I·C·T

INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

COMPUTER MANUAL

I·C·T

INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

THE I.C.T. ATLAS 1 COMPUTER PROGRAMMING MANUAL FOR ATLAS BASIC LANGUAGE (ABL)

68 NEWMAN STREET,
LONDON, W.1.

LIST CS 348A
JANUARY 1965

INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

THE I.C.T. ATLAS 1 COMPUTER

PROGRAMMING MANUAL

FOR

ATLAS BASIC LANGUAGE

(ABL)

68 NEWMAN STREET,
LONDON, W.1.

LIST CS 348A
JANUARY 1965

PREFACE

This manual supersedes the manual CS 348, "The Atlas Provisional Programming Manual", January 1963. It provides information for the programming of the Atlas 1 computer in the language known as Atlas Basic Language (ABL). It is a self-contained document, providing sufficient information about the Atlas 1 computer to enable programmers to write and develop programs in ABL without recourse to any other documents about Atlas 1.

The Atlas 1 Computer is the latest result of a long-standing collaboration between Manchester University and Ferranti Ltd. A later version of the computer, known as Atlas 2, has been developed jointly by Cambridge University and Ferranti Ltd. In September 1963 the Computer Department of Ferranti Ltd., was acquired by I.C.T. Ltd., who now manufacture and market the Atlas computers.

Atlas Basic Language (ABL) is a symbolic input language close to "machine language". Each ABL instruction corresponds to one machine instruction, and each part of an ABL instruction to each part of a machine instruction. In its simplest form an ABL instruction consists of four numbers corresponding exactly to the internal machine representation, but extensive facilities are also provided in ABL for the use of a variety of parameters and symbolic expressions which are evaluated by the ABL compiler. ABL also provides a comprehensive system of directives to control the assembly of a complete program. Finally ABL provides facilities for the input, conversion and storage of fixed-point numbers, floating-point numbers and character strings for use by the program.

In this manual no attempt is normally made to differentiate between those facilities which are a basic part of the machine (e.g. the instruction repertoire) and those which are a part of the particular language ABL (e.g. the formats for writing instructions). This is partly because it is impossible to separate them completely - any feature of the machine itself needs a language in which to describe it, and in this case that language is ABL - and partly because it is not normally necessary or helpful for a programmer to be conscious of the distinction. However, inasmuch as certain facilities of the machine itself are described here, parts of this manual are relevant and interesting to users of other Atlas programming languages. In particular, Chapter 10 "Preparing a Complete Program" applies to all Atlas languages.

A word must be said about the enumeration of binary digits: throughout this manual the convention adopted is to number bits as 0, 1, 2 ...., starting always with bit number 0 at the more significant end. This convention differs from that used in documents on the Supervisor and in engineering documents, in which only the accumulator is numbered as here, and in all other cases bit 0 is the least-significant bit.

CONTENTS

Date of Issue

Date of Issue

Chapter 1

AUTOMATIC DIGITAL COMPUTERS

1.1 Introduction

Atlas 1 is a very fast, automatic digital computer with built-in time-sharing facilities enabling a considerable number of problems to be processed simultaneously. This manual is intended for those who will prepare programs giving the machine detailed instructions for the step-by-step solution of individual problems. It is likely that they will have some previous knowledge of computers: should the ideas outlined in this chapter be unfamiliar, the reader is advised to consult an introductory text for further clarification.

1.2 Electronic Computers

The application of electronics has led to the development of the modern high-speed computer; we must distinguish two types of electronic computers:-

Analogue computers represent quantities by some analogous physical quantity and solve problems by working with an actual physical model obeying the desired theoretical equations. Since the quantities involved can be evaluated only by measurement, the attainable precision is necessarily limited. The slide-rule is a familiar example of an analogue computer, using lengths to represent the logarithms of numbers.

Digital computers operate upon numbers in some coded-digit form and make use of standard computational techniques to obtain direct numerical solutions to problems. By increasing the number of digits with which numbers are represented in the machine, the precision may be extended without limit. A desk calculator is a simple form of digital computer.

1.2.1 Digital Computers.

As a more complete form of digital computer, we may consider the combination of a desk calculator and its operator as a single computing unit; this will enable us to introduce the essential features of a typical digital computer:-

(a) *Input and Output.* This is one of the roles performed by the operator of a desk calculator, who must set up numbers in the machine before they can be operated on and also read results from the machine, recording them elsewhere. For an electronic computer, information is transferred to and from the machine by automatic equipment.

