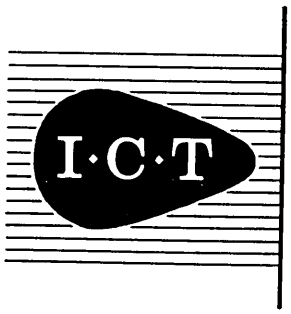
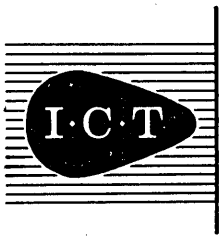
I.C.T. ATLAS 2 SUPERVISOR AND OPERATING SYSTEM



COMPUTER PUBLICATIONS



COMPUTER
PUBLICATIONS

I.C.T. ATLAS 2 SUPERVISOR AND OPERATING SYSTEM

INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

68 NEWMAN STREET · LONDON W1

First Edition.....November 1964

5056

Printed in Great Britain by

International Computers and Tabulators Limited,

London.

ATLAS 2

SUPERVISOR AND OPERATING SYSTEM

CONTENTS

5056(11.64)

CONTENTS Continued

5056(11.64)

Chapter 1

INTRODUCTION

GENERAL 1.1

Atlas 2 is a new, smaller version of the I.C.T. Atlas Computer. Atlas 1 (Ref. 1), the original version, was developed jointly by Ferranti Ltd. and the Department of Electrical Engineering at Manchester University, where the first machine came into operation in January 1963. Atlas 2 (Ref. 2) has been developed in conjunction with the Mathematical Laboratory at Cambridge University, where the first machine is now coming into operation (Summer 1964). The Computer Department of Ferranti Ltd. was acquired in September 1963 by I.C.T. Ltd., who now manufacture and market the Atlas computers.

Atlas 2 reproduces many of the important features of Atlas 1, including the comprehensive time-sharing of programs and peripheral transfers, and it is capable of speeds approaching that of the larger machine. Like Atlas 1, all its activities are controlled by a master program called the Supervisor, but, unlike Atlas 1, it has no fixed store, subsidiary store, or drum store, so that all the Supervisor routines, working space, and bufferage, must be either in the main core store or on magnetic tape (or on magnetic discs if a disc file is attached). Also, like Atlas 1, there is a special store, called the V-store, which is in fact a collection of flip-flops throughout the computer, which can be read, set and reset by reading from, or writing to, particular store addresses, and which are used to obtain information from, and control the activities of, the peripherals and certain other parts of the machine.

The majority of the Central Processing Unit, including the whole of the Arithmetic Unit, is the same as in Atlas 1, so that a very high degree of program compatibility can be achieved, but, because of the absence of special stores and drum store, the premium on efficient utilization of core store is very much higher. Also, whereas Atlas 1 has Page Address Registers (Ref. 3) permitting each programmer to use a floating address system for blocks of 512 words, the blocks being numbered arbitrarily in the range 0 to 1791, Atlas 2 provides each programmer with a single area of store addressed continuously from zero upwards relative to some origin, and normally during the running of a program this is fixed in size and position. For this and other reasons, the Atlas 2 Supervisor is internally very different from that of Atlas 1 (Ref. 4), although it uses similar techniques in many instances, and its purpose is to provide comparable facilities and to make Atlas 1 and Atlas 2 look almost the same to the programmer. It turns out that the machines also look similar to the operator.

In this document we are not concerned with presenting the hardware, but certain aspects must be mentioned in order to make clear the workings of the Supervisor. The core store may have 32k, 64k, or 128k words (k=1024), the word length being 48 bits. A word can represent a single floating-point number, two 24-bit half-words (which may be interpreted either as fixed-point integers or as logical bit-patterns), eight 6-bit characters, or a single instruction. 24-bit half-words are used for addressing the stores of Atlas 2. Of these, the top three bits are used to distinguish the stores and the mode of addressing (see Chapter 3): these are normally zero for object program addressing; the next bit is normally zero; the next 17 bits are used to address 48-bit words within the stores; the next bit is used to permit half-word addressing, and this and the last two bits are used to permit character addressing.

There are 128 index registers, or B-lines, each 24 bits long. There are three control registers, referred to as "main control", "extracode control", and "interrupt control", which are also B-lines 127, 126, and 125. Main control is used by object programs. When main control is active, access to the V-store, to the Supervisor, or to other programs sharing the store is prevented by hardware,

and this ensures that object programs cannot interfere with each other or with the Supervisor. Extracode control is used by a number of built-in subroutines, called extracodes, which can be activated by the programmer by a single instruction (called an extracode instruction), and also by the major routines of the Supervisor. When extracode control is active, access can be made to the V-store and to the whole of the main store, thus permitting control of the peripheral devices within the extracode subroutines and the Supervisor, and permitting the Supervisor to control the placing of programs etc. Interrupt control is used in short routines within the Supervisor which deal with activities requiring immediate attention. These routines are entered, for example, when action is required by a peripheral, or when a program or machine fault occurs; the program using main or extracode control is interrupted, and continues when the interrupt routine is completed.

While interrupt control is active, further interrupts are inhibited by an "Inhibit Interrupts flip-flop". This flip-flop can also be set, if required, by the Supervisor, for routines obeyed on extracode or main control.

A wide range of peripheral equipments can be attached, including card readers and punches, paper tape readers and punches, line-printers, 1" and $\frac{1}{2}$" magnetic tape decks, magnetic disc files etc., and the design is such that all of these can operate simultaneously.

THE PHILOSOPHY 1.2

The purpose of the Supervisor is to control all the activities of the computer, and to control them in such a way as to achieve high internal efficiency, thereby maximizing "through-put" and minimizing the cost-per-operation, and in such a way as to provide a system which is easy to operate, and easy to program.

Atlas 2 is a time-sharing computer, that is there is provision for the execution of more than one program simultaneously, as well as the independent handling of input and output activities. To be safe, a time-sharing system must incorporate full protection, that is it must be impossible under any circumstances for a program to affect any other program with which it is being time-shared. To be convenient, the system must be designed so that programs are written and compiled separately, without the programmer or the compiler having to take account of other programs. Also the time-sharing must not dictate a rigid pattern of day-to-day operation of the installation. Protection can only be provided by hardware. Convenience is provided primarily by software, though suitable hardware facilities help. Flexibility is provided entirely by software, and implies a complex mode of operation.

To obtain maximum efficiency it is necessary that the Central Processor of the computer shall be occupied for the greatest possible proportion of the time. This can be hindered by delays due to

(a) slow peripheral devices,

(b) program loading, and

(c) magnetic tape transfers.

It is only when these delays become large compared with the average execution time for a program that they assume any significance. Thus it is only fast computers, such as Atlas, that are affected.

Computers have for some time been substantially faster than the slower peripheral devices, that is paper tape and card equipment, etc. Once these devices became too slow to connect on-line, the most common development was the use of "satellite" equipment to buffer information to and from magnetic tape. At first, special-purpose convertors were used, but nowadays the use of a small satellite computer in a large installation is almost standard practice. The satellite system also helps to remove the second cause of delay, program loading, since a stream of programs can be assembled on a tape and executed in sequence. However, this gain in efficiency is at the price of some inconvenience, since it is not easy to execute programs other than in the order in which they appear on the tape. This means, for example, that special expedients are needed to run an urgent program of high priority, and also that any scheduling is the responsibility of the operator who loads programs on the satellite computer. The faster the machine, the more difficult it is, and the less efficient it becomes, to rely on an operator for this scheduling.

5056(10.64)

The alternative to the satellite system is the "self-contained time-sharing" system, in which peripheral devices are controlled by time-sharing on an interrupt basis, and the operation of the computer is controlled by a Supervisor program. Such a system allows reasonable scheduling to be done internally. It also reduces the load on the operators by the need for programs to be loaded as units; the system can accomplish the necessary sorting of multiple input streams into separate programs.

The third type of delay, magnetic tape transfers, only becomes significant for a machine of the "microsecond" class such as Atlas. To remove this source of inefficiency there is no alternative to time-sharing between programs in order to absorb the waiting time, and this implies more or less elaborate internal scheduling, since the jobs which are to be time-shared must be carefully selected. Effective scheduling can be done by a self-contained system, but not by a conventional satellite system.

Efficiency cannot be regarded as an end in itself. Equally important is the cost of obtaining it, since absolute efficiency is of little interest if it is too expensive to achieve. The true purpose of time-sharing is not solely to increase efficiency, but to increase the ratio of efficiency to cost, in other words to increase the "value for money" of the installation.

To compare the cost of the satellite system with that of the self-contained system, both the direct and the hidden costs must be taken into account. The direct costs of the satellite system include the cost of the satellite computer and the cost of the tape decks used on the main computer for the satellite tapes; the direct costs of the self-contained system include the cost of the percentage of the central processor time taken by the Supervisor program, the cost of the tape decks used by the Supervisor for input and output buffering, and the cost of the part of the core store used by the Supervisor for its own routines and for core store buffering. The hidden costs of the satellite system include the need for operators to change and transport magnetic tapes and the indirect costs due to the inflexibility of the operating system; the hidden costs of the self-contained system include the cost of preparing the complex Supervisor program.

In choosing a system for Atlas 2 both efficiency and cost in their widest senses have been considered. Satellite systems were rejected because of their inefficiency and inflexibility. The particular self-contained system which has been evolved provides the required efficiency and other benefits at less cost, although provision is made for the use of satellites when desirable (for example when there are remote stations, although even here an on-line data link system may be preferable). The potential of the chosen system is far greater than that of a satellite system, for with comparatively little extra equipment, it can make the computer easy to use, and it can perform the intelligent internal scheduling that makes time-sharing between programs economically advantageous. However, the system does not depend on time-sharing between programs, nor require it, when it is not the most economic solution.

Most existing supervisor or monitor systems use a fixed amount of core store and a fixed number of tape decks. This means that many installations do not obtain the best value from this equipment because their requirements from the system are such that some of the tape decks, or parts of the core store, are hardly ever used. It is impossible to meet all requirements in the most efficient manner with a fixed allocation of equipment. There are essentially four variables to take account of:

(a) type of jobs to be run,

(b) facilities to be provided,

(c) amount of core store to be used, and

(d) number of tape decks to be used.

If efficiency is ignored, these are independent variables; as soon as an efficiency constraint is imposed, the variables interact in a complicated manner, and a compromise "best" solution must be sought. The Atlas 2 Supervisor allows a wide range of choice for items (c) and (d); thus an installation with very large programs might restrict the Supervisor to, say, 10k of store and two tapes, whilst another installation with a very large throughput of small programs might obtain best efficiency from a Supervisor using 6 tapes and 25k of store.

The hardware configuration for a particular installation is chosen to meet either the maximum or the "normal" requirements there. If the installation processes only one type of job this means that it is fairly straightforward to achieve high efficiency, but if there is a wide variety of jobs then the hardware requirements of the object programs will vary. Many will not use all the nominally available core store or all the available tape decks, but there will be occasional jobs which require as much store or as many tape decks as possible. To prevent decks and core store being under-utilized, the Supervisor has been designed so that it automatically varies its requirements for tapes and core store dynamically so as to absorb any surplus capacity; thus while small jobs are being run it will expand

to use the available core store, but when a big job comes along it will automatically contract. Similarly spare tape decks are taken over but released when required; the changeover is smooth and rapid, often taking only a few seconds, during which the system continues to operate efficiently. By thus taking over surplus equipment, the Supervisor can increase efficiency by, for example, holding more input information and more Supervisor routines in core store so that there is no waiting time when these are required. This full utilization of equipment maximizes the "value for money" of the installation.

