MOLECULAR 18

Hardware Reference Manual

This manual applies only to configurations with serial numbers above 7000.

202(5.73) ISS.1

Issued May 1973

The policy of Business Computers Limited is one of continuous research and development of its products and services and therefore the right is reserved to alter the information contained in this manual without notice.

While every effort is made to ensure the accuracy of the information contained in the manual, Business Computers Limited does not accept liability for any error or omission and the figures quoted in the manual do not form part of any contract or agreement.

SECTION 1 - Computer Description

List of Contents

COMPUTER DESCRIPTION

Introduction

The BCL Molecular 18 enhanced version is a high speed, modular processor designed to provide maximum computing power at minimum cost to the user. It is a general purpose computer system able to operate in a normal office environment; the cycle time is 1.2, 1.4 or 1.6 microseconds and an 18-bit word length is incorporated, the 18th bit being used as a parity check.

The primary storage of the computer is a "core memory", composed of magnetic cores which record binary information by the direction in which they are magnetized. The core memory system is organised into Pages, each containing 1024 words (1K). There are several sizes of core available ranging from 4K to 32K in modules of 4K.

Peripherals

The input/output hardware allows a program to address up to sixty-three peripherals, and Section 4 of this Manual describes many of these in detail. The following list shows some of the peripherals available with the enhanced version :-

various types of input/output writers
paper tape or card input/output
discs
magnetic tape units
optical mark readers
etc.

Software

The Molecular 18 enhanced version is supported with a programming package to ease the task of the programmer. Basic software programs include the Assembler, Operating System, Linkage Editor, Library Update and Debug.

Assembler

The MK.II Assembler allows the user to write efficient and time-saving symbolic programs in modular form, which are translated by the Assembler into Object Modules. The Assembler accepts a source module consisting of user-coded instructions, and outputs it as an object module which can then be linked to Library modules, etc. by the Linkage Editor.

The Assembler is designed to operate with a minimum of 16K of core memory, and can be adapted to almost any configuration of I/O devices to optimise assembly speed. In addition to op-code mnemonics, Assembler Mk.II has the capability of accepting user symbols, and has a flexible constant definition format.

Operating System

The operating system provides an efficient loading and input/output control capability for relocatable programs produced by Assembler Mk.II. The system is modular in design and may be constructed to fit each user's hardware configuration. Its general purpose is to take over from the operator and programmer the execution of a number of ordinary error-prone tasks, and to organise the running of each program in the most efficient way possible. The main functions of the operating system are to :-

a) Monitor all peripheral transfers
b) Organise and control multi-programming
c) Control, execute or initiate all communications with the operator via a control peripheral device.
d) Execute Extracode functions.

Linkage Editor

The Linkage Editor provides a means of combining the relocatable object modules generated by Assembler Mk.II with each other, automatically linking in any required Library Modules, to form a loadable program.

Library Update

The Library is a set of standard modules for use by any program. The Library can be created and updated by use of the Library Update routine, which can also delete any out-of-date library modules.

Debug Package

Debug is an interactive octal debugging aid which contains many valuable program testing features. Among these are the display and modification of memory locations and/or the accumulators. Also included is a breakpoint feature which allows the programmer to set, reset and recognise breakpoints throughout his logic and enter his program at any point.

COMPUTER DESCRIPTION

The basic Molecular 18 system includes the Central Processing Unit or CPU, the core memory and an input/output coupler to connect the processor to the peripheral chassis.

The CPU is the control unit for the whole system: it governs all peripheral in-out equipment, performs all arithmetic, logical and data handling operations and also sequences the program. It is connected to the core memory by a memory bus and to the peripheral equipment by an in-out bus. In order to store, retrieve, control and modify information and to perform the required logical, arithmetic and data processing operations, the core memory and the CPU employ the logic complement described in the following paragraphs :-

Core Memory

Core stores are used primarily for holding program, but also provide a fast random-access storage medium for data to be processed or distributed. Each core stack used in the processor has a 4K capacity, holding 4096 18-bit words. A word of 17 binary digits or bits (plus a parity bit which may be disregarded) can represent signed or unsigned binary quantities from Ø to 1111111111111111 (i.e. 65,535 in decimal). The bits of a word are numbered 17 to 1, left to right, as are the bits in the registers or accumulators that handle the words. The 'leftmost' 1 in a binary number is called the most significant digit, the least significant digit being the extreme right digit (Bit 1).

Core stores used for numeric data use Bit 17 as a sign bit, its presence indicating that the number is negative (see Fig.1.1 below).

Numeric Data Word

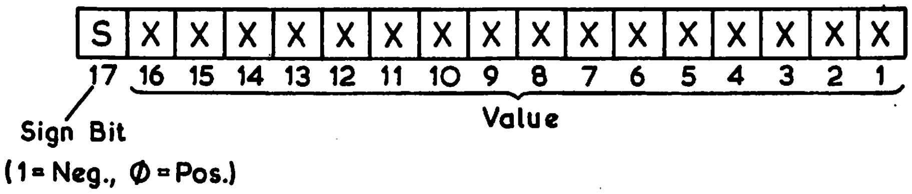


Fig. 1:1

Accumulators

Two internal registers of importance to the programmer are the 17-bit accumulators, Registers A and B, which are used by program for logic and arithmetic operations and can also be addressed as Steps 00 and 01 respectively on Zero Page. It is important to note that these are in fact flip-flop hardware registers and not part of the core storage. They are registers each capable of storing a number made up of sixteen binary digits plus a sign bit, and are called accumulators because they can accumulate partial sums during the operation of the computer. Data can be moved in either direction between any core memory location and either accumulator. Although words in the core memory can be incremented or decremented, all other arithmetic and logic operations are performed via the accumulators, the accumulator receiving the result. In addition, all programmed information transfers between the processor and the peripherals are passed through these registers, with the exception of data-channelling.

Accumulator A - this holds the results of arithmetic and logical operations such as Boolean logic operations, comparison for equality with any core store, shifting or rotating of bits left or right, testing individual bits, complementing, swapping the left and right halves (Bits 16 to 9 with Bits 8 to 1) and testing the contents for a skip. It is also used for transfers to and from peripheral devices, and by the Privileged Instruction 'Mask' for masking out.

Accumulator B - this has almost the same capabilities as Accumulator A, except that the three Boolean logic instructions (ANDA, IORA and XORA) and also the Privileged Instruction 'Mask' can apply only to Accumulator A, while the Privileged Instruction 'ACKINT' can apply only to Accumulator B. Also, on a JSBR instruction, the return address is always placed in Accumulator B.

Carry Flag

Associated with the Accumulators is the Carry flag (sometimes referred to as the Overflow). This is a one-bit register which is used to extend the arithmetic facilities of the Accumulators. When it is in a non-zero state it could indicate either that a carry has occurred from Bit 16 to Bit 17 of an accumulator, or that an overflow has occurred from Bit 1 downwards (as in Right Shift, see Page 3:25). Under program control the Carry flag may be cleared, complemented, tested or rotated as part of an Accumulator.

Greater Than flag

There is also an indicator known as the Greater Than flag, which is associated with the CMPA and CMPB memory reference instructions (see Page 3:14). Under program control, this flag may be tested for, cleared or set by certain Register instructions.

Program Counter (PC)

The PC is a 16-bit register which is used to control the program sequence; in other words, the order in which program instructions are performed is determined by the PC. The PC register contains the current relative program address, the address of the core memory location from which the instruction will be taken. When the instruction has been completed,the PC is incremented by one, the information in the PC then being transferred to the Memory Address Register (MA) to determine the core memory address from which each instruction is taken. Sequential program flow can be altered by changing the contents of PC, either by incrementing it by two, as in a test skip instruction, or by replacing its contents with the address specified by a JUMP or JSBR instruction (see Page 3:5).

Memory Address Register (MA)

The MA is a 16-bit register which is used for absolute core addressing. When using the Operating System, the MA = PC + Base. Bits 12 to 1 are used for selection of core words, while Bits 13 to 16 are decoded to select the correct core stack. The MA will differ from the PC during the processing of multi-phase instructions, since the MA will be changed by memory references in the instruction (which may be several in the case of indirect addressing), whereas the PC remains constant until completion of the instruction.

Memory Buffer Register (MB)

The MB is an 18-bit register that is used for all information transfers between CPU registers and core memory. The 18th bit is used for parity checks.

Instruction Register

This is a 5-bit register which contains the operation code (Bits 17 to 13) of the instruction currently being performed by the machine. The five most significant bits of the current instruction are loaded into the Instruction Register from the MB. The contents of the Instruction Register are decoded to produce the conditions for program execution. This register is not accessible by program or via the control panel.

DCH Memory Address

This is a 16-bit register which is used purely for core access during data channelling; it is not accessible either by program or via the control panel.