(b) *Control.* Here again, it is the operator who must control the sequence of operations on a desk calculator. An automatic computer is controlled by a program consisting of a sequence of detailed instructions in coded form. In the important case of a stored-program computer, the whole program is stored within the machine before any of it is obeyed; the speed of the computation is then not restricted to that of the input devices.

(c) *Arithmetic Unit*. The mechanism of a desk calculator carries out the individual operations in accordance with the key depressed. Similarly, the arithmetic unit of an electronic computer performs the functions called for by the successive instructions in the program. There will usually be included an accumulator in which the result of each step first appears, similar to the long register on the carriage of a desk machine.

(d) *Storage*. The keyboard and certain other registers of a desk calculator constitute a working store, insofar as the mechanism of the arithmetic unit is able to operate directly upon numbers contained in these registers. An electronic computer commonly has a working store capable of holding several thousand numbers. At any time each of these numbers is immediately available to the arithmetic unit, and so one speaks of a "random access" or "fast" store.

This type of storage is relatively expensive and so if still larger amounts of storage are required this is normally provided by some cheaper form of "backing" store. This will inevitably involve a longer access time, but, when required for computation, data can be transferred in blocks of several hundred numbers at once from the backing store to the working store.

1.3 Addresses

Each of the locations in the backing store and in the computing store is assigned an address. The address is a number, and it is important to distinguish the number which is the address of a location from the number which is contained in that location. As a means of distinction, we shall denote addresses by capital letters and contents by small letters and will assume, for example, that the number s is the content of the location whose address is S. In certain contexts we shall find it necessary to use the notation C(S) as an alternative to s. (Note that C(S + 1) is not the same as s + 1; the notation S^* is sometimes used to denote S + 1, so that $s^* = C(S + 1)$). The contents of S after an operation will be written s′, so that, for example, the equation

$$s' = s + b$$

denotes the operation of adding the number b to the contents of S.

1.4 Instruction Code

We have so far spoken of the contents of store locations as numbers, but they may also be instructions in coded form. Both numbers and coded instructions may be referred to as "words"; it is for the programmer to ensure that no attempt is made to interpret one type of word as the other. An instruction word will usually contain one or more addresses to specify the data to be operated on; there will also be a coded number specifying the operation to be performed. The correspondence between the elementary operations which may be directly carried out in the arithmetic unit and the code numbers which control them constitutes the "order-code" or "instruction-code" of the computer.

1.5 Jump Instructions

Instructions will generally be obeyed in the same sequential order as their addresses occur in the store. However, it is possible to specify by an instruction the address of the next instruction to be obeyed, and hence one may arrange to "jump" out of sequence to an instruction at any desired address. Instructions are provided to make such a jump conditional upon the sign of certain numbers in the machine; these conditional jumps provide the ability to take elementary decisions.

1.5.1 Looping.

By repeatedly jumping back to the same instruction, the computer can be made to obey a "loop" of instructions over and over again; this is a vital feature of high-speed computing, making it possible for a program of reasonable length to control the machine for relatively long periods of time.

1.5.2 Modification.

The utility of computing loops is greatly enhanced by the facility known as "modification", whereby different data is processed in each iteration of the loop. The store address of the number to be operated on is modified before use by the addition of an index stored in one of several special registers. Thus, if the index is increased by unity before each successive iteration, one may operate upon a list of numbers held in the store, and so, for example, form their sum or average.

1.6 Binary Numbers

The storage mechanisms used in electronic digital computers are normally made up of devices having two possible states, just as a switch may be either OFF or ON. If we associate with these two states the symbols 0 and 1 respectively, we are led to adopt the binary number system: the string of 0's and 1's stored on a row of two-state devices is interpreted as a succession of coefficients of powers of two in a polynomial. This is exactly analogous to the conventional decimal notation based on powers of ten. The binary digits 0 and 1 are commonly referred to as "bits".

1.6.1 Negative Numbers.

If a desk calculator is used to subtract some small numbers from zero, the result is characterised by a string of 9's at the more-significant end; the same operation with binary numbers produces a string of 1's. We have here a naturally occurring representation of negative numbers, which is made unambiguous by restricting the range of positive numbers to those having 0 as their most-significant digit. This then becomes a sign digit, and the presence of a 1 in this position will indicate a negative number whose actual value is obtainable by subtracting 2^r, r being the number of bits in the number.

