Mikrokomputer K-202

Dokumentacja archiwalna polskiego minikomputera K-202

K-202 modular

computer system

Skanowanie i opracowanie:
Ryszard Zenker
www.zenker.com.pl/k-202
e-mail: k-202@tlen.pl
Puszczykowo - 2008

K202 modular computer system

COMPUTER DIVISION MBM

M B M – K 2 0 2

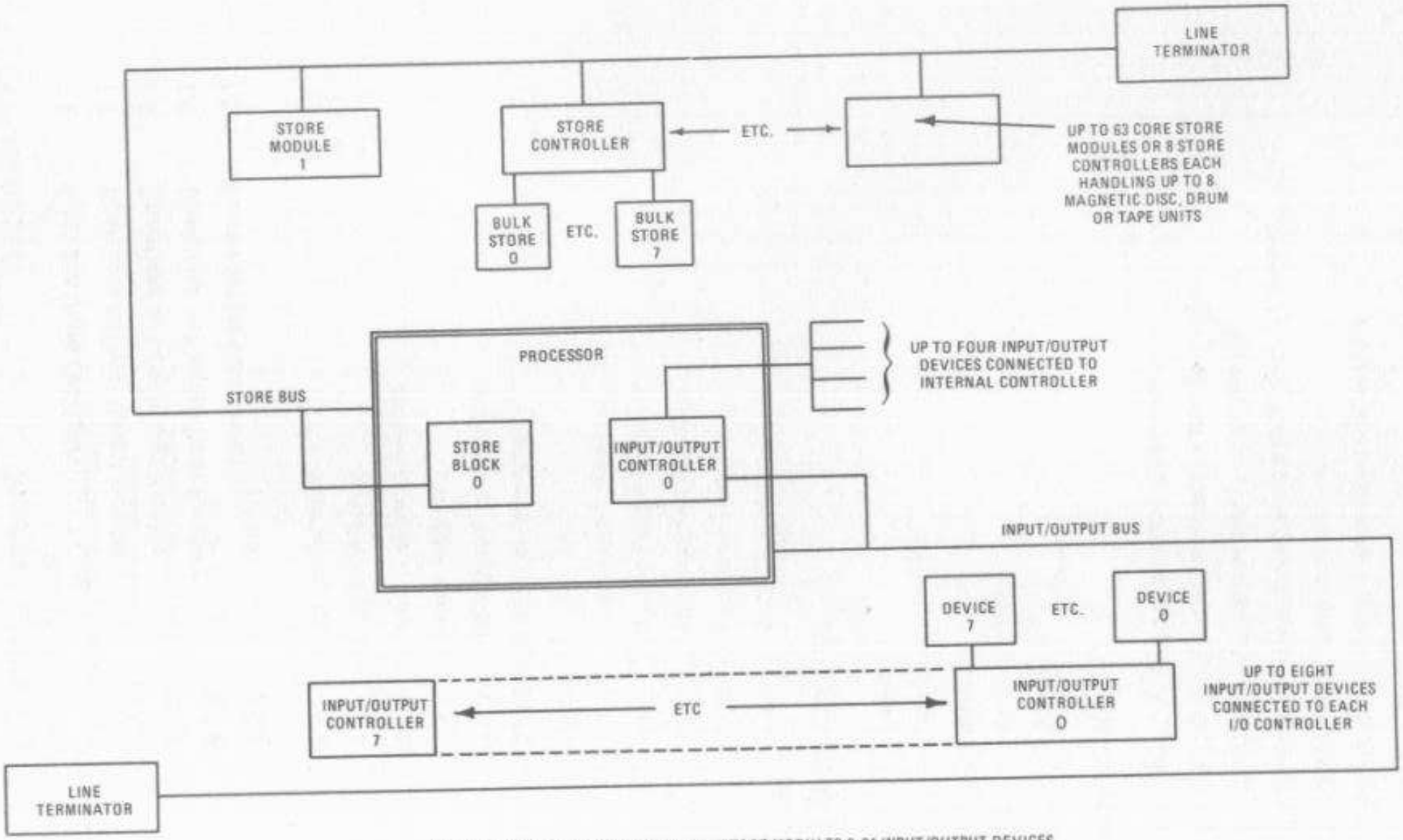
MODULAR COMPUTING SYSTEM

M.B. Metals Limited
Portslade, Sussex,
England.

Telephone: Brighton 46981

K 202 HANDBOOK

7. THE EXTRACODE SET.



THE PROCESSOR CAN ACCOMMODATE 64 STORE MODULES & 64 INPUT/OUTPUT DEVICES
THE FIRST MODULE OF EACH KIND IS ESSENTIAL IN EVEN THE
MINIMUM SYSTEM, & IS LOCATED INSIDE THE PROCESSOR.

1. INTRODUCTION

K 202 is a fourth generation computing system based on a fast, asynchronous processor having a sixteen bit word length and a dual bus which provides for very comprehensive bulk storage and input/output facilities.

Hardware and software are completely modular, and can be assembled into any configuration from a mini-computer with 4K store and one input/output device, to a large multi-processor system having hardware floating point units, up to four million words of directly addressable core store, and an almost unlimited number of peripheral devices. Bulk storage peripherals all have autonomous data transfer facilities, and all input/output devices work on a time-sharing basis. All peripherals have individual interrupts, with hardware pointers to their interrupt routines.

Thirty two levels of interrupt in the basic machine may be extended to 272 levels to provide very powerful facilities: multi-programming is a feature of all systems having more than one block of core store; multi-access is inherent in any system having more than one input/output peripheral.

The processor provides seven universal registers which serve as accumulators, indexes, pointers etc. A unique and sophisticated control logic combines fail-safe and auto-diagnostic properties with an economy of components which keeps the cost below that of machines offering less facilities.

A comprehensive range of software includes an Operating System, which combines an Executive program with Maths, Peripheral and Error routines; the ASSK Assembler; compilers for BASIC, FORTRAN IV, ALGOL 60, and C.S.L. high level languages, and TRAFOK, a conversational language for algebraic problems. A wide range of applications programs is available to all users.

2. THE OVERALL SYSTEM

This section describes how the K–202 hardware and software modules may be assembled into any configuration from the minimum to the maximum system. It also describes the multi-programming and multi-access facilities which are inherent in the system and explains the advantages of operating two or more processors on a common data highway.

2.1 MODULAR STRUCTURE

The K–202 system is completely modular, and may be built up from a minimum to a maximum system by simply plugging in extra modules. Systems software is also modular, and with each additional item of hardware is supplied the necessary software routines to update the operating system programs to cater for the new devices.

It is not necessary to switch off an installation whilst connecting new modules, but once connected it is usual to run systems tests before bringing the equipment back into service. The total time consumed in this procedure is less than a quarter of an hour; and it is never necessary to return equipment to the factory in order to add new modules.

Apart from the range of standard modules and peripherals, any make or type of non-standard peripheral or bulk storage device can be fitted with the aid of a simple custom built interface board. Provision is made for such boards in the peripheral controllers.

2.2 MINIMUM SYSTEM

The minimum system consists of the processor with operators control panel, and power supplies, coupled to 4096 words of read/write core store, and one input/output device such as a teleprinter. Apart from the teleprinter, this whole system is built into a single case size 19" wide by 8" high by 22" deep or may be mounted in a standard 19" rack. Also included in the basic unit are the controller and connectors for three further input/output devices, room to expand the internal store to 16K words, and bus connections to facilitate expansion up to the maximum system.

2.3 MAXIMUM SYSTEM

The maximum system includes the whole of the minimum system, plus a fast arithmetic unit; sixty-four storage or process control modules connected to the store bus, and sixty-four input/output modules connected to the input/output bus.

Modules which may be connected in any combination to the store bus include:-

1. Additional core stores, each having 16K to 64K words of 16 bits, and read/write cycle time of, 0.7 uS or 1.5 uS.
2. Bulk storage such as magnetic disc, drum or tape units. For every group of each such units, a store controller is required.

3. Multiplexors, each providing up to 32K inputs and/or outputs of 1, 8 or 16 bits for process control.

Modules which may be connected to the input/output bus include:-

1. Readers, Punches, printers, ploters and display units etc. For every group of eight such units, an input/output controller is required.
2. Multiplexors, each providing up to 16K inputs and/or outputs of 1 or 8 bits for process control.

2.4 MULTI-PROGRAMMING

Any system which includes more than one block of external core store is capable of operating in a multi-programming mode. That is, a number of separate programs can operate concurrently, each program having a level of priority which is allocated by the operator. The number of programs which can run concurrently is equal to the number of external blocks of operating store provided in the system.

The program which has control of the processor at any instant is that program which is of highest priority of those programs which are ready to take control. A program is deemed to be ready to take control of the processor if it has available all the data and peripheral devices which it needs at that moment to perform its required function. When a program becomes unready due to unavailability of data or peripheral devices, then the next highest priority program of those which are ready assumes control.

Priority control of multi-programs is a function of Executive, and the individual users programs do not themselves differ in any way from those which run on a single program machine. Executive ensures that programs cannot interfere with each other, and that no user can 'steal' data from another users program unless this has been specifically authorised.

2.5 MULTI-ACCESS

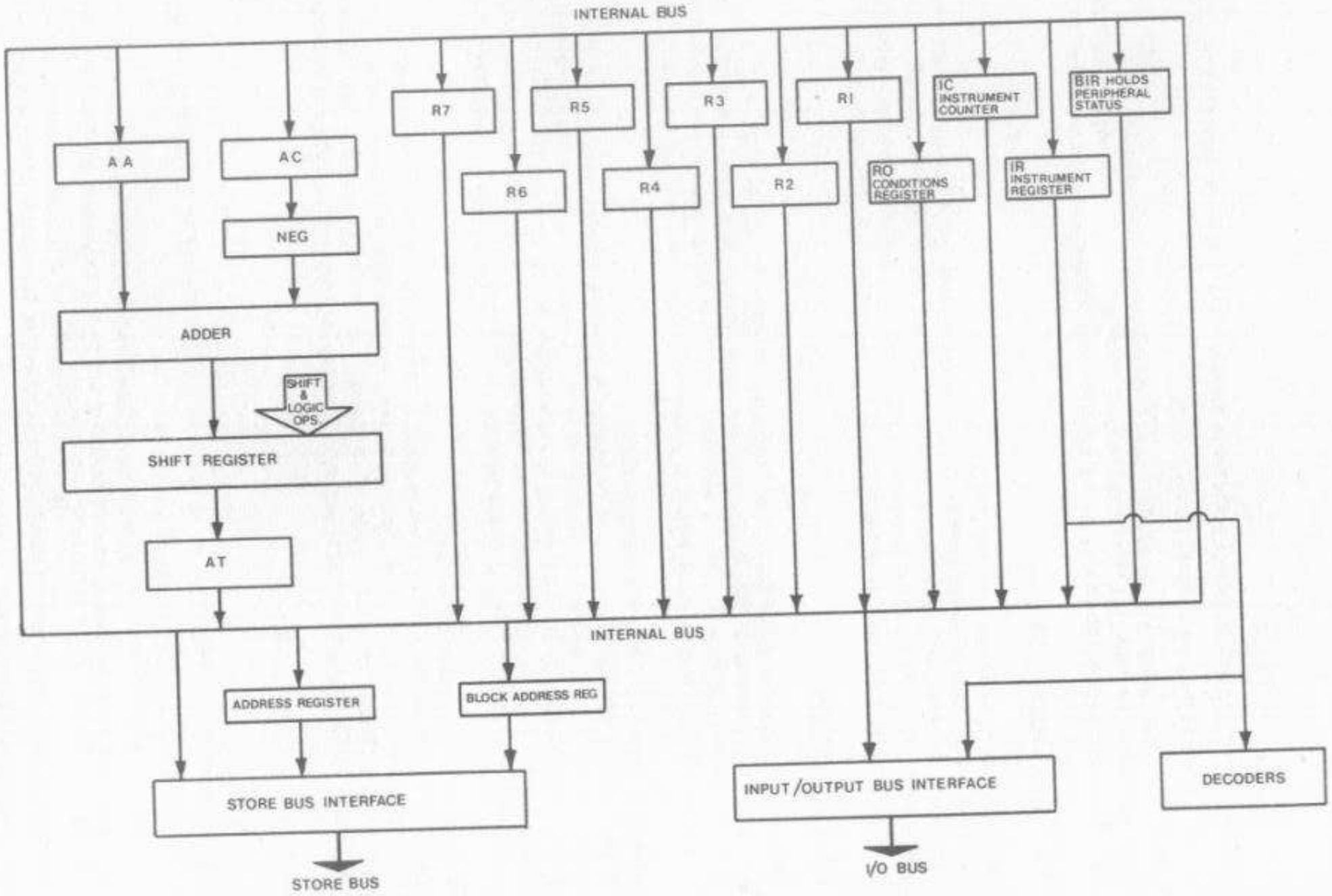
A facility which is desirable, if not essential in a multi-programming system, is multi-access. This means that two or more operators can concurrently but independently access the processor from their own particular input/output devices, either each controlling his own programs, or all controlling the same programs. K–202 provides for up to 64 separate access peripherals, all of which may be used concurrently.

2.6 MULTI-PROCESSORS

It is possible to connect up to four, or in some cases more, processors to the same store bus, so that they share the same bulk store and operating store, and may or may not share the same input/output peripherals. The advantages of this multi-processor system are:-

1. All processors have access to the same central data bank.
2. Systems software library is available to all processors, and is not separately stored for each.

3. Each processor has its own Executive program, which will be smaller and very much faster than that required for a single, large processor of the same computing power.
4. An overloaded system may be expanded by additional processors, without any change in techniques or systems as an organisation expands.
5. Two or more processors may be run in parallel to provide standby in the event of failure of one processor, where absolute reliability is demanded.


INTERNAL BUS
R7
R5
R3
R1
IC
INSTRUMENT
COUNTER
BIR HOLDS
PERIPHERAL
STATUS
A A
AC
R6
R4
R2
RO
CONDITIONS
REGISTER
IR
INSTRUMENT
REGISTER
NEG
ADDER
SHIFT
&
LOGIC
OPS
SHIFT REGISTER
AT
INTERNAL BUS
ADDRESS REGISTER
BLOCK ADDRESS REG
STORE BUS INTERFACE
INPUT /OUTPUT BUS INTERFACE
DECODERS
STORE BUS
I/O BUS

3. MODULES OF THE SYSTEM

This section describes the hardware modules which make up a K–202 system.

The processor as described in 3.1 is a complete computer in its own right, in that it includes basic power supplies, control panel, the basic minimum amount of core store, and an input/output controller. Add to this a peripheral device such as a teleprinter, and it becomes the minimum useable system, as described in section 2.2.

The remainder of section 3 goes on to describe the other modules which may be connected in any combination to this basic computer in order to build it up into a large and comprehensive system.

3.1 PROCESSOR

The processor can be considered in six parts:-

1. Register System
2. Control Logic
3. Interrupt priority logic
4. The internal store
5. The internal I/O Controller
6. Interfaces

3.1.1 Register System

All registers are connected to the internal bus, which is in turn connected to the external bus. This system provides the universal means of data transfer both inside and outside the processor.

Ten registers are directly available to the programmer. They are:-

R0 – Which holds the set of conditions upon which conditional orders depend.

R1 to R7 – Universal registers which serve as accumulators, modifiers, indexes, or pointers as required.

IC – The instruction counter

BAR – The block address register. This holds the address of the block of store in current use.

A further six registers are used in the logic, but are not directly available to the programmer. They are:-

AA & AC – Which hold the two operands at the input to the adder.

AT – Output register for the adder.

IR – The instruction register

AR – The address register

BIR – Holds the peripheral status word.

3.1.2 Control Logic

The control logic decodes the instructions and uses the information obtained to generate register gating and other signals to control the adder, store and peripheral logic.

Control of each instruction is achieved by generating in the logic a series of micro-instructions. The logic is switched consecutively through a series of 'states'. A state is said to exist when one of the 24 state bistables is set, only one of which can be set at any time.

The purpose of each state bistable is to generate a number of signals, which are levels, not pulses. Of the signals which it can generate, some are inhibited at any time dependent upon conditions set up earlier.

The first state to be set for a given instruction depends upon a decode of the bits of the instruction, together with conditions prevailing at that time in registers and on control and peripheral lines.

Choice of the next state depends upon similar parameters plus the presence of all the signals which should have been generated by the first state.

Any state may be used more than once in the chain, its effect differing each time according to the conditions set up earlier. An instruction chain may contain any number of states.

There is no central timer. The amount of time that a state remains set is simply that amount of time that it requires to generate its control signals. A logical sum of these forms the signal which steps to the next state.

Thus the system is fail safe: it cannot proceed to the next state until the first one has been completed satisfactorily. If signals are slow to come up, the system waits for them; if they fail to come up, the system stops. When this happens, a monostable terminates the faulty state after a few microseconds delay, and brings up the next state, but at the same time it generates an error signal to show where the fault occurred. The system can be stopped at any intermediate stage of the instruction chain by inhibiting any signal, leaving all signals available as steady, D.C. levels for inspection.

The resultant system is extremely fast in operation, completely asynchronous, and faults are self-diagnostic.

3.1.3 Interrupt Priority Logic

Thirty two levels of priority are provided in the interrupt system of the basic machine. This can be extended to 272 levels, by the addition of sub-levels, each having a hardware pointer to its interrupt routine.

The basic 32 levels are normally allocated in blocks of eight. The eight highest levels of priority are given to the system error interrupts. The next eight to the bulk store controllers; the third block of eight are allocated to the input/output controllers, and the last block are available for any other devices which may be connected to the bus. Each input/output controller and store controller has 16 sub levels of interrupt priority to cater for the devices it controls.

The chances of all interrupts occurring simultaneously are so remote as to be virtually non-existent, but if they did, then the one of lowest priority would be kept waiting only micro-seconds before being serviced.

3.14 Internal Store

At least one block of core store is essential in even the smallest system. This is called block 0 and is located inside the cabinet of the processor. It may vary in size from 4K to 16K words of 16 bits, and its cycle time may be 0.35 uS, 0.7 uS or 1.5 uS depending upon the requirements of the system. Apart from its location within the processor, and its limited size, it is in all other respects similar to the external stores described in section 3.4.

3.1.5 Internal I/O Controller

At least one input/output controller is essential in even the smallest system. This is called Controller 0 and is fited inside the cabinet of the processor. Unlike the other controllers, this one handles only four devices. However, it may be extended to eight devices by mounting the additional logic in a separate cabinet. Logically, it is similar to the external I/O controllers. Each device that it handles may provide for input or output or combined input and output.

3.1.6 Interfaces

Two interfaces are provided, one each for the store bus and the input/output bus. Each consists of a set of line terminators which provides two way communication between the internal bus and the respective external bus.

3.2 FAST ARITHMETIC UNIT

The fast arithmetic unit is an option which may be provided with the processor. If fitted, it goes inside the processor, occupying five logic board positions.

The function of this unit is to carry out floating point operations by hardware. It is about 36 times faster than the extracode instructions which would otherwise carry out these operations.

The inclusion of a fast arithmetic unit in a processor makes no difference to the programming. The program simply calls for the operation it requires, and Executive will utilise the fast arithmetic unit if fited, but otherwise will automatically call up Extracode. Use of a fast arithmetic unit does of course reduce the number of software modules required in the Operating System, as less extracodes are needed.

3.3 THE DUAL BUS

Two independent bus systems are provided. One caters for transfer of whole words, and is used primarily for communication between the processor, core store modules, and bulk storage peripherals. The other caters for transfer of eight-bit characters and is used by the processor to communicate with input/output devices. Both are connected to the internal bus, which in turn is connected to all the registers. The overall system of buses provides the universal means of communication both inside and outside the processor.

3.3.1 Why Not a Single Bus?

For simplicity and economy it is convenient to use a single bus on small computers. However, where a system has the expansion capabilities of the K–202, a single bus suffers from a number of inherent disadvantages.

(a) The user cannot be given freedom to attach any peripheral of his choice to a single bus, because unbuffered peripherals would stop the processor for relatively long periods by occupying the bus whilst preparing to accept or transmit their data. Providing a buffer increases the cost of interfaces for non-standard peripherals.

(b) Most input/output devices require only eight bit character transfers. The apparent economy of a single bus system is offset by the cost of cables and interfaces which provide for the transfer of 16 bits, half of which are not required.

For these reasons it was decided to utilise the more complex dual bus system. The store bus handles word transfers between the processor and blocks of store, including bulk storage units such as magnetic drums, discs and tape. The input/output bus deals exclusively with input and output character transfers, and should it be held up by a peripheral, the processor and store are not affected.

3.3.2 The Store Bus

The store bus consists of fifty pairs of balanced lines which are common to the processor, all core stores and bulk store controllers. Of these lines, 16 carry a word of data, 16 carry the address, 8 carry the block number and the remaining 10 are control and interrupt lines.

All transfers via the store bus are autonomous. That is, they are initiated by the processor, but then take place automatically whilst the processor carries on with other work. When the transfer of a whole block of words is complete, the processor will receive an interrupt to signal that the data is now available at its destination.

Any number of data transfers can take place concurrently, the various store controllers alternating in their use of the bus; as each store goes through its read/write cycle, it releases the bus for others to use.

Time taken for each data transfer is approximately one microsecond per word, but this can vary as the system is asynchronous. Stores with a slow cycle time will of course transfer data only as fast as the store itself can work.

3.3.3 Input/Output Bus

The input/output bus consists of 29 pairs of balanced lines which are common to the processor and all input/output controllers. Of these lines, 8 carry a character of data, 6 carry an address which consists of two octal numbers identifying controller and device, respectively, and the remaining 15 are control and interrupt lines.

All transfers via the input/output bus operate on a time sharing basis. That is, the users programs can call for transfer of different strings of characters to or from any number of peripheral devices, and they will all take place concurrently, without stopping the users programs from running; the various devices alternating in their use of the bus. The Executive program keeps a record of the progress of these transfers, and informs the various users programs when each block of data has been input or output as required.

Time taken for each character transfer is approximately one microsecond but of course the time interval between transfer of two consecutive characters will depend upon the speed of the peripheral device itself.

3.4 CORE STORES

Up to sixty-four blocks of 16 bit ferrite core store may be fited to a K–202 system, numbered 0 to 63 inclusive, this number being the address of the block for programming purposes.

Block 0 is essential in even the minimum system. It is located in the processor cabinet and is described in the processor section.

The remaining 63 blocks may each contain from 16K to 64K words, with cycle times of 0.7 uS or 1.5 uS, access time being about 40% of this figure in each case. Operation is on the 3 wire principle.

Each block is mounted, together with its power supplies in a frame which fits into a cabinet similar to that which houses the processor; or it may be mounted in a standard 19" rack. Three connectors are provided on each store module: a mains power connector, and bus input and output connectors. Where a block of core store is the last module on the bus, its bus output will be connected to a matched line terminator.

3.5 STORE CONTROLLERS

The function of a store controller is to operate up to eight magnetic disc, drum or tape units.

These may be identical units or a mixture of the different kinds. The controller is housed in a frame similar to the one which holds the processor. Eleven connectors are provided; these are bus input and output, mains power input, and eight connectors for the peripheral devices. Where a controller is the last module on the bus, its bus output is connected to a matched line terminator.

When a program calls for transfer of data between a bulk storage peripheral and a block of core store, the processor sends the request to the relevant store controller via the store bus. The controller then transfers the block of data via the bus, whilst the processor carries on with its other work. When the transfer is complete, the controller sends an interrupt signal which informs the processor that the data is now available at its destination. Each controller can handle transfers between any of its eight devices and any block of core store. All controllers can operate concurrently. An individual level of interrupt is provided for each controller, and within the controller is logic which allocates sub-levels of interrupt priority to each of the eight devices.

3.6 INPUT/OUTPUT CONTROLLERS

The function of an input/output controller is to operate up to eight input/output peripherals. These may be any combination of three kinds of device; input devices, such as paper tape or card readers; output devices such as punches or printers; or combined input/output devices, such as typewriters, teleprinters and digital display units.

The controller is housed in a frame similar to the one which holds the processor. Eleven connectors are provided; these are bus input and output, mains power input, and eight connectors for the peripheral devices.

The processor may transfer a character of data to or from any device, via the controller, and when the device is ready to send or receive the next character it will interrupt the processor to request the next transfer. In this way, all eight devices can transfer blocks of data concurrently and all controllers can operate at the same time. An individual level of interrupt is provided for each controller, and within the controller is logic which allocates sub-levels of interrupt priority to each of the eight devices.

3.7 COMBINED CONTROLLERS

In certain system configurations it is convenient to use a store controller or input/output controller of less than the maximum capacity. Also store blocks of limited size may be called for. For these reasons, bulk store and input/output controllers are available which each handle only four peripheral devices instead of the usual eight; and store blocks limited to 16K words are available.

Each of these units occupies only half a standard module cabinet; any two of them may be mounted together. Thus a single cabinet may contain controllers for 4 bulk storage peripherals and 4 input/output peripherals; or 16K words of core store may be mounted together with a controller for 4 peripheral devices.

4. FORMAT OF INSTRUCTIONS AND DATA

This section describes how instructions and data should be written in a program sheet, and how they are stored in the computer. It is, however, only a general introduction and is not intended to be a programming guide. For a complete specification of programming see the ASSK Assembly Language manual, and the manuals of the various high level programming languages.

4.1 INSTRUCTION FORMAT

A typical instruction in the K–202 assembly language specifies an operation to be performed between two operands. Some instructions require only one operand, or none at all, in which case those not required are omitted when writing the instructions.

The written instruction is translated on input to the computer into a series of binary numbers which are stored in from one to three consecutive computer words.

4.2 THE WRITTEN FORM OF INSTRUCTION

The general instruction format is:

OPERATION, A K

The Operation is specified by any of the mnemonics chosen from the instruction list.
A is the address of the general register which holds the first operand.

K specifies the second operand.

4.2.1 The register A, holding the first Operand will normally be used as an accumulator. A is specified by a label which has already been declared as having a value in the range 0 to 7, or by a number between 0 and 7. It is separated from the operation by a comma:

OPERATION, LABEL. K

or

OPERATION, N. K

4.2.2 The Second Operand K may be specified in a number of different ways, depending on the mode of address required. This is explained in the next paragraphs.

4.2.3 Address Modes

The second operand is usually specified in terms of the address of the register or store location in which it is to be found. Such addresses fall into two distinct groups:

1. Register Addresses. The address given for the second operand is either a number in the range 0 to 7, or a label which has been declared as having a value in the range 0 to 7. This number indicates which general register holds the second operand or a pointer to it. A register address is identified by enclosing it between a comma and a full stop thus:

```
OPERATION, A, ADDRESS.
```

2. Absolute Addresses. The address given for the second operand is either a number of up to 16 bits, or a label which has been or will be declared as having a value of up to 16 bits. This number indicates which store location holds the second operand or a pointer to it. An absolute address is identified by enclosing it in round brackets thus:

```
OPERATION, A (ADDRESS)
```

Within these two groups are provided a variety of different modes of address which are designed to facilitate the addressing of individual elements of data held in arrays, lists or stacks. Each of the following examples is shown first as a register address, and then as an absolute address.

4.2.4 Direct Address

The operand is specified as a label or number indicating the address of the register or store location in which it is to be found.

```
OPERATION, A, LABEL.
OPERATION, A (LABEL)
```

4.2.5 Indirect Address

The operand is specified as a label or number indicating the address of the register or store location in which will be found a pointer to the operand. Use of a pointer is indicated by an apostrophe:

```
OPERATION, A, LABEL'.
OPERATION, A (LABEL')
```

4.2.6 Indexed Address

The operand is specified as a label or number indicating the address of a register or store location, and a second label or number indicating the address of an index register. The contents of the index register are added to the contents of the first register or store location to form the operand. Use of an index is shown thus:

```
OPERATION, A, LABEL 1 & LABEL 2.
OPERATION, A (LABEL 1 & LABEL 2)
```

4.2.7 Indirect Indexed Address

The two previous facilities may be combined, and would then be written in the form:

OPERATION, A, LABEL 1 & LABEL 2'.
OPERATION, A (LABEL 1 & LABEL 2')

The contents of register or store location specified in label 1 and of register specified in label 2 are added together to form a pointer to the store location which holds the operand.

4.2.8 Group Address

It is often convenient to load a group of registers from a group of store locations or vice versa. The two instructions 'log' (load group) and 'stg' (store group) provide for this. In these instructions, the first operand specifies which group of registers should be dealt with, and the second operand indicates the first of a group of consecutive store locations to be used. The second operand can be generated by any of the above methods.

4.2.9 Conditional Instructions

Any instruction can be made conditional. This means that the contents of the conditions register R0 are compared with a parameter word which follows the instruction, and the instruction will be skipped unless R0 contains ones in the bit positions where the parameter word contains ones.

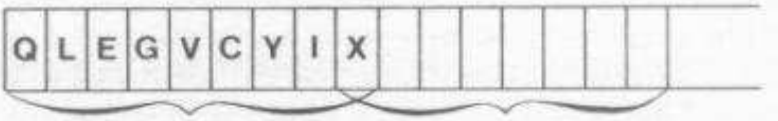
An instruction is made conditional by writing into the second operand a question mark followed by the parameter word.

examples:

OPERATION, A, LABEL 1 & LABEL 2 ? 0124020.
OPERATION, A, N' ? 011021.
OPERATION, A (LABEL ? 4096)
OPERATION, A (LABEL 1 & LABEL 2' ? 8192)

N.B. In the first two examples, the parameter word is shown in octal notation. In the second two examples the parameter word is shown in decimal notation. See para. 4.4.5 for format of octal numbers.

Layout of bits in the Conditions register, R0:



9 bits set by hardware

8 bits available to programmer

N.B. X can be set by hardware or the programmer.

The conditions register R0 has nine bits set by hardware to record the information shown below. X and the other seven bits may be set by the programmer for use as flags or indicators. R0 is used in conditional instructions to compare with a mask set by the programmer, and the instruction is carried out only if the comparison is valid.

Q – If Q = 0 then block zero of store in use and BAR does not modify addresses.

L – In the last comparison, operand II was less than operand I

E – In the last comparison, operand II was equal to operand I

G – In the last comparison, operand II was greater than operand I

C – Carry

V – Overflow

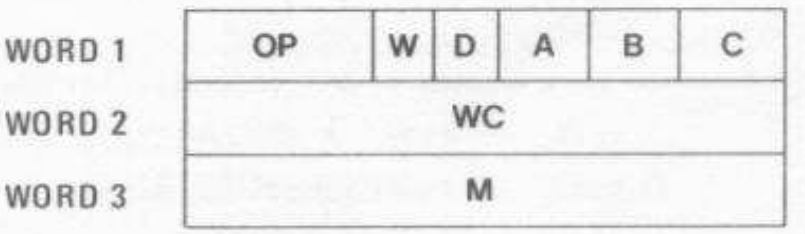
Y – Holds the bit which overflows at either end of the register during shift instructions.

I – Interrupts allowed if 1 = 0.

X – Set by hardware or software and used in shifts.

4.3 THE STORED FORM OF INSTRUCTION

When a program of instructions is writen in the form shown in the foregoing section, it is assembled into the computer store in such a way that it is held as a set of binary numbers.



Each instruction occupies from one to three computer words. The first word is essential to all instructions, and is split into six fields, each with a discrete meaning. The two remaining words are optional, and either or both may be omited. If M is used without WC, then it moves up to form the second word.

In practice, most instructions require only the first word, and very few require all three.

The meanings of the various fields are:

OP (5 bits)	Is a binary representation of the operation specified in the written instruction.
W (1 bit)	Is set to 1 if the instruction is conditional. This is shown in the writen instruction by the presence of a question mark.
D (1 bit)	Is set to 1 if the instruction is indirect. This is shown in the writen instruction by the presence of an apostrophe.
A (3 bits)	Is the address of the general register which holds the first operand and is to serve as an accumulator. It is specified by the number or label appearing in field A of the writen instruction.
B (3 bits)	Is the address of the general register which holds an index for the second operand. It is specified in the writen instruction by the number or label following the symbol '&'. If no index is specified then B = 0.
C (3 bits)	Is the address of the general register which holds the second operand or its pointer. It is specified by a number or label preceded by a comma in the K field of the writen instruction. If an absolute address was specified instead, then C = 0.
WC (16 bits)	Is present only if W = 1. If present, then it holds the parameter word which followed the question mark in the writen instruction.
M (16 bits)	Is present only if the written instruction specified an absolute address, in which case it holds that address.

4.4 FORMAT OF DATA

Numbers are represented in fractional ferm, negative numbers being held as their two's complement. The degree of accuracy with which a number can be represented depends upon the quantity of bits used to describe it.

4.4.1 Single Length Numbers

Single length numbers utilise 16 bits. Of these, bit 0 is the sign digit and the remaining 15 bits represent the number. Positive numbers can be represented with an accuracy of one part in 32,767, negative numbers to a slightly higher degree of accuracy.

4.4.2 Double Length Numbers

Double length numbers utilise 32 bits. Of these, bit 0 is the sign digit and the remaining 31 bits represent the number. Positive numbers can be represented with an accuracy of one part in 2,147,483,647, negative numbers to a slightly higher degree of accuracy.

4.4.3 Floating Point Numbers

Floating point numbers are held in the form $N = A \times 2^{b}$, where the mantissa, A, is held as a double length number; and the exponent, b, is held as a single length number. The largest number which can be accurately represented in this form is so great that this book is not large enough to hold it in print!

Floating point numbers may be writen into program or data sheets in the form of figures with decimal points, being separated from each other by commas thus:

3.1416, 2.331, 997.37,

Alternatively, they may be writen as a mantissa with exponent to the base ten thus:

31416 E–4, 2331 E–3, 99737 E–2

4.4.4 Format of Text

Text consisting of upper and lower case characters is stored in ISO 7 code, two characters per word.

Text which is to be copied from the program tape or cards straight onto the output medium without being stored is written in the program enclosed in triangular brackets thus:

TEXT

Text which is to be stored by the program is written into the program preceded by an exclamation mark and terminated by any symbol which is not a character thus:

! TEXT.

4.4.5 Octal Numbers

A number which appears as a constant in the program may be writen in Octal form. That is to say, each group of 3 binary digits, starting from the least significant end of the number, may be considered to be the binary equivalent of the numbers 0 to 7, and is written down as such. Thus the binary number 101 becomes 5 and 111101 becomes 75 etc. Octal numbers are identified in the program by preceding them with a figure 0.

5. THE OPERATING SYSTEM

The operating system or OPSYS consists of a suite of programs whose functions are to control the users programs and peripheral devices, to give warning of system errors, and to provide additional operations over and above those which are built into the logic.

The whole of OPSYS is modular, and its size in any particular installation depends upon the hardware configuration and the use to which it is put. In a typical small system, it would occupy about 2K words, and could be as much as 16K words in a very large and complex system.

In normal use, OPSYS is held in block 0 of store. However, in a large installation modules of it which are not required at any given time may be dumped onto a magnetic disc or drum, to leave more space in the core store.

The proportion of processor time occupied by OPSYS depends on the system configuration, and the quantity and type of users programs being run. For a medium to large configuration with an average mix of programs it should not exceed about 20%.

The various modules of the operating system can be considered in the following five groups:

5.1 EXECUTIVE

Executive is the program which keeps a record of the priority and ready status of each of the users programs, and at all times ensures that the processor operates under the control of the highest priority program of those which are ready. It also ensures that interrupts are dealt with in accordance with the priorities built into the peripheral controllers, and changes the ready status of any program as necessary in accordance with the availability of its peripheral devices.

5.2 PERIPHERAL ROUTINES

The peripheral routines are the blocks of program which control the peripheral devices. When a users program calls for transfer of data to or from a peripheral device, the number of the device and the address and quantity of data to be transferred are given to the appropriate peripheral routine, which carries out the transfer. All peripheral routines can run concurrently with the users programs and with each other.

5.3 MATHS ROUTINES

The maths routines are blocks of program which carry out standard mathematical operations such as multiplication, division, double precision or floating point arithmetic, and calculation of trig. functions. The user can call up each maths routine by means of a single instruction. All maths routines are re-entrant; that is, they can be used concurrently by several different users programs.

5.4 ERROR ROUTINES

A considerable amount of error checking circuitry is built into the hardware to continuously ascertain that the equipment is functioning properly. If a malfunction is detected, then this circuitry causes an interrupt to be given to the processor, which transfers control to the appropriate error routine. Also, certain aspects of the feasibility and legality of the users programs are checked and any infringement of the permited conditions causes transfer to an error routine.

The error routines are blocks of program which print out on the control teleprinter a statement as to the nature of the malfunction or infringement which was detected. Following an error statement, the running of users programs may or may not be suspended, depending on whether or not their continuation would cause irrepairable damage to the program, its data, or the equipment.

5.5 ADMIN

Admin is the program which communicates with the operators via their control teleprinters or alphanumeric display units, and co-ordinates the operations of all the other operating system programs.

X \ Y	0	1	2	3
0		Group F	Group Bo	Group B1
1	st	stb	lob	los
2	and	or	lo	jpar
3	orn	clmo	clbo	lom
4	ad	adc	su	co
5	adt	adot	adjt	cot
6	sts	lts	Lot	Group E
7	Group Go	Group G1	Group C	puf

Considering the operation codes as two octal numbers XY ranging from 00 to 73, the individual operations are shown here.

Where a single code is shown for a whole group, the individual operations within the group are specified by one of the other fields of the instruction.

Group G and certain other operations are reserved for exclusive use of the operating system.

6. THE INSTRUCTION SET

Basic programs for K–202 are composed of a list of the instructions shown in this chapter. The meaning of the operations part of each instruction is given beside it. The methods of expressing the operands A and K are described in Chapter 4.

6.1 GROUP A — Operations having two operands

lo, AK	load register A with operand K
los, AK	load operand K into the store location whose address is held in register A
ad, AK	Add to the contents of register A, the operand K. In the conditions register, C is set according to the most significant carry bit in the arithmetic unit, and V is set to 1 if overflow occurs.
adc, AK	The same as ad, but the original contents of C are also added to A.
Su, AK	Subtract operand K from the contents of register A. C and V are treated the same as in ad.
co, AK	Compare contents of reg. A with operand K and set, L, E or G in the conditions register, depending on whether A is less than, equal to or greater than K. Contents of A and K are not changed.
and, AK	Form in register A a logical sum of its original contents and operand K.
or, AK	Form in register A a logical mix of its original contents and operand K.
clmo, AK	Compare contents of register A with parameter word and if they agree, skip the next instruction. Contents of A and K are not changed.
clbo, AK	Compare contents of register A with operand K and if they agree, skip the next instruction. Contents of A and K are not changed.
orn, AK	Set Zero's into register A in bit positions marked by zero's in operand K.
lom, AK	Overwrite the contents of register A with operand K in the bit position where ones are set in the mask stored in register 7.
lob, AK	Load register A with operand K which is taken from the block of store specified in the block address register.
st, AK	Store the contents of register A in store location K.
stb, AK	Store the contents of register A in location K of the store block indicated in the block address register.
jpar, AK	Jump, seting link. Register A is loaded with the number which was in the instruction counter. The instruction counter is loaded with the contents of K.

6.2 GROUP B	– Operations without a first operand. Field A serves as an extension of the operation code.
jp K	Jump to the address K. K is writen into the instruction counter.
jpl K	Jump to the address K if L is set.
jpe K	Jump to the address K if E is set.
jpg K	Jump to the address K if G is set.
jpr K	Jump to subroutine: Store the contents of the instruction counter in place of K, and jump to K + 1.
jprl K	As above, but only if L is set.
jpre K	As above, but only if E is set.
jprg K	As above, but only if G is set.
sbar K	Store block address register contents in bits 8 to 15 of location K.
mod K	Modify the next instruction; K is stored in an internal register to be added to the second operand of the next instruction.
ex K	Jump to extracode. Also the contents of the instruction counter are loaded into store location 0; contents of Conditions register are loaded into store location 1.
reex K	Exit from Extracode. Also, the contents of the instruction counter and Conditions register are returned to these registers, from locations 0 and 1, but if the extracode caused overflow to be set, then this is left in the conditions register after exit.
ados K	Add one to K and skip the next instruction if K becomes zero.
zs K	Set location K to zero.
6.3 GROUP C	– Operations having no second operand. Field K serves as an extension of the operation code. In this section the leters QLEGVCX refer to individual bits in the conditions register R0
neg, A	Negate the contents of register A.
nega, A	Negate the contents of register A seting C and V.
nec, A	Negate the contents of register A and add C to it. Set C and V according to the new contents.
shl, A	Shift left. Contents of register A are shifted 1 place to the left.
shv, A	Shift left seting V
shly, A	Shift left seting Y.
shvy, A	Shift left seting V and Y.
shlx, A	Shift left with X.
shvx, A	Shift left seting V and X.
shr, A	Shift right: Contents of register A are shifted one place to the right.

shry, A	Shift right setting Y.
shrx, A	Shift right with X, seting Y
stxa, A	Set X according to the least significant bit in register A.
stxz, A	Set X according to the most significant bit in register A.

6.4 GROUP D — Instructions having a short second operand

$-63 \quad K \quad 63$

lot, A. K	Load K into register A
adt, A. K	Add K to the contents of register A. Set C and V.
adot, A. K	Add K to the contents of register A and if the result is zero then skip one instruction.
adjt, A. K	Add 1 to the contents of register A and if the result is not zero, then skip K locations.
cot, A. K	Compare K with contents of register A and set L, E or G. (Contents of register A are not changed).
lts, A. K	Load into register A the contents of the store location whose address is given by K plus the contents of the instruction counter.
sts, A. K	Load the contents of register A into the location whose address is given by K plus the contents of the instruction counter.

6.5 GROUP E — Jump instructions. Field A extends the operation code.

jpt, K	Skip K locations
jptl, K	Skip K locations if L is set
jpte, K	Skip K locations if E is set.
jptg, K	Skip K locations if G is set.
jptv, K	Skip K locations if V is set, and reset V.
jptx, K	Skip K locations if X is set.
jpty, K	Skip K locations if Y is set.
jpti, K	Skip K locations if I is set.

6.6 GROUP F — Group transmission instructions. Field A is used to extend the operation code.

The following instructions mean: Load the specified registers from a group of consecutive store locations starting with location K of the store block specified in the Block Address Register:-

log, 0 K	Registers 1 and 2
log, 1 K	Registers 1 to 3

log, 2 K	Registers 1 to 7
log, 3 K	Registers 5 to 7

The following instructions mean: Load the specified registers from a group of consecutive store locations starting with location K of store block 0.

log, 4 K	Registers 1 and 2
log, 5 K	Registers 1 to 3
log, 6 K	Registers 1 to 7
log, 7 K	Registers 5 to 7

The following instructions mean store the contents of the specified registers in a group of consecutive store locations starting with location K of the store block specified in the Block Address Register.

stg, 0 K	Registers 1 and 2
stg, 1 K	Registers 1 to 3
stg, 2 K	Registers 1 to 7
stg, 3 K	Registers 5 to 7

The following instructions mean store the contents of the specified registers in a group of consecutive store locations starting with location K of store block 0.

stg, 4 K	Registers 1 and 2
stg, 5 K	Registers 1 to 3
stg, 6 K	Registers 1 to 7
stg, 7 K	Registers 5 to 7

7. THE EXTRACODE SET

Extracodes are a set of instructions available to the user, which provide facilities over and above those catered for directly by the hardware. Each extracode consists of a single instruction, which when used calls up a small block of program held in the Operating System. This block of program consists of several of the basic instructions assembled together in such a way as to perform a standard operation such as computation of a trig. function or input or output of data in a standard format.

The extracodes are all re-entrant: they may be used by several programs concurrently, without causing delays or interference of one programme by another.

7.1 Extracodes for Input and Output of Data

EXTRACODE	EFFECT
ex(n)rec.I.	Read one character from input device n into the least significant eight bits of register I
ex(n)prc.I.	Output one character from the least significant eight bits of register I to output device n.
ex(n)redw.I.	Read a short decimal integer from input device n into register I. Number must be within the range −32768 to + 32767
ex(n)reow.I.	Read a short octal number from device n into register I.
ex(n)prdw.I.M.	Output a short decimal integer from register I to output device n. The integer will be preceded by one space, and either a second space or a minus sign. M specifies the number of decimal digits to be output.
ex(n)prow.I.	Output a short octal number from register I to output device n. The number is preceded by a space and includes all leading zeros.
ex(n)redi.	Read a long decimal integer from input device n into registers 1 and 2. Number must be within the range −2147483648 to +2147483647
ex(n)prdi.M.	Output a long decimal integer from registers 1 and 2 to output device n. The number is preceded by one space and either a second space or a minus sign. M specifies the number of digits to be printed.
ex(n)prt.K.	Output to device n the text starting in address K. Any amount of text may be output, its end being indicated in store by a character consisting of eight zero bits.
ex(n)redf.	Read a floating point number from input device n to registers 1, 2 and 3. The exponent is held in register 1 and the mantissa in registers 2 and 3.
ex(n)prdf.M.	Output on device n the floating point number is held in registers 1, 2 and 3. The number of decimal digits to be output is specified by M.

7.2 Extracodes for Operations on Short Integer Numbers.

NOTE: K is the second operand

EXTRACODE	EFFECT
ex,K.mplw.I.	Multiply the contents of register I by K and leave the result in register I. Set bit V in conditions register if overflow occurs.
ex,K.divw.I	Divide the contents of register I by K and leave the result in register I and remainder in register 4. V may be set.

7.3 Extracodes for Operations on Long Integer Numbers.

NOTE: Numbers must be in the range. −2147483648 to +2147483647. Operand 1 is held in registers 1 and 2. Operand 2 is held in store locations K and K+1.

ex,K.addi.	Operand 2 is added to Operand 1 and the result left in registers 1 and 2. V may be set.
ex,K.subi.	Operand 2 is subtracted from operand 1 and the result left in registers 1 and 2. V may be set.
ex,K.mpli.	Operand 1 is multiplied by operand 2 and the result left in registers 1 and 2. V may be set.
ex,K.divi.	Operand 1 is divided by operand 2 and the result left in registers 1 and 2. The remainder is lost. V may be set.
ex,4.absi.	Generate in registers 1 and 2 the modulus of their original contents.
ex,K.comi.	Compare operand 1 with operand 2 and set L, E or G in RO. Operands are not changed.

7.4 Extracodes for Operations on Floating Point Numbers.

NOTES: The first operand is held as an exponent in register 1 with a mantissa in registers 2 and 3. The second operand is held as an exponent in store location K with a mantissa in locations K + 1 and K + 2. The result is left in registers 1, 2 and 3.

ex(K)addf.	Add operand 2 to operand 1. If overflow occurs this is indicated on the control typewriter.
ex(K)subf.	Subtract operand 2 from operand 1. If overflow occurs this is indicated on the control typewriter.
ex(K)mplf.	Multiply operand 1 by operand 2. If overflow occurs this is indicated on the control typewriter.
ex(K)divf.	Divide operand 1 by operand 2. If overflow occurs, or if operand 2 is zero, these facts are indicated on the control typewriter.
ex(K)comf.	Compare operand 1 and operand 2. Set L, E or G in the conditions register. No change of operands.

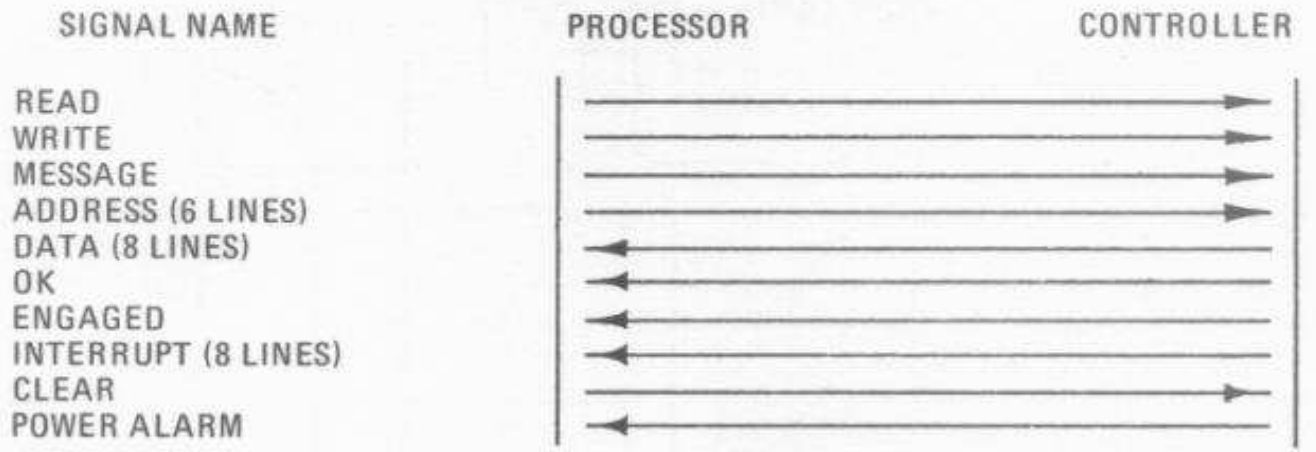
ex,4.abs	Generate the modulus of the floating point number held in registers 1, 2 and 3.
ex,4.sgn.	Floating point number held in registers 1, 2 and 3 is replaced by its sign only, held in floating point form.
ex,4.ent.	Generate the whole part of the floating point number held in registers 1, 2 and 3; the fractional part being lost.
ex,4.sin.	Generate the sin of floating point number held in registers 1, 2 and 3.
ex,4.cos.	Generate the cos of floating point number held in registers 1, 2 and 3.
ex,4.sqrt.	Generate the square root of the floating point number held in registers 1, 2 and 3. If the number was negative, then this is indicated on the control typewriter, and the square root of its absolute value is used in the program.
ex,4.ln	Generate the natural log of the floating point number held in registers 1, 2 and 3
ex,4.exp.	Generate the exponent of the floating point number held in registers 1, 2 and 3.
ex,4.arct.	Generate the arctan of the floating point number held in registers 1, 2 and 3.
ex,4.actg.	Generate the arc cotangent of the floating point number held in registers 1, 2 and 3.

7.5 Extracodes for other functions.

ex,l.flow.	Change short integer in register l into a floating point number to be placed in registers 1, 2 and 3.
ex,4.floi.	Change the long integer held in registers 1 and 2 into a floating point number to be placed in registers 1, 2 and 3.
ex,4.intw.l.	Change the floating point number held in registers 1, 2 and 3 into a short integer to be placed in register l. If the permitted range is exceeded this will be indicated on the control typewriter.
ex,4.inti.	Change the floating point number held in registers 1, 2 and 3 into a long integer to be placed in registers 1 and 2. If permitted range is exceeded, this will be indicated on the control typewriter.

APPENDIX 1

INPUT/OUTPUT BUS



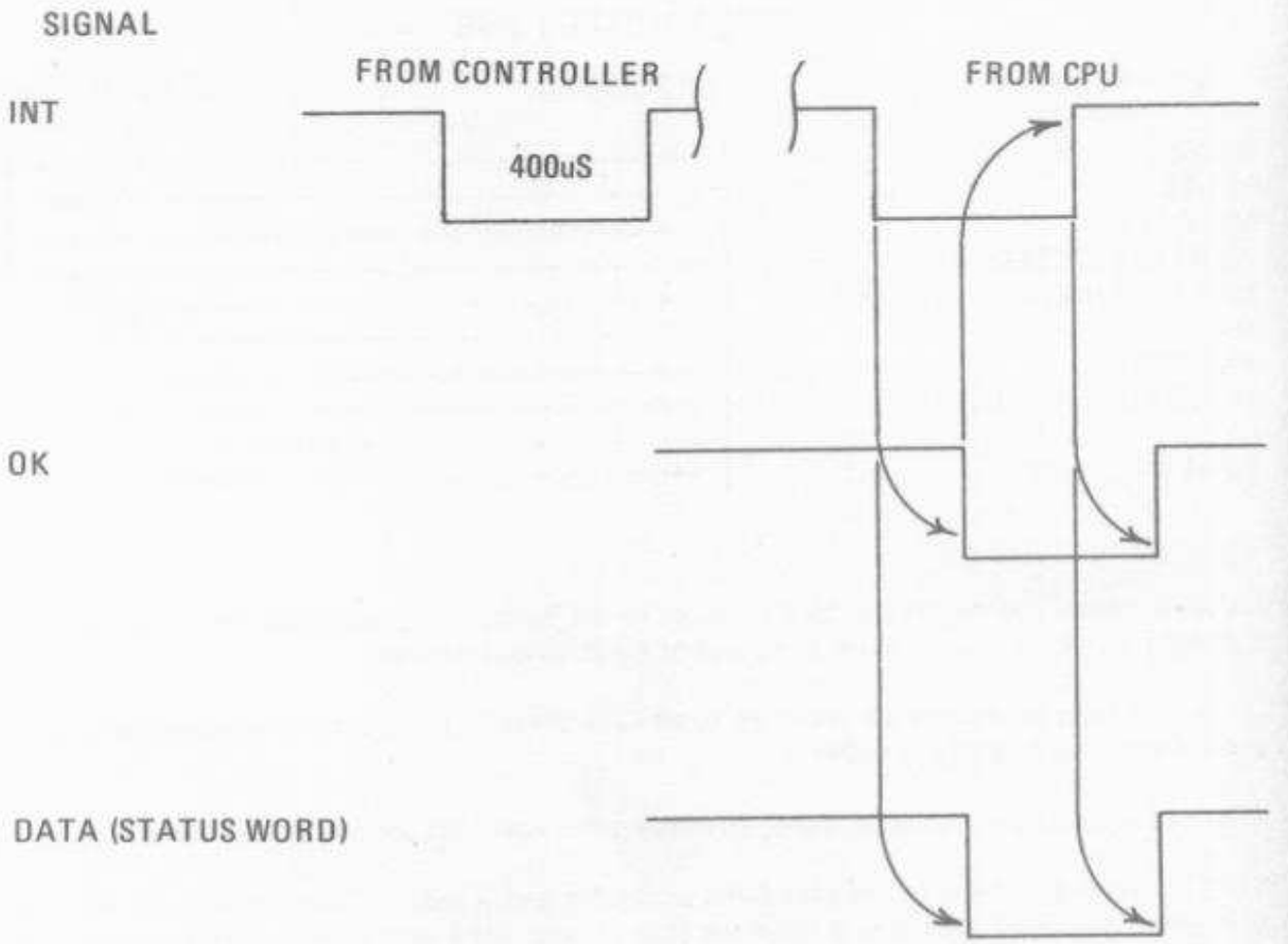
Eight peripheral controllers can be connected to the input/output bus. Each controller handles 8 peripherals which may be input or output or input/output devices.

The 29 lines of the bus are common to all controllers, but of the eight interrupt lines, only one is connected in each controller.

ADDRESS consists of two octal digits, identifying the controller and the device respectively.

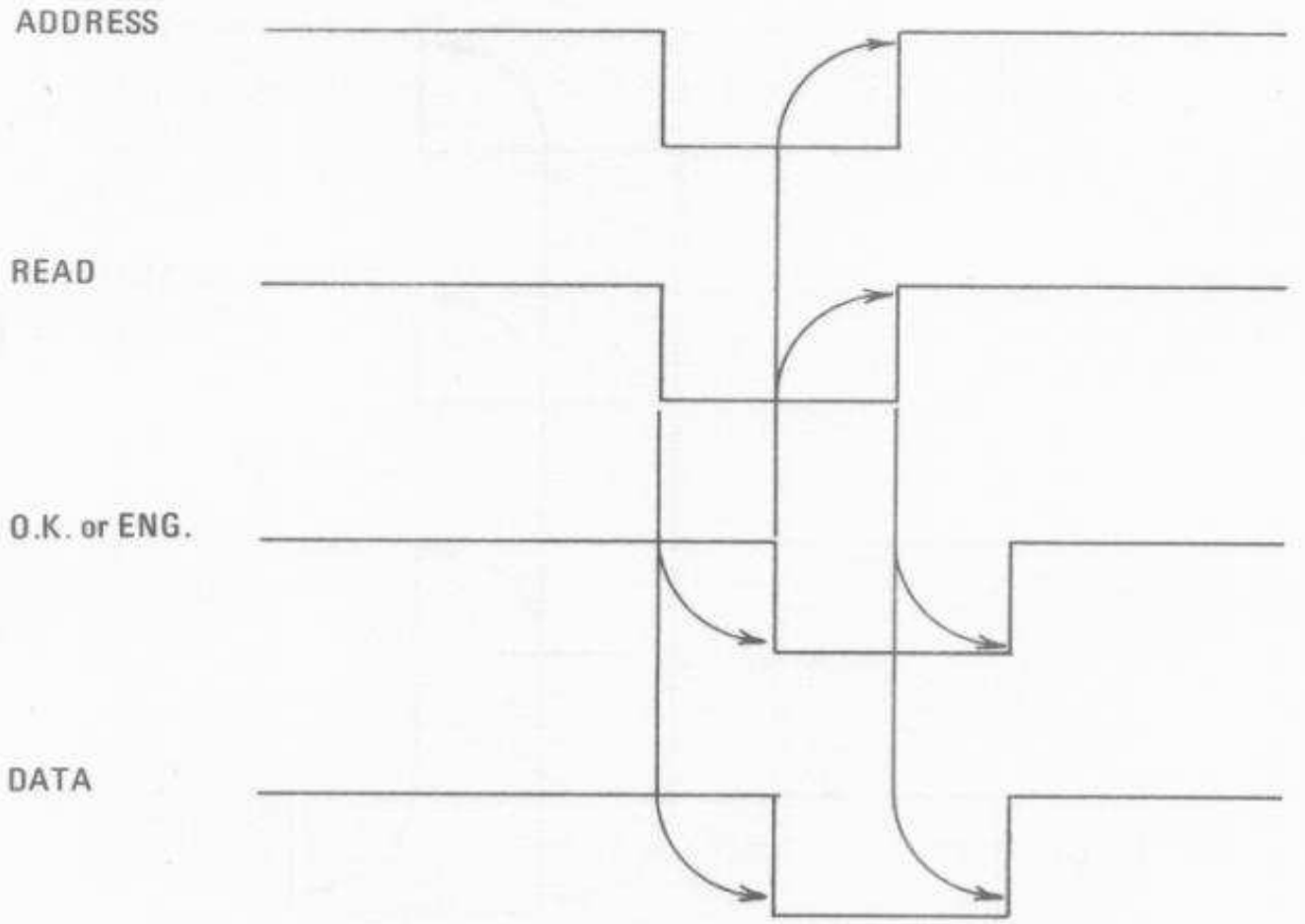
STATUS WORD is held in the peripheral controller and is sent to the processor via the data lines when requested. Bits 0 to 3 show the class of item. Bit 4 indicates whether it is normal or error interrupt. Bits 5 to 7 are the number of the device.

INPUT STATUS WORD



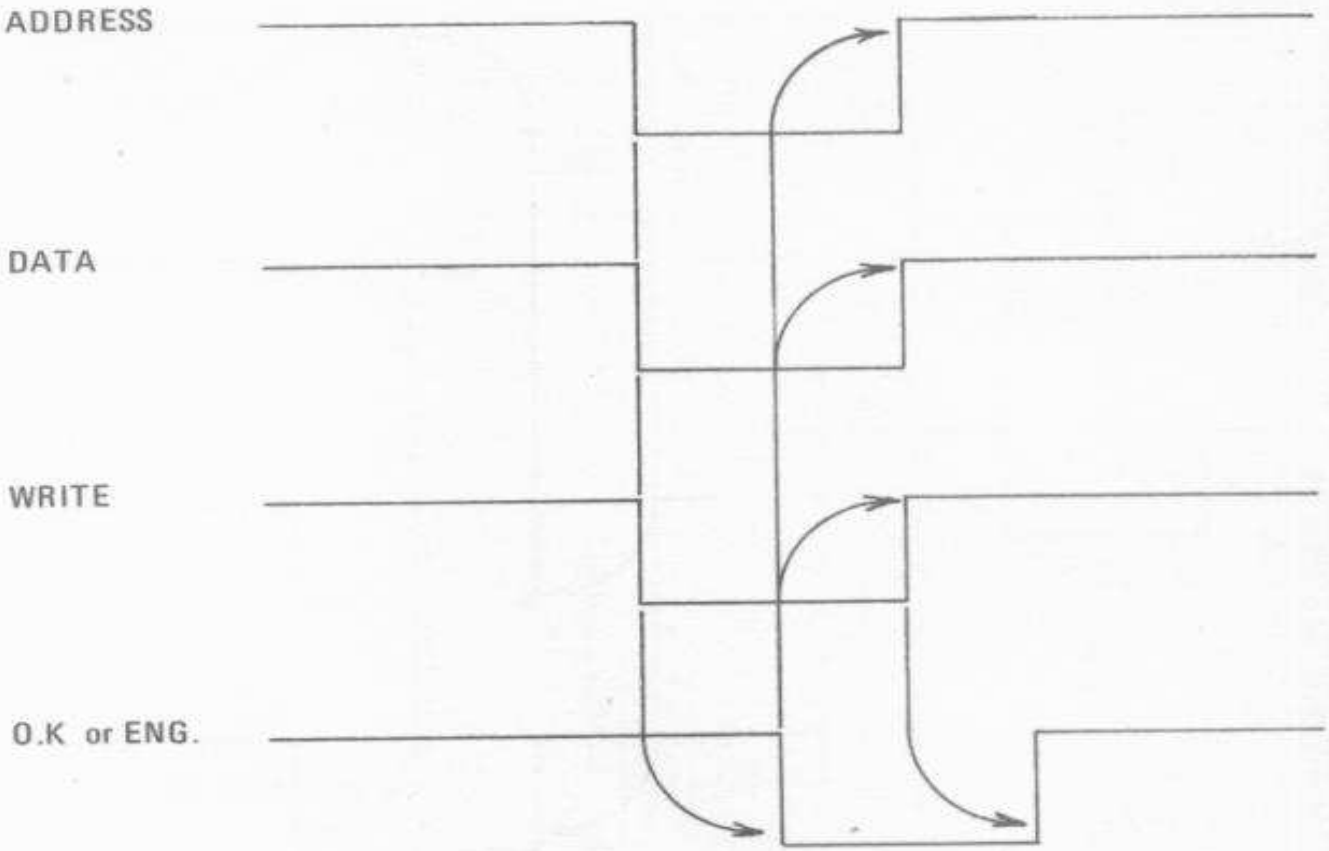
1. The end of the previous operation gives an interrupt from the device when it is available.
2. The CPU will reply on the interrupt line when it is ready for the status word.
3. The peripheral controller will respond by outputting the status word on the data lines, accompanied by either OK or ENGAGED.
4. The status word shows:
   (a) the device number
   (b) the type of interrupt i.e. manual, input or output
   (c) whether normal or error.
5. The small arrows in the diagram show which signal causes each subsequent one to be generated.

INPUT DATA



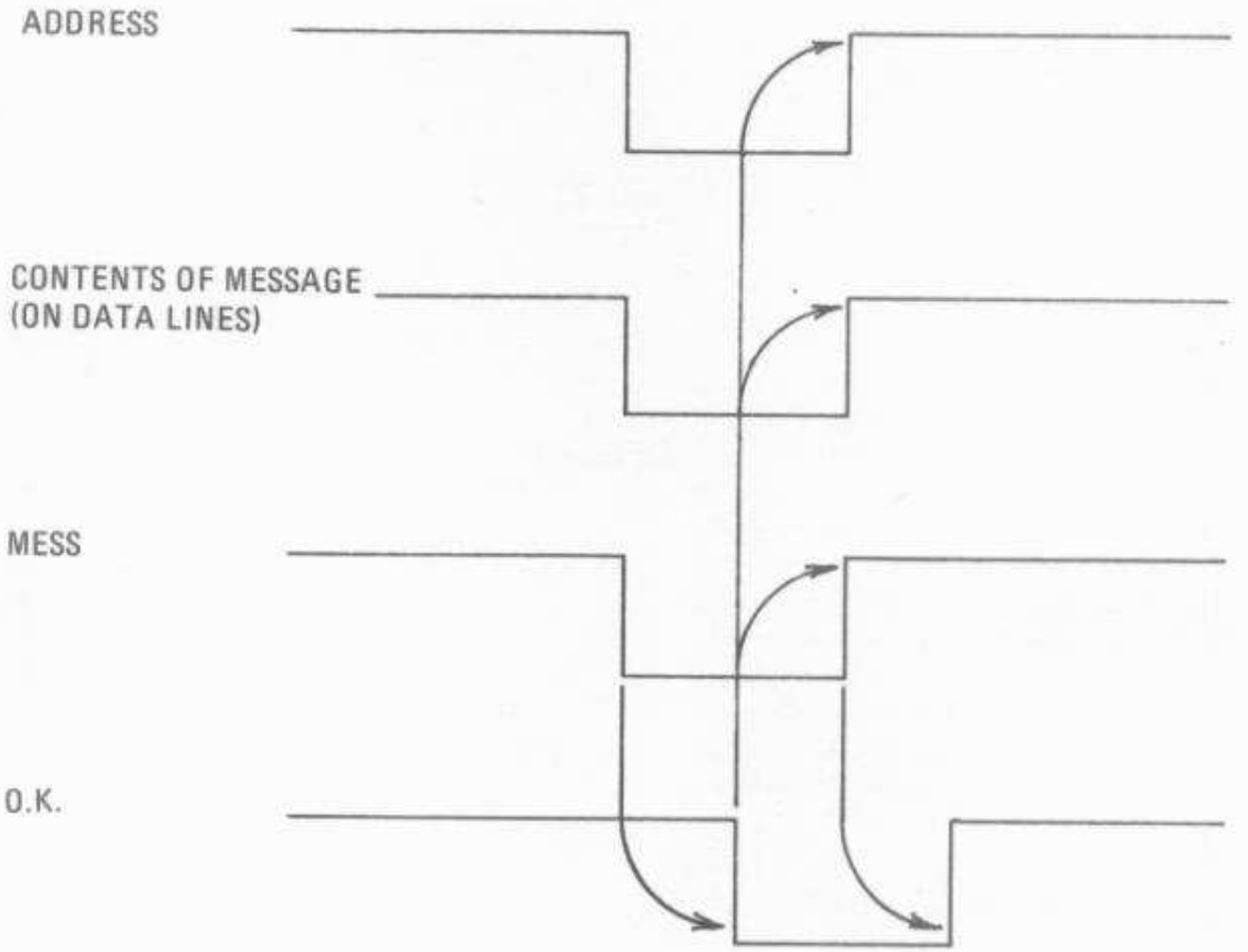
1. CPU outputs ADDRESS of peripheral and READ.
2. Peripheral replies with DATA and OK.
3. If the peripheral has no data ready, it replies with ENGAGED.

OUTPUT DATA



1. CPU outputs ADDRESS of peripheral, DATA, and WRITE.
2. Peripheral accepts the data and replies with OK.
3. If the peripheral could not accept the data it replies with ENGAGED.

OUTPUT MESSAGE

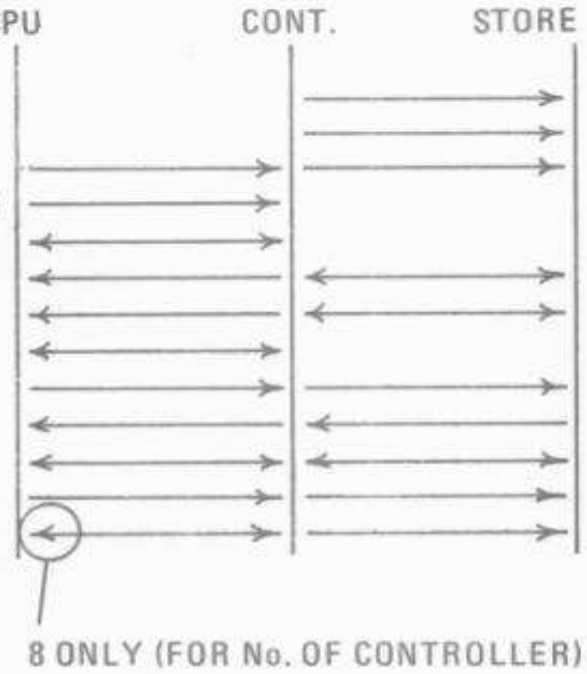


1. CPU outputs ADDRESS of peripheral, contents of message on data lines, and the signal MESSAGE.
2. Peripheral accepts the message and replies with OK. If no reply is received within a pre-set time, then a monostable in the CPU generates a signal which indicates that the peripheral is not connected.

APPENDIX 2

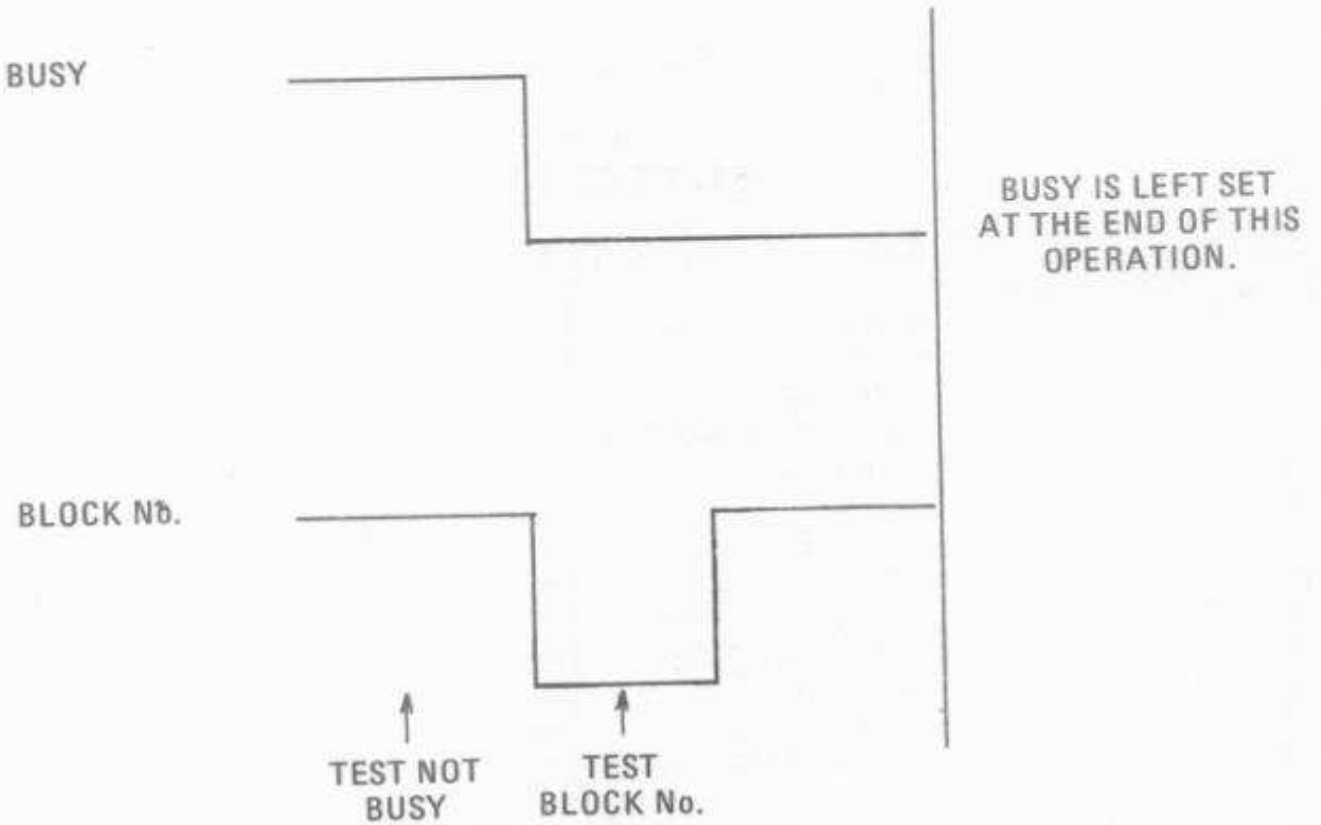
STORE BUS

SIGNAL	NAME
M – R	READ
M – W	WRITE
M – M	MESSAGE
M – P	(INITIATE TRANSFER)
M – IN	INTERRUPT
M – OK	OK
M – EN	ENGAGED
M – BU	BUSY
M – CL	CLEAR
M – PA	POWER ALARM
M – DT	DATA (16)
M – NB	BLOCK No. (8)
M – AD	ADDRESS (16)