Base Register

This is a 16-bit register which contains the start address of whichever program is running in the machine at the time. The Executive always occupies an area of core starting from the beginning of the Zero Page. However, each program written for the machine is assembled as if from Address Zero also, so that all memory instructions will refer to addresses 'relative' to the start of the program. In order that the hardware can convert these 'relative' addresses to actual machine addresses, the absolute address is obtained by adding the contents of the base register to the relative address. In normal operation the base register will be set to any value which is a multiple of 128.

When an interrupt occurs, the base register is set to zero after stacking.

Limit Register

This is a 16-bit register which, on Unstacking, is loaded with Word 00 of the program, which always specifies the length of the program at present running in the machine. If a program attempts to access a relative address which exceeds this limit, an internal interrupt will be called, eventually causing the program to go to location 000214 (see Page 1:10).

Addressing Mode Memory

This is a one-bit memory which is provided for purposes of indicating and controlling the addressing mode, such that absolute addressing or relative addressing may be achieved. The Addressing Mode Memory will be set either to 1 or Ø :-

Privileged Mode (i) When set at 1, absolute addressing is achieved (and Privileged Instructions enabled).

User Mode (ii) When set at Ø, the relative addressing mode is achieved, such that any address contained in an instruction has the contents of the Base Register added to it, and is further checked against the set 'Limit' before that instruction is carried out. Should an instruction be found to exceed the set limit, an internal interrupt occurs.

When mains is first switched on, the Addressing Mode Memory will be set to the 1 (absolute addressing) state. Thereafter :-

a) When any interrupt occurs, the contents of the Addressing Mode Memory is propagated into the sign bit (Bit 17) of the 2nd word of the appropriate Interrupt Stack. The Addressing Mode Memory itself will then be set to the '1' state.

b) The 'Unstack' process will use the sign bit of the 2nd word in the appropriate Interrupt Stack to set the Addressing Mode Memory back to the correct state.

Arithmetic Logic Unit (ALU)

The ALU is that part of the processor which can perform various logic and arithmetic functions on the data retrieved from store locations addressed by the PC; the operands are located in the accumulators and/or core memory. Any operands specified are fetched from core or accumulators, the operation carried out and the results then stored. Arithmetic data is fixed point and is treated as a signed sixteen-bit integer. Extended precision arithmetic may be accomplished via the Library software routines. Negative numbers are represented in two's complement form with a sign bit of one, the numeric value of zero, however, always being deemed non-negative.

The time taken to 'fetch' a word of data from core, operate on it and to regenerate it in core is called the machine cycle time; for the Molecular 18 processor a typical time is 1.6 µs. Each machine cycle enables a data transfer from and to core of one word, the significance of which is determined by the setting of the Phase Control Register, which governs the application of this data :-

Phase 0 - Idle - There is no processing in this phase; the machine waits for a Start signal from the Control Panel switches (see Page 2:2) to continue processing. The Load/Examine switches may be operated in this phase (see Phases 6 and 7 below).

Phase 1 - Fetch - Assuming the computer is in a fully automatic, running condition, core access in this phase results in a word transfer from core to the Memory Buffer and Instruction Register, from where it is decoded to set the required processing conditions for successful operation. During the 'Fetch' phase, Control, Register and I/O instructions may be performed, as no further access is required. At the end of this phase for these instructions, the PC is incremented and the CPU restarts Phase 1 for the next instruction.

For memory instructions, further core cycles must be initiated to retrieve the operand as specified by the address in Bits 10-1 of the instruction. However, if Bit 12, the indirect bit, is set, an additional core cycle is required prior to execution to locate the true address of the operand. This will cause Phase 2 to set.

Phase 2 - Indirect - This phase is used if a binary 1 is present in Bit 12 of a memory reference instruction decoded in Phase 1. This state causes the CPU to obtain the full 16-bit address of the operand from the address in either the current page or Page Zero, as specified by Bits 11 to 1 of the instruction. Should the word accessed during this phase have the indirect bit (Bit 17) set, Phase 2 will be repeated; this can go on almost indefinitely, except that if more than 15 indirects are specified an internal interrupt will be called, causing the program to go eventually to location 000213. If the indirect bit of the accessed word is not set, the processor will set Phase 3 at the end. The one exception to this is an Indirect Jump or JSBR which can be executed in Phase 2 and will then return to Phase 1 for another instruction.

Phase 3 - Execute - The Execute phase is entered for all memory reference instructions except JUMP and JSBR. Memory instructions may be completed in this phase after accessing core for an operand or modifying a core word, depending on the instruction to be performed. At the end of this phase the PC will be incremented, the processor then returning to Phase 1 for the next instruction unless there is an Interrupt or Data Channel Request pending.

Phase 4 - Interrupt - When an interrupt occurs, this phase sets the Memory Address Register to 000030 to access the Interrupt Stack Pointer address, which is read from core and sets the MA to the address of the current Interrupt Stack. The next five machine cycles are taken up by the stacking of the Interrupt Stack, the Interrupt Stack Pointer address is then incremented by five, and the MA set to the appropriate Interrupt Address (see Page 1:10).

Phase 5 - Data Channel - Devices requiring high speed I/O data transfers may 'steal' machine cycles at any time during processing by raising Data Channel Request, which for one machine cycle enables one core word to be accessed for direct transfer to or from a device. Other control lines determine the direction of data flow and the core address involved.

Phases 6 and 7 - Control Panel - These two phases are used for Control Panel Load/Examine operations (see Page 2:4), and can only be reached from Phase 0. i.e. while there is no processing in progress. When the Load/Examine Mem. Next switch is operated, Phase 6 increments the MA, and then sets Phase 7 to access core.

Memory Organisation

Some areas of memory are dedicated to certain hardware and software functions when in the Absolute Addressing Mode. The following paragraphs described these areas, giving their purposes and specific locations in octal :-

000000 Accumulator A, as described on Page 1:4

000001 Accumulator B, as described on Page 1:4

000002 The Memory address for Auto-Start.

When power is restored to the Molecular 18 after a Mains fail, interrupt is off.

If the Key switch is in Normal Mode, PC and MA will both be set to 000002 and the program will immediately commence from this point. If Interrupt is then turned on without first performing the instruction 'Skip if Mains Return Interrupt' (see Page 3:21), a Mains Return Interrupt will occur, which will eventually send the program to the Interrupt Address 000207 (see Page 1:10).

If the Key switch is in Manual Mode, PC and MA will be set to 000002 as above, but the machine then awaits further orders.

000010 to 000017

When a location between and including 000010 and 000017 in Page Zero of the core memory field is addressed indirectly (when in Absolute Addressing Mode) by any Memory Reference instruction except JUMP and JSBR, the contents of that location are taken as the effective address of the instruction, then incremented by one and rewritten in the same location. This feature is called Auto-Increment.

e.g. If location 000012 contains the octal number 001234, and the instruction STA I 000012 is given, the contents of Accumulator A will be deposited in core memory location 001234 and the number 001235 then rewritten in location 000012.

000020 to 000027

When a location between and including 000020 and 000027 in Page Zero of the core memory field is addressed indirectly (when in Absolute Addressing Mode) by any Memory Reference instruction except JUMP and JSBR, the contents of that location are taken as the effective address of the instruction, then decremented by one and rewritten in the same location. This feature is called Auto-Decrement.

000030 - The Interrupt Stack Pointer

Associated with each interrupt line (level) is a 5-word set of consecutive memory locations situated anywhere in the core memory. Accordingly, on the Molecular 18 enhanced version, 40 words of core are reserved as there are eight interrupt levels. These reserved core addresses are referred to as 'the Interrupt Stack' and are accessed from the Interrupt Stack Pointer at 000030. When the processor honours an interrupt, it saves the following information in the appropriate position in the Interrupt Stack :-

Word 1 - Carry and Greater Than flags (Bit 1=Carry, Bit 2=G.T.)
Word 2 - Program Counter (+ Addressing Mode in Bit 17)
Word 3 - Base Address
Word 4 - Contents of B Register
Word 5 - Contents of A Register

The Interrupt Stack Pointer address (000030) is incremented by five, and the computer then causes the next instruction to be read from a fixed memory location, according to the type of interrupt it was. This location in memory is referred to as the 'interrupt address', and is reserved for that particular type of interrupt. A routine can thus be entered to perform the process required by the interrupt. In the enhanced version of the Molecular 18, the built-in Executive program will handle all hardware interrupts.

000205 to 00217 - Interrupt Addresses

000205	Continuous Interrupt Switch Interrupt
000206	MA=SW Interrupt
000207	Mains Return Interrupt
000210	Extra Code Interrupt
000211	I/O Interrupt
000212	Illegal Operation Interrupt
000213	More than 15 Indirects Interrupt
000214	Memory Boundary (Limit) Interrupt
000215	Memory Parity Interrupt
000216	Mains Fail Interrupt
000217	Overstack (Stack Full) Interrupt

Input/Output Hardware

The input/output hardware allows the program to address up to sixty-three peripheral devices, which are connected to the CPU through a highway known as the input/output bus. The 'bus' system allows one set of data and control lines to communicate with all I/O devices; no additional connections to the computer are required.

