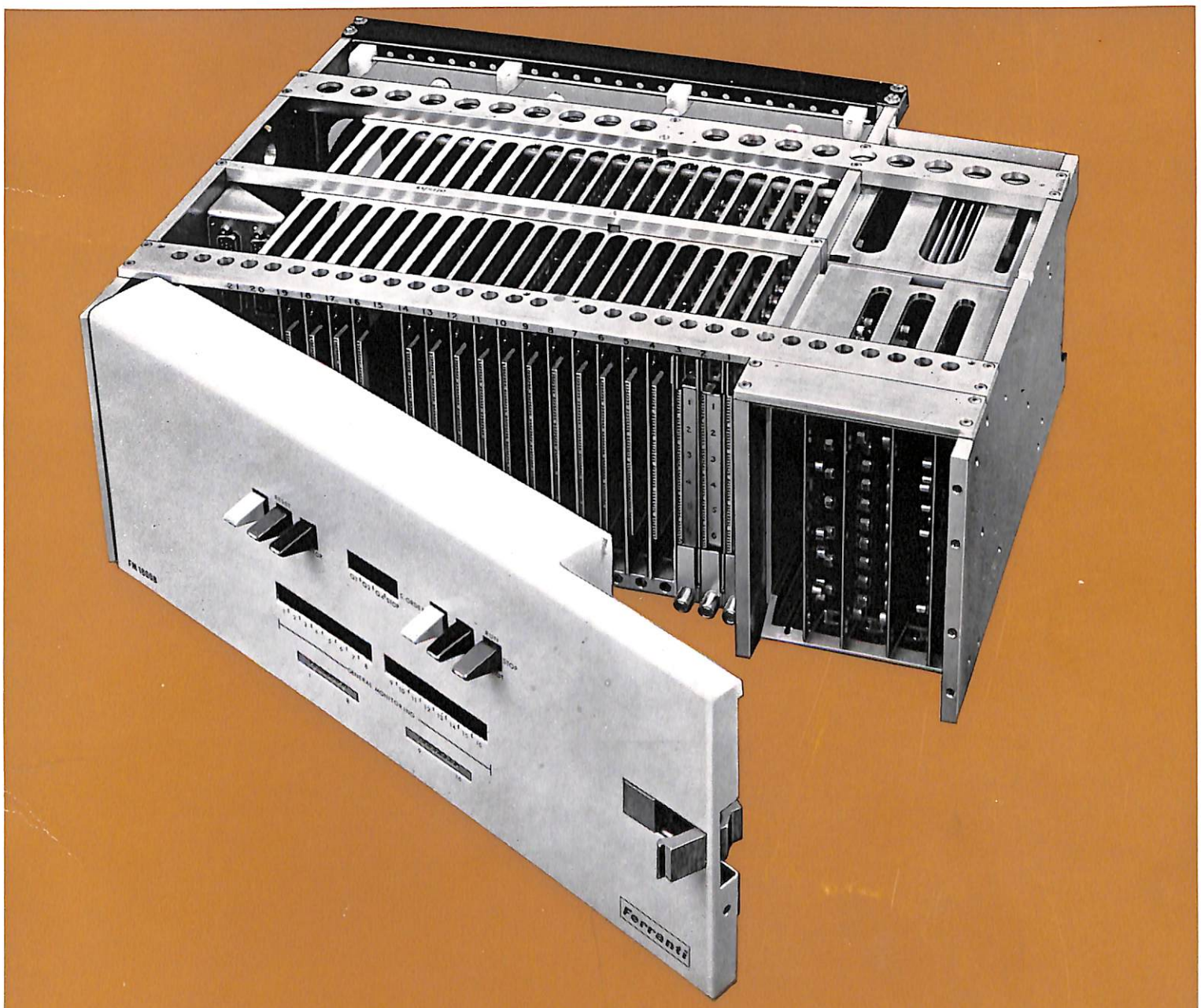
