MCD/M/68/2

ELLIOTT

MOBILE COMPUTING DIVISION
ELLIOTT BROTHERS (LONDON) LIMITED
ELSTREE WAY . BOREHAMWOOD . HERTS
Telephone: 01-953 2030

Issue 2, October 1968

ELLIOTT

MCS 920 M FACTS

GENERAL INFORMATION

The Elliott M.C.S. 920M is a rugged, reliable, microminiature high speed computer. Its design meets the stringent requirements of airborne, naval and military applications where small size is essential. It has been chosen for the inertial guidance system of the EUROPA 1 satellite launch vehicle.

Manufactured by Elliott Automation, a company with twenty years' experience in the design and production of digital computing systems, the M.C.S. 920M has been developed from the successful M.C.S. 920B computer, of which a considerable number are now in service.

The 920M—which is functionally interchangeable with all 900 Series computers—embodies major advances in speed, reliability, cost effectiveness and flexibility, and, though smaller and offering an improved performance, uses the same order code so that proven types of software can be used.

The major design aims for the 920M computer have been:

1. High reliability
2. Reduction in size and weight
3. Reduction in power consumption
4. Ease of servicing
5. Competitive price

Reliability has been achieved by adopting a method of assembly which involves a minimum number of components, by reducing power wherever possible, and by ensuring adequate design margins on all components, so that they are generally operated at less than 10% of their rating.

The 920M incorporates the latest micro-circuit forms of the well proven DTL type of logic circuit which has proved so successful in previous 920 Series computers.

PHYSICAL CHARACTERISTICS

M.C.S. 920M	*Dimensions*	*Volume cu. ft.*	*Weight lb.*	*Power dissipation*
Computer with internal 8192-word store, 5 μS	12·6″ x 7·5″ x 7·5″ (32 cm x 19 cm x 19 cm) (Short ¾ ATR)	0·42	32	45 watts
Additional store, 8192 words	12·6″ x 7·5″ x 3·65″ (32 cm x 19 cm x 9.2 cm)	0·2	12·5	20 watts

SPECIFICATION

Type	General purpose.
Mode	Parallel.
Word length	18 bits.
Number representation ..	Binary—negative numbers stored as 2's complement.
Memory	8192 words internal store, extendable to 32,768 words.
Type	Random access
Store cycle	5 µS.
Order structure	Single address.
Order code	Up to 60 functions
Priority programming ..	4 levels of priority.
Operation times	Gross (including accessing the order and incrementing the SCR). Add 19 µS Subtract: 21 µS Multiply: 38 µS Divide: 39 µS

Environment

Ambient temperature range ..	–10°C to +55°C.
Non-derangement temperature	–40°C to +100°C.
Vibration	±7½g 0–2000 c/s (Type tested to ±10g).
Shock	25g without breaking loose from its mounting.
Altitude	No limit.
Humidity	0–95% RH.

The 920M conforms to the general requirements of DEF 133 A2 and MIL-E-5272C, for airborne equipments.

INPUT/OUTPUT EQUIPMENT

A wide range of compatible input/output units is available for the 920M. They include:

- Paper Tape and Teleprinter Controller
- Tape Reader (250 or 500 ch/sec.)
- Tape Punch (110 ch/sec.)
- Teleprinter
- Magnetic Tape Store
- Magnetic Drum Backing Store
- Military Program Loading Unit
- C.R.T. Displays
- Keyboards
- Digital Plotter

ANCILLARY EQUIPMENT

Other units associated with the operation of a 920M system include:

CONTROL UNIT

Provides either manual or automatic control of the computer's operation. Can also be used, in conjunction with the Display Unit, for system or program development, or to test the computer independently of its peripherals.

DISPLAY UNIT

Contains a full set of indicators to monitor the state of every register or control bistable within the processor.

MARGINAL TEST UNIT

Enables voltages, currents and timings in the computer to be manually varied above and below the specified limits; and thus makes possible a continuous check on the performance and reliability of the system.

FACTS FOR PROGRAMMERS

WORD FORMAT

In the computer each word of information consists of 18 binary bits which may represent either a computer instruction or an operand of an instruction.