To make best use of the equipment a novel method of amalgamating a variable number of magnetic tapes into a semi-random access backing store has been devised. This is combined with an elaborate system of dynamic store allocation, which is assisted by the fact that all programs are relocatable, the relocation being done by hardware at run time.

THE PLAN 1.3

The following chapters present details of the various main parts and features of the Atlas 2 Supervisor. Chapters 2 to 6 present various aspects of the basic structure of the Supervisor, its own internal organization, the store organization and how the magnetic tapes and the slow peripherals are handled and driven, whilst Chapter 7 presents the Operating System. The Operating System is logically very largely separate from, and independent of, the basic structure of the Supervisor. The structure provides the framework without which no Operating System could operate, but the structure in no way dictates the Operating System. Thus, for example, an operating system could be provided, using the same basic structure, to implement purely serial non-time-sharing operation, or a simple satellite -type mode of operation. However, the structure provides facilities whereby a complex mode of self -contained time-shared operation is possible, and it is this mode which is being implemented and which is described here. The Operating System is, as it were, superimposed on the basic structure of the Supervisor, and can fairly easily be extended or modified if or when required.

In addition, the individual routines of the Operating System are largely independent of each other, so that it is possible to change the inner workings of a single routine and thereby perhaps considerably change the Operating System without any of the other routines being affected. For example, it is possible to change the algorithms affecting the Scheduling of jobs (that is, the decision as to which job should be executed when), or to change the rules affecting the priorities of store or backing store utilization, and thereby perhaps make the system more efficient for a specific situation, in each case without affecting any other part of the Operating System. Accordingly the System described in Chapter 7 should be regarded as a particular Operating System rather than the only possible Operating System which could be provided by the nature of Atlas 2. It is believed that the system described is an extremely powerful and flexible one, and one that should meet the requirements of all kinds of users. But it must be stressed that it is not built deeply into the structure of the Supervisor and is designed in such a way that not only does it have a very considerable amount of flexibility built-in but beyond that it is also fairly easy to extend or change it.

Chapter 2 describes the general organization of the Supervisor - how the various routines of the Supervisor are linked together, the system of priorities, the various levels of activity, and the general facilities. It describes how interrupts occur to the current program when a peripheral requires attention, how this leads to an interrupt routine, how these are queued when two or more interrupts occur more or less simultaneously, and how these are limited in length to ensure a quick response. It describes how longer Supervisor routines are requested by the interrupt routines or by extracode instructions, how these are queued, and how they are obeyed on extracode control. It also describes how object programs are effectively low-priority branches of the Supervisor, the rest of the Supervisor being normally dormant but taking priority when active. Finally it describes the error detection facilities provided within the Supervisor.

Chapter 3 describes the organization of the store - the various kinds of addressing available, the lockout system, how the Supervisor uses the store for its own purposes, and how the Supervisor optimizes the use of object program store areas. It describes how a system of relative addressing, which is interpreted dynamically by hardware at run time relative to a base set by the Supervisor, is used for object programs, and how an upper limit to the object program area is also checked by hardware, but how other modes of addressing, both absolute and unchecked relative, are available for the Supervisor. It describes how lockout is provided within a program area to protect magnetic tape and disc transfers, and how special tunnel-diode stores are used to speed up programs. It describes how the most important

basic routines of the Supervisor are stored permanently at the highest addressed end of the store, but how the remaining parts of the Supervisor are brought into any available parts of the store from the backing store as and when required. Finally it describes how the Supervisor takes advantage of the characteristics of programs to use for its own purposes parts of the store within the program areas.

Chapter 4 describes the magnetic tape system - the facilities available, how transfers are handled, and the checks provided. It describes how transfers, once initiated, proceed autonomously, and how a system of queuing within the Supervisor permits programs to continue after requests for transfers have been made. It describes how the system of using pre-addressed tape permits selective over -writing and off-channel searching. It describes how tapes are titled to prevent erroneous use and how tape failures are identified and isolated. It also describes how IBM-compatible $\frac{1}{2}$" tape is connected into the system.

Chapter 5 describes how a magnetic disc file may be used.

Chapter 6 describes the slow-peripheral system. It describes how transfers, once initiated, may proceed autonomously, and how the peripherals are started or stopped by the operator or by the Supervisor.

Chapter 7, describing the Operating System, introduces the concept of a "document", describes how a variable number of magnetic tapes may be used to provide a powerful and flexible backing-store system for the storage of input and output, called the Well, introduces the Job Description which must be provided by the programmer with each job and tells the Supervisor of the requirements of the job, describes the scheduling activities of the Supervisor, and describes the facilities provided for communication between the operator and the Supervisor and the various facilities provided by the Supervisor for making program development easy.

Finally, Chapter 8 summarizes the principal features of the Supervisor and of the operating system, and the final page lists the various publications describing aspects of Atlas 1 and Atlas 2 which are referred to in the text.

Chapter 2

THE ORGANIZATION OF THE SUPERVISOR

GENERAL 2.1

The central parts of the Supervisor, consisting of those parts without which no program can run, are stored at the highest addressed end of the store, in what is known as the "Fixed Area" (F.A.). Certain other routines which are usually but not always required and which for various reasons must be stored in fixed addresses are stored adjacent to the F.A. in what is known as the Auxiliary Fixed Area (A.F.A.). The A.F.A. is also used for the small double buffers which are filled or emptied by the slow peripherals. The remaining parts of the Supervisor, and the remainder of the input and output information, are normally stored on some backing store medium (tape or disc) and are brought into any vacant part of the store as and when they are required. Thus the F.A. is always non-available to object programs, the A.F.A. is normally non-available if the full facilities of the Supervisor are in use (e.g. time-sharing, use of all peripherals including magnetic tape, etc.), but can be relinquished if necessary for large programs, and most of the remainder of the store is normally available for object programs, it being usually required by the Supervisor only for short intervals of time, whenever, for example, a new job is fed into the system or the execution of a program is completed. A certain amount of the store will normally be needed to provide bufferage for input and output information, assuming that there is information to be fed into the computer, and that there is output awaiting dissemination and that it is normally necessary or desirable that the peripherals should be allowed to operate. All the store which is neither F.A. nor A.F.A. is called the Space-Routine Domain (S.R.D.), since its use is controlled by routines of the Supervisor called the Central Space Routines.

The Supervisor controls all those functions of the system that are not obtained merely by allowing the central processor to proceed with obeying an object program, or by allowing peripheral equipments to carry out their built-in operations. The Supervisor therefore becomes active on frequent occasions and for a variety of reasons - in fact whenever any part of the system requires attention from it. It becomes activated in several different ways. Firstly, it can be entered as a direct result of obeying an object program. Thus, a problem being executed calls for the Supervisor whenever it requests an action that is subject to control by the Supervisor, such as a request for transfer to or from magnetic tape; the Supervisor is also activated when an object program requires monitoring for any reason such as exponent or division overflow, or if store or time allocation is exceeded. Secondly, the Supervisor may be activated by various items of hardware which have completed their assigned tasks and require further attention, Thus, for example, 1" magnetic tapes call the Supervisor into action whenever the transfer of a 512-word block to or from the core store is completed; likewise, the input and output peripherals require attention when their assigned core store buffers have been emptied or filled. Lastly, certain failures of the central processor, store, and peripheral equipments call the Supervisor into action.

The central processor thus shares its time between these Supervisor activities and the execution of object programs, and the design of Atlas 2 and its Supervisor is such that there is mutual protection between object programs and all parts of the Supervisor. The Supervisor consists of many branches which are normally dormant, but which can be activated whenever required. The sequence in which the branches are activated is essentially random, being dictated by the course of an object program and the functioning of the peripheral equipments.

The principal branches of the Supervisor are organized into what are called Supervisor Extracode Routines (SERs). These are routines obeyed on extracode control. They have various priorities, but within these priorities they are executed in the order in which they are requested. They will be discussed in detail in Section 2.3.

The means by which peripheral equipments initiate Supervisor activity is by causing "interrupts" to the current program (which may be on main or extracode control and may be an object program or an SER). These interrupts cause Interrupt Routines to be obeyed on interrupt control; the interrupt routines perform any action which is immediately necessary and, normally, request the execution of an SER to take more elaborate action but which can wait its turn in the SER queues. The intention of the system is to keep the interrupt routines as short as possible since interrupt routines cannot themselves be interrupted, so that their maximum length imposes a limit on the response time to any high-speed peripherals which might be attached; the interrupt routines perform only such actions as must be taken immediately the interrupt occurs and all other necessary action is taken in SERs. As a counterpart to this intention, the hardware has been designed so that very little action is essential within the interrupt routines, and if any necessary action within an SER cannot be taken even within the normal expected time-scale (for example because of a central processor fault) then the peripherals will automatically stop in such a way that no harm is done and no information lost. Thus the system is designed to be "Fail Safe". The Interrupt Routines are discussed in more detail in Section 2.2.

THE INTERRUPT ROUTINES 2.2

The Interrupt Routines are the most rapidly activated parts of the Supervisor. When a peripheral equipment requires attention, for example, an interrupt flip-flop (called a Look-At-Me or L.A.M.) is set which is available to the central processor as a digit in the V-store; a separate L.A.M. is provided for each reason for interruption. If an L.A.M. is set and interruptions are not inhibited, then before the next instruction is started, control is switched to interrupt control and a jump occurs to one of eight fixed addresses in the Fixed Area (according to the priority of the interruption). Further interruptions are inhibited until control reverts to main or extracode control. Under interrupt control, the Supervisor instructions at the eight fixed addresses detect which L.A.M. has been set and initiate an appropriate interrupt routine in the Supervisor. If more than one L.A.M. is set, that of the highest priority is dealt with first, the priority being built-in corresponding to the urgency of the action required. By the use of special hardware attached to one of the B-registers, B123, the source of any interruption can be determined as a result of obeying two or four instructions.

The interrupt routines so entered deal with the immediate cause of the particular interrupt, and take any action immediately necessary. For example, when one half of the 160-character core store buffer assigned to a card reader has been filled, the appropriate L.A.M. is set and the "Card Reader Interrupt Routine" is entered. This requests an SER to deal with the characters contained in the buffer, arranges that if this SER has not been completed by the time the other half-buffer is filled then the reader will automatically stop (for details see Section 6.1) so that the system is "Fail Safe", and puts out the L.A.M. The card reader meanwhile proceeds to read the next card into the other half of the double -buffer. Separate interrupt routines are provided to deal with each type of peripheral equipment, including magnetic tapes and discs. The interrupt technique is also employed to deal with certain exceptional situations which occur when the central processor cannot itself deal adequately with a problem under execution, for example when an overflow occurs. There are therefore L.A.M.'s and interrupt routines to deal with such cases. Further routines are provided to deal with interruptions due to detected computer faults.