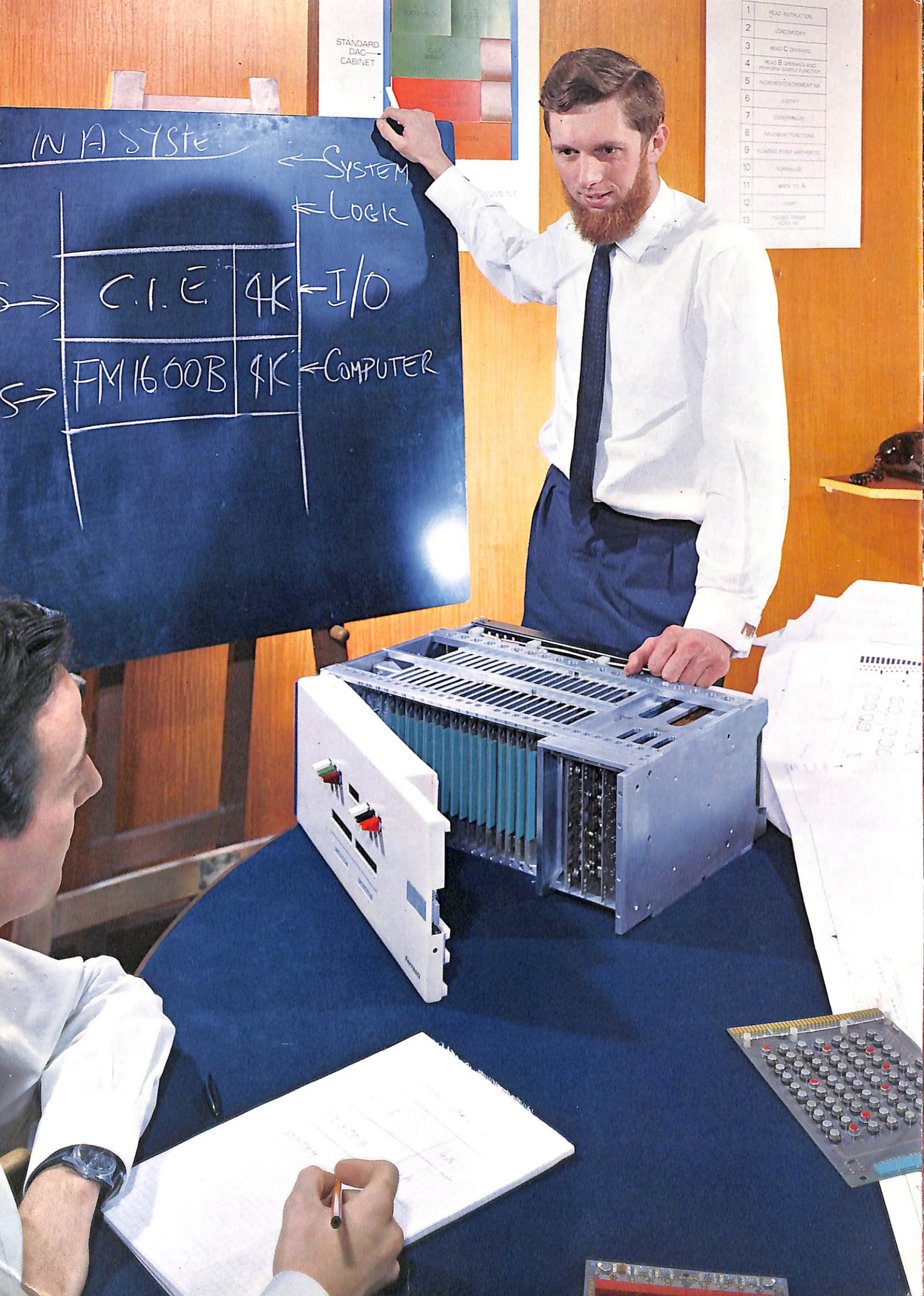
FERRANTI