Chapter 2

THE ATLAS 1 COMPUTER

2.1 A General Description of Atlas 1

The main parts of Atlas 1 consist of:-

(a) The control unit
(b) Two arithmetic units
(c) The Supervisor
(d) The storage system
(e) Input and output devices.

2.1.1 The control unit produces in correct sequence the control signals necessary to call for an instruction, to decode it, to modify the address, to obtain the operand from store and to perform the arithmetic operation. The address of the current instruction is held in one of three special index registers, called control registers. Before the current instruction is decoded the contents of the control register in use are increased by one in anticipation of the next instruction.

2.1.2 Arithmetic is mainly done in the accumulator, which is a double-length floating-point register. The accumulator arithmetic unit can obey 49 different instructions, including different types of addition, subtraction, multiplication and division, transfers, tests, shifts etc.

For small integer arithmetic, modification and counting, there are also 128 index registers. These are known as B-registers and have their own arithmetic unit. The B-register arithmetic unit can obey 51 different instructions, including addition, subtraction, logical operations, shifts, tests, counts etc.

2.1.3 Any peripheral transfer on Atlas 1 has only to be initiated, after which it proceeds independently, leaving the central computer free to continue obeying instructions. Suppose there to be only one program in the computer, which might be reading characters from the tape reader and sorting them on magnetic tape, one per word in units of 512 words. The tape reader operates at 300 characters per second and so reads a character once every 3,333 microseconds (μs); a magnetic tape transfer of 512 words takes 46 milliseconds (ms). Between reading characters it would be possible for the central computer to obey about 2,000 instructions, and while executing a magnetic tape transfer, about 30,000 arithmetic instructions could be obeyed. If the computer were to be idle during transfers because the information was wanted immediately (in the next instruction) obviously its utilisation would be very inefficient. Note that if the slow peripheral equipments could always transfer information at the rate required by the central computer for any problem no difficulty would arise. As they cannot, special operating methods have to be used. The method on Atlas 1 is to have a special program called the Supervisor which controls the flow of programs through the computer. The Supervisor is simply a program which attempts to run Atlas in an efficient way, that is, it tries to keep all the parts of Atlas busy. To achieve this, it shares computing time between programs, and manages all peripheral transfers, including input and output as well as drum and magnetic tape transfers. The Supervisor is described in more detail later; at this stage it must be remarked that although it is not part of the "hardware" in the sense that the core store is, it is the most important single feature of Atlas and quite indispensable.

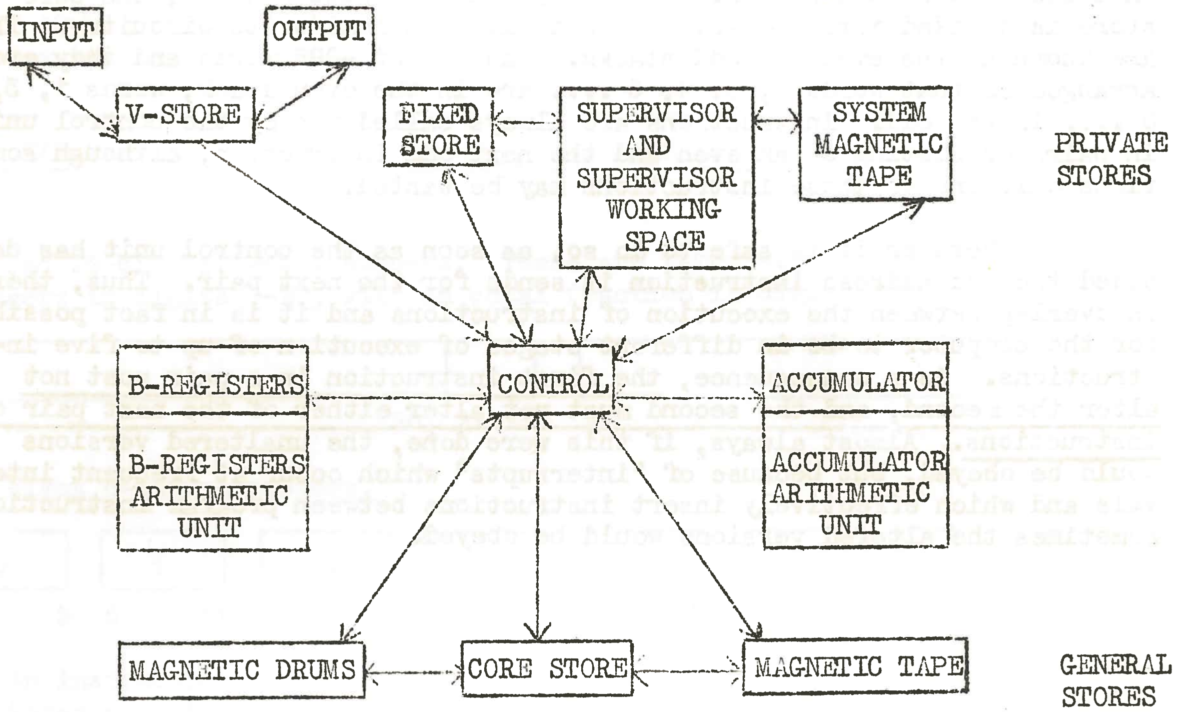
2.1.4 The main store on Atlas 1 consists of core store in units of 8192 words and magnetic drum store in units of 24,576 words. A total of about one million words are directly addressable. Up to 32 magnetic tapes can also be used as a backing store. The main store can consist of core store and drums in any proportion. The programmer treats the store as if it were all core store; he will in fact not know what parts of his program are in the core store at any one time. The Supervisor manages drum transfers behind the scenes as required, and attempts to keep the most used blocks of program always in the core store, by means of a "drum learning" program. This idea of a fast and a slow store appearing as a single fast store is called the "one-level-store" concept.