During the course of an interrupt routine further interruptions are inhibited. Normally, at the completion of an interrupt routine, control returns to the point in main or extracode control which was current before the interruption. However, there are two circumstances in which this does not, at any rate immediately, occur. The first is when there are further L.A.M.'s set; in this case special hardware ensures that a further jump under interrupt control to one of the eight fixed addresses occurs before any further instructions are obeyed under main or extracode control. The second is when the interrupt routine just completed has requested an SER and the current main or extracode program is not a Supervisor routine. In this case a part of the Supervisor known as the Co-ordinator is activated, and this arranges for the current value of extracode control (B126) to be preserved, for B126 to be set to the entry address of the required SER, and for extracode control to become operative when the current interrupt routine or routines are completed. If the interrupt routine requests an SER and

5056(10.64)

the current program before the interruption was a Supervisor routine, then the required SER is queued. This is discussed in more detail in the next section.

In order to avoid interference with object programs or Supervisor routines, interrupt routines use only restricted parts of the core store, and only restricted parts of the central processor, namely the interrupt control register (B125), B123, and B113 to B118 inclusive. No lock-out is imposed on these B-lines, but interrupt routines make no assumptions concerning the original contents of the B-lines, and hence, at worst, erroneous use of interrupt B-lines by an object program can only result in erroneous functioning of that particular program. Switching of control to and from an interrupt routine is rapid, since no preservation or resetting of working registers in required.

SUPERVISOR EXTRACODE ROUTINES 2.3

Supervisor Extracode Routines (SERs) form the principal "Branches" of the Supervisor. They are activated either by interrupt routines or by extracode instructions in an object program. They operate under extracode control, the extracode control of any current object program being preserved and subsequently restored. Like the interrupt routines, they use private B-lines, in this case B-lines 100 to 112 inclusive; if any other B-lines are required, the SERs themselves preserve and subsequently restore the contents of such registers. The SERs thus apply mutual protection between themselves and an object program.

These branches of the Supervisor may be activated at random intervals. They can moreover be interrupted by interrupt routines, which may in turn initiate further SERs. It is thus possible for several SERs to be activated at the same time, in the same way as it is possible for several interrupt flip -flops to be set at the same time. Although several SERs may be activated, obviously not more than one can be obeyed at any one moment; the rest are queued awaiting execution. This matter is organized by a part of the Supervisor called the "Co-ordinator". Activation of an SER always occurs via the Co-ordinator, which arranges that any SER in progress is not interrupted by any other SERs. As these are activated, they are recorded by the Co-ordinator in chain-lists, and an entry is extracted from one of these lists whenever an SER ends or halts itself. Once started, an SER is always allowed to continue if it can; a high priority SER does not "interrupt" a low priority SER but is entered only on conclusion or halting of the current SER. The Co-ordinator has the rôle of the program equivalent of the "Inhibit Interrupts Flip-Flop", the lists of activated SERs being the equivalent of the setting of several interrupt flip-flops. The two major differences are that no time limit is placed on an SER, and that an SER may halt itself for various reasons; this is in contrast to interrupt routines, which observe a time limit and are never halted.

In order that the activity of each branch of the computing system be maintained at the highest possible level, the SERs awaiting execution are held in four distinct lists. Within each list, the routines are normally obeyed in the order in which they were activated, but the lists are assigned priorities, so that the top priority list is emptied before entries are extracted from the next list. The top priority list holds routines initiated by high-speed devices, and also routines entered as a result of computer failures such as core store parity. The second list holds routines arising from magnetic tape interruptions, and the third holds routines arising from peripheral interruptions. The lowest priority list contains one entry for each object program currently under execution, and entry to an SER through an extracode instruction in an object program is recorded in the list. On completion of an SER, the Co-ordinator selects for execution the first activated SER in the highest priority list.

The central processor is not necessarily fully occupied during the course of an SER. The routine may, for example, require the transfer of a block of information from the discs to the core store, in which case it is halted until the disc transfer is completed. Furthermore, there may be no space available for extending the chain-list of requests for disc transfers (see Chapter 5), in which case the SER making the request must be halted. When an SER is halted for this or similar reasons, it is placed in one of several "Halt Lists" according to the reason for halting, and the next highest priority SER which is free to proceed is activated by the Co-ordinator. Before an SER is halted, a restart point is specified. A halted routine is made free to proceed when the cause of the halt has been removed - for example by the SER which controls disc transfers and the extraction of entries from the disc queue. There are thus lists for holding SERs awaiting execution and lists for halted SERs; interrupt routines are written in such a way that the number of such SERs activated at any one time is limited to one per object program and one or two per interrupt flip-flop, depending upon the particular features

of each interrupt routine. When an SER is finally concluded, as distinct from halted, it is removed from the lists and becomes dormant again.

Although SERs originate in many cases as routines to control peripheral equipment, magnetic tapes and discs, it should not be supposed that this is the sole function of these routines. Entrances to SERs from interrupt routines or from extracode instructions in an object program initiate routines which control the entire operation of the computing system, including the transfer of information between the store and the peripherals, communication with the operators and the engineers, the initiation, termination and, where necessary, monitoring of object programs, the monitoring of central processor and peripheral failures, the execution of test programs and the accumulation of logging information. Each branch of Supervisory activity is composed of a series of SERs, each one activated by an object program or an interrupt routine and terminated usually by initiating a peripheral transfer or by changing the status of an SER list or object program list. The most frequently used routines are in the F.A. and A.F.A.; routines required less frequently are held on magnetic tape or discs and are transferred to core store when required. Supervisor routines in core store and backing store are protected from interference by object programs by use of hardware lock-out and by the backing store organization routines of the Supervisor respectively.

EXTENDED INTERRUPT ROUTINES 2.4

Besides Interrupt Routines and SERs there is a third category of Supervisor Routines known as Extended Interrupt Routines. These come logically between Interrupt Routines and SERs, having some of the characteristics of each. SERs form the principal branches of the Supervisor. Every time an SER is requested, initiated, halted or terminated, several instructions must be obeyed in the Co-ordinator. One of the main reasons why these instructions are needed is because SERs can be halted, so that it is necessary for the Co-ordinator to preserve a restart point in case the SER is halted. There are certain SERs which, as is known when they are written, will never give rise to a halt, and moreover are of known maximum duration. These routines can then be written to more restrictive rules than SERs, and if the maximum permitted length of interrupt routines for the peripheral devices currently activated is greater than the known maximum length of the SER, then the Co-ordinator arranges to obey the routine with interrupts inhibited, so that it cannot be interrupted and no preservation of registers is required. These routines are called E.I.R.'s. If the maximum permitted length of interrupt routines for the peripheral devices currently activated is less than the known maximum length of the E.I.R., then it is obeyed as an ordinary SER, i.e. with interrupts permitted. The choice as to how to run each E.I.R. is made dynamically by the Supervisor according to the peripherals activated. The paths through the Co-ordinator for E.I.R.'s are faster than those for SERs.

OBJECT PROGRAMS 2.5

The function of all Supervisor activity is in general terms to organize the progress of problems through the computer with the minimum possible delay whilst maintaining a high level of activity of all possible parts of the system. Object programs are initiated by SERs, which insert them into the object program list; they are subsequently activated by the Co-ordinator routine, effectively as branches of lower priority than any SER. Although object programs are thus logically subprograms of the Supervisor, they may function for long periods using the computer facilities to the full without reference to the Supervisor. For this reason the Supervisor may be regarded as normally dormant, and activated, using the central processor, for only a small proportion of the available time.

In order to allow object programs to function with the minimum of program supervision, they are not permitted to use extracode control or interrupt control directly. This enables protection of main programs and Supervisor programs to be enforced by hardware. Object programs use the main control register, B127, and are therefore forbidden access to the V-store, the Supervisor and other object programs. Any attempt to refer to any of these is detected either by hardware or software and causes entrance to the Supervisor.

Access to the peripherals, the V-store generally, and the Supervisor routines, is only obtained indirectly by use of extracode instructions, which cause control to be switched to extracode control and entry to be made to one of a possible maximum of 512 extracode routines within the Supervisor. The extracode routines form simple extensions of the basic order code, and also provide specific entry to Supervisor routines to control the transfer of information to and from the core store and to carry out necessary organization. Such specific entrances to the Supervisor maintain complete protection of the object programs. Protection of magnetic tapes, peripheral input and output data, and magnetic disc regions is obtained by the use, in extracode functions, of logical tape, disc and data numbers which the Supervisor identifies for each program with absolute tape deck numbers and disc regions and actual information streams, by means of tables within the Supervisor. The program specifies the name of a magnetic tape or information stream or the amount of disc storage required and specifies logical numbers for these, and the Supervisor allocates actual tape decks to the tapes and actual disc areas to the logical ones specified.

An object program is halted (by SERs) whenever access is required to a block of information not immediately available in the core store. This occurs whenever the block is involved in a magnetic tape or disc transfer. In such a case the program is halted until the transfer is completed. In the case of information involved in peripheral transfers, such as input data or output results, the Supervisor buffers the information in core and backing store, and "direct" control of a peripheral equipment by an object program is not allowed. In this way, immobilization of large sections of store whilst a program awaits a peripheral transfer can be avoided. A program may however call directly for transfers involving magnetic tapes or discs by use of extracode functions, which cause entrance to the relevant Supervisor routines. Queues of such requests are held by the Supervisor, in order to allow the object program to continue and to achieve the fullest possible overlap between tape and disc transfers and the execution of an object program.

Whilst one program is halted, for example awaiting completion of a magnetic tape transfer, the Co-ordinator routine switches control to the next program in the object program list which is free to proceed. In order to maintain full protection, it is necessary to preserve and recover the contents of working registers common to all programs, such as the B-lines and the accumulator, and to change the core store regions locked out. The SER to perform this switching from one object program to another occupies the central processor for about 800 microseconds, in contrast to the time of around 30 microseconds to enter and return from an SER and even less to switch to and from an interrupt routine. It is therefore obvious that the most efficient method of obtaining the maximum overlap between input and output, magnetic tape and disc transfers, and computing is to reduce to a minimum the number of changes between object programs and to utilize to the full the rapid switching to and from Supervisor routines. The method of achieving this in practice is discussed in Chapter 7

Compilation of programs is treated by the Supervisor as a special case of the execution of object programs, the compiler comprising an object program which treats the source language program as input data and, with this data, "computes" the compiled program. Facilities are provided to permit either "load and go" or the separation of the compilation and execution phases, thus permitting batch -compiling and batch-execution when appropriate.

ERROR CONDITIONS 2.6

In addition to programmed entrances to the Supervisor, entrance may also be made in the event of certain detectable errors arising during the course of execution of a problem. A variety of program faults may occur and be detected by hardware, by programmed checks in extracodes, and in the Supervisor. Hardware causes entry to the Supervisor by means of interrupt flip-flops in the event of overflow of the accumulator, use of an unassigned instruction, or reference to the V-store or those parts of the core store reserved for the Supervisor or for other programs. Extracode routines detect errors in the range of the argument in square root, logarithm and arcsin/cos instructions. In the extracodes referring to peripheral equipment or magnetic tapes, a check is included that the logical number of the equipment has been previously defined, and in extracodes for discs and working space tapes that a requested transfer is not outside the logical areas assigned to the program.