FM1600B Microcircuit Computer



FERRANTI DIGITAL SYSTEMS


STANDARD DAC CABINET
SYSTEM
LOGIC
C.I.E
4K
I/O
FM1600B
4K
COMPUTER

Ferranti FM1600B Microcircuit Computer

Contents

FM1600B
Microcircuit Computer

Basic Features

Structure	Central Processor formed of 19, 6-layer printed circuit panels which plug into a 12-layer printed circuit Backboard.
Logic	Ferranti Micronor II circuits.
Size and Weight	Central Processor including 4 K,/1 μsec Ferranti store occupies one shelf 19×16×7 in. and weighs 40 lb.
Storage	Up to 65 536 words in 4096 word increments.
Word Length	24 bits.
Mode	Parallel.
Clock Rate	3 MHz. Each logic beat takes ⅓ μsec.
Arithmetic	Fixed Point and Floating Point.
Fast Shift	Single length shift, left or right, with or without end around carry takes ⅓ μsec.
Instruction Code	F1600, variable address.
Link Nest	Hardware 'push down' or link nest store.
Standard Interface	Up to 22 'Christchurch' Ferranti B input/output channels via modular Computer Interrupt Equipment.

Introduction

The FM1600B is the latest machine in the series of F1600 general purpose real-time digital computers developed by the Digital Systems Department of Ferranti Limited of Bracknell, Berkshire. It is the 'basic' version of the recently introduced FM1600 microcircuit computer and is intended for use in systems where the power of the larger machine is not required.

Certain new features relating to the basic control of the machine and to the design of the function unit, have been patented and incorporated into the FM1600B. As a result, the FM1600B is about one fifth the size and yet retains about half the power of the FM1600.

Together these two machines satisfy the requirements of real-time systems ranging from large multiple computer automatic data handling and control systems down to specific individual control applications in the military and civil fields.

The computers have been developed as two items in a range of modular equipment for handling inputs and outputs from a multiplicity of sources as encountered in :

- ★ **Naval Data Handling and Weapon Control Systems**
- ★ **Air Defence Systems**
- ★ **Message Switching Systems**
- ★ **Air Traffic Control Systems**
- ★ **Tactical Trainers**
- ★ **Simulators**
- ★ **Program Development Centres**

The modules all use similar circuitry and mechanical construction and cover such operations as :

- ★ **Automatic Radar Data Extraction**
- ★ **Shaft Angle Encoding and Decoding**
- ★ **Control of C.R.T. displays**
- ★ **Interfacing to digital data links**
- ★ **Interfacing to backing stores**
- ★ **Interfacing to line printers and digital plotters**

The computers have been engineered to meet British and NATO Defence specifications. These reflect the exacting environmental conditions of military systems, but it must be stressed that the computers are general purpose machines suitable for most on-line, real-time, civil and military applications.

The F1600 instruction code is used. This code is well proven, having been in use for more than five years' and the experience so gained is reflected in the performance of the FM1600B. The full range of compilers, the sub-routine library and general support software developed over this period is immediately available for use with the FM1600B.

Main Features

The FM1600B central processor is formed of 19, 6-layer printed circuit panels which plug into a 12-layer printed circuit backboard. On the panels are mounted Ferranti Micronor II integrated circuits, a fast DTL logic with typical signal rise times of 8 nsecs, in T05 cans. The panels and the backboard have the necessary transmission line characteristics for handling signals with such fast rise times, and the use of a multi-layer backboard allows much complex backwiring to be avoided.

The central processor, including one 4096 word block of Ferranti 1 μsec core store, complete with drive and address circuits, fits in a single shelf module of dimensions 19×16×7 in. (49 cm × 41 cm×18 cm). This is normally mounted as one shelf of a standard rack which holds six equipment shelves together with a common power supply and cooling unit.

For systems requiring up to 8192 words of store a second 4096 word block may be mounted immediately above the first block ; the remaining space being occupied by a ¾ in. shelf Computer Interrupt Equipment. When larger stores are necessary these 4096 word blocks are not fitted and the required storage is fitted as a separate one, two or three shelf module.

Basic Features

Word Length 24 bits.

Mode Parallel.

Representation Binary, two's complement for negative numbers.

Clock Rate The basic clock rate of the computer is 3 MHz. Logic beats are carried out at the rate of one beat every ⅓ μs.

Instruction Code F1600, variable address.