The peripheral devices are 'slow' compared with electronic operations within the CPU. The control unit deals with them by starting them and then turning to another instruction while waiting for them to complete their operation. A single instruction can transfer a word between an accumulator and a device and at the same time control the device operation. Each device is linked to the I/O bus by an interface board, which usually holds buffers for temporary storage of data in or out. Through this buffering, Input/Output can be overlapped with processing to take full advantage of available processing time. The devices empty or fill these buffers at their own pace, the buffers then transferring their contents at very much higher electronic speeds to or from the Accumulators; this avoids the tying up of a working register during the relatively 'long' transfer periods. The actual number of buffer bits (up to 17 flip-flops) will depend on the device for which the interface is intended. If the buffer is less than 17 bits in length, data is transferred to or from the least significant bits of Accumulators A or B (bits 8-1). Data is transferred one character at a time in most cases, between the interface buffers and core via the Accumulators, the transfer logic being controlled by the CPU.

Included in the in-out system, as mentioned before, are facilities for program interrupts and high speed data transfers. The interrupt system facilitates processor control of the peripheral equipment by allowing any device to interrupt the normal program flow on a priority basis.

While most devices are addressed directly by the CPU, disc drives and magnetic tapes are accessed through a device controller, controlling up to four devices, although only one of these may be accessed at any one time.

High-speed devices such as disc, magnetic tape, line-printer, can gain direct access to memory under data channel control. This is connected to the appropriate devices through interface boards which contain buffers and also addressing logic, allowing more than one character at a time to be transferred directly to or from core without using the accumulators. The CPU carries out the transfer by 'stealing' machine cycles during instructions without disturbing program execution.

E.C.M.A. (European Computer Manufacturers' Association) No.6 code is used wherever possible for the exchange of information between various peripherals and between peripherals and processor. This code is the U.K. version of I.S.O. No. 7 and I.T.A. No. 5., all these codes being based on ASCII.

Peripheral devices which operate directly connected to the CPU via the I/O bus are said to work 'on line'. No operations at all may be carried out 'off program'. The input and output on writers are completely separated, and no writer can be used when the CPU is switched off. Likewise, no regeneration of tape can be achieved 'off program'.

Octal Notation

It will be apparent that the binary number system, although suitable for computers, is rather complex from the programmer's point of view. To overcome this problem the Octal or Base 8 numbering system can be used to make the 17-bit words of the Molecular 18 easier to handle; indeed in Debugging or in object program print-outs all references to instructions or addresses are made in octal notation. Each binary word is divided into 6 groups, 5 3-bit and 1 2-bit, each group being represented by a single octal digit, using the table of equivalents shown in Fig. 1:2 :-

Decimal-Octal-Binary Equivalents

Decimal	Octal	Binary
0	0	000
1	1	001
2	2	010
3	3	011
4	4	100
5	5	101
6	6	110
7	7	111

Fig.1:2

A binary number 1101111010111101 is separated into groups as stated previously, by starting from the least significant digit end (the extreme right hand end) of the number, and supplying leading zeros if necessary.

i.e. 01 101 111 010 111 101

Each binary group is then replaced by its octal equivalent :-

01	=	1
101	=	5
111	=	7
010	=	2
111	=	7
101	=	5

the binary number 1101111010111101 thus being converted to its octal equivalent, 157275.

An octal number can be expanded to a binary number using the same system and table of equivalents.

e.g. 105307 Octal = 01000101011000111 (Binary)

Note :-

It should be remembered that only binary numbers are recognised by the computer, and that the Octal representation used for purposes of simplicity does not itself exist within the computer.

SECTION 2 - Computer Controls

List of Contents

COMPUTER CONTROLS

On the processor console is a set of data switches through which an operator can supply words and addresses to the program. This is :-

The Control Panel

The Manual Control Unit consists of a panel which is situated in the chassis. It provides all necessary manual control over the computer.

The panel includes the following set of control elements as indicated in Fig. 2:1 on the next page :-

1	Power On/Off pushbutton
1	Key Operated Security Lock
7	Power-Pack Indicators
27	Indicator Lamps
17	Data Switches
14	Control Switches

A functional description of each of these follows :-

The Power On/Off button should be pushed in to switch on the CPU - it will then lock in position and light up. To switch off, the button should be pushed once again.

The security lock is a two-position, Key operated locking switch which can disable all switches on the Panel except the 17 Data Switches, which can then only be accessed from program by the Register Instruction 'Enter Switch Register into Accumulator A or B'. This lock eliminates any accidental manual input to the system.

In the Normal position the Key is ejected and, when switched on, the Program will start from address 000002 (which is set on PC and MA) with Interrupt Off.

When in 'Manual' Mode, all Control Switches on the Panel are enabled. To change to Manual operation the Key should be inserted and turned towards 'Manual', leaving it in this position. The program will not then start automatically, although the PC and MA will be set to address 000002 when first switched on.

CONTROL PANEL

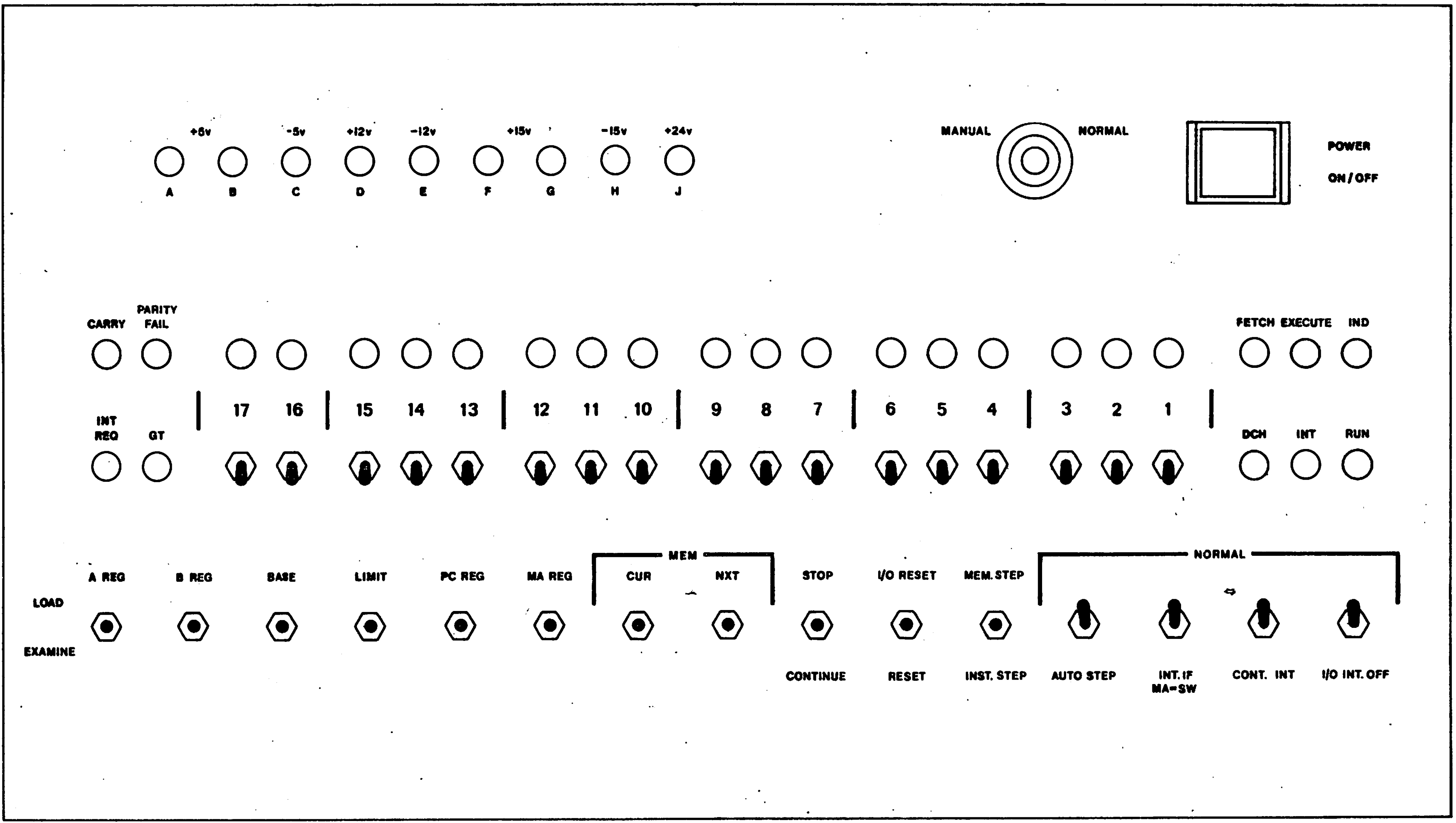


Fig.2.1

The 9 blue lights on the top left-hand side are power supply indicators.