The Supervisor also detects any error in connection with the use of store and any attempt to exceed allocated execution time or amount of output. All problems must supply information to the Supervisor on the amount of store required, the amount of output, and the expected duration of execution. This information is supplied before the program is compiled, or may be deduced after compilation. The Supervisor sets a lock-out register to the upper bound of the store area assigned to a program and an interrupt occurs if any attempt is made to refer to an address above this. In addition, an interrupt flip-flop is set by a clock at intervals of a pre-set multiple of 10 milliseconds. This causes entrance to the Supervisor which enables a program to be "monitored" to ensure that the allocated time has not expired. Also, as output is generated by the program and handed to the Supervisor by extracodes, a check is performed to see that the total output does not exceed that assigned. The purpose of these last two checks is not so much to stop short a program which may be functioning correctly but has nevertheless exceeded the programmer's estimates, as to terminate a program which is malfunctioning, e.g. by becoming stuck in a computing or output loop. With a computer of the speed of Atlas, it is possible for many millions of instructions to be obeyed in a short space of time before any operator becomes aware that anything is amiss. Moreover, with the facility on Atlas for executing more than one program simultaneously and in view of the fact that input and output are handled in an indirect way, it is not possible to tell easily which programs are in execution and to what extent, and so a program stuck in a loop might go unnoticed for a very considerable time. Likewise, a vast amount of nonsense could be printed on a high-speed line-printer, before a program stuck in an output loop was noticed.

The clock interrupts are also used to initiate routines to carry out regular timed operations such as logging of computer performance and initiation of routine test programs. These test programs are also called in to check the performance of the computer, whenever all the current object programs are halted and unable to proceed and no SERs are active, thereby taking advantage of time which would otherwise be wasted in an idling loop.

The action taken by the Supervisor when a program "error" is detected depends upon the conditions previously set up by the programmer and the nature of the error. Certain errors may be individually "trapped", causing return of control to a preset address; alternatively a private monitor sequence may be entered before the program is suspended to enable a program or a compiler to obtain diagnostic printing; if neither of these is specified, some standard information is printed by the Supervisor and the program is suspended and may be dumped in backing store to allow storage space for another program whilst permitting recovery of the program at a later time for continuation, correction or further diagnostic printing if required.

Chapter 3

STORE ORGANIZATION

STORE ADDRESSING 3.1

The core store of Atlas 2 is provided with a method of automatic "re-location" or "relativizing" whereby the contents of a special register (called r1) specify a base address which is normally added automatically to all store references whether for operands or instructions. Thus each object program is written, compiled, and stored in the computer as if it occupies addresses from zero upwards, but the Supervisor may place the program anywhere (almost - see below) in the store, and set r1 appropriately, and the program will be executed correctly. Because the adding in of the contents of r1 is done by hardware as the store references are made, rather than added explicitly into any store references within a program, it is possible for the Supervisor to move the program from one part of the store to another at any time without having to change the internal form at all. This is used to enable the Supervisor to obtain large contiguous areas of store for new programs as old ones are finished, and is also used if programs request more store or free store during the course of execution.

A second special register (called r2) is used to specify the upper bound of the area assigned to an object program. If any attempt is made in an object program to refer to a (relative) address greater than the contents of r2, than an automatic interrupt occurs. Thus, by means of r1 and r2, the program is prevented from referring to parts of the store outside its own assigned area.

Two further registers (called r3 and r4) are used to lock out parts of the store within an object program area, for example when a magnetic tape or disc transfer has been requested or is in progress, to prevent access to the store region involved until the transfer is complete. Specifically, if m is a (relative) store address, then an interrupt, leading to the program being halted, will occur if a store reference is made to an address where r3> m> r4.

Strictly r1 is not *added* to the program relative address, it is "or-ed". This is to make the operation fast, since it occurs with every store reference. Thus in order to ensure that the correct result is obtained the highest program relative address must be less than the least significant binary digit in r1. Thus if a program requires an area of store of length n words, where $2^k > n > 2^{k-1}$, then r1 may be set to any address which is a multiple of 2^k. This is organized automatically by the Supervisor.

In addition to the normal "relative" addressing used by object programs, it is also possible to address the store "absolutely". The type of addressing required is indicated by the three most significant binary digits of the address. If these are zero then the addressing is the normal program relative, and the 17 store address bits (see Chapter 1) are or-ed with r1 to obtain an absolute store address. If, however, the three most significant bits are ones (octal-7), then an absolute address is indicated and the 17 store address bits are interpreted directly. If an attempt is made to make a store reference of this type when main control is active then an interrupt occurs, but the reference is permitted when extracode or interrupt controls are active. Thus an object program cannot use absolute addressing, but the extracode routines, the Supervisor, and the interrupt routines can, and can thus obtain access to the whole of the store including their own operating regions and the object program regions. When an octal-7 address is used the checks with r2, r3 and r4 are not performed.

There is also a further type of addressing (octal-3) in which the relativizing with r1 is performed, but the checks with r2, r3 and r4 are omitted. This kind of addressing is also only permitted on extracode and interrupt controls, and is used when the Supervisor has routines or tables which can be moved around in the store (i.e. which are not within the F.A. or the A.F.A.). Thus in order to gain access to these regions, for example in an SER, the Supervisor has only to reset r1 and may

leave r2, r3 and r4 as set for the current object program, thereby saving instructions obeyed on entry to and exit from the SER.

Octal-6 and octal-5 addresses are used for the V-store, and are again only acceptable when used on interrupt or extracode controls.

Octal-1 and octal-2 addresses are used for a special facility available to the object program and called the "Window" facility. Addresses of this type permit selective lock-out of fixed regions of an object program's store area and can be used for example for "double buffering" of long magnetic tape transfers whilst leaving r3 and r4 free for lock-outs associated with other transfers elsewhere in the store.

Finally, octal-4 addresses are not normally used on Atlas 2, with the exception of the special address where the remaining bits are all zero. On Atlas 1 this address is used as standard whenever a "dummy address" is required for an instruction; this causes (on Atlas 1) a reference to the *Fixed* Store, and thereby prevents any slowing down of the program due to an unnecessary main store reference. This facility is carried over to Atlas 2 to retain compatibility, although no fixed store exists. Thus this special address is simply treated as a dummy by the hardware. A reference to any other address beginning octal-4 will cause the program to be monitored.

THE TUNNEL-DIODE STORES 3.2

In addition to the main core store and the V-store, there are two small stores constructed of tunnel diodes, called the Fast Operand Store and the Slave Store.

The Fast Operand Store consists of 8 48-bit registers addressed (by octal-0 type addresses) as the first eight words of an object program area. These have much faster access than the main core store and can, therefore, be used by programs for frequently used operands to speed up execution.

The Slave Store consists of 32 48-bit registers with an additional 12-bit register associated with each. The Slave Store cannot be addressed directly, but operates entirely automatically and continually. Whenever an instruction is extracted from main store for execution, then while it is being executed it is also being copied into an appropriate register of the slave store, the appropriate one being determined by its position modulo 32 in the main store. At the same time the number of the block of 32 instructions from which it was taken is placed in the associated 12-bit register. Whenever an instruction is requested, the Slave Store is automatically consulted by hardware to see if the 12-bit register in the appropriate position contains the number of the 32-word block containing the required instruction, and, if it does, then the instruction is taken from there and no main store access is made; otherwise a main store access is made in the usual way. Since the Slave Store has a very much shorter access time than the main store, this permits the speeding up of loops of program. Full benefit of the Slave Store is obtained for loops of 32 instructions or less, but some benefit is obtained for all loops of less than 64 instructions. The Slave Store operates only for instructions and not for operands, although if an instruction is modified by program, the contents of the Slave Store are also modified if necessary.

The Slave Store is not used by instructions obeyed on interrupt control. This is because interrupt routines do not normally contain loops, and therefore could not benefit from use of the Slave Store. If, however, interrupt instructions did overwrite object program instructions in the Slave Store, then, when control returned to the object program, and if the object program was in a loop which would normally be operating in the Slave Store, the next cycle of instructions in the loop would have to be taken from the main store again, thus increasing the execution time for that cycle. Thus the interrupt routine would not only not benefit from the Slave Store but would actually cause the object program following it to be slower than otherwise, which is obviously undesirable.

THE SPACE-ROUTINE DOMAIN 3.3

All the store which is not within either the Fixed Area or the Auxiliary Fixed Area forms the Space-Routine Domain (S.R.D.), which is administered by the Central Space Routines of the Supervisor. The S.R.D. is divided into Sections and Blocks.

Sections consist of n consecutive words of store, where n is 512 or a multiple of 512, starting at address 0, 2^m, or a multiple of 2^m, where $2^m \geq n$ (i.e. r1 can be set to it). Octal-0, octal-1, octal-2 and octal-3 addresses are normally used within Sections.

Blocks consist of 512 consecutive words of store, starting at address 0, 512, or a multiple of 512. A Block can be either a particular case of a Section (i.e. a one-block Section) or a subdivision of a larger Section. Octal-7 addresses are in some cases used for one-block Sections.

One-block Sections can be divided into Blocklets. A Blocklet consists of 64 consecutive words, starting at 0, 64, or a multiple of 64. Blocklets are intended for data and tables referred to by octal-7 addresses, and for scattered storage of Supervisor routines which must be reconstructed into a section before use, and chains of blocklets are used as buffers for input and output.

Supervisor routines request and relinquish space by entering closed subroutines which form part of the Central Space Routines. The Central Space Routines grant requests firstly from free space if available, secondly by overwriting material of which another copy exists in backing store, and thirdly by dumping material in backing store.

In order to meet a request for a Block or Section, the Central Space Routines may shift material around the Space-Routine Domain. *Any* material in the domain may be shifted though if it is "locked" for a tape or disc transfer the shift is delayed. However each request carries a "Requesting Priority", and each occupier of space carries a "Keeping Priority" (both priorities being dynamically variable), and material is only *removed* from core to make way for a Requesting Priority greater than its own Keeping Priority. If it is necessary to overwrite or dump material, then that with the lowest Keeping Priority is chosen.

All requests are labelled, in their entry specification, as "haltable" or "unhaltable". If the request cannot be met immediately, then if it is haltable the requesting routine is delayed and the request queued, and if the request is unhaltable the requesting routine is simply told that the request cannot be met. Queued requests are met in order of descending Requesting Priority. A third form of request is available whereby the return is immediate even when the request has to be queued; this is used mainly to anticipate the arrival of material retrieved from backing store.

All occupiers of Sections, Blocks and Blocklets are pointed at, either directly or via a chain of pointers, by pointers in Fixed Area. Whenever Space Routines shift or remove material they reset the relevant (chain of) pointers. If also r1 is pointing at the material, they reset r1.