Weight The central processor with 4096 words of store weighs 40 lb. (18 kg).

Power Consumption

Central Processor	
Storage : First 4096 words	50 watts
For each additional 4096 words	110 watts
	12 watts

Storage Up to 65 536 words of core store, occupying three shelves, may be attached to the computer. At present Ferranti 1 μs storage is available in units of 4096 words. The computer is designed to make efficient use of store speeds of up to 600 nsec and storage of this speed, under development by Ferranti, will be available by 1969 in 16 384 word units.

Input Output The FM1600B input/output requirements conform to the Ferranti B (Christchurch) Standard Interface specification. Up to 20 Standard Interface input/output channels can be handled by the central processor, via modular Computer Interrupt Equipment.

Special Features

Fast Shift Facility

A new feature in the function unit of the FM1600B is the powerful fast shift network. In one pass through the function unit a 24-bit word can be shifted any number of places left or right with or without end around carry. The time taken for such a shift is one logic beat of ⅓ μs. Besides increasing the speed of shift instructions, this feature speeds up bit testing, bit changing, and normalising instructions. As about a fifth of the instructions in typical tactical and fire control programs are of this type significant speed increases are obtained.

Floating Point Arithmetic

Advantage has been taken of the fast shift facility to provide floating point arithmetic which is fast, automatic and cost effective.

Floating point instructions are specified in the Extracode section of the F1600 instruction code list. Operands are held in the computer packed into single words with the format :

Bits 0–5, Exponent
Bits 6–23, Mantissa

So far as is known the FM1600B is the only small computer currently available with floating point operation as a basic feature of the central processor.

Link Nest Store

An address pointer is provided which enables an automated link nest to be placed anywhere in the core store. Address or data words can be sequentially stored and extracted in the reverse order to that in which they were entered. This provides a neat method for handling entry and exit points of interrupt programs and sub-routines ; each time a program or sub-routine is interrupted the continuation address is automatically stored and, when the interrupt program has been completed, is extracted. This enables multi-activation of sub-routines to take place without increase in Supervisor program load, and enables economies in storage space to be realised, typically between 5 and 11 words of store for each sub-routine and interrupt program.

Single Address Function

A special Single Address Function has been built into the FM1600B. This function is specified in the Extracode section of the instruction code and by transferring data from random core store locations to an accumulator, extends the ability of the FM1600B to deal with the comparatively rare occasions on which true random access is required.

Instruction Times

Typical instruction times, assuming the use of 1 μs storage, are given below. These are for two and three address operations between directly addressable locations and are often equivalent to two or more instructions in a single address code. For this reason these times are not directly comparable with times quoted for single or one and a half address computers.

The full implications of this instruction format in relation to single and one and a half address operations are discussed in the next section.

	Time in Microseconds	
Function	Fixed Point	Floating Point
Add/subtract	2·7– 4·3	6·3–7·0
Multiply	11·3–13·3	14·0
Divide	13·0–16·0	15·3
Shifts	2·7– 8·7	—
Jumps	2·0– 5·7	—

Instruction Format

One of the most advanced features common throughout the F1600 series machines is the instruction code. The techniques used in this code offer a number of advantages of increasing significance to all types of real-time system.

In the development of the code, priority was given to three main factors:

(a)

The desirability of improved methods of addressing to make more efficient use of costly quick access storage. This has always been a major factor in achieving cost effectiveness, but now that the cost of a modern compact central processor has been reduced to little more than that of 4000 words of core store the efficient use of storage assumes even greater significance.

(b)

The need for a comprehensive function list so that maximum advantage may be gained from a powerful function unit.

(c)

The problem of developing computers of different sizes and capability but with complete instruction code compatibility.

The code has been well proven in over five years' use and its effectiveness is reflected in the performance of the FM1600B. Very high bit efficiency is achieved by the address structure and this, together with the value of a comprehensive function list, reduces the amount of program store required in representative systems. Recent studies have shown, for example, that for typical programs the FM1600B uses less than half the program store required by a single address computer of relatively limited function repertoire.