The 6 lights on the right-hand side, below the Power On/Off button, display control conditions. A few of these indicators display useful information while the processor is running, but most change too frequently, and are therefore described in terms of the information they display when the processor has stopped.

The top 3 lights, Fetch, Execute and Indirect, also the DCH light in the bottom row, are phase indicators in that they specify the phase (the type of cycle) the processor will enter if operations are continued.

Fetch indicates that the next machine cycle will be used to 'Fetch' an instruction from memory.

Execute indicates that the next processor cycle will be used to reference memory for an operand in a move data or modify memory instruction.

Indirect indicates that the next processor cycle will be used to Fetch an address word in an indirectly addressed memory reference instruction.

DCH indicates that the next processor cycle will be used by data channel for direct access to memory by an in/out device.

Interrupt indicates that an Interrupt is being taken.

Run indicates that the processor is in normal operation with one instruction following another. This light goes off when 'Stop' is pressed, when a 'Halt' instruction is executed, or when an abnormal change occurs in the internal power supplies.

There are twenty-one (21) other indicator lights, nineteen of them being in a row above the Data switches. The red light on the extreme left-hand side is equivalent to the Carry flag. The red light next to it, when on, indicates a parity error has occurred in the memory system, the parity bit being used by the system for internal checking purposes.

The other 17 lights in this row are 'bit indicator' lights numbered 17 to 1 from left to right, forming a register with Bit 17 being the most significant and Bit 1 the least significant. This register can represent and display the contents of any hardware register or core location.

Below the bit indicator lights is a set of Data switches used for direct communication with the central processor. This is a set of 17 toggle switches used to specify binary numbers, which are then either loaded into the appropriate registers when other switches on the console are operated or loaded into an accumulator at Program command (ESWRA or ESWRB). The 17 positions represent a 17-bit binary word. When a switch is up it represents a binary '1' and is considered set; conversely, when a switch is down it represents a binary '0' and is considered reset.

Note :- The bit indicator lights do not change or light up as the Data switches are operated.

On the left of the Data switches two more indicator lights are situated. The one on the extreme left, when on, indicates that an Interrupt has been requested. The other indicator light is equivalent to the Greater Than flag.

The data switches may be used in conjunction with any of the 8 bottom row switches on the left hand side. These are operating or control switches and are interlocked so that if they are operated by mistake while the security lock is in the Normal position, they will have no adverse effect on the running program. Each of the operating switch levers is actually a three positional spring-loaded switch with a common off position in the centre. Lifting a switch lever up loads the contents of the data switches into the specified register. Pressing a switch down displays the contents in the bit indicator lights, thus enabling the user either to examine what has just been loaded or to examine the contents of the specified register.

e.g. Starting at the left-hand side:-

1) With switch labelled 'A.Reg', the operator can load and/or examine the contents of the A Register.

2) With switch labelled 'B.Reg', the operator can load and/or examine the contents of the B Register.

3) With switch labelled 'Base', the operator can load and/or examine the contents of the Base Register.

4) With switch labelled 'Limit', the operator can load and/or examine the contents of the Limit register.

5) With switch labelled 'P.C.Reg', the operator can load and/or examine the contents of the Program Counter, which represents the address of the next instruction to be fetched out of memory.

6) With switch labelled 'M.A.Reg', the operator can load and/or examine the Memory Address, which represents the address of the memory word being examined or loaded.

7) With switch labelled 'CUR', having previously loaded the Memory Address with the required address, the operator may now load the Memory Buffer and core with the contents of the data switches and/or examine the contents of the current location.

8) With switch labelled 'NXT' having once loaded the "Current" address with data from the switches, the operator may continue loading the next consecutive word by resetting the Data switches to the required pattern and pressing the 'NXT' switch up. This operation may be repeated as many times as necessary, because each Load operation of the 'NXT' switch increments the MA by 1.

Each of the next three switch levers is also a three positional spring-loaded switch with a common off position in the centre. Lifting the lever up initiates the condition printed above it. Pressing it down initiates the condition printed below :-

STOP When the switch is pressed up, the STOP switch halts program execution at the end of the instruction in progress, when the run light will go off.

CONTINUE When the switch is depressed, the run indicator light will be turned on, and the processor will begin normal operation, executing the instruction at the location specified by both the PC and MA.

I.O. RESET When the switch is raised, the I.O. Reset switch clears the control flip-flops, including Busy, Done and Interrupt Disable, in all devices connected to the I/O Bus.

RESET When the switch is depressed, the Reset switch causes a Mains Return condition, i.e. PC and MA will be set to 000002, a Mains Return Interrupt will be pending and all peripherals cleared down as in I/O reset.

MEM-STEP Each time the switch is raised, the processor executes instructions one memory cycle at a time in the state indicated by the phase indicator lights, displaying the MA on the bit-indicator lights, and then stops. The phase indicator lights will then indicate the next phase to be executed.

N.B. The use of the Examine A, B, PC, etc. switches between memory steps within an instruction, can display information without altering the conditions necessary for the successful execution of the remainder of the instruction.

INST.STEP When this switch is depressed, the processor will execute one complete instruction from wherever the PC and MA have previously been set. In other words it will operate from the beginning of a FETCH, completing the current instruction. The address displayed on the bit indicator lights after each Instruction Step is the address of the next consecutive instruction word.

Now follows a description of the 4 toggle switches in the bottom row, right hand side. These are kept in the 'up' position for the normal running of the processor and are only depressed when in use.

AUTO-STEP This switch provides slow speed operation of the Instruction Step (an average of 25 instructions per second).

INT IF MA=SW Having first set an address on the Data switches, this switch may then be depressed; the program is then set to Continue until the MA equals this address, when an internal interrupt is called.

N.B. This equalisation could happen during any machine phase, but the interrupt would not occur until the instruction had been completed.

CONT.INT. When this switch is depressed, an internal interrupt is called after every step, assuming Interrupt is on.

I.O. INT.OFF When this switch is depressed, all Input/Output external interrupts are disabled, (but internal processor interrupts are still enabled).

SECTION 3 - Program Instructions

List of Contents

PROGRAM INSTRUCTIONS

Introduction

There are three types of basic instructions which are grouped according to the bit format of the instruction word. These types are :-

a) Memory Reference Instructions which deal mainly with the transfer of information to/from Accumulators A and B to/from the core stores.

b) Register and Control Instructions which deal mainly with shifts, clears, rotates, negative tests, zero tests, etc. on the Accumulators.

c) Literal Instructions (I/O, Shifts, etc.) All Input/Output instructions are privileged, while the literal functions deal with multiple shifts and rotates.

The following pages describe in detail each of the instructions in the three groups. Functions of bits appearing in the form A/B, Z/C, I/D, L/R, T/F, S/R, L/S or A/L throughout these specifications are invariably obtained by coding a 1 or Ø respectively (1/Ø). Thus, for example, A is specified by a one-bit, and B by a zero-bit. The following defines the abbreviations used :-

A/B	Accumulator A/ Accumulator B
Z/C	Zero Page/Current Page
I/D	Indirect/Direct
L/R	Left/Right
T/F	True/False
S/R	Shift/Rotate
L/S	Long/Short
A/L	Arithmetic/Logical

a) MEMORY REFERENCE INSTRUCTIONS

The 22 Memory Reference instructions carry out some operation involving core locations, such as transferring information in or out of a core memory location or checking the memory location contents. Memory Reference instructions have the following general format in Machine Language, as shown in Fig.3:1 below :-

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
OPERATION CODE(02-27)					I/D	Z/C	MEMORY ADDRESS									

Fig. 3:1

The instruction word uses five bits (Bits 13-17) to encode the 22 instruction commands in this group. The address referenced is determined by a combination of ten memory address bits (Bits 1 to 10), plus two bits (Bits 11 and 12) to identify one of four addressing modes :-

Direct in Page Zero
Direct in Current Page
Indirect through Page Zero
Indirect through Current Page

Because there are only ten bits available to specify the memory address, a Memory reference instruction can directly reference only Octal 4000 words, 2000 on the Zero Page (the base page, consisting of locations 000000 through 001777) and 2000 on the Current Page (the page in which the instruction itself is situated). Bit 11 in a Memory Reference instruction specifies one or the other of these two pages as follows :-

1 = Zero Page
Ø = Current Page

a page being defined as the largest block of memory which can be addressed by the ten memory address bits of a Memory instruction. The Molecular 18 core memory is logically divided into pages of 1024 words (decimal) each. Octal addresses of the pages are as follows :-

Page Ø	-	000000 to 001777
Page 1	-	002000 to 003777
Page 2	-	004000 to 005777
Page 3	-	006000 to 007777
Page 4	-	010000 to 011777
and so on through to :-		
Page 77	-	176000 to 177777

It will be noted that each page starts on an even thousand (Octal). Appendix 15 lists the octal addresses of all pages from Ø to 77 in full.

