmit"
210 PRINT TAB(10,12);"3..Terminal"
220 PRINT TAB(10,14);"4..EXIT"
230 PRINT TAB(10,20);
240 REPEAT
250 MS=INSTR("1234",GET$)
260 UNTIL MS>0
270 IF MS=1 THEN PROCedit
280 IF MS=2 THEN PROCtransmit
290 IF MS=3 THEN PROCterminal
300 IF MS=4 THEN PROCexit
310 ENDPROC
320 DEFPROCsave
330 CLS
340 IF PT=0 THEN PRINT "No File in Editor":ENDPROC
350 INPUT "FILE NAME ",SVM$:IF SVM$="" THEN SVM$=FLN$
360 CHN=OPENIN SVM$
370 IF CHN<>0 THEN INPUT"File Exists - Overwrite (Y/N "A$
:IF LEFT$(A$,1)="N" THEN 350 ELSE CLOSE#CHN
380 CLS
390 PRINT TAB(10,10);”Saving File “;SVM$
400 CHN=OPENOUT SVM$
410 FOR X=0 TO PT-1
420 FL$=SN$(X)+SL$(X)
430 PRINT# CHN,FL$
460 ENDPROC
470 DEFPROCload
480 CLS
490 INPUT"File Name ";FLN$
500 IF FLN$="" THEN ENDPROC
510 CHN=OPENIN FLN$
520 IF CHN=0 THEN PRINT TAB(10,10):"File doesn't exist":ENDPROC
530 X=0
540 REPEAT
550 INPUT# CHN,CS$
560 PROCstrip
570 SN$(X)=SN$:SL$(X)=SL$
580 X=X+1
590 PRINT SN$;" ";SL$
600 UNTIL EOF# CHN
610 CLOSE# CHN
620 PT=X:MX=VAL(SN$(X-1))
630 ENDPROC
640 DEFPROCtransmit
650 PROCheader
```

```
660 PRINT TAB(10);"Transmit"
670 PROCscreen
680 IF PT=0 THEN 780
690 PRINT "Edit File (Y/N) ";
700 C$=GET$
710 IF C$="N" THEN 780
720 PROCsetup
730 FOR X=0 TO PT-1
740 C$=SN$(X)+SL$(X)
750 PROCsend
760 NEXT
770 ENDPROC
780 INPUT "File name ";TFN$
790 IF TFN$="" THEN ENDPROC
800 CLS
810 PROCsetup
820 CHN=OPENIN TFN$
830 REPEAT
840 INPUT# CHN,CS$
850 PROCsend
860 UNTIL EOF# CHN
870 CLOSE# CHN
880 ENDPROC
890 DEFPROCexit
900 *FX18
910 VDU26
920 CLS
930 CLOSE#0
940 END
950 ENDPROC
960 DEFPROCterminal
970 PROCheader
980 PRINT TAB(10);"Terminal"
990 PROCscreen
1000 CLS
1010 PROCcomline
1020 PRINT" f0     f1     f2"
1030 PRINT"MENU   ECHO   LIST";
1040 PROCscreen
1050 EF=0
1060 *KEY 0 "|("
1070 *KEY 1 "|E"
1080 *KEY 2 "LIST"
1090 *FX4,1
1100 *FX229,1
1110 *FX7,7
1120 *FX8,7
1130 IF EF=0 THEN *FX3,5
1140 IF EF=1 THEN *FX3,7
1150 *FX2,2
1160 A=INKEY(1):IF A=-1 THEN 1200
1170 IF A=27 THEN 1260
1180 IF A=5 THEN EF=EF+1:IF EF>1 THEN EF=0
1190 VDU A
1200 REM READ PORT
1210 *FX3,4
1220 *FX2,1
1230 A=INKEY(1):IF A=-1 THEN 1130
1240 VDU A
1250 GOTO 1130
1260 *FX3,0
1270 *FX2,0
1280 *FX4,0
1290 *FX229,0
1300 ENDPROC
1310 DEFPROCheader
1320 VDU28,0,4,39,2
1330 CLS
1340 ENDPROC
```