The F1600 code uses a multiple address format which caters for the maximum of three variables in typical autocode instructions and provides for both direct and indirect addressing. A maximum of 23 accumulators, the first 23 locations of the core store in the FM1600B, may be directly addressed and the main store may be indirectly addressed via index registers which hold the required full length addresses. A generous allowance of 9 bits for the function field makes for a comprehensive function list containing combinations of a main and a subsidiary function.

Three addresses and the large function field are handled within a single 24-bit instruction in the following format:

FUNCTION	3 ADDRESSES
F I D S	A B C
9 bits	5 bits each

F, I, D, S are the sub-fields which constitute the main and subsidiary function fields, and give 512 combinations, of which about 350 have been used in practice. Typically B and C identify the operands upon which the function is performed and A supplies the address at which the result is stored. In certain instructions, such as 'jump storing link', part of the S field and the A, B and C fields are used to hold the jump address.

In the F1600 code the required combinations of main and subsidiary functions have been selectively spaced throughout the 512 available codings, as opposed to being formed into a consecutive list. The positioning of the undefined codings has been exploited, in conjunction with the Executive Register, to simplify the computer control logic. The Executive Control System is explained in the next section.

The addressing structure of the code exploits the fact that address references in real programs are never distributed in a truly random fashion. In practice related data are located either in limited areas of store or in a patterned sequence ; consequently, addressing usually takes the form of random access within these limited areas or of a controlled sweep through the store. Most of the address of a required location can therefore be predicted in advance so that the specification of a full length address in each instruction would entail redundancy and consequent waste of storage space. In the F1600 code this redundancy is avoided by the use of short addresses and indirect addressing via index registers.

If it were necessary to use a fresh instruction to load an index register prior to every new store access, then the advantage of indirect addressing would be lost. It follows therefore that the efficiency of indirect addressing depends on the facilities available for setting and changing the contents of the index registers. These facilities are provided by three subsidiary functions which can be specified and carried out concurrently with the main function. These subsidiary functions are :

Load Functions These set an index register to a main store address. The loaded address is then immediately available for use as an operand source or destination in the main function.

Index Functions These increment or decrement the contents of an index register by unity.

Modify Functions These add a constant within the range of 0–31 to the address held in an index register prior to the address being used in the main function. The index register either retains its original contents or is given the modified value at the programmer's discretion. Thus either random access within a block of 32 words in the main store or a controlled sweep of regularly spaced stored data is available.

The effectiveness of these subsidiary functions is illustrated by a recent analysis which showed that of over 15 000 words of proven real-time software only 7% of the instructions used involved the loading of an index register.

Control Hardware

The Executive Register

Overall control of the computer is organised through a patented Executive Control System in which use is made of the fact that any instruction included in F1600 code can be executed by performing in a fixed sequence a sub-set of 12 basic computer actions. The system is centralised in a special 12-bit register, known as 'Executive', in which each bit corresponds to one such basic action. The basic actions within the Executive Register format are given in the following table.

FM1600B Executive Format

Executive Register Bit	Instruction
1	Load/Modify Index Registers
2	Read C Operand
3	Read B Operand and perform simple function
4	Increment/Decrement Index Registers
5	Justify
6	Denormalise
7	Multibeat functions
8	Floating point arithmetic
9	Normalise
10	Write to A
11	Jump
12	Pseudo-instruction

As the instruction is read from the store the function bits are inspected and the executive actions appropriate to the main and subsidiary functions selected. After completion of the read-instruction beat the relevant executive bits are set and, since these bits define not only the required actions but also their sequence, the instruction can then be processed automatically. The operations are always carried out in numerical sequence from 1 to 12 although, since typical instructions involve only three or four executive actions, only those called for are performed.

A simple priority network continuously inspects the executive register and defines the action currently being performed. On completion of this action the corresponding bit is cleared and the priority network automatically selects the next action required. When all the bits are clear the current instruction has been completed and the next instruction is accessed.