The Supervisor occupies most of a special store called the fixed store, and some blocks of the main store. The fixed store is a "read only" store in multiples of 4096 words where binary ones and zeros are represented by ferrite and copper slugs in a wire mesh. It is used to represent permanent programs which will not be changed, and besides the Supervisor it holds the "extracodes". These are extensions to the basic instructions, described later. For working space the Supervisor has a subsidiary core store of 1024 words, in which it keeps parameters associated with programs in the machine, peripherals, etc. The Supervisor also uses three magnetic tape units, called "system tapes" as part of its input/output organisation. All the stores used by the Supervisor are known as private store, and it is not possible for ordinary programs to interfere with them.

2.1.5 A large variety of input and output devices are allowed on Atlas 1. Each type of device is connected to the central computer via the peripheral co-ordinator, which contains buffer registers and information registers concerned with the transfer of data. These registers, which are at different places in the computer, and also some registers connected with the arithmetic units, are collectively referred to as the V-store. They are only accessible to the Supervisor, and form part of the private store. The peripheral co-ordinator allows the following types of input/output equipments to be attached.

ICT TR5 paper tape readers	300 ch/sec
ICT TR7 paper tape readers	1000 ch/sec
Teletype paper tape punches	110 ch/sec
Creed 3000 paper tape punches	300 ch/sec
Creed 75 teleprinters	10 ch/sec
ICT card readers	600 cards/min
ICT 582 card punches	100 cards/min
Anelex line printers	1000 lines/min
Graphical outputs	
Clock	

The following diagram shows the component parts of Atlas 1:-



2.2 The Main Store

Within the programmers store, registers are numbered consecutively from 0 upwards. Registers are arranged in blocks of 512 words called pages, and transfers between the core store and drums or magnetic tape take place in units of 512-word blocks. To increase the computing speed, the core store is divided into stacks, each with its own read/write circuitry. These are known as the even and odd stacks. Each is of 4096 words and they are arranged so that words 0, 2, 4, 6 .... are in the even stack, words 1, 3, 5 .... in the odd. Instructions are always called for by the control unit in pairs consisting of an even and the next odd instruction, although sometimes only one of these instructions may be wanted.

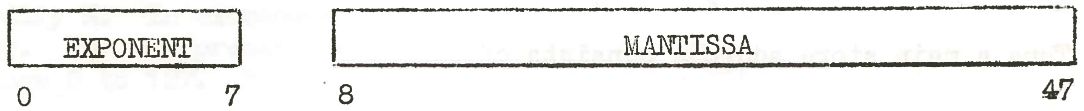
Wherever it is safe to do so, as soon as the control unit has decoded the odd address instruction it sends for the next pair. Thus, there is overlap between the execution of instructions and it is in fact possible for the computer to be in different stages of execution of up to five instructions. As a consequence, the first instruction in a pair must not alter the second, and the second must not alter either of the next pair of instructions. Almost always, if this were done, the unaltered versions would be obeyed, but because of "interrupts" which occur at frequent intervals and which effectively insert instructions between program instructions sometimes the altered versions would be obeyed.