In the arithmetic functions of addition, subtraction, multiplication and division, the absolute value attributed to each bit of the word is:

b^{18}	b^{17}	b^{16}	b^{15}	..	..	b^{4}	b^{3}	b^{2}	b^{1}
-1	$+2^{-1}$	$+2^{-2}$	$+2^{-3}$	..	..	$+2^{-14}$	$+2^{-15}$	$+2^{-16}$	$+2^{-17}$

The range of numbers that may be represented in the computer is from -1 to $+1-2^{-17}$.

When representing instructions, the 18 bits are divided into three integers representing the modifier B, the function F and the address N, as follows:

b^{18}	b^{17}	b^{16}	b^{15}	b^{14}	b^{13}	b^{12}	b^{11}	b^{10}	..	b^{1}
2^{0}	2^{3}	2^{2}	2^{1}	2^{0}	2^{12}	2^{11}	2^{10}	2^{9}	..	2^{0}
B		F					N			

$0 \leqslant B \leqslant 1 \qquad 0 \leqslant F \leqslant 15 \qquad 0 \leqslant N \leqslant 8191$

REGISTERS

The registers which are addressed by the various functions are as follows:

A Register:	This is an accumulator with a capacity of 18 bits which is used to hold the result of calculations prior to transfer to the store or output channel.
Q Register:	This is a register with a capacity of 18 bits which is used to hold information temporarily during a computation and also as an extension of the accumulator for numbers containing more than 18 significant bits. In this case it extends the less significant end of the accumulator by 17 bits, using bits 2 to 18 of the Q register. The register used in this way is known as the Auxiliary register.
B Register:	This 18 bit register holds the number which is added to the N bits of the instruction to form the address if the B bit of the instruction is a 'one'.
S Register:	This register, with a capacity of 16 bits, controls the extraction of instructions from the store; it is incremented as each instruction is extracted from the store so that instructions are obeyed in numerical sequence.

PROGRAM LEVELS AND INTERRUPT FACILITY

Provision is made for four program priority levels, which are arranged to operate on an interrupt basis. If while the computer is running an interrupt signal is received for a higher level program, the program currently being obeyed will be temporarily suspended while the higher level program is obeyed. When this higher level program is terminated by a program terminate instruction the computer will revert to the lower level program and continue from the point at which it was interrupted.

Each priority level has its own sequence control register (SCR) and modification register (B register). These registers are held in the computer store and can be referred to by program in the normal way. They are held in locations 0–7 as follows:

		SCR	B Register
Program level	1	0	1
	2	2	3
	3	4	5
	4	6	7

Program level 1 is the highest priority and cannot be interrupted.

Program level 2 may be interrupted only by level 1.

Program level 3 may be interrupted by levels 1 and 2.

Program level 4 may be interrupted by levels 1, 2 and 3.

Interruption occurs as a result of a signal received from a peripheral equipment.

TRACE FACILITY

The tracing of faults or errors in programs is simplified by use of the trace facility. The TRACE program is operated on program level 1 and the program under examination on program level 4. A repetitive interrupt signal is produced for level 1 and the timing of the interrupt logic is such that when program 1 terminates, a single instruction of level 4 program is obeyed before the computer returns to level 1. Thus the level 1 program is obeyed after each instruction of the level 4 program and is able to examine the effect of each level 4 instruction. The trace program may operate on any one of levels 1, 2 and 3 and may examine programs on any level lower than itself.

STORE ADDRESSING

The store is divided into blocks of 8192 words each; numbers or instructions may be placed in any location of any available block. The N bits of an unmodified instruction specify a location of the block in which the instruction itself is located; i.e. the absolute address of the location is formed by adding the N bits to bits 14-16 of the Sequence Control Register, the latter being those which specify the block in which the instruction is located.

INSTRUCTION MODIFICATION

The address specified by an instruction may be modified by adding the contents of the B register to the unmodified address prior to implementing the instruction. This does not affect the instruction held in the store. Where the instruction is required to be modified, this is signified by a one in bit 18 position of the instruction. Only the least significant 16 bits of the sum will be used as the address: by this means the range of storage locations which may be specified by the computer instructions is increased to 32,768 words.

INITIAL INSTRUCTIONS

Locations 8180 to 8191 inclusive are used to contain the initial instructions, which facilitate the reading of program tapes into the computer. These are permanently held in store and cannot be overwritten.