Two Address Instructions

In practice some instructions occur in which the destination address A is the same as address B. In the FM1600B the Executive Register Control takes advantage of this to save significant time in all simple functions. In instructions of this type, bit 10 (WRITE TO A) of Executive is not set. Instead the READ B OPERAND micro-action is extended by ⅓ μs to allow time for the function to be performed between the read and write halves of the store cycle.

Software

The Digital Systems Department incorporates a Programming Section employing upwards of one hundred programmers. The section has an in-house Computer Centre comprising a multi-access computer suite with a quad magnetic tape unit and a million word drum store. Up to four programmers may have simultaneous use of the suite which is also equipped with high speed line printers and digital plotters.

During the last five years' both Fixed Point and Floating Point Autocodes together with a large Standard Sub-routine Library, have been developed and used in many applications. In addition compilers for a number of high level languages including Algol, Fortran and Coral are available.

Autocodes:

FIXPAC

This is a Fixed Point Autocode, enabling computer programs to be written in a conveniently simple and economical form, saving the programmer the considerable effort required in machine language programming to compose complex sequences for direct input to the computer.

As shown in the brief description of the instruction format, the three address fields of the instruction code correspond to the maximum of three variables of a typical Fixed Point Autocode instruction. Programs written in FIXPAC therefore exhibit a close correspondence with programs written in F1600 machine code, and hence are virtually as efficient as machine code programs.

FLOPAC

This is a double word length Floating Point Autocode which has been designed to give flexibility in complex mathematical calculations. In FLOPAC the mantissa and exponent can each be up to 24 bits in length.

High Level Languages:

ALGOL

An Algol compiler which accepts a sub-set of Algol 60 is available. A full description is given in a separate publication entitled 'Ferranti Algol'.

FORTRAN

The version of Fortran implemented by Ferranti is based on the ECMA specification, and has been designed to provide a Fortran facility for a small machine. Compilation works from a paper tape, produces machine code in a single pass and can be used in a computer having 8000 words of store.

CORAL

CORAL is a real-time language which is derived from the well established JOVIAL language. The original definition of CORAL (called CORAL 64) was made by a committee at the Royal Radar Establishment, Malvern, England, primarily for use in the real-time radar data processing field. The design of Ferranti's implementation of CORAL is based on this specification, and on discussions with officers of the Royal Radar Establishment. It includes features not present in CORAL 64, for example, floating point arithmetic.

Great attention has been given to providing good 'debugging' facilities. Rigorous syntax and semantic checks are applied at compile-time and one of the intermediate codes produced can be run interpretively with extensive run-time diagnostic capability.

Input/Output

Computer Interrupt Equipment

The FM1600B central processor is designed to work with modular Computer Interrupt Equipment (C.I.E.) which is constructed in the same technology as the computer. The basic C.I.E. module is a three-quarter shelf unit capable of handling up to 12 Ferranti B (Christchurch) Standard Interface Channels. A larger, full shelf, C.I.E. module is also available which can handle up to 22 Standard Interface Channels.

Standard Interface

The Ferranti B (Christchurch) Standard Interface was designed to cater for the requirements of real-time systems in which a variety of fast independent peripherals may work together. These needs are met by ensuring that all transfers to or from the computer are entirely autonomous and independent of the program currently running. Transfers across this interface are carried out on a handshake basis. This avoids strict timing rules and restrictions on computer to peripheral cable lengths.

Each Standard Interface Channel can handle a single complex peripheral, such as a multi-console alpha-numeric display system, or a number of simpler peripherals multiplexed on to the one channel. Examples of groups of peripherals which can be serviced by a single channel are :

A few hundred single bit input and output lines.
A few tens of teleprinters.
A Master/Slave quad magnetic tape system.
A single Shaft Angle Encoder which accepts inputs from up to 28 shafts on a time multiplexed sampling basis and
eight Shaft Angle Decoders, each capable of accurately controlling the position of an independent shaft.

The autonomous transfer requests fall into two classes : Data Interrupts and Program Interrupts.

Data Interrupts

Data interrupts are allowed within instructions between executive beats and also between the individual beats of multi-beat functions, e.g. multiply, divide. This gives a maximum waiting time for data interrupts of 3 μs.