To address locations in any page other than Zero or Current, indirect addressing is used via the Zero or Current Page, as explained beneath :-

Indirect/Direct Addressing

Memory Reference instructions include a bit (Bit 12) reserved to specify direct or indirect addressing. Direct addressing combines the operation code and the effective address into one word, permitting a Memory Reference instruction to be executed in two machine cycles (Fetch and Execute), except for Jump, which takes only one machine cycle. Indirect addressing uses the address part of the instruction to access another word in memory which is taken as a new memory reference for the same instruction. This new address is a full 17 bits long, 16 bits of address plus another bit (Bit 17) which is used as a further Indirect/Direct bit. The 16-bit length of the address permits access to any location in up to 64K of core. If Bit 17 again specified indirect addressing still another address is obtained; this multiple-step indirect addressing could be carried out to any number of levels (up to 15), but this of course would not be practical. The first address obtained in the Indirect phase which does not specify another indirect level becomes the effective address for the Memory instruction. Memory instructions with indirect addresses are therefore executed in a minimum of three machine cycles (Fetch, Indirect and Execute).

Indirect or Direct addressing is specified by Bit 12 (or Bit 17) as follows :-

1	=	Indirect
Ø	=	Direct

Note that since Accumulators A and B can be addressed, a Memory Reference instruction can apply to either of these registers, both directly or indirectly (except in the case of a direct JUMP or JSBR), as well as to core stores.

Fig. 3:2 gives instruction codes and mnemonics for all Memory Reference instructions :-

		17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Mnemonic *	Octal	INSTRUCTION					I/D	Z/C	MEMORY ADDRESS									
JUMP △	02	0	0	0	1	0												
JSBR △	03	0	0	0	1	1												
INSZ △	04	0	0	1	0	0												
DESZ △	05	0	0	1	0	1												
ANDA △	06	0	0	1	1	0												
IORA △	07	0	0	1	1	1												
XORA △	10	0	1	0	0	0												
ADA △	11	0	1	0	0	1												
ADB △	12	0	1	0	1	0												
SFA △	13	0	1	0	1	1												
SFB △	14	0	1	1	0	0												
ADAC △	15	0	1	1	0	1												
ADBC △	16	0	1	1	1	0												
SFAC △	17	0	1	1	1	1												
SFBC △	20	1	0	0	0	0												
LDA △	21	1	0	0	0	1												
LDB △	22	1	0	0	1	0												
CMPA △	23	1	0	0	1	1												
CMPB △	24	1	0	1	0	0												
STA △	25	1	0	1	0	1												
STB △	26	1	0	1	1	0												
UNSTK △	27	1	0	1	1	1												

Fig. 3:2 Memory Reference Instructions

* The Mnemonic code is meaningful to and translated by the Assembler into binary code. In Usercode, Indirect is specified by attaching I to the function mnemonic before the space, which is mandatory.

There now follows an explanation of each instruction in the Memory Reference Group. In each individual instruction description, the name will be given first, followed by the appropriate assembler mnemonic.

Unconditional Jump - JUMP

Unconditional JUMP to the word of core specified by the operand. The JUMP instruction loads the effective address of the instruction into the Program Counter (PC), thereby changing the program sequence, since the PC specifies the next instruction to be performed. The next instruction is then taken from that location, the program continuing operation from there. The JUMP instruction does not, in any way, affect the contents of the Accumulators. It should be noted that a JUMP Direct to an Accumulator will give unpredictable results.

Jump to Subroutine - JSBR

On this instruction the program will Jump to the word of core specified by the operand. During execution of the instruction the current PC is incremented by one and stored in Accumulator B, replacing or over-writing any original contents. After the subroutine is executed, this pointer address (of the JSBR instruction + 1), in other words the return or link address, identifies the next instruction to be executed. Thus, the programmer has at his or her disposal a simple means of exiting from the normal flow of the program to perform an intermediate task, and a means of return to the correct location on completion of that task, perhaps ending with a Jump Indirect via Accumulator B. It should be noted that a JSBR Direct to an Accumulator will give unpredictable results.

Increment, and Skip if Zero - INSZ▵

This instruction adds one to the contents of the word of core specified (the operand), all 17 bits, and then examines the result of the addition.

If the result is zero, the instruction following the INSZ is skipped and the Carry flag will be set.

If the result is not zero, the program will proceed normally to the instruction immediately following the INSZ (the next word of program in sequence).

The following points should be kept in mind with reference to the INSZ instruction :-

1) The contents of Accumulators A and B are not disturbed unless the instruction specifies A or B as the operand.

2) The original word in the referenced memory location is replaced by the incremented value.

3) The INSZ instruction performs the incrementation first and then checks for a zero result.

4) The Carry flag is set whenever a transfer occurs between Bits 16 and 17 in any store as a result of the INSZ instruction.

Decrement, and Skip if Zero - DESZ▵

This instruction subtracts one from the contents of the word of core specified (the operand), all 17 bits, and then examines the result of the subtraction.

If the result is zero, the instruction following the DESZ is skipped.

If the result is not zero, the program will proceed normally to the instruction immediately following the DESZ (the next word of program in sequence).

Should the specified store overflow as a result of this instruction the Carry flag will be set.

The following points should be kept in mind with reference to the DESZ instruction :-

1) The contents of Accumulators A and B are not disturbed unless the instruction specifies A or B as the operand.

2) The original word in the referenced memory location (in other words the operand), is replaced by the decremented value.

3) The DESZ instruction performs the decrementation first and then checks for a zero result.

4) The Carry flag is set whenever a transfer occurs between Bits 17 and 16 in any store as a result of the DESZ instruction. (e.g. If the operand contains Bit 17 only and is decremented, it will afterwards contain Bits 16 to 1 inclusive, and the Carry flag will be set).

"And" to A - ANDAΔ

The ANDA instruction causes a bit-by-bit Boolean AND operation between the contents of Accumulator A and the contents of the operand specified. The result is left in Accumulator A, replacing its original contents, but the operand specified is not altered.

Figure 3:3 below, showing a simple circuit with two switches, explains the AND operation. Should current be allowed to flow through a switch, the switch is said to have a value of '1'. Where current cannot flow through because the switch is open, the switch is said to have a value of 'Ø'. When the whole circuit is considered it will be noted that current may only flow through it (therefore giving it a value of '1'), when both switches are '1'. This is the AND operation.

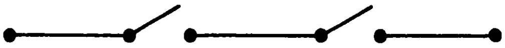


Fig. 3:3

When this is applied to binary numbers, a binary 1 will result if a binary 1 appears in the relevant position of the two numbers, but a Ø will result if only one of the bits has this value. Accordingly, the ANDA instruction can be used to Mask Out a portion of a number or word.

```
e.g.          To be       : retained for
            Masked   Out  :  future use
                          :
            10111010101   : 010101          (17-bit word in Accumulator A)
            00000000000   : 111111          (Mask in operand)
            00000000000   : 010101          (Result left in Accumulator A)
```

The following points sum up the ANDA instruction :-

1) A '1' is left in Accumulator A only when a '1' is present in the corresponding position of both Accumulator A and the specified word of core (Mask).

2) The Carry flag is not affected, neither is the Greater Than flag, as the operation is performed on a bit-for-bit basis.

3) The operand specified remains unaltered.

Acc. A	Operand	Result in A
O	O	O
O	I	O
I	O	O
I	I	I

Fig. 3:4

"Inclusive OR' to A - IORAΔ

The IORA instruction causes a bit-by-bit Boolean OR (or Inclusive OR) operation between the contents of Accumulator A and the contents of the operand specified by the IORA instruction. The result is left in A, replacing its original contents, but the operand is not altered.

The circuit diagram in Figure 3:5 illustrates the use of the OR connective when two words are combined. It can be seen that current is allowed to flow to Z whenever EITHER X or Y switches (or both) is closed.

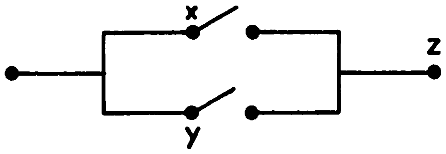


Fig. 3:5

i.e. Z = 1 if X = 1 or Y = 1
Z = 1 if X = 1 and Y = 1

Correspondingly, the IORA instruction results in the value of 1 if either or both relevant bits of Accumulator A and the operand is 1.

e.g.	
A contents	110110
Operand	011010
Result (in A)	111110

The following points sum up the IORA instruction :-

1) A '1' is inserted in Accumulator A if a 1 is present in the corresponding position of <u>either</u> Accumulator A <u>or</u> the operand.