The initial instructions and their respective addresses are tabulated below:

Address		*Instruction*		*Effect*
'N' digits	'B'	'F'	'N'	
8180	/	14	8189	(–3)
8181		0	8180	(Set B-Register to –3)
8182		4	8189	(Set Accumulator initially)
8183		15	2048	(Shift and input tape character)
8184		9	8186	(Jump to 8186 if Accumulator is negative)
8185		8	8183	(Jump to 8183 if Accumulator is positive)
8186		15	2048	(Shift and input final tape character of word)
8187	/	5	8180	(Store word read in)
8188		10	0001	(Count in B-Register)
8189		4	0001	(Read B-Register)
8190		9	8182	(Jump to 8182 if Accumulator is negative)
8191		8	8177	(Jump to 8177 if Accumulator is positive)

Notes: When entered at 8181 the routine initially reads words into 8177, 8178 and 8179. Control is then transferred to location 8177. If these instructions set the B-Register to –n and then transfer control to 8182, words can then be read into the sequence of n locations ending at 8179.

INSTRUCTION CODE

The 18-bit word, when used to represent instructions, is made up of 1 modifier bit, 4 function bits and 13 address bits.
A comprehensive instruction code containing up to 60 functions is realised by extending the effects of the four function bits in the following three ways:

(a) By direct use of certain addresses with the 14 and 15 functions to specify different 'addressless' instructions, or instructions which require only a small part of the full address range.*

(b) By giving certain functions secondary effects which do not interfere with the primary effect but are nevertheless extremely useful.

(c) By taking advantage of the fact that the modifier register and sequence control register of all four program levels (including the one currently active) and the accumulator and auxiliary register of the three program levels not currently active are held in the computer store and can be addressed as for any other store location.

**Note* that for functions 14 and 15 only the 13 least significant bits of N are meaningful where N is the absolute address.
The resulting instruction set is given in the following tables:

Note: a represents contents of the accumulator (A).
b represents contents of the modifier register (B).
q represents contents of the Q register (Q).
q_{2-18} represents contents of the auxiliary register.
s represents contents of the sequence control register (S).
n represents any contents of store location N.

Any letter followed by ′ represents the new contents of the appropriate register or store location.

TRANSFERS BETWEEN REGISTERS AND STORE

Instruction	*Effect*	*Function and Address Representation*	*Specification*
Set A	$a' = n$	4 N	Place the contents of the store location specified by N in the accumulator.
Set B	$b' = n$	0 N	Place in the B-register of the current program level, the contents of the store location specified by N. This instruction places the same information in the Q-register.
Set Q	$q' = n$	0 N or 2 N	Place in the Q-register the contents of the store location specified by N. If instruction O is used the same information is placed in the B-register, but the accumulator is not affected. If instruction 2 is used, the B-register is unaffected but (n-a) is placed in the accumulator.

Instruction	*Effect*	*Function and Address Representation*	*Specification*
Store A	$n' = a$	5 N	Place the contents of the accumulator in the store location specified by N.
Store S	$n'_{1-13}=s_{1-13}$ $q'_{14-16}=s_{14-16}$	11 N	Place bits 1-13 of the sequence control register of the current program in the store location specified by N, setting bits 14-16 of this location to zero (bits 17 and 18 are not defined). Place bits 14-16 of the sequence control register in the Q-register, setting the remaining bits of Q to zero.
Store Aux.	$n'_{1-17}=q_{2-18}$	3 N	Place the contents of the auxiliary register in the least significant 17 bits of the store location specified by N. The most significant bit of the store location is made zero.

INTER-REGISTER TRANSFERS

Instruction	*Effect*	*Function and Address Representation*		*Specification*
A to B	$b'=a$	5	1 or 3 or 5 or 7 according to current program level.	Place contents of accumulator in B-register of current program level.
B to Q	$q'=b$	0	1 or 3 or 5 or 7 according to current program level.	Place contents of B-register of current program level in the Q-register.
S to B	$b'_{1-13}=s_{1-13}$ $q'_{14-17}=s_{14-17}$	11	1 or 3 or 5 or 7 according to program level	Place bits 1 to 13 of the sequence control register of the current program level in any B register, setting bits 14-18 to zero. Place bits 14-17 of the sequence control register in the Q-register, setting the remaining bits of Q to zero.
B to A	$a'=b$	4	1 or 3 or 5 or 7 according to current program level.	Place the contents of B-register in the accumulator.
Aux. to B	$b'_{1-17}=q_{2-18}$	3	1 or 3 or 5 or 7 according to current program level.	Place the contents of the auxiliary register in the least significant 17 bits of any B register, making the most significant bit of the B-register zero.