2.3 Storage of information in Atlas 1

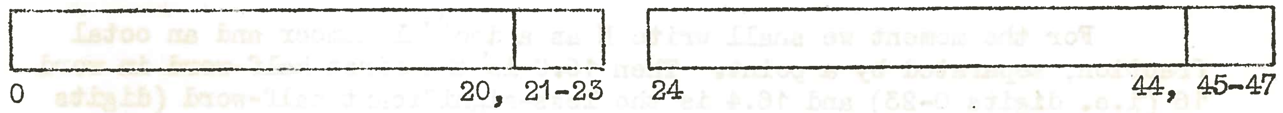
An Atlas 1 register, or word, contains 48 binary digits. These are conventionally numbered from 0 (most-significant) to 47 (least-significant).

A single word can be used to represent any of the following:-

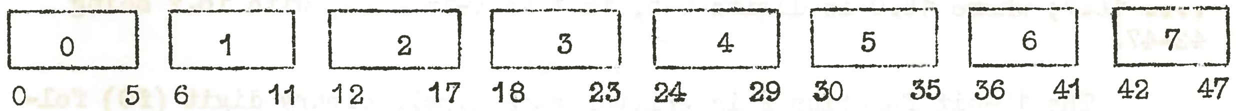
(a) A 48-bit floating or fixed point number, with the 8 most-significant bits representing the exponent and the other 40 the mantissa.



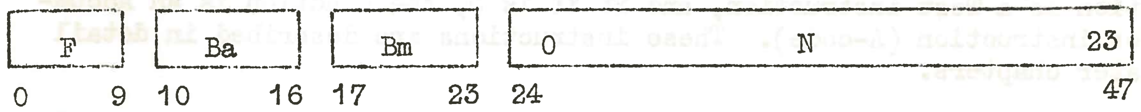
(b) Two 24-bit half-word numbers. These are taken usually as 21-bit signed integers in digits 0-20, with an octal fraction in digits 21-23.



(c) Eight 6-bit characters.



(d) An instruction, specifying a function F (most-significant 10 bits), two index registers Ba and Bm (7 bits each) and an address N (least-significant 24 bits).



Throughout this manual the binary digits of a word are numbered from the most-significant end, starting with bit 0. The engineers' numbering system is the reverse of this, and is used in some documents describing basic programs such as the Supervisor and Engineers' programs: in these documents bit 0 is the least-significant bit.

2.4 Instructions in the machine

N can be taken as an operand directly in some instructions, in which case it is known as n. When N is used as an address, bits 1-20 specify a word in the main store, bit 21 specifies a half-word address and bits 22 and 23 specify a character address within a half-word (for the moment renumbering the bits of N as 0-23). Instructions ignore irrelevant digits in the address. Thus an instruction involving half-words ignores digits 22 and 23 in the address. Digits 12-20 specify the word address within a block, and digits 1-11 specify the block.

Thus a main store address consists of

```
  Block number   Word number in block   Half-word address   Character address
1             11 12                  20               21        22          23
```

For the moment we shall write N as a decimal number and an octal fraction, separated by a point. Then 16.0 is the first half-word in word 16 (i.e. digits 0-23) and 16.4 is the less-signifioant half-word (digits 24-47), in an instruction which uses a half-word. In instructions for handling characters, the characters in word 16 are 16.0, 16.1, 16.2, ... 16.7; where 16.0 is digits 0-5, 16.1 is 6-11 etc., with 16.7 being 42-47.

The 10-bit function F is written as a single binary digit (f0) followed by three octal digits. For all basic functions f0 is zero, and may be omitted in the written form. The basic functions fall into three categories, depending on the first octal digit (f1 to f3); if this digit is 1, the function is a B-register instruction (B-code), if it is 2, the function is a Test instruction, and if it is 3, the function is an Accumulator instruction (A-code). These instructions are described in detail in later chapters.

The basic order-code is extended by the provision of "extracodes". These are routines, written in basic instructions, which are positioned in the fixed store and which carry out many operations usually managed by a subroutine library. Extracodes are recognised by having f0 = 1. When this is encountered, main control is halted and the program continues under a special "extracode" control at an address in the fixed store. The final instruction in this routine (which is recognised by f1 = f3 = 1 and obeyed as if f1 = 0) has the effect of returning to main control at the program's next instruction. Thus extracodes are subroutines with automatic entry and exit; to the programmer they appear as one instruction.