Thus users of space need not normally be concerned with the whereabouts of material, provided that when they use octal-7 addresses for material outside F.A. and A.F.A. they pick up the address via the (chain of) pointers, and, if the material has a Keeping Priority less than the maximum, they check for "not pointing" (i.e. not in core). In practice they should rarely have to follow down a long chain of pointers except when deliberately sorting material.

There are two main types of Section:

(a) Object-Program Sections and Gaps, and

(b) Supervisor Sections.

Object-Program Sections and Gaps are complete, i.e. they extend to a multiple of 512. Note that Gaps are defined as empty Sections, so that they are units of space which could be used for storing object programs; for example free space from 3k to (10k-1) would be regarded as three Gaps, namely 3k to (4k-1), 4k to (8k-1) and 8k to (10k-1). Supervisor Sections are liable to be incomplete. For example, a one-block Supervisor Section might contain routines and their working space in addresses 0 to 300, in which case 320 to 383, 384 to 447, and 448 to 511 would be made available as blocklets for general Supervisor use. Such blocklets are always at the high-address end of a Section. A special case of a Supervisor Section is a block consisting of eight blocklets.

Each *Chapter* of the Supervisor (that is, unit of Supervisor which requires a Section) has a *Base* in Fixed Area, which records among other things its position in backing store, for retrieval, and its required setting for r1. Central routines in F.A. are available for (initiating retrieval and) setting r1 whenever a routine in a Chapter is required.

When compilation or loading of a program is started, the compiler or loader concerned is placed in the store immediately above the highest address required by the programmer, and r2 is set to an appropriate address beyond the end of the compiler, so that the compiler may operate on main control. Normally, the compiled program is placed serially from the lowest addressed end of the program area, so that during compilation the area between this and the compiler is not used, this area decreasing during the course of compilation.

To take account, and advantage, of this fact, a technique is adopted whereby the Supervisor uses this area as part of the Well to store the input document which is the source of the instructions which the compiler is compiling. This is done by making use of the lockout registers r3 and r4. During compilation r4 is set to a suitable address just above that into which the instructions are being compiled, and r3 is set to an address just below the start of the compiler. Thus this area *cannot* be referred to by main control and may be used freely by the Supervisor. The Supervisor then brings down as much as possible of the input document into this area, so that information is immediately available for the compiler without waiting for transfers from backing store, and without using any additional store area. Should the compiler attempt to refer to this area, either because the instructions are not being compiled in a purely serial fashion or because they have completely filled the area below r4, then the Supervisor adjusts the setting of r4 and allows the program to continue. Normally the information in the area thus handed over will already have been used, and there is no wastage, but if this is not the case then the information is simply brought down from backing store again to another area of core store.

At the beginning of execution of the object program, the program is not initially given the whole of the store area as requested in the Job Description, although the Supervisor ensures that it is available if required. This is because:

(a) object programs also tend to start by using the low addresses first and then extend upwards, and

(b) the programmer's estimate of his store requirements tends to be a generous upper limit and the program may never in fact use the whole amount.

The Supervisor therefore takes advantage of this by initially setting r2 to a value less than that requested by the programmer, and using the remainder of the area for its own purposes, i.e. either for the storage of Supervisor routines or tables, or as part of the Well. Should the program attempt to refer to an address above the limit set by r2, then an interrupt will occur and entry will be made to the Supervisor which will check that the address referred to is not outside the limit specified by the programmer, and, provided that it is not, move r2 just enough to allow the program to continue, continuing to use the smaller remainder of the area for its own purposes. If the address is outside the limit specified by the programmer then, of course, the program is monitored. It should be noted however that provision is made for the program to request or relinquish space explicitly by extracode, so that the upper limit specified in the Job Description does not necessarily apply throughout the duration of the program.

Chapter 4

MAGNETIC TAPE

ONE - INCH TAPE 4.1

Facilities 4.1.1

The standard tape system for Atlas 2 uses the Potter MT120 tape decks with 1" tape. There are sixteen tracks across the tape - twelve information tracks, two clock tracks, and two tracks used for reference purposes. The tapes are used in a fixed-block, pre-addressed mode. Information is stored on tape in blocks of 512 48-bit words, followed by a 24-bit checksum with end-around-carry. Each block is preceded by a block address and block marker, and terminated by a block marker; the leading block address is sequential along the tape and the trailing block address is always zero. Tapes are tested and pre-addressed by special routines before being put into use, and the fixed position of the addresses permits selective overwriting and simple omission of faulty patches on the tape. Blocks can be read when the tape is moving either in the forward or reverse direction, but writing is only possible when the tape is moving forwards. The double read and write head is used to check-read when writing to the tape. When not operating the tape stops with the read head midway between blocks.

There may be either two or four magnetic tape channels on Atlas 2. Decks may be attached directly to the channels or via switching units. The standard switching unit permits up to eight decks to be connected to a pair of channels, each deck being capable of operating through either channel, but it is also possible to have a switching unit for up to eight decks attached to a single channel, or to have a two-channel switching unit to which more than eight decks can be attached. Each channel can operate simultaneously, controlling one read, write or short search operation. In addition, wind, rewind, and searches can proceed, once initiated, independently, i.e. without being connected to any channel. Transfer operations are performed directly between the Tape Co-ordinator and the core store, and thus the execution of programs is not interrupted by them. The instruction execution rate is affected by a tape transfer only when the central processor and the Tape Co-ordinator request access to the same stack of core store at the same instant or within one core store cycle of each other. In such cases the program hesitates momentarily.

The total area of core store involved in all tape transfers proceeding at any instant is locked out from reference by any program by means of the special registers r3 and r4, or by use of the window facilities.

A block on the tape always contains 512 words. This block may either be transferred between a continuous area of core store or may be split up into a number of groups of 64 words, called "blocklets", scattered throughout the core store. The transfer of a block is controlled by "code-words": there are 8 for each channel, one for each blocklet. The codewords are in fixed positions in the store, and, once a block transfer has been initiated, are accessed autonomously by the Tape Control as each blocklet is transferred. The mode of the transfer (i.e. read or write) is specified by writing to the V-store when the transfer is initiated, but the code-words specify the areas of core store involved and whether each particular blocklet transfer is to be performed or suppressed (thus permitting selective blocklet reading and writing) or whether a single continuous 512-word block transfer is required. The scattered blocklet mode is only available to Supervisor routines (see Chapter 6), only the continuous block mode being available to the object programs by extracode.

Although the tape transfers, once initiated, proceed autonomously, an interrupt is caused when a block transfer is completed, and the interrupt routine entered notifies the Supervisor that the transfer is complete, so that the Supervisor can take any appropriate action (see 4.1.2. overleaf). Interrupts in

fact occur whenever a leading or trailing block marker is encountered, and the interrupt routine determines which kind of block marker it was, checks the block number, and takes note of whether the check-sum was correct. This interrupt routine is called the Block Mark Interrupt Routine.

Organization 4.1.2

Magnetic tape operations are initiated by entrance to the tape control routines in the Supervisor either from extracode instructions in the case of tape operations required by object programs, or from SERs in the case of tape operations required by the Supervisor for its own purposes. From a table maintained by the Supervisor the logical tape number is converted to the actual mechanism number, and the request either implemented immediately if possible, or queued by the Supervisor. Control normally returns to the program immediately the transfer has been initiated or queued and it is then free to proceed. A tape instruction may request the transfer of one or several blocks, and the Supervisor immediately ensures that the store areas concerned are "locked out" by extending the area covered by r3 and r4 if necessary. The Supervisor also notes the actual areas involved in the transfers. If any part of the store area covered by the transfer request is already involved in another transfer which is not yet completed, the program initiating the request is halted. Similarly, the program is halted if the queue of tape requests is full. A program may thus request a number of tape transfers without being halted, allowing virtually the maximum possible overlap between the central processor and the tape mechanisms, during execution of a program.

Separate queues are used for the requests for each tape mechanism, and a "points system" is used to assign

(a) a priority to each request, and

(b) a priority to each mechanism.

This latter priority is the sum of the priorities for all requests relating to that mechanism. When the last block address involved in an operation has been read, the mechanism with the currently highest priority which can be connected to that channel is identified, and the highest priority request for that mechanism is located. If it involves the same mechanism as the current request, and if tape motion is in the same direction, then the Supervisor takes action to ensure that the tape is not stopped but continues to move, implementing the new request. Thus, whenever possible, stopping the tape is avoided. As each transfer is completed, the Supervisor adjusts the limits of lock-out covered by r3 and r4, and any object program previously halted through reference to part of any area thus unlocked is made free to proceed.

Searching for specified blocks off-channel is possible with Atlas one-inch tape because the tape is pre-addressed. When a search is required, then, if it is short, it is performed off-channel by synchronizing the tape movement with that of an on-channel transfer. If it is long (say over 200 blocks), it is performed at high speed off-channel, and the movement is timed by means of the clock interrupt routine, until the tape is roughly in position, when the tape is brought back to normal speed, the next block address is read, and then the search is completed as for short searches.

The Title Block 4.1.3

The first block on each tape is reserved for use by the Supervisor, and access to information in this block by an object program is through special extracode instructions only. This block contains the serial number of the tape, a *name* for the tape and a record of any faulty blocks and of the number of block labels which have been put on the tape. The serial number and the name together are called the *title* of the tape.

The serial number is recorded on the tape when it is initially addressed and is normally never changed during the life of the tape, although it can be changed either during any re-addressing run which may be desired or required, or by an operator message.

The name is assigned by the programmer. When a tape is first addressed, the name FREE is placed in the title block by the Supervisor. Whenever a programmer acquires a fresh magnetic tape he must specify to the Supervisor not only the serial number of the tape, but also what name he wishes to be written to the tape. Whenever he subsequently uses the tape he must again specify the name which is then checked by the Supervisor. The use and checking of names of magnetic tapes guards against

5056(10.64)

erroneous loading of tapes by operators and erroneous requesting of tapes by programmers. The programmer may, if he desires, change the name of a tape during the course of a program, thereby, for example, ensuring that a subsequent program cannot use the tape until the current one has been successfully completed.

The Supervisor maintains, in the title block, a list of faulty blocks, and ensures that, in any reading or writing operations, these blocks are ignored and the subsequent block numbers interpreted accordingly, so that the program "sees" a continuous sequence of block numbers, without the tape having to be re-addressed. This list is in two parts, the first of blocks which are to be ignored, and the second of blocks discovered to be faulty since the tape was last made FREE. Thus the programmer may continue to use the tape after new faulty blocks have been discovered, but, when he next releases the tape, the Supervisor will incorporate the second list into the first and thus implicitly re-address the tape before making it available for re-use.

Finally the title block records the number of block labels which have been put on to tape. Normally a full-length tape contains 5000 blocks, providing $2\frac{1}{2}$ million words of storage, but if, for any reason, a shorter tape is used or less than the full number of blocks have been pre-addressed, then this is noted in the title block so that the programmer can know the number of available blocks.