2) The Carry flag is not affected, neither is the Greater Than flag.

3) The specified word of core (operand) remains unchanged.

Acc. A	Operand	Result in A
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 3:6

"Exclusive OR" to A - XORAΔ

The XORA instruction causes a bit-by-bit Boolean "Exclusive OR' operation between the contents of Accumulator A and the contents of the operand specified by the XORA instruction. The result is left in Accumulator A, replacing its original contents, but the operand specified is not altered.

The circuit diagram in Fig. 3:7 below illustrates that the Exclusive OR is similar to the Inclusive OR with the exception that one set of conditions has been altered or excluded. This exclusion has been shown by connecting the two switches mechanically together so that it is impossible for them to be closed simultaneously, although they may be open simultaneously or singly.

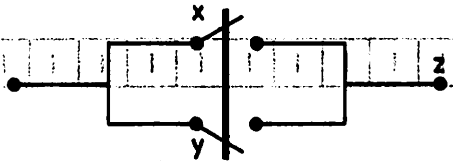


Fig. 3:7

i.e. $Z = 1$ if $X = 1$ and $Y = \emptyset$

$Z = 1$ if $X = \emptyset$ and $Y = 1$

Correspondingly, the XORA results in the value of 1 if only one of the relevant bits of Accumulator A and the operand is 1.

e.g.		
	A Contents	110110
	operand	011010
	Result (in A)	101100

The following points sum up the XORA instruction :-

1) A '1' is inserted in Accumulator A only if the two corresponding bits of the Accumulator and the operand differ.

2) The Carry flag is not affected, neither is the Greater Than flag.

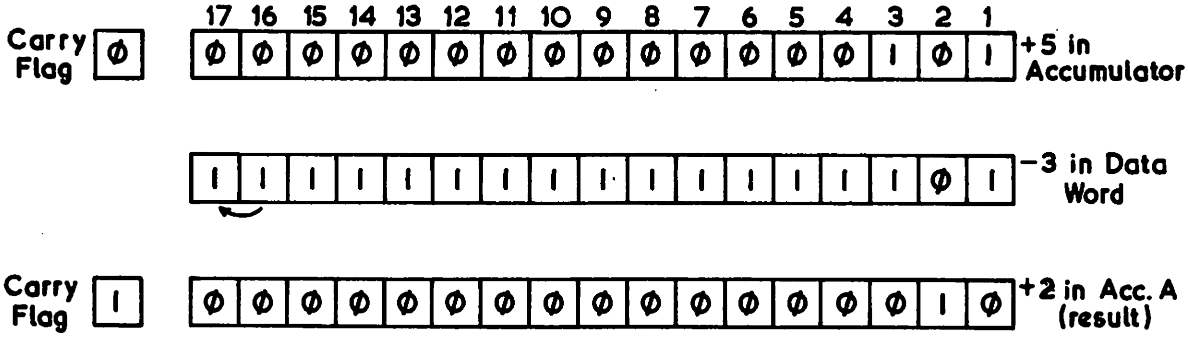
3) The operand remains unaltered.

Acc. A	Operand	Result in A
O	O	O
O	I	I
I	O	I
I	I	O

Fig. 3:8

Add to A - ADA△

The ADA instruction performs a binary addition between the specified word and the contents of Accumulator A, all 17 bits, leaving the result of the addition in Accumulator A. The specified word of core will remain unchanged, but the result of the addition could set the Carry flag. The following figure (Fig. 3:9) illustrates the operation of the ADA instruction :-



		17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Carry Flag	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	Ø	1	+5 in Accumulator
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Ø	1	-3 in Data Word
Carry Flag	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	Ø	+2 in Acc. A (result)

Fig. 3:9

In the example above, a carry has occurred between Bits 16 and 17, and this has caused the Carry flag to be set. This could be detected with a Register Instruction 'Skip if Carry'. Arithmetically the sign bit (Bit 17) behaves in exactly the same way as the other 16 bits. As can be seen above, the most significant bit of the answer was lost off the end of the register, as would also happen if two stores containing Bit 17 only were added together.

Add to B - ADB△

The ADB instruction performs a binary addition between the specified word and the contents of Accumulator B, all 17 bits, leaving the result of the addition in Accumulator B. The specified word of core will remain unchanged, but the result of the addition could set the Carry flag.

Subtract from A - SFA△

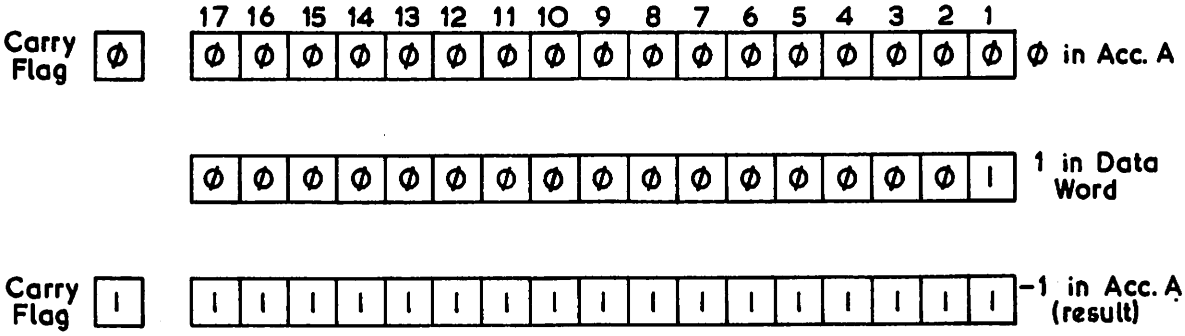
The SFA instruction performs a binary subtraction, subtracting the contents of the specified word of core from the contents of Accumulator A, all 17 bits, leaving the difference in Accumulator A. The specified word of core will remain unchanged, but the result of the subtraction could set the Carry flag.

Arithmetically, binary numbers may be directly subtracted in a manner similar to decimal subtraction. The essential difference is that if a 'borrow' is required, it is equal to the base of the system of 2.

```
i.e.                    110-   =   6   (decimal)
                        101    =   5
                        ───        ─
                        001        1
```

To subtract 1 from Ø in the first column, a borrow of 1 was made from the second column, which effectively added 2 to the first column. After the borrow, 2 - 1 = 1 in the first column; in the second column Ø - Ø = Ø; and in the third column 1 - 1 = Ø.

The following Figure 3:10 illustrates the operation of the SFA instruction :-



		17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Carry Flag	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø in Acc. A
		Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	1 in Data Word
Carry Flag	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1 in Acc. A (result)

Fig. 3:10

In the above example the 'borrow' is lost off the end of the register; also a carry has occurred between Bits 16 and 17 which has caused the Carry flag to be set. Again this could be detected with the register instruction 'Skip if Carry'.

Subtract from B - SFB∆

The SFB instruction performs a binary subtraction, subtracting the contents of the specified word of core from the contents of Accumulator B, all 17 bits, leaving the difference in Accumulator B. The specified word of core remains unchanged, but the result of the subtraction could set the Carry flag.

Add with Carry (to Accumulator A) - ADACΔ

Multiple register addition is possible using the ADAC instruction. This type of arithmetic is needed when a number that is too large to be contained in one word (i.e. more than 65,535 in decimal), has to be added to another similar number, or when the result of an addition may be too large for a single register.

e.g. Two numbers 65,535 and 65,537 are to be added together

```
65,535  =  00000 0000 0000 0000 01111 1111 1111 1111
65,537  =  00000 0000 0000 0001 00000 0000 0000 0001
```

Let us assume that the 65,535 is double stored in Stores 0235 and 0236, the 65,537 is double stored in Stores 0251 and 0252, and that the answer is to be stored in 0276 and 0277. The first bit of program could read :-

```
CLC            (Clear Carry)
LDA    0236    (Load A with 0236)    01111 1111 1111 1111
ADA    0252    (Add to A 0252)       00000 0000 0000 0001
                                     --------------------
               (Result in A)         10000 0000 0000 0000
```

As Bit 17 is left on the presence of this bit denotes a negative number, therefore it must be cleared before being stored in the answer store :-

```
CLSA           (Clear Sign of A)
STA    0277    (Store A in 0277)     00000 0000 0000 0000
```

The first register has been added, and because a carry occurred between Bits 16 and 17 the Carry flag will be set. Using ADAC for the addition of the second pair of stores, the "with carry" will cause the Carry flag to be added to the A register before the addition is started. The addition is then completed normally.

```
LDA    0235          00000 0000 0000 0000
                                        1  - Carry flag
ADAC   0251          00000 0000 0000 0001
                     --------------------
STA    0276          00000 0000 0000 0010
```

The total left in Stores 0276 and 0277 now equals :-

```
00000 0000 0000 0010 00000 0000 0000 0000  =  131,072 (Decimal)
```

To sum up, on an ADAC instruction, the contents of the Carry flag (if any) will be added to the contents of Accumulator A, the Carry flag will be cleared, and the contents of the word of core specified will then be added to Accumulator A. The store specified will remain unchanged.