ARITHMETIC BETWEEN ACCUMULATOR AND STORE

Instruction	*Effect*	*Function and Address Representation*	*Specification*
Add n to a	$a'=n+a$	1 N	Add the contents of the store location specified by N to the contents of the accumulator.
Subtract a from n	$a'=n-a$	2 N	Negate the contents of the accumulator and add the contents of the store location specified by N. The contents of N are also placed in the Q-register by this function.
Multiply a by n	$(a\ aux)'=axn$	12 N	Multiply the contents of the accumulator by the contents of the store location specified by N, and place the result in the accumulator and the auxiliary register.
Divide (a aux) by n	$a'_{2-18}=(a\ aux)/n$ $a'_1=1$	13 N	Divide the contents of the accumulator and the auxiliary register by the contents of the store location specified by N and place the result in the accumulator, making the least significant bit a one so that it is correctly rounded to 17 bits. The result is also placed in Q, with the least significant bit a zero.
Collate a with n	$a'=a\ \&\ n$	6 N	Place ones in the accumulator in only those bit positions in which both the contents of the accumulator and the contents of the store location specified by N are ones.

ARITHMETIC BETWEEN ACCUMULATOR AND MODIFIER REGISTER

Instruction	*Effect*	*Function and Address Representation*		*Specification*
Add b to a	$a'=b+a$	1	1 or 3 or 5 or 7 according to current program level.	Add the contents of the B-register of the current program level to the contents of the accumulator.
Subtract a from b	$a'=b-a$	2	1 or 3 or 5 or 7 according to current program level.	Negate the contents of the accumulator and add the contents of the B-register of the current program level. The contents of B are also placed in the Q-register by this function.
Multiply a by b	$(a\ aux)'=axb$	12	1 or 3 or 5 or 7 according to current program level.	Multiply the contents of the accumulator by the contents of the B-register of the current program level, and place the result in the accumulator and and auxiliary register.

Divide (a aux) by b	$a'_{2-18}=(a\ aux)/b$ $a'_1=1$	13	1 or 3 or 5 or 7 according to current program level.	Divide the contents of the accumulator and the auxiliary register by the contents of the B-register of the current program level, and place the result in the accumulaor, marking the least significant bit a one so that it is correctly rounded to 17 bits. The result is also placed in Q, with the least significant bit a zero.
Collate a with b	$a'=a\ \&\ b$	6	1 or 3 or 5 or 7 according to current program level.	Place ones in the accumulator only in those bit positions in which both the contents of the accumulator and the contents of the B-register of the current program level are ones.

JUMPS

Instruction	*Effect*	*Function and Address Representation*	*Specification*
Jump if zero	If $a=0$ then $s'_{1-13}=N$ $s'_{14-17}=s_{14-17}$	7 N	If the contents of the accumulator are zero, place N in the sequence control register of the current program level.
Jump	$s'_{1-13}=N$ $s'_{14-17}=s_{14-17}$	8 N	Place N in the sequence control register of the current program level.
Jump if negative	If $a<0$ then $s'_{1-13}=N$ $s'_{14-17}=s_{14-17}$	9 N	If the contents of the accumulator are negative, place N in the sequence control register of the current program level.

INTERLEVEL INSTRUCTIONS

Any instruction which refers to a store location can also be used to refer to another level's sequence control register, modifier, accumulator or auxiliary register. Strictly this adds 48 possibilities (12 instructions referring to store, each of which can refer to any of the 4 stored registers). All these are not of general value, but 23 warrant inclusion.

Note: It may not be safe to refer to a priority level higher than the one currently active.

The asterisk * indicates the register of another priority level, the corresponding store location being as follows:

S * is location 0, 2, 4 or 6 according to level.

B * is location 1, 3, 5 or 7 according to level.