Magnetic Tape Failures 4.1.4

All failures detected by the Block Mark Interrupt Routine cause the tape to be stopped when possible, and then entry to be made to tape fault routines in the Supervisor. These routines are SERs designed to minimize the immediate effect on the central processor of isolated faults in the tape system, to inform maintenance engineers of any faults, to diagnose as far as possible the source of a failure, and, if possible, to permit the program using the tape involved to continue. Failures may be caused by the tape, the tape mechanism, the channel electronics, or the electronics common to all channels. In attempting, for example, to isolate the cause and the effect of an apparent check-sum failure on reading a block from tape, these SERs will take action such as to cause repeated attempts to read the block, including attempts to read with reduced bias level, or to cause attempts to read the block using the other channel to which the deck is connected. If recovery is successful, the program continues, but the operator is notified, so that action can be taken if these failures occur frequently. Other faults are monitored in a similar manner, and, throughout, the operators, and thence the engineers, are informed of any detected faults. Provision is made for the programs using the tape to "trap" persistent tape errors and thereby to take action suitable to the particular problem, which may be more straightforward and efficient than the standard Supervisor action.

Addressing of new tapes, and re-addressing of old tapes, are carried out on the computer by Supervisor routines called in by the operator, and are operations which can be performed whilst normal programs are running, including the use of tape decks and tape channels other than those actually involved in the addressing. A tape deck is switched to "addressing mode", which prohibits the normal tape operations but permits the Supervisor to write freely in all twelve tracks on the tape. Thus the block marks and block addresses can be written and spaced correctly on the tape. When a tape is addressed, addresses are written sequentially along the tape and the area between leading and trailing block addresses is checked by writing to all digit positions and detecting failures on reading back. Any block causing failure is erased and the tape spaced suitably. On completion, a special block address is written to indicate "end of tape", and the entire tape is then checked again by reading backwards. Any failures cause re-entry to the addressing routines. Finally, a title block is written containing the serial number, the name FREE, the number of blocks addressed, and an empty list of faulty blocks, and the tape is made available for use.

HALF - INCH TAPE 4.2

Half-inch IBM-compatible tape mechanisms may be attached to Atlas 2 in addition to the standard one-inch system. These operate through channels which are separate from, and additional to, the one-inch tape channels. Like the one-inch tape system, the transfers, once initiated, proceed autonomously, under the control of codewords. Provision can be made for reading or writing of any

length. The normal character code is an extended form of standard I.B.M. Binary-Coded Decimal (B.C.D.) and hardware is provided for automatic conversion from this to standard Atlas Internal Code. Provision is also made for reading and writing pure binary records. The half-inch tape can be used in two entirely different modes. The first is as normal backing store in the same way as one-inch tapes, and extracodes are provided for requesting direct transfers of records, with or without code conversion, between core store and half-inch tape. The second is as an input/output device for the input of programs and data and the output of results, for use in association with a "satellite" computer. In this case the information is handled by input/output extracodes (see Chapter 6) in exactly the same way as information coming directly from, or going directly to, normal on-line input/output peripherals. Thus there are SERs associated with half-inch tape to provide for all facilities comparable to those provided for one-inch tape, such as monitoring, queuing of requests, etc. and, in addition, there are SERs to link the half-inch tape system with the input/output system.

Chapter 5

MAGNETIC DISC FILES

Provision is made on Atlas 2 for attaching Data Products Disc Files. Each disc has a capacity of 262,144 words. The maximum number of discs per module is 16, so that a module can hold up to 4,194,304 words, but it is possible to attach several modules. Transfers between core store and the discs, once initiated, are autonomous, and are in units of one or more "sectors" of 512 words or of complete "bands" of 4096 words. The revolution time is 52ms; thus instantaneous transfer rate is 4096 words in 52ms,i.e. one word in about 12.7 microseconds. The disc file is attached to the computer by a separate channel, so that disc transfers can proceed independently of any other peripheral device. Extracodes are provided so that programmers may request transfers between core store and the disc file. In addition, on an Atlas 2 equipped with a disc file, provision is made for the Supervisor to make extensive use of the disc file, for the storage of its own routines, the compilers and input/output bufferage, as an alternative to, or in addition to, the magnetic tape system. Object programs have "logical" areas of the disc store assigned to them by the Supervisor, and the Supervisor monitors any attempts to gain access to parts of the file outside these areas. Thus complete protection is provided by the Supervisor between the areas of the disc store used by different object programs and between these and the areas used by the Supervisor. Moreover, because each programmer uses addresses from zero upwards within each area, he does not have to be aware of the specific areas assigned to him nor of the fact that other programmers are sharing the file. This system is used to provide programmers with space on the disc file during the execution of an object program. A different system is used for programmers who require long term storage for programs or data from one run to another. This is discussed in Chapters 6 and 7.

Chapter 6

SLOW PERIPHERAL EQUIPMENTS

PERIPHERAL TRANSFERS 6.1

As mentioned in Chapter 1, a large number and variety of peripheral equipments may be attached to Atlas 2. However, the amount of electronics associated with each equipment is kept to a minimum, and a system is used whereby transfers proceed autonomously between one-character buffers associated with each equipment and pre-set multi-character buffers in the main core store, under the control of "code-words" held in the core store.

Each peripheral has a fixed word in the Fixed Area of the main core store for its codeword, which it accesses autonomously for each character transfer, increasing the address contained there by one (half-word) on each occasion. The code-word also specifies the length of the pre-set main-store buffer, and the hardware causes an interrupt when the last character from or to this buffer has been transferred. The code-word is in two halves, the first associated with the current multi-character transfer, and the second with the next. There is a bit in the current half of the code-word to specify whether the other half has been set up. If it has, then when the current multi-character transfer has been completed, in addition to causing an interrupt, the hardware arranges for the second code-word to be transferred to the position of the first, and for the transfer to be continued using the new main -store buffer specified. If the bit in the current half of the code-word specifies that the other half has not been set up, then the hardware automatically stops the peripheral. The interrupt routine initiated by the completion of the current multi-character transfer informs the Supervisor, so that, in case of an input device, arrangements can be made to translate and pack the information and transfer it to backing store, and, in the case of an output device, to label the area as Free. Also the Supervisor must take steps to permit the transfer to continue, which means, in the case of an input device, finding an area of vacant core store to be set up as a multi-character input buffer, and, in the case of an output device, to ensure that the next section of output is available in core store in a suitable buffer area. When the SERs which perform these operations are completed, they set up the new second-half code-word, and set the bit in the first-half code-word to indicate that the new second -half code-word is set up. Thus the first-half code-word is always *set up* with the indicator bit set to "Stop", but this may be changed to "Continue" by the SERs before the multi-character transfer is completed, if it has been possible for the SERs to complete their operations in time (which may not always be possible, either because, through the existence of higher priority SERs, there has not been time to implement them, or because there has not been time to bring down a next section of output from backing store, or because there is no vacant core store to assign as a new buffer). So, in the normal way the peripherals operate continuously, but if for any reason the Supervisor is not able to permit them to do so, then they will automatically stop, and the system is "Fail Safe".

The maximum size of main store area which may be used as a single multi-character buffer for a peripheral device is 1024 half-words (one block), but it is normal to use units of 32 half-words for the paper-tape equipment and 128 half-words for the card equipment, in order to reduce the amount of core store needed for buffering, whilst keeping small the percentage of computer central processor time used in organizing the transfers.

For the ordinary slow peripherals, the normal system is to use a fixed double-buffer in A.F.A. for each peripheral as the primary buffer and for SERs performing code-conversion and packing or unpacking to do transfers into or out of chains of blocklets in S.R.D. within the Well. Thus normally if the first half-buffer has not been emptied (or replenished) by the time the second half-buffer is filled (or emptied), the Supervisor will stop the peripheral rather than assign a third buffer area.

However, provision is made for transfers to be made directly into or out of S.R.D. (useful, for example, for high-speed on-line devices) or for multiple buffers.

The code-word system is used for the card readers and punches, the paper-tape readers and punches, and the teleprinters. A different system is used for the line-printers (Anelex Series 4/1000), which have a built-in buffer store for the characters of one line of 120 characters. With the line-printers, an interrupt occurs when this buffer has been emptied (i.e. when a line has been printed) and the interrupt routine thus initiated calls for an SER which sends characters one at a time directly (i.e. not autonomously) to a one-character buffer in the V-store, whence they are sent to the line-printer's own buffer. When the SER has sent the last character of a line, it sends a special End-of-line character to the line-printer V-store, and the line-printer control then initiates the printing of the line. Normally this SER is completed before the line-printer has finished feeding the paper after the previous line, so that in effect the line-printer operates continuously, but if for any reason (such as those above) the SER has not been completed before then, the line-printer will simply wait until it is completed. Thus the line-printer system is also normally non-stop, but is "Fail Safe".

So all the peripherals operate autonomously, whilst the central processor is occupied with the execution of object programs, and only infrequently (e.g. every 100 ms. in the case of the card reader, or every 60ms. in the case of the line-printer, etc.) is there an interruption leading to an SER to deal with a line or a card of input or output.

ATTENTION BY OPERATORS 6.2

Whenever equipment needs attention it is "disengaged" from the computer. In this state, which is indicated by a light on the equipment and a corresponding bit in the V-store, it automatically stops and cannot be restarted by the computer.

The operator may engage or disengage an equipment by means of a button so labelled. The equipment may also be disengaged by the computer by writing to the appropriate V-store bit, but the computer cannot engage it.

The "engage/disengage" button does not iself cause an interruption to the central processor. Instead, the "engaged" bits in the V-store are examined about every second (by a routine activated by the clock interruption), and any change activates an appropriate SER. Disengaging a peripheral causes its activities to cease when its current operation is completed; for example, if the operator disengages a card machine in mid-cycle to replenish the magazine or to empty the stacker, the cycle is completed correctly.

There are also other special controls for particular equipments, e.g. a run-out key on card machines, and a 5/7/8-track tape width selector on paper tape equipments.

Most devices have detectors that indicate when cards or paper are exhausted or running low. These correspond to bits in the V-store that are read by appropriate SERs. The paper tape readers however have no such detector, and should a paper tape pass completely through a reader (due to the absence of terminator characters - either because of mis-punching or because the tape has torn) this is detected simply as a failure to encounter a further character within the normal time interval. This condition is detected by the same timed interrupt routine as detects changes in the state of the engaged/disengaged bits.

Chapter 7

THE OPERATING SYSTEM

GENERAL 7.1

The high computing speed of Atlas 2 and the use of multiple input and output peripheral equipments enable the computer to handle a large quantity and variety of problems. These will range from small jobs for which there is no data outside the program itself, to large jobs requiring several batches of data, possibly arriving on different media. Other input items may consist of amendments to programs, or results generated by an earlier program, or requests to execute programs already stored within the system. Several such items may be submitted together on one deck of cards or length of punched tape. All must be properly identified for the computer.

To systematize this identification task, the concept of a "document" has been introduced. A document is a self-contained section of input information, presented to the computer consecutively through one input channel, or a self-contained section of output information intended for dissemination consecutively through one output channel. Each document carries suitable identifying information, which consists mainly of a title, to identify the document uniquely within the system, but which also includes a means of telling the Supervisor what kind of document it is (see below).