For example,

Function	113	is a B-code used to store a B-register
	234	is a test-code, transferring n to Ba if the contents of the accumulator are zero
	374	is an A-code, dividing the accumulator by the contents of a store address
	1250	is an extracode, to unpack a 6-bit character from store and place it in Ba.

The B-registers are used in different ways depending on the type of instruction. There are 128 B-registers, B0 to B127, most of them of 24 bits. B120-127 are special purpose B-registers, the rest are general purpose. In A-codes, the contents of Ba and the contents of Bm are added to N to give a modified address. That is, the address S used in the instruction is N+ba+bm. In most B-codes, ba is used as an operand, so only bm is used to modify N, and then S = N+bm. (There are two exceptions, in which bm is also an operand, so no modification takes place.) For most test codes, bm is used as a further operand; where it is not it is used to modify N. In extracode instructions, Ba and Bm are treated in a special way. For the present we shall write Ba and Bm as decimal numbers in the range 0 to 127. B-register arithmetic is described in Chapter 4.

2.5 The written form of Instructions

Instructions are written on one line, with the component parts separated:

```
        F      Ba     Bm     S
Thus   113     1      0     16.4   stores b1 in the half word
                                   at address 16.4 (B0 is
                                   always zero)

If b2 = 1.4 say, then
       113     1      2     16.4   stores b1 at 18.0

If b1 = 3.0 say, then with b2 = 1.4,
       374     1      2     16.4   divides the accumulator
                                   by the number at 21.0
```

Instructions are read into Atlas through the media of punched paper tape, either of 5-track or 7-track width, and 80-column punched cards. 7-track paper tape (the most common input medium) is prepared on a Flexowriter, 5-track paper tape on a Creed teleprinter and cards on a card punch. These three equipments have slightly different sets of characters. In the punched form, the parts of an instruction are punched as written and separated by "multiple spaces" or commas. Multiple space is two or more spaces on the teleprinter, or a tabulate (TAB) on the Flexowriter.

2.6 The Full Range of Atlas 1 Addresses

As explained in 2.4, address bits 1 to 11 represent the number of 512 word main store block to which that address refers, so that an Atlas 1 main store address refers to one of 2048 blocks. In octal (for the J notation see 4.3) these block addresses are J0000, J0001, J0002,...., J3777 and in their decimal representation (see 5.6) 0:, 1:, 2:,..., 2047:. The ABL compiler and the program it is compiling share the same range of block numbers, with ABL occupying blocks in the range J3 to J34. Consequently to avoid over-writing itself ABL will refuse to compile program into any of the 256 blocks 1536: to 1791: Once the program is compiled and ABL has withdrawn from the store these blocks become available again and can be used as working space by the program.

The remainder of the main store block numbers J34 to J4 are illegal. ABL will not store program in block J3 or above and the Supervisor will fault the program if the compiled program attempts to refer to block J34 or above.

In fixed store and private store addresses bit 0 is a 1. There are therefore another 2048 blocks that can be addressed and they have octal addresses J4000 to J7777. The first 16 of these (J4 to J4017) are the block numbers of the fixed store and may be referred to by the programmer if he wishes, although there is generally no reason why he should do so. The block numbers J4020 to J4777 are also quite legal. In effect these block numbers refer to 31 consecutively stored copies of the fixed store. For example either of the instructions

```
101     3     0     J4017777
101     3     0      -1J5
```

would place the same half-word in B3, namely the left half of the last word of the fixed store. In other words addresses in the range J4 to J5 are in effect masked with J40177777 before execution (for a definition of masking, or what is the same thing collating, see 4.3)

The private store block addresses J5 through J7777 are completely forbidden to the ordinary programmer. These addresses can be referred to on extracode control. However it is impossible for the programmer to force an extracode to refer to the private store. He is prevented by faulting an extracode instruction in which the modified address is in the private store but the unmodified address is not. Thus for example a program containing the instruction

```
1730     0     0     J6     am' = sin (J6)
```

would be thrown off because the first instruction of the extracode routine is

```
324     0     119     0     am' = (b119)
```

(B119 will contain the address J6 upon entry to the extracode. See 7.2.)