A * and Aux * are locations allocated by the programmer.

Instruction	*Effect*	*Function Representation*
A to A *	$a^{*\prime} = a$	5
A to B *	$b^{*\prime} = a$	5
A to Aux. *	$aux^{*\prime} = a$	5
A to S *	$s^{*\prime} = a$	5
A * to A	$a' = a^*$	4
B * to A	$a' = b^*$	4
Aux * to A	$a' = aux^*$	4
S * to A	$a' = s^*$	4
B * to B	$b' = b^*$	0
S * to B	$b' = s^*$	0
S to S *	$s^{*\prime}_{1-13} = s_{1-13}$	11
Add A * to A	$a' = a^* + a$	1
Add B * to A	$a' = b^* + a$	1
Add Aux. * to A	$a' = aux^* + a$	1
Add S * to A	$a' = s^* + a$	1
Subtract A from A *	$a' = a^* - a$	2
Subtract A from B *	$a' = b^* - a$	2
Subtract A from Aux.*	$a' = aux^* - a$	2
Subtract A from S *	$a' = s^* - a$	2
Collate A * with A	$a' = a\ \&\ a^*$	6
Collate B* with A	$a' = a\ \&\ b^*$	6
Collate Aux.* with A	$a' = a\ \&\ aux^*$	6
Collate S * with A	$a' = a\ \&\ s^*$	6

SHIFT DOUBLE LENGTH

Instruction	*Effect*	*Function and Address Representation*	*Specification*
Left Shift AQ	$(aq)'=aqx2^{N}$	14 N Where $0 \leqslant N \leqslant 36$	Shift the double-length contents of A and Q left N places (multiply by 2^{N}).
Right Shift AQ	$(aq)'=aqx2^{-N}$	14 8192-N Where $1 \leqslant N \leqslant 36$	Shift the double-length contents of A and Q right N places (divide by 2^{N}).

MISCELLANEOUS

Instruction	*Effect*	*Function and Address Representation*	*Specification*
Increment Store	$n'=n+1x2^{-17}$	10 N	Increment the contents of the store location specified by N by 2^{-17}.
Increment Modifier	$b'=b+1x2^{-17}$	10 1 or 3 or 5 or 7 according to current program level.	Increment the B-register of the current program level by 2^{-17}.
Read Paper Tape Character	$a'_{1-7}=t_{1-7}$ $a'_{8}=t_{8}$ OR a_{1} $a'_{9-18}=a_{2-11}$	15 2048	Read one 8 bit character from paper tape into accumulator. Shift the previous contents of the accumulator left 7 places and place the input character in the 8 least significant bits of the accumulator.
Punch Paper Tape Character		15 6144	Output the least significant 8 bits of the accumulator to the tape punch.
Block Transfer into store		14 N where $2048 \leqslant N \leqslant 4095$	Transfer n words of information from the device specified by bits 1-11 of N into store locations a to a+n–1, where a is the contents of the accumulator and n is the contents of the Q-register. $(n \leqslant 4095)$. The contents of A and Q are altered.

Instruction	*Effect*	*Function and Address Representation*	*Specification*
Block Transfer out of store		14 N where $4096 \leqslant N \leqslant 6143$	Transfer n words of information to the device specified by bits 1-11 of N from store locations a to a+n–1 where a is the contents of the accumulator and n is the contents of the Q-register. $(n \leqslant 4095)$. The contents of A and Q are altered.
Peripheral word input		15 N where $0 \leqslant N \leqslant 2047$	Input to the accumulator one 18 bit word from the device specified by bits 1 to 11 of N.
Peripheral word output		15 N where $4096 \leqslant N \leqslant 6143$	Output from the accumulator one 18 bit word to the device specified by bits 1 to 11 of N.
Program Terminate	$s'=s^{*}$ $b'=b^{*}$	15 7168	Terminate current program level.
Read Teleprinter	$a'_{1-7}=t_{1-7}$ $a'_{8}=t_{8}$ OR a_{1} $a'_{9-18}=a_{2-11}$	15 2052	Input one 8 bit character from the teleprinter. Shift the previous contents of the accumulator left by 7 places and place the input character in the least significant 8 bit positions of the accumulator.
Output to Teleprinter		15 6148	Output the least significant 8 bits of the accumulator to the teleprinter.