In order that the Supervisor can identify and collect the input documents required for the execution of each job, organize the mounting of any necessary magnetic tapes when execution is initiated, as well as determine the best time for execution which will ensure a high overall efficiency for all parts of the system, the programmer has to supply with each job, in addition to the program and data documents, a special document known as the Job Description document. The Job Description contains the Job title, and includes the names of all input documents required, the number and/or names of all magnetic tapes required, estimates of the computing time and amount of store needed, and indications of the types and amounts of output expected. The Job Description is discussed further in Section 7.3.

INPUT AND OUTPUT - THE WELL 7.2

All input awaiting execution and all output awaiting dissemination is stored in a combination of core store and backing store known as the Well, whose activities are controlled by a part of the Supervisor known as the Well Master. It is the function of the Well Master to receive information from an input peripheral or output from a program, and to make this information available when it is required by a program or by an output peripheral with the minimum of delay, whilst minimizing the amount of core store needed as bufferage. The Well system is designed to use a variable number of magnetic tapes, a variable amount of the disc file (if one is attached) and a variable amount of core store, according to what is available after the need of current programs or compilers have been met. The Well system does not distinguish between input information (i.e. programs awaiting execution or data for these programs) and output information (i.e. results awaiting dissemination). All is stored in a suitable part of the Well according to its length and the length of time before it is likely to be required. In this section therefore we shall use the words Input and Output to mean Input To and Output From the Well, i.e. Input will mean either input from peripherals of programs awaiting execution or of data or output of results from a program and Output will mean the giving of programs to a compiler or data to a program, or the giving of results to an output peripheral.

The Well Master operates in co-operation with the Scheduling routines of the Supervisor, in order to determine when a document will be required and what parts of the system it may use for the Well.

Magnetic Tapes may be used by the Well either as Swinging Tapes (possibly backed by Feeder Tapes) or as Serial Tapes, or as File Tapes. The Swinging Tapes are used for storing all short and medium -length documents, the Serial Tapes for long documents, and the File Tapes for standard programs and for programs currently under development. The Well as a whole can be treated as a single "black box" by the rest of the system, information being simply given to it for storage or requested from it, and any movements within it of information between core store and backing store or between one part of backing store and another are entirely its own concern; however provision is also made for other parts of the Supervisor to influence the internal activities of the Well when this is desirable.

The Atlas 2 Swinging Tapes operate on basically the same principles as the Atlas 1 Swinging Tapes (Ref. 4). A Swinging Tape operates in conjunction with two core store buffers, the Reading and the Writing buffers, and at any stage the information which has been Input but not yet Output is contained in a certain region, called the Scanning Area, the space before this containing information which has been Input and Output, and the space beyond this being as yet unused. The tape swings over the Scanning Area, and each cycle consists of three phases: Reading, Traversing, and Writing. The reading phase is initiated when the tape is at the upper end of the scanning area and moving backwards. As the area is scanned, any items that are required to be read down are transferred to the reading buffer. At the end of the scan (when the tape has reached the lower end of the scanning area), the tape is reversed and is run forward to the upper end: this is the traversing phase. The contents of the writing buffer are now written to the tape, thereby extending the scanning area; the tape is then reversed again and the next reading phase starts. During the reading and traversing phases, material for the tape accumulates in the writing buffer; during traversing and writing phases the reading buffer is emptied. The buffers must be large enough to deal with the information flowing to and from the tape during one swing. Each writing phase uses a new section of the tape, and, since items are normally read only once, the scanning area gradually creeps forward. (For this reason, the motion of the tape is sometimes described as "Creeping Swing".) The maximum permissible length of the scanning area is determined by the size of the core-store buffers and the information rate, and therefore it is necessary that items are not required to stay on the tape longer than a critical time, determined by the swing time and the average rate of arrival of information. If this condition is not met, special steps have to be taken.

A single Swinging Tape can be used as the sole backing store for the Well, if the back-log and the rate of flow of information into and out of the Well are not too great, but if these become large it is not very satisfactory. For example, suppose that a storage capacity of about 800,000 characters is required, and that the mean rate of flow to and from the tape is 3,000 characters per second (which is about the rate for one card punch, one card reader, and one line-printer, all operating continuously), then the time of swing would be about 30 seconds, and core-store bufferage of about 200,000 characters, or 25,000 words, would be needed, and this is normally unacceptably large. Also, requests for documents would have to be scheduled up to 30 seconds ahead, which may be difficult to do accurately. Finally, if some individual documents are very large, then large amounts of core-store bufferage would be needed from time to time. The situation can be improved greatly by bringing further tapes into the Well. These may operate as subsidiary Swinging Tapes, or as Feeder Tapes, or as Serial Tapes.

A subsidiary Swinging Tape buffers information on and off the main Swinging Tape. This considerably reduces the core-space required for a given information rate (in the above example to about 8,000 words) and also means that accurate scheduling need only be done a short time ahead, with less accurate scheduling over the period of the swing of the main tape.

A Serial Tape does not swing, but, having had a document written to it, rewinds to the start of it and remains there until the document is required, when it unloads it, block by block, as required. Thus the Serial Tape normally requires less than 1k of core-store bufferage, but can normally handle only one document at a time. It is useful for very long documents which would otherwise require large amounts of core-store bufferage.

Feeder tapes can be provided either for an isolated Swinging Tape or for either or both of a main Swinging Tape backed up by a subsidiary Swinging Tape. Whilst its associated Swinging Tape is in the reading or traversing phase, information destined for it is written to the Feeder, which moves progressively from a fixed base line. When the Swinging Tape starts the writing phase, the Feeder is read backwards, and its contents transferred. Material arriving for the Swinging Tape during the writing phase goes straight to it, the transfer from the Feeder being temporarily halted. The use of Feeder Tapes again reduces the core-store requirements for buffering, since, if a Feeder is

5056(10.64)

provided for a Swinging Tape, there is always a tape available to accept information coming to it. For example, using a single Feeder Tape together with a pair of Swinging Tapes a ratio of Well capacity to core store needed of about 35 is achieved when the rate of information flow is about 3,000 characters per second, and using Feeder Tapes for both these tapes the ratio becomes about 60. Even with a rate of information flow of 6,000 characters per second a ratio of 16 is obtained with four tapes altogether.

The Well is made up dynamically of tapes of these types according to the number of tapes available and the kind of demands being made upon the Well (the rate of flow, the distribution of the lengths of documents, and the pattern of the scheduling). It should not be thought that the division between the various kinds of tapes is absolute, although it is convenient to described them as if it were. In fact tapes change almost imperceptibly and continually from one type to another. For example, a Swinging Tape becomes a Serial Tape if it is arranged that only one document is unloaded from it for a while and no new information put to it during this time; since a Swinging Tape does not swing unless there is a definite immediate need for it to do so, this is then at the same time behaving as a normal Swinging Tape. When the document is completely unloaded, then perhaps other material will be sent to it and other documents will be requested from it, so that it will again begin swinging. All this activity is controlled by the scheduling routines of the Supervisor in conjunction with the Well Master.

Thus the Well expands and contracts as tapes become available or are required, and uses its tapes in such a way as to provide convenient Input for information requiring to be stored in the Well and to have available information from it when it is required as Output. The actual way in which the tapes are used at any instant of time depends on the details of all the activities being performed by, and required of, the Well, but the purpose is always to maintain high efficiency and to minimize the amount of core store required.

To change the number of tapes used by the Well is easy: it is only necessary to divert the flow of information to or from a tape, and it will fill up or drain within a few seconds by the normal working of the Well. Thus it is possible to use any tapes that are available, so that an otherwise idle tape can temporarily increase the efficiency of the Well system.

Documents are stored in the Well as chains of blocklets, the first two words of each blocklet being used for chain links and identifying information. It is not necessary to keep a record of what is on a feeder tape, since it must inevitably be transferred to the associated swinging tape within a short time, but a dictionary is needed for each swinging tape, giving the position of every document currently on the tape. These dictionaries are updated whilst information is being written to the tape, and are consulted by the scheduling routines before each read sweep.

Parts of the disc file can also be incorporated in the Well. It can be used in a similar way to the swinging tapes, storing mixed documents on each band and bringing them down as required, or it can be used in a similar way to the serial tapes for the serial storage of long documents, or it can be used as a feeder either for other parts of the disc or for tapes. The amount of disc storage thus used, its distribution between the various possible modes of use, the distribution of information between the disc storage alone or tape storage alone or a combination of the two, would be decided dynamically by the Well Master in co-operation with the schedulers.

The Well was designed initially as an input-output buffer, but it can be used for many other purposes, for example for holding the inter-pass streams of multi-pass compilers. It provides, in an economical manner, efficient input-output with many convenient user-facilities, in particular multiple input -output streams, and it enables the system to provide all the advantages of buffered input-output with the convenience that to the programmer the peripheral devices appear to be on-line.

TITLES, DOCUMENTS AND JOB DESCRIPTIONS 7.3

Titles 7.3.1

Every job, every input document, and every 1" magnetic tape must be uniquely identified by its title. When referred to, titles are always written between round brackets. A tape title is in two parts, the serial number and the name, and these are separated by a stroke, thus:

(C1345/PSD Sort Results)

A document or job title also divides into two main parts also separated by a stroke. In this case, the first part is called the code, has a maximum of 16 characters, and must in itself uniquely identify the document or job; the second part is called the body, may consist of up to 48 characters, and may be used as a descriptive title by the programmer to identify further the document or job. Often part of the code consists of a "job number" assigned by an installation to a programmer for logging and costing purposes; for this reason the code is further divided by another stroke, so that the "job number" can be separated from the programmer's part of the code. Typical document or job titles are:

(C4567/1AB3/Matrix Division)
(GGHJ43//Field Tables)

Documents 7.3.2

Every input document is preceded by its identifying information, as mentioned in Section 7.1. In the case of program or data documents this consists simply of the title of the document, as above. In the case of a Job Description document this consists of the name of the compiler language or multi-language system in which the program is written followed by the title of the job, thus:

ALGOL (CRMN486/Random Coding)

All output documents are also preceded by a title. This may simply be the title of the job which generated the output or it may be a title requested by the programmer both to aid identification and to provide a suitable title for re-input, should this be required.

All input documents must normally be terminated by one of several standard *markers*. In the case of paper-tape documents this consists of three asterisks followed by an identifying letter. Thus, ***Z indicates the end of a document to the Supervisor and causes it to disengage the reader ready for the next document; ***T indicates a temporary stop and causes the Supervisor to disengage the reader and to treat the information read in when the reader is re-engaged as a continuation of the same document. In the case of punched cards, the marker consists of a punching of rows 7 and 8 in column 1 of a card, and the identifying letter is punched in column 80.

Normally input is in a standard code for each device, and the information is translated to standard internal code by the Supervisor's input routines, but provision is also made for input of pure binary information. This is introduced by other special markers. For example on paper tape when ***B is encountered the computer reads the information following on the same document in binary, whilst ***F causes the reader to be disengaged and, when it is re-engaged, the Supervisor treats the new information as binary. Facilities also exist for outputting pure binary information.

Job Descriptions 7.3.3

The Job Description consists of several sections, namely those specifying INPUT, OUTPUT, TAPE, STORE, TIME, and NOTES.