Since requests from peripherals are all independent and asynchronous, allowance has to be made for the presence of multiple requests. Prior to carrying out a data interrupt, therefore, a priority assessment of all requests present is performed. Advantage is taken of the fast shift capability of FM1600B to save the additional hardware which is normally required to carry out such priority assessments.

When one or more data interrupt requests are present, the C.I.E. seeks to interrupt the central processor. As soon as the interrupt is established (within 3 μs) a request word is input to the central processor. This word consists of the complete set of request staticisers and is normalised by the central processor. The number of shifts required to normalise this word is available within ½ μs and is equal to the channel number of the highest priority request present.

The time taken by each data interrupt is dependent on the number of store cycles required by the interrupt and on the number of extra logic beats required for indexing and address transfers. Since a handshake principle is used for the transfer of data across the standard interface, a further variable time may be added to the interrupt time depending on the distance between the computer and the peripheral. For peripherals within 40 ft. of the computer this time can usually be absorbed within the interrupt. The following are some typical times for peripherals within 40 ft. of the computer.

Fast Input	2 μs
Slow Input	3⅓ μs
Fast Output	3⅓ μs
Slow Output	4⅔ μs

When multiple requests of the computer are present at one time, the complete set of data interrupts required are carried out without any gaps between them.

Program Interrupts

Program interrupts are only accepted between instructions. The waiting time for the highest priority program interrupt, when no other interrupt is already in progress, is therefore no greater than the longest instruction time (19 μs).

When a program interrupt occurs, the interrupt staticiser, Q2, is set and all further program interrupts are locked out until Q2 has been cleared. It is normal practice to clear this staticiser within the interrupt program, as soon as any urgent work has been carried out. It should be noted that data interrupts are not locked out while Q2 is set.

The program interrupt mechanism is similar to that for data interrupts. An interrupt is established at the end of an instruction when one or more program interrupt requests are present. Prior to actually accepting the interrupt, the central processor carries out a store link procedure identical to that performed by the jump and store link process, except that Q2 is also set. Assuming no data interrupt requests have occurred during the store link process, a procedure similar to that of a data interrupt is entered on behalf of the program interrupt. The actions carried out are : the assessment of the program interrupt priority (identical to that for a data interrupt) ; the loading of the instruction number register with the first address of the interrupt program ; the transfer of the status word. The total time taken in establishing the interrupt and entering the new program is about 7 μs.