INSTRUCTION TIMES

The following are nominal instruction times in micro-seconds. They apply within ±10% over the whole operating temperature range.

Function	*Operation*	*Time µS* *Un-modified*	*Modified*
0	Set B register	22	28
1	Add	19	25
2	Negate and Add	21	27
3	Store Aux Register	20	26
4	Read	19	25
5	Write	20	26
6	Collate	19	25
7	Jump if A zero (A = +ve)	17	23
	(A = –ve)	16	22
	(A = 0)	21	27
8	Jump	19	25
9	Jump if A negative (A > 0)	15	21
	(A < 0)	19	25
10	Count in store	20	26
11	Store S.C.R.	25	31
12	Multiply	38	44
13	Divide	39	45
14	Shift (n places)	16+n	22+n
,,	Block transfer (n words)	18+10n	24+10n
15	Input-output	20 (min.)	26 (min.)
,,	Program terminate	20	26

SOFTWARE

The M.C.S. 920M is program-compatible with all other 900 Series computers.

Again with compatibility in mind, the IFIP subset of ALGOL 60 was chosen for implementation on the 920 Series. ALGOL, the international language that allows programs to be written in a combination of English words and mathematical expressions, greatly increases the computing power of the 920M and also permits the interchange of programs with different computers.

There is also a facility which allows ALGOL programs to call up procedures written in SIR code.

The use of SIR (Symbolic Input Routine) enables programs to be written in modified machine code form, with addresses referred to by invented names. It simplifies editing, as the insertion or removal of instructions does not require the wholesale modification of addresses in the rest of the program.

SYMBOLIC INPUT ROUTINE

Symbolic Input Routine (SIR) enables programs to be written in modified machine code form. Essential details of SIR are given below. Full details of the routine can be obtained from the SIR Handbook.

SIR Symbols used for the Major Facilities

(i)	IDENTIFIER Group of up to six letters or numbers commencing with a letter.	
(ii)	CONSTANTS	
	(a) Integer or fraction	± e.g. –71032
	(b) Octal groups	& e.g. & 770123
	(c) Alphanumeric groups	£ e.g. £ a23
(iii)	ADDRESSES	
	(a) Absolute	an unsigned integer
	(b) Block relative address	N;
	(c) SIR relative	;±N
	(d) Identified address	Identifier ± Integer (if required)
	(e) Literal address	Any constant as in (ii) above or = followed by an instruction (absolute address only)
(iv)	SKIPS	+ N
(v)	COMMENT	( ) e.g. (entry point)
(vi)	OPTION	*+N where N is the sum of the following:

Value	*Effect*
1	Punch list labels
2	Load and go operation (otherwise punch relocatable binary tape).
4	Clear the store up to the assembler.
8	Punch a binary loader tape.
16	Continue assembly at location 8.
32	Set dictionary below the program.

(vii)	PATCHES	+ N
(viii)	END OF SIR PROGRAM	%

When more than 1 tape is used only the last tape ends with % the others end with 'stop code' on a new line.

Note: Every new line symbol must be followed by at least four blanks.

START ADDRESSES

Initial assembly of S.I.R. tapes.

The S.I.R. assembly tape is read in under initial instructions. S.I.R. tapes are then read in at:

6017 (1011110000001) (START) for the first tape of a program.

6018 (1011110000010) (CONTINUE) for all other tapes.

The binary loader tape is read in by initial instructions. The RLB tapes can then be read in at:

7816 (1111010001000) (START) A)	for the first tape of a program.
7817 (1111010001001) (START B)	for other tapes of the same program.
7818 (1111010001010) (START C)	for the first tape of a new program when the previous program is to be left in the store.

ERROR INDICATIONS

Assembling

E0	F > 15 or quasi-instruction wrong.
E1	Contextual error.
E2	Octal or alphanumerical error.
E3	Label used twice.
E4	Global identifier list error.
E5	Store full or patch error.
E6	Number overflow.
E7	Buffer overflow.
E8	Illegal character.
E9	Stop code not first on line.
EU	Unlocated identifier.

Loading

FA } FD }	Misread or mispunched tape.
FC	Label used twice.
FE	Store full.
FF	Check sum failure.
UF	Unlocated identifier.