N.B. It should be pointed out that the Carry flag could be set again by the addition.

This instruction is used for double or multiple store arithmetic.

Add with Carry (to Accumulator B) - ADBC△

Exactly as for ADAC, except that the contents of the Carry flag (if any) will be added to the contents of Accumulator B, the Carry flag will be cleared, and the contents of the word of core specified will then be added to Accumulator B. The store specified will remain unchanged.
N.B. The Carry flag could be set again by the actual addition.

This instruction is used for double or multiple store arithmetic.

Subtract Store from A with Carry - SFAC△

The contents of the Carry flag (if any) will be subtracted from the contents of Accumulator A, the Carry flag will be cleared, and the contents of the word of core specified will then be subtracted from Accumulator A. The store specified will remain unchanged.
N.B. The Carry flag could be set again by the actual subtraction.

This instruction is used for double or multiple store arithmetic.

Subtract store from B with Carry - SFBC△

The contents of the Carry flag (if any) will be subtracted from the contents of Accumulator B, the Carry flag will be cleared, and the contents of the word of core specified will then be subtracted from Accumulator B. The store specified will remain unchanged.
N.B. The Carry flag could be set again by the actual subtraction.

This instruction is used for double or multiple store arithmetic.

Load into A - LDA△

LDA stores the contents of the referenced location in Accumulator A, over-writing the original contents of Accumulator A. The specified word of core remains unaltered.

Load into B - LDB△

LDB stores the contents of the referenced location in Accumulator B, over-writing the original contents of Accumulator B. The specified word of core remains unaltered.

Compare store with A (skip if unequal) - CMPAΔ

This instruction compares the contents of the word of core specified with the contents of Accumulator A, all 17 bits. If the two words are different the next instruction will be skipped (i.e. the PC is advanced by two instead of one). If both words are identical, the program will proceed normally to the next instruction in sequence. The contents of both the specified word of core and Accumulator A remain unaltered.

Should the contents of Accumulator A be greater than the contents of the word of core addressed, the Greater Than flag will be set (but NOT the Carry flag).

Should the contents of Accumulator A be less than or equal to the contents of the word of core addressed, the Greater Than flag will be cleared (but NOT the carry flag).

Compare store with B (skip if unequal) - CMPBΔ

This instruction compares the contents of the word of core specified with the contents of Accumulator B, all 17 bits. If the two words are different the next instruction will be skipped (i.e. the PC is advanced by two instead of one). If both words are identical, the program will proceed normally to the next instruction in sequence. The contents of both the specified word of core and Accumulator B remain unaltered.

Should the contents of Accumulator B be greater than the contents of the word of core addressed, the Greater Than flag will be set (but NOT the Carry flag).

Should the contents of Accumulator B be less than or equal to the contents of the word of core addressed, the Greater Than flag will be cleared (but NOT the Carry flag).

Store A - STAΔ

The STA instruction stores the contents of Accumulator A in the word of core specified, over-writing the original contents of the referenced location. Accumulator A remains unaltered.

Store B - STBΔ

The STB instruction stores the contents of Accumulator B in the word of core specified, over-writing the original contents of the referenced location. Accumulator B remains unaltered.

Unstack - UNSTK△

This is a Privileged Instruction, and may be used only in the system software. It has been assigned the function "Unstack", the address specified normally referring to 000030, thus pointing to that part of the Interrupt Stack indicated by the Interrupt Stack Pointer.

The 'Unstack' instruction first restores the contents of Accumulators A and B, the Base Register, the Program Counter and the Carry and Greater Than flags and then :-

(i) Uses the sign bit of the 2nd word in the appropriate stack to set the Addressing Mode Memory (see Page 1:6) to the correct state.

(ii) Reduces the Interrupt Stack Pointer by 5.

(iii) Uses Step 01 relative to the Base to set the Limit Register.

(iv) Sets Interrupt on.

(v) Causes a JUMP to the step specified in the PC.

b) REGISTER AND CONTROL INSTRUCTIONS

These instructions, in general, manipulate bits in the A and B accumulators. There is no reference to memory, thus the instructions are executed in only one machine cycle. Register and Control Instructions have the following general format in Machine Language :-

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
OPERATION CODE (OO)					MODE		A/B	REGISTER MICRO - INSTRUCTIONS								

Fig. 3:11

The instruction word uses Bits 17-13 to specify the type of instruction (being all zeros for Register and Control instructions), Bits 12 and 11 to specify the Mode, Bit 10 to specify the relevant Accumulator and Bits 9 to 1 to combine in 'micro-instructions', with the resulting multiple instruction operating on the A or B accumulators as a single-word instruction.

Micro-instructions may be combined under the following general rules :-

1) No instructions may be used which combine micro-instructions from different Modes.

2) References to both A and B registers cannot be mixed in the same micro-instruction.

3) If more than one of the micro-instruction bits of a particular Mode is set at any one time, the sequence of execution is left to right (Bit 9 to Bit 1).

4) If two (or more) skip functions are combined, the skip will only occur if all conditions are fulfilled. One exception exists: In Mode 1 only, the skip will occur if either or both conditions are met.

5) Shift and Rotate or Increment and Decrement must not be mixed in the same micro-instruction - the effect of doing so is undefined.

MODE - there are four modes - the mode alters the meaning of the micro-instruction bits.

Mode 0	Mode 2
Mode 1	Mode 3

Mode 0

The Mode 0 instructions contain 27 basic instructions, only 7 of which will be used by the Applications programmer, the other 'control' instructions being 'Privileged', and only used by the system software. Each Mode 0 instruction has its own binary structure as shown in Fig. 3:12 below. These instructions are not micro-programmable.

	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Mnemonics	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	Octal Representation
NOP;	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	000000
HALT;	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	000001
MASK;	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	000002
ACK;	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	000003
ION;	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	000004
IOF;	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	000005
SKON;	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	000006
SKOF;	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	000007
SKMF;	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	000010
SKMR;	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	000011
SKMP;	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	000012
SKPM;	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	000014
SKSW;	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	000015
SKIC;	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	000016
RIO;	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	000017
SK7L;	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	000020
SK15;	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	000021
SKTI;	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	000022
SKIF;	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	000023
SKEX;	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	000024
SRLD;	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	000540
SRAD;	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	000560
SLLD;	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	000740
SLAD;	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	000760
EXC;	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	001000
SETGT;	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	001001

Fig. 3:12

The operation of the 7 Mode 0 instructions in general use is first described below :-

No Operation - NOP;

If all bits are zero, no operation is performed and program control is transferred to the next instruction in sequence.

Shift Right Logical Double Length (one place) - SRLD;

This instruction causes a double length single shift, in that a double length Accumulator (comprised of Accumulators A and B) is shifted right one place, as per Fig. 3:13 below :-

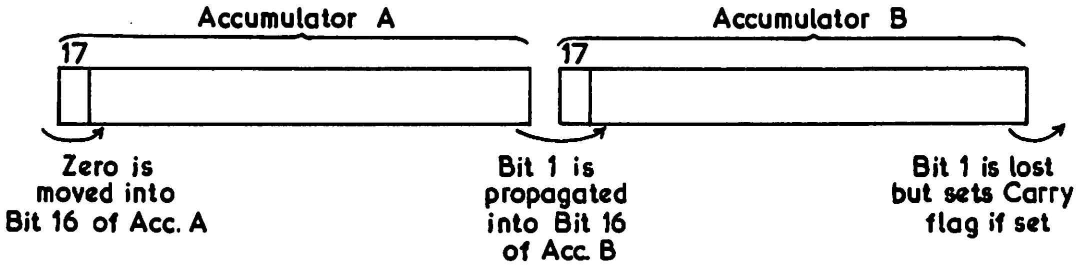

Accumulator A
Accumulator B
17
17
Zero is moved into Bit 16 of Acc. A
Bit 1 is propagated into Bit 16 of Acc. B
Bit 1 is lost but sets Carry flag if set

Fig. 3:13

Bits 16 to 1 of both accumulators are shifted one place to the right, Bit 17 (the sign bit) of each remaining stationary. A zero is moved into position 16 of Accumulator A. The contents of Bit 1 of Accumulator A are propagated into Bit 16 of Accumulator B, but without affecting the Carry flag. Should there be a '1' bit in position 1 of Accumulator B the Carry flag will be set, the '1' bit being otherwise lost. (If there is a 'Ø' bit in position 1 of Accumulator B the Carry flag, should it be set already, will not be overwritten.)