A Full Table of Fixed-Point Autocode Instructions

Operation	Description	Nominal instruction time in μs	
Basic Operations			
1. νA = νB	Transfer from register B to register A	3·3	
2. νA = νB + νC	Add contents of two registers	4·3	
3. νA = νB − νC	Subtract contents of two registers	4·3	
4. νA = νB & νC	AND operation (logical multiplication)	4·3	
5. νA = νB ≠ νC	NOT EQUIVALENT operation (logical addition)	4·3	
Multiplication			
6. νA = νB × νC	Double length multiplication – result in A and (A + 1)	13·3	
7. νA = νB × νC,F	Single length product (rounded) – fractional interpretation	12·3	
8. νA = νB × νC,I	Single length product (unrounded) – integer interpretation	12·3	
Division			
9. νA = νB/νC,Q	Division ; store quotient in A	15·0	
10. νA = νB/νC,R	Division ; store remainder in A	15·0	
11. νA = νB/νC,QR	Division ; store remainder in A and quotient in (A+1)	16·0	
12. νA = νB/νC,IQ	Integer division ; store quotient in A.	14·0	
13. νA = νB/νC,FQ	Fractional division, single length divisor ; store quotient in A	14·0	
Arithmetic Shifts			
14. νA = νB (νC),LD	Double length shift left νC places of ν(B,B + 1)	8·7	
15. νA = νB (νC),RD	Double length shift right νC places of ν(B,B + 1)	8·7	
16. νA = νB (νC),L	Single length shift left νC places of νB	6·7	
17. νA = νB (νC),R	Single length shift right νC places of νB (rounded)	6·7	
Logical Shifts			
18. νA = νB((νC)),L	Single length shift left νC places of νB	4·3	
19. νA = νB((νC)),R	Single length shift right νC places of νB	4·3	
20. νA = νB((νC)),E	Single length right end-around shift νC places of νB	4·3	
21. νA = νB((νC)),LD	Double length shift left νC places of ν(B,B + 1)	8·3	
22. νA = νB((νC)),RD	Double length shift right νC places of ν(B,B + 1)	8·3	
Special Shifts			
23. νA = νB → N	Normalize νB, number of shifts in A ; normalized form in (A + 1)	5·0	
24. νA = νB → C	Count number of ones in B	11·3	
Indexing			
25. νA = νB, ni ± 1	Transfer order with simultaneous indexing of specified N register (i = 1,2,3)	4·7	
26. νA = νB + νC, ni ± 1	Add with indexing	5·7	
νA = νB − νC, ni ± 1	Subtract with indexing		
νA = νB & νC, ni ± 1	AND with indexing		
νA = νB ≠ νC, ni ± 1	NOT EQUIVALENT with indexing		
Load			
27. ni = νA	Load index register Ni	7·3	
28. ni = νA, ni = νC	Load Ni and transfer		
29. ni = νA, ni = νB ± νC	Load Ni and write the result of an operation to main store		
ni = νA, ni = νB & νC			
ni = νA, ni = νB ≠ νC			
30. ni = νB, νA = ni + νC	Load Ni and write to store (Examples only)		
ni = νC, νA = νB − ni			
Modification (0≤K≤31)			
31. νA = ν(ni + K)	Modified transfer	7·3	
32. ν(ni + K) = νC	Modified transfer		
33. ν(ni + K) = νB ± νC	Modified write to store		
Bit Operations			
34. νB(νC) = 0	Clear bit νC of νB	4·0	
≠ 0	Set bit νC of νB	4·0	
35. QC = 0	Clear Q stat C	2·3	
≠ 0	Set Q stat C	2·3	
Unconditional Jumps			
36. → Label	Unconditional jump	2·0	
37. → Label, L	Unconditional jump storing link in nest	4·3	
38. → L	Obey link	3·7	
39. → Sn	Jump to subroutine n, normal entry	3·7	
40. → Sn, m	Jump to subroutine n, entry m	3·7	
Conditional Jumps			
41. → Label, νB ≥ 0	Test νB, jump if positive	2·7, 3·3	Longer time if jump > 16 places ; independent of whether satisfied
42. → Label, νB ± νC < 0	Jump if result < 0	3·7, 4·3	
43. → Label, νB & νC = 0	Jump if result = 0	3·7, 4·3	
44. → Label, νB = νB ± νC ≥ 0	Jump if result is zero and store result	4·0, 4·7	
45. → Label, νB = νB ≠ νC ≠ 0	Jump if result non zero and store result	4·0, 4·7	
46. → Label, νB(νC) = 0	Jump if specified bit clear	3·7, 4·3	
≠ 0	Jump if specified bit set	3·7, 4·3	
47. → Label, νB(νC) = 0	Jump if specified bit set and clear it	4·0, 4·7	
48. → Label, QC = 0	Jump if Q statiscisor clear	2·0, 2·7	
≠ 0	Jump if Q statiscisor set	2·0, 2·7	
Miscellaneous			
49. νA = DATA (Label)	Read labelled data and set address in N_1	7·7	
50. STOP C	Stop identified by number C		
51. WAIT	Optional Stop		
52. END	Last instruction in program		
53. νA · BIT νC	Clear νA and set it equal to bit νC only	4·3	
54. νA MASK νC	Set register to νC. Zeros at most significant end and ones elsewhere	4·3	
55. νA MASK νC	Set C to the inverse pattern of the above	4·3	

NOTE ALL the times quoted are between accumulators.

FERRANTI

For further details contact :

Ferranti Limited,
Digital Systems Department,
Western Road,
Bracknell,
Berkshire,
England.

Telephone : 0344 3232
Telex : 84117

List DSD/68/6. Printed in England PLH. September 1968. Revised October 1968