```
1350 DEFPROCscreen
1360 VDU 28,0,22,39,5
1370 CLS
1380 ENDPROC
1390 DEFPROCcomline
1400 VDU 28,0,24,39,23
1410 CLS
1420 ENDPROC
1430 DEFPROCsim
1440 CN=OPENIN TFILE$
1450 *FX8,7
1460 *FX3,7
1470 REPEAT
1480 INPUT#CN,TS$
1490 PRINT TS$;CHR$(131);
1500 UNTIL EOF#CN
1510 CLOSE#CN
1520 *FX3,0
1530 ENDPROC
1540 DEFPROCdup
1550 ENDPROC
1560 DEFPROCofon
1570 ENDPROC
1580 DEFPROCedit
1590 PROCheader
1600 PRINT TAB(10);"File Editor";
1610 PROCcomline
1620 *KEY 0 "MENU¦M"
1630 *KEY 1 "LOAD¦M"
1640 *KEY 2 "LIST "
1650 *KEY 3 "AUTO¦M"
1660 *KEY 4 "¦M"
1670 *KEY 5 "RENUMBER "
1680 *KEY 6 "SAVE¦M"
1690 PRINT TAB(0,1);" f0    f1    f2    f3    f4    f5    f6"
1700 PRINT "MENU LOAD LIST AUTO OFF RENUM SAVE";
1710 PROCscreen
1720 PRINT">>";
1730 IF AF THEN PRINT STR$(BL);:INPUT LINE" " CS$:GOTO 2040
1740 INPUT LINE"" CS$
1750 LC$=LEFT$(CS$,3)
1760 IF LC$="LOA" THEN PROCload:GOTO 1720
1770 IF LC$="SAV" THEN PROCsave:GOTO 1720
1780 IF LC$="AUT" THEN AF=-1:GOTO 1720
1790 IF LC$="REN" AND PT<>0 THEN GOTO 1860
1800 IF LC$="NEW" THEN RMX=MX:RPT=PT:MX=0:PT=0:GOTO 1720
1810 IF LC$="OLD" AND PT=0 THEN PT=RPT:MX=RMX:GOTO 1720
1820 IF LC$="LIS" THEN 2330
1830 IF LC$="MEN" THEN GOTO 2560
1840 IF VAL(CS$)>0 THEN 2040
1850 PRINT "error":GOTO 1720
1860 REM **RENUM
1870 IF LEN(CS$)>8 THEN N$=RIGHT$(CS$,LEN(CS$)-8) ELSE N$="10"
1880 N=VAL(N$):IF N=0 THEN N=10
1890 FOR X=TO PT-1
1900 SP=INSTR(SL$(X),"GO")
1910 IF SP=0 THEN 1980
1920 PN=INSTR(SL$(X)," ",SP)
1930 DL$=RIGHT$(SL$(X),LEN(SL$(X))-PN)
1940 Y=-1:REPEAT
1950 Y=Y+1:UNTIL VAL(DL$)=VAL(SN$(Y)) OR Y=PT
1960 IF Y=PT THEN PRINT"Failed at line ";STR$((X+1)*N)
ELSE DL$=STR$((Y+1)*N)
1970 SL$(X)=LEFT$(SL$(X),PN)+DL
1980 NEXT
1990 FOR X=0 TO PT-1
2000 SN$(X)=STR$((X+1)*N)
2010 NEXT
2020 MX=VAL(SN$(PT-1))
```

```
2030 GOTO 1720
2040 REM ** INSERT
2050 IF CS$="" THEN AF=0:GOTO 1720
2060 IF AF THEN CS$=STR$(BL)+CS$
2070 PROCstrip
2080 IF SL$="" THEN 2470
2090 IF VAL(SN$)=MX THEN PT=PT-1
2100 IF VAL(SN$)<MX THEN 2170
2110 MX=VAL(SN$)
2120 SN$(PT)=SN$
2130 SL$(PT)=SL$
2140 PT=PT+1
2150 IF AF THEN BL=BL+LI
2160 GOTO 1720
2170 REM ** REFORM
2180 X=-1
2190 REPEAT
2200 X=X+1
2210 UNTIL VAL(SN$)<=VAL(SN$(X))
2220 IF VAL(SN$)=VAL(SN$(X)) THEN 2310
2230 FOR Y=X TO PT-1
2240 TN$=SN$(Y):TL$=SL$(Y)
2250 SN$(Y)=SN$:SL$(Y)=SL$
2260 SN$=TN$:SL$=TL$
2270 NEXT
2280 SN$(PT)=SN$:SL$(PT)=SL$
2290 PT=PT+1
2300 GOTO 1720
2310 SN$(X)=SN$:SL$(X)=SL$
2320 GOTO 1720
2330 REM ** LIST
2340 LL=0:UL=0:L%=LEN(CS$)
2350 LL=VAL(RIGHT$(CS$,L%-4))
2360 SP=INSTR(CS$,",")
2370 IF SP=0 THEN 2390
2380 UL=VAL(MID$(CS$,SP+1,L%-SP))
2390 IF UL=0 AND LL>0 AND SP=0 THEN UL=LL
2400 IF UL=0 THEN UL=MX
2410 FOR X=0 TO PT-1
2420 IF VAL(SN$(X))<LL THEN 2450
2430 IF VAL(SN$(X))>UL THEN X=PT-1:GOTO 2450
2440 PRINT SN$(X);" ";SL$(X)
2450 NEXT
2460 GOTO 1720
2470 IF VAL(SN$)=MX THEN PT=PT-1:MX=VAL(SN$(PT-1)):GOTO 1720
2480 X=-1
2490 REPEAT:X=X+1
2500 UNTIL VAL(SN$)<=VAL(SN$(X)) OR X=PT-1
2510 IF VAL(SN$)<>VAL(SN$(X)) THEN 1720
2520 FOR Y=X TO PT-2
2530 SN$(Y)=SN$(Y+1):SL$(Y)=SL$(Y+1)
2540 NEXT
2550 PT=PT-1:GOTO 1720
2560 ENDPROC
2570 DEFPROCsetup
2580 *FX7,7
2590 *FX8,7
2600 ENDPROC
2610 DEFPROCsend
2620 *FX3,5
2630 L%=LEN(CS$)
2640 FOR Y=1 TO L%
2650 A$=MID$(CS$,Y,1)
2660 PRINT A$;
2670 FOR Z=1 TO 5:NEXT
2680 NEXT
2690 PRINT CHR$(13);
2700 *FX3,0
2710 PRINT CHR$(10);
```

```
2720 *FX3,5
2730 T=TIME:REPEAT:UNTIL TIME>T+L%*10
2740 ENDPROC
2750 DEFRPROCstrip
2760 L%=LEN(CS$):SP=0
2770 REPEAT
2780 SP=SP+1
2790 N$=MID$(CS$,SP,1)
2800 UNTIL INSTR("0123456789",N$)=0 OR SP=L%+1
2810 SN$=LEFT$(CS$,SP-1)
2820 SL$=RIGHT$(CS$,L%-SP+1)
2830 ENDPROC
```