Shift Right Arithmetic Double length (one place) - SRAD;

This instruction causes a double length single shift, in that a double length accumulator (comprised of Accumulators A and B) is shifted right one place, as per Fig. 3:14 below :-

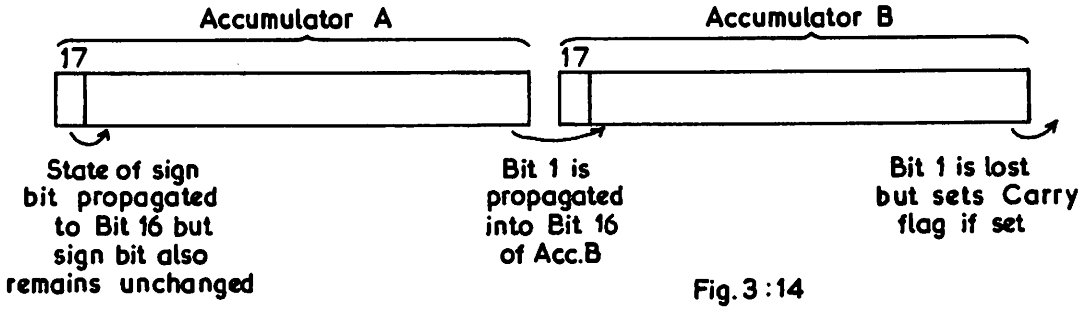

Accumulator A
Accumulator B
17
17
State of sign bit propagated to Bit 16 but sign bit also remains unchanged
Bit 1 is propagated into Bit 16 of Acc.B
Bit 1 is lost but sets Carry flag if set

Fig.3:14

Bits 16 to 1 of both accumulators are shifted one place to the right, Bit 17 (the sign bit) of each remaining unchanged, although the state of the sign bit of Accumulator A is propagated into Bit 16 of Accumulator A. The contents of Bit 1 of Accumulator A are propagated into Bit 16 of Accumulator B, but without affecting the Carry flag. Should there be a '1' bit in position 1 of Accumulator B, the Carry flag will be set, the '1' bit being otherwise lost. (If there is a 'Ø' bit in position 1 of Accumulator B, the Carry flag, should it be set already, will not be overwritten.)

Shift Left Logical Double length (one place) - SLLD;

This instruction causes a double length single shift, in that a double length Accumulator (comprised of Accumulators A and B) is shifted left one place as per Fig. 3:15 below :-

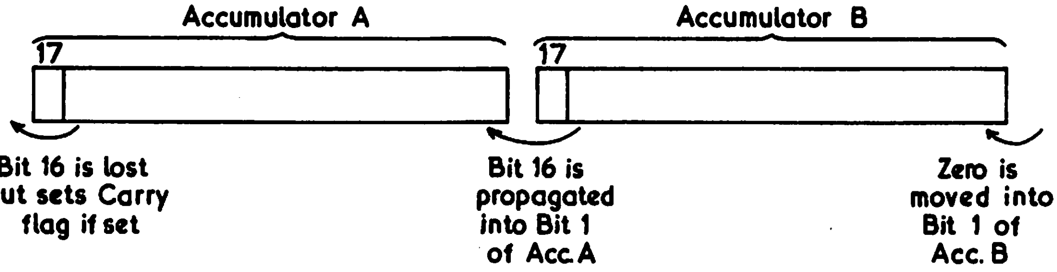

Accumulator A
Accumulator B
17
17
Bit 16 is lost but sets Carry flag if set
Bit 16 is propagated into Bit 1 of Acc.A
Zero is moved into Bit 1 of Acc. B

Fig. 3:15

Bits 16 to 1 of both accumulators are shifted one place to the left, Bit 17 (the sign bit) of each remaining stationary. A zero is moved into position 1 of Accumulator B. The contents of Bit 16 of Accumulator B are propagated into Bit 1 of Accumulator A, but without affecting the Carry flag. Should there be a '1' bit in position 16 of Accumulator A, the Carry flag will be set, the '1' bit being otherwise lost. (If there is a 'Ø' bit in position 16 of Accumulator A the Carry flag, should it be set already, will not be overwritten.)

Shift Left Arithmetic Double length (one place) - SLAD;

This instruction is identical to SLLD (above), as per Fig. 3:16 below :-

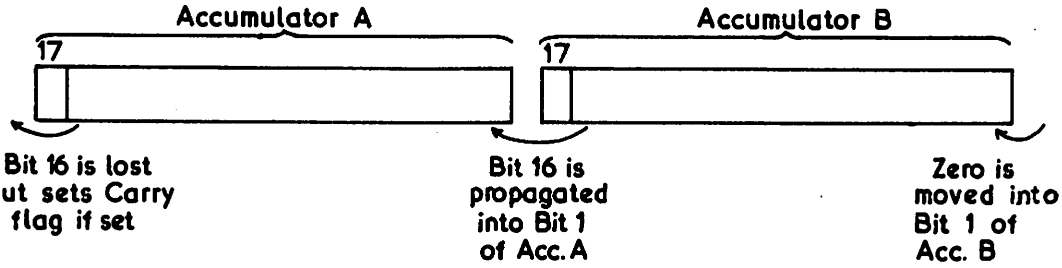

Accumulator A
Accumulator B
17
17
Bit 16 is lost but sets Carry flag if set
Bit 16 is propagated into Bit 1 of Acc.A
Zero is moved into Bit 1 of Acc. B

Fig. 3:16

User call of Executive - EXC;

This instruction is used when Executive is needed to carry out a task. It gives rise to an internal interrupt, which eventually causes the program to go to location 000210.

Set Greater Than Flag - SETGT;

This instruction causes the Greater Than flag to be set.

The operation of each individual 'Privileged' control instruction is briefly described below :-

Halt - HALT;

If Bit 1 is set and all other bits are zero, the computer will stop at the conclusion of the current machine cycle.

Mask Out - MASK;

This instruction sets up the Interrupt Disable flags of each device, according to a pattern or mask set up in Accumulator A - each device's Interrupt Disable flag is set or cleared as the corresponding bit in the Mask is 1 or Ø. (The Mask Bit No. chart is shown in Fig. 3:17 below.)

Mask Bit Numbers

Device	Mask Bit No.
Tape Reader	1
Card Reader	1
Single Shot	1
Line Printer	2
Serial Printer	2
Display (Alpha/Numeric)	2
Tally Roll Printer	2
Keyboards	3
Card Feeds	4
Form Feeds	5
Punch	6
80 Column Card Reader	6
IBM Input	7
IBM Output	8
Modem Coupler transmit	9
Modem Coupler receive	10
Disc	16

Fig.3:17

The Mask Bit numbers refer to I/O Bus Bits 1 to 17 numeric with ascending Binary order. Every device is wired to a particular data line on the in-out bus and hence to a particular bit of the mask. Although slower devices are assigned to the higher numbered bits in the mask, there is no established priority as the program can use any mask configuration.

Acknowledge Interrupt - ACK;

This instruction determines which is the highest priority device awaiting service by reading its device code into Accumulator B, assuming an I/O Interrupt is pending. ACK can read the code of only one device at a time, whichever of those waiting has the highest priority. This is normally determined by the positions of the I/O Boards in the chassis, i.e. the further from the processor, the lower the priority. When two devices share a single I/O Board (e.g. Keyboard and Fast Serial Printer/ Display) the relative priority is fixed (Keyboard higher).

Interrupt On - ION;

This instruction sets the Interrupt On flag, to allow the processor to respond to interrupt requests. If interrupt is disabled when this instruction is given, the CPU executes the next instruction (step) and then enables interrupt.

Interrupt Off - IOF;

This instruction clears the Interrupt On flag to prevent the processor from responding to I/O interrupt requests, and also prevents all internal processor interrupts.

Skip if Interrupt On - SKON;

This will skip the next instruction if interrupt is on.

Skip if Interrupt Off - SKOF;

This will skip the next instruction if interrupt is off.

Skip if Mains Failure Interrupt - SKMF;

This will skip the next instruction in sequence when an internal interrupt is called by a power failure. If the skip is taken, the Interrupt will be reset.

Skip if Mains Return Interrupt - SKMR;

This will skip the next instruction in sequence when an internal interrupt is called when power is restored. If the Skip is taken, the Interrupt will be reset.

Skip if Memory Parity Interrupt - SKMP;

This will skip the next instruction in sequence when an internal interrupt is called in the case of a memory parity failure. If the skip is taken, the interrupt and also the lamp will be reset.

Skip if Memory Boundary Interrupt (Limit) - SKPM;

This will skip the next instruction in sequence when an internal interrupt is called by the program selecting an address outside the bounds of the Limit Register. If the skip is taken, the interrupt will be reset. (The Memory Boundary Interrupt is only applicable when in User mode.)

Skip if MA = Switch Register - SKSW;

This will skip the next instruction in sequence when an internal interrupt is called when the memory address equals an address previously set on the Data switches. If the skip is taken the interrupt