50 lines in all

BUS ACCESS WAVEFORMS



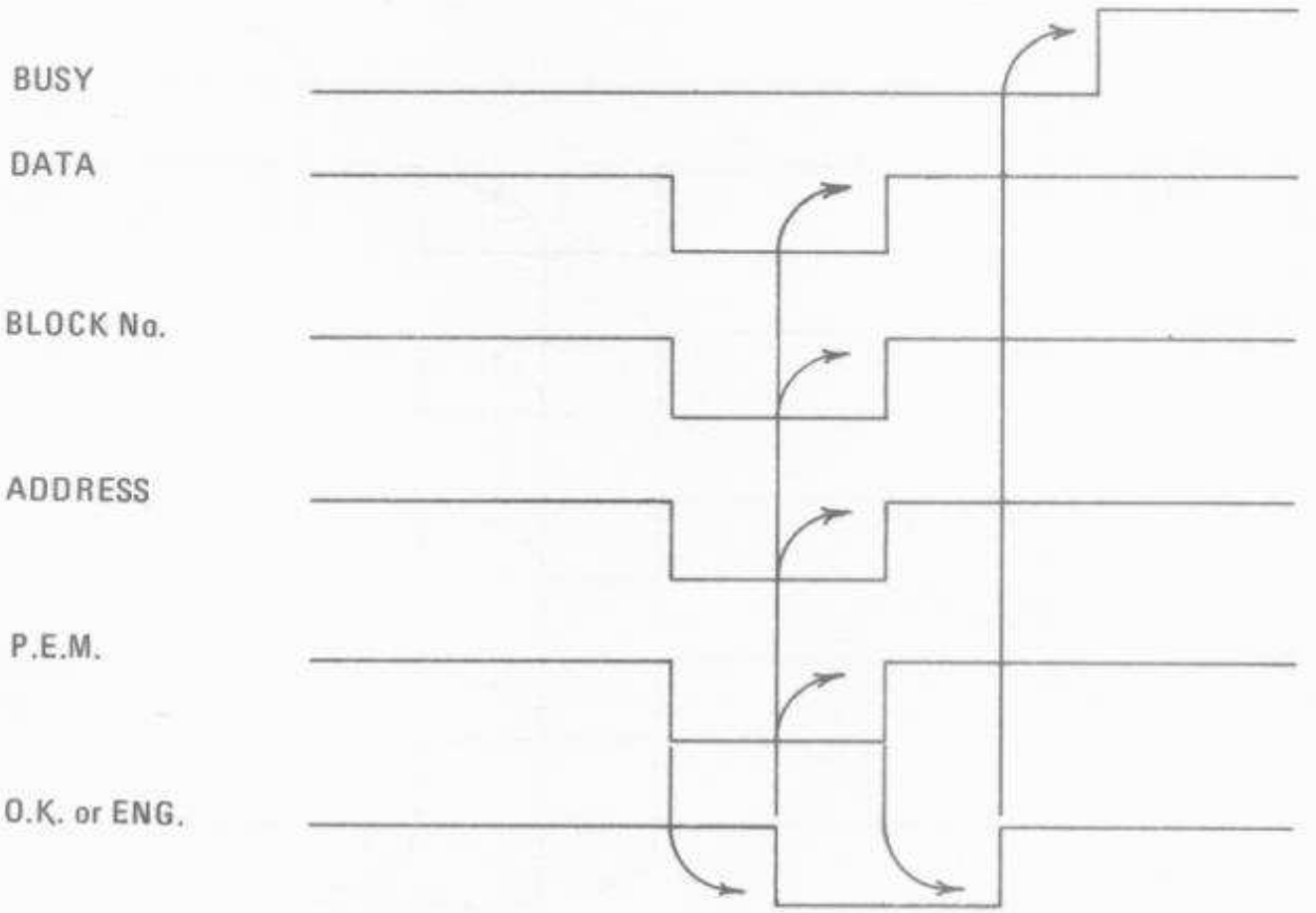
1. Any processor or store controller which wishes to access the bus will first check that the busy line is up; i.e. the bus is not occupied. If occupied it will keep testing until free.
2. When the bus is free, the controller will transmit a signal on the busy line, and its own block number on the block no. lines.
3. It will then check the block no. to see it is correct. This is to ensure that no other controller tried to claim the bus at the same time. If the number on the lines is incorrect, the controller will revert to step I. Otherwise, it will remove its address from the block no. lines, leaving the busy signal set.

N.B. This procedure is carried out by:

(a) A store controller, once for each word it wishes to send or receive via the bus, and once for each interrupt it wishes to send to the processor.

(b) A processor, once for each word it wishes to transfer via the bus, and once for each ADT it wishes to initiate.

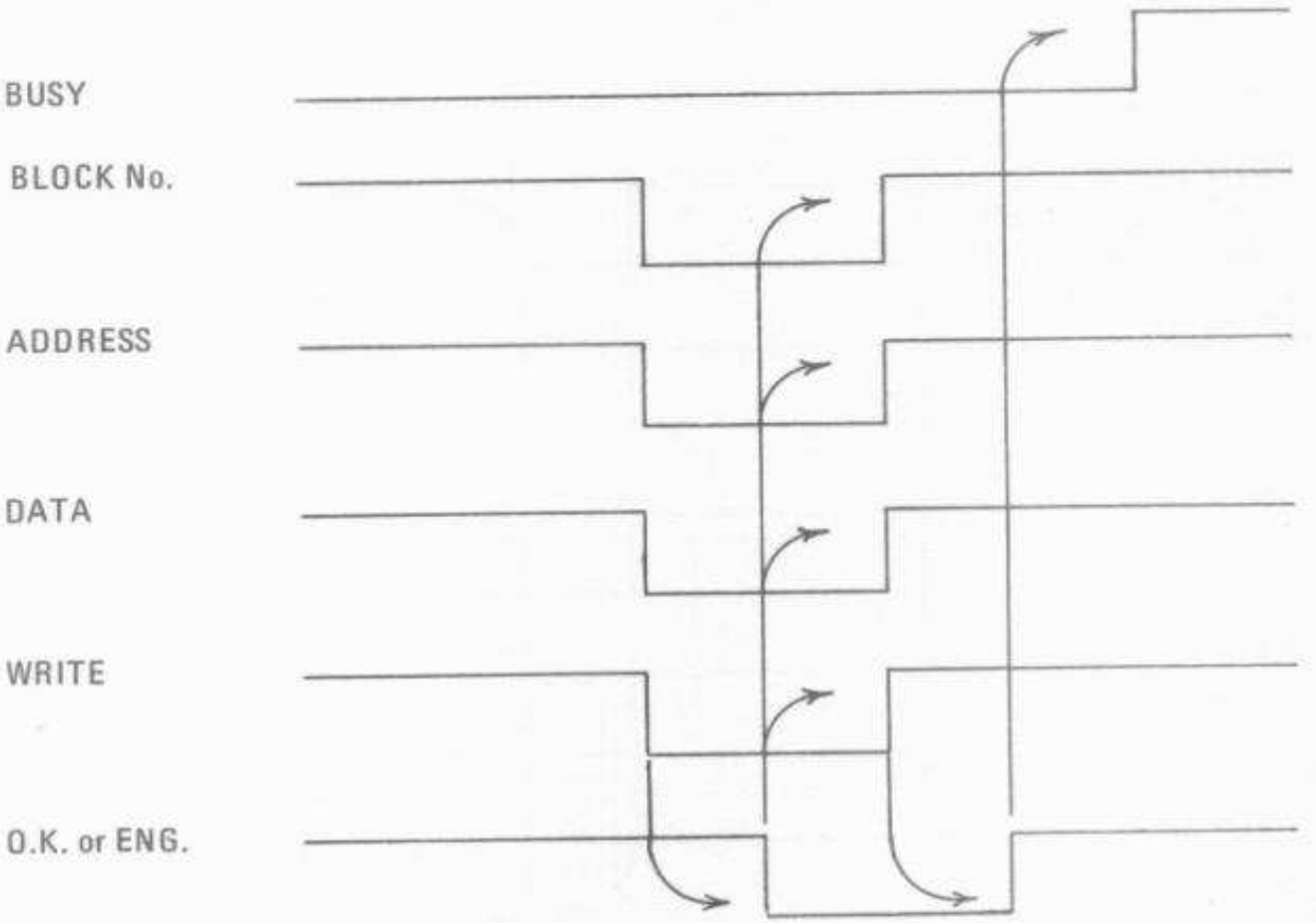
WAVEFORMS TO INITIATE A DATA TRANSFER



1. BUSY was left set by the bus access procedure.
2. Processor sends on bus the following signals:
   on DATA lines: Block No. of store holding control field.
   on BLOCK No. lines: Block No. of controller which is to be master.
   on ADDRESS lines: Address within block of control field.
   PEM: The signal which instructs controller to operate.
3. The required controller identifies itself from the BLOCK No. lines, stores the information on the DATA & ADDRESS lines, and sends back the signal OK, or if unable to handle the transfer it sends back ENGAGED.

N.B. This procedure is carried out by a processor when it wants to initiate a data transfer.

DATA TRANSFER FROM MASTER TO SLAVE



1. BUSY was left set by the bus access procedure.
2. Controller sends on bus the following signals.
   on BLOCK No. lines: Block No. of slave.
   on ADDRESS lines: Address of data with slave block.
   on DATA lines: Word of data to be transferred.
   WRITE: Instruction to write data in the address space.
3. The required slave identifies itself from the Block No. lines, and writes the word of data into the location specified on the address lines, and sends back the signal OK, or if unable to accept the data sends ENGAGED.

MB METALS LIMITED Victoria Road Portslade Sussex England telephone Brighton 46981