The INPUT section specifies the titles of all input documents (program and data) required by the job, and assigns logical numbers to them by which they are identified during the execution of the program. It may specify that one or more of the documents is to be read from magnetic tape. A prefixed P or D to each entry indicates whether the document is Program (and thus required at compile time) or Data (and thus not required until execute time).

The OUTPUT section specifies all the output documents, the type of output device required for each and the expected length of each document, and assigns a logical number to each one. It may also specify that one or more of the documents is to be written to magnetic tape. As the required output device, in addition to PRINTER, CARD PUNCH etc., it may specify ANY, where it is immaterial what type of device is used, and NONE, which is useful, for example, to suppress output only required during program development. A title required at the head of an output document is also specified here.

The TAPE section specifies the numbers and/or names of all magnetic tapes required and assigns logical numbers to them. If the tapes are not required at the beginning of the job, their mounting may be requested by extracode during the execution of the program and their logical number assigned then; such tapes are assigned a logical *letter* in the Job Description which is used by the Mount extracode.

A tape may be already "owned", that is, it may be already titled and may contain information required by the program, in which case the number and the name must be specified; or it may be a new tape requiring to become owned, in which case the number is specified and the title requiring to be written to the tape is specified; or it may be an untitled tape (or part of tape) for use as working space which is relinquished when the program finishes, in which case the programmer need not specify number or name.

The STORE section specifies the amount of store required by the program during *execution* (the amount required by the compiler during compilation is communicated to the Supervisor by the compiler). The TIME section consists of two items, namely an estimate of the actual time taken by the central processor in obeying the instructions in the program, and estimate of the total real time required for completion of the job, taking account of time waiting for tape transfers etc.

The NOTES section is not decoded by the Supervisor, but is provided to permit the programmer to convey conveniently information to the compiler such as, for example, requests to compile trace instructions, or to record post-mortem information.

The Job Description system makes it easy to specify the requirements of even the most complex job. However, it might appear from the above that even for the most simple job a lengthy Job Description must be prepared. This is not so, because facilities are provided for "taking as read" certain standard items and for certain other simplifications in simple cases, so that for a considerable proportion of simple jobs the Job Description will consist of perhaps only three or four items. For example, if the TIME and/or STORE sections are omitted then standard allowances will be made; if there are no tapes required, the TAPE section may be omitted completely; if no output length estimate is given then a standard allowance will be made; if the Job Description is prepared at the same time as the program (or the data), or in the case of cards if these may conveniently be placed one after the other, then they may be put together as a single document, with the word FOLLOWS replacing the program or data document title in the INPUT section of the Job Description. The following is an example of a typical simple complete Job Description:

```
ALGOL  (CRMN486//Random Coding)
INPUT
P1  FOLLOWS
D2  (CECB//Magnetic Fields)
OUTPUT 1 PRINTER
```

The following is an example of a more complex Job Description:

```
ABL (F1234/1AB3/Survey Update)
INPUT
P1 FROM TAPE (F3452/EMKE Surveys)/100(F1234/AB/Survey Program)
D2 FOLLOWS
D3 (F1234/PQ, 17-9-64/North Harrow)
OUTPUT
1 PRINTER 500 SHEETS (Main Results)
2 PRINTER (Summary)
3 CARDS (F1234/XY, 17-9-64/Updated Parameters)
TAPE
1 (F6753/EMKE Statistical Tables)
2 BLOCKS 200
STORE 50k
COMPUTING 10 MINUTES
EXECUTION 25 MINUTES
```

SCHEDULING 7.4

The parts of the Supervisor concerned with determining the order of execution of jobs fed into the system, the number of tape decks etc. which may be used for the Well, the distribution and priorities of the various activities in the computer, etc., are called the Scheduling Routines, or the Schedulers. It is the function of the Schedulers to organize all the activities of the computer in such a way as

(a) to ensure the efficient utilization of all parts of the system as far as is possible with the mixture of jobs fed into the system, and thus to provide as high a "throughput" as possible, and

(b) to provide facilities whereby high-priority jobs and short development runs may be brought quickly to execution and completion, in order to provide a convenient service to users.

The algorithms of the Schedulers have been devised so that, as far as possible, these two aims are simultaneously achieved, complementing, rather than conflicting with, each other. Indeed it might be stated that the whole of the Supervisor, and not only the Scheduler routines, has been designed so as to achieve as efficient a system as possible not only in the computer itself, but in the whole computer installation, including operators and programmers, making the computer easy to run, and providing efficient facilities for the development and running of programs.

Perhaps the most important aim in all the Scheduling routines is to achieve efficient utilization of the core store. The Scheduler concerned with determining the order of execution of jobs is of course concerned with efficient utilization of the central processor, of the output peripherals and of the tape decks, as are, to a lesser extent, many of the other Scheduler routines, but the principal feature of the Schedulers, which determines the way they operate, and to a large extent the whole structure of the Supervisor, is the way in which they organize the use of core store. Every activity of the computer and every branch of the Supervisor is given a priority by the Schedulers, which determines whether it obtains and retains space in the core store. Everything is geared to making it possible for the Supervisor to operate in the minimum possible amount of core store, and yet to take advantage of any spare core store to increase its own efficiency or simplicity of operation. The way in which r3 and r4 are used to make good use of otherwise wasted space in the middle of a program's area during compilation is typical of the kind of measures taken to ensure efficient store utilization.

The Job Scheduler has available to it all the information in Job Descriptions to ensure efficient utilization of all parts of the computer. However, the efficiency of the system is not made to be dependent on the accuracy of this information. Thus, for example, advance preparation is always made as far as possible for the beginning of a new job in case any of the current jobs should finish prematurely, either because of a program fault or because of an inaccurate time estimate in the Job Description; for example the beginnings of the Input streams for the new job and the first part of the required compiler will be brought into core store or near backing store. Also the activity of jobs is monitored dynamically during the course of execution and, if necessary, their dynamic priorities adjusted so that high central processor utilization is maintained.

OPERATOR COMMUNICATION 7.5

Provision is made for communication between the computer operators and the Supervisor, via input/output typewriters or teleprinters or normal input and output peripherals, so that, for example, the loading of magnetic tapes may be requested by the Supervisor, the operators may be informed of the progress of jobs, and the operators may assign special priorities or request special actions for specific jobs. Each message from the operator to the Supervisor is acknowledged by the Supervisor so that the operator can know that it has been correctly received and interpreted.

The variety of messages for which provision is made is very great, and each has a necessary place in the operation of the computer. However, in practice, in the normal course of operations, very few messages from the operators to the Supervisor should be necessary and there should be very few messages from the Supervisor to the operators apart from routine requests for the mounting and dismounting of tapes and routine logging information, listing the jobs as they are executed together with their execution times etc. Normally from the moment the Supervisor is brought into the machine at the beginning of the day, it should be possible for jobs to be fed into the system, as they arise, at any vacant peripheral, and the only other operator activities should be to handle magnetic tapes and collect and distribute the results.

PROGRAMMING AIDS 7.6

A number of facilities are provided in the Supervisor specifically to aid program development. Apart from normal monitoring information in the event of a program or machine fault, these include the automatic dumping on backing store of the entire store area associated with any program in the event

5056(10.64)

of a program fault, so that the programmer can at a later time arrange for post-mortem information to be printed out or for the program to be corrected or for it to be re-entered at some suitable point, without having to keep the program in store all the time. Also an extracode is provided by means of which the programmer can establish a re-start point. This is a point at which execution should re-start in the event of a subsequent machine fault. Programmers with long programs are encouraged to write their programs in such a way that re-start points can be established at frequent intervals, so as to reduce the loss of time when a fault occurs. Otherwise, in the event of a machine fault current programs will be recovered from the Well backing store and re-entered at their beginnings, except in the case of jobs using magnetic tape, when a complete re-start is not necessarily possible (when information on tape is overwritten - such programs should always be designed with suitable re-start points).

Provision is also made for communication directly by programmers with programs in an on-line manner by means of any peripheral via the Supervisor, although the practical use of this would have to be restricted if efficient computer utilization were to be achieved. This facility could be of use either during the development of a program for the eradication of errors or the feeding in of trial data, or during the running of a developed program, controlling its course of action in the light of its output during computation.

Chapter 8

CONCLUSION

The Atlas 1 and Atlas 2 Supervisor programs are perhaps the most advanced examples so far existing of a program involving many parallel activities, all closely interconnected. They are also the most complex examples of a program designed to control the activities of a general-purpose computer. In such a program the overall structure is of prime importance: only if this structure is adequate, sound, and systematic, will it be possible to provide a flexible operating system which will not only operate efficiently in the circumstances and for the configuration for which it was originally designed, but which can be easily modified and extended to cope with configurations and purposes originally unforeseen. The structure of both the Atlas 1 and Atlas 2 Supervisors, involving the use of interrupt routines and "Supervisor Extracode Routines" controlled by a co-ordinating routine, has proved eminently satisfactory as a basis for every supervisory task that has so far been envisaged, including those conceived long after the Supervisors had been designed, and it is expected that it will be able to cope with all future variations that may be called for.

The design of the Well is such that it can contract to require little or no backing store and cope efficiently with a low data rate, or expand to occupy several magnetic tape decks and large areas of the disc file in order to cope with very high data rates, very long documents, very large "back-logs", and to provide every facility for the permanent storage of results for re-input and of standard programs, whilst at the same time being able to contract should large complex jobs arise. Thus the Well system should be suitable and efficient for all kinds of Atlas 2 and all kinds of load pattern.

The Store organization system is such that the computer can operate efficiently with either a stream of time-shared small jobs, needing a variety of compilers and making considerable demands on the ingenuity of the Supervisor, or single large jobs demanding the maximum of store with the minimum of Supervisor interference.

The Operating System has been designed to combine the highest possible internal efficiency with a high regard for user convenience and service. The Scheduling system and the techniques of operator communication make it possible for efficiency to be achieved with the minimum of human interference and external decision-taking, so that all the activities of the system may be controlled and directed internally with all information about the up-to-the-microsecond state of all the programs in the system available to ensure that correct and meaningful decisions are taken at every stage and at the full speed of the computer, in a way that would be impossible externally.

Finally, the whole of the Supervisor has been planned, developed and tested within a complex environmental testing system, so that, as routines are added, all their entries and exits, and their entry and exit conditions, are fully checked. Thus when the Supervisor as a whole is brought into operation, it can be hoped that all, or almost all, logical or coding errors will have been eliminated. This is essential in a program of the complexity of the Atlas 2 Supervisor, particularly because, by its very nature (its activities depending on the interaction of many independent programs and many asynchronous events) many conditions arising in the course of its operations can never be repeated or recaptured.

References

1 Ferranti Computing System - Atlas (1), List DC46, I.C.T. Ltd.

2 Ferranti Computing System - Atlas 2, List DC48, I.C.T. Ltd.

3 One-Level Storage System, by T. Kilburn, D.B.G. Edwards, M.J. Lanigan, and F.H. Sumner, List R67, I.C.T. Ltd.

4 The Atlas Supervisor, by T. Kilburn, R.B. Payne, and D.J. Howarth, List R58, I.C.T. Ltd.

All the above publications are obtainable from the I.C.T. London Computer Centre, 68 Newman Street, London, W.1.

5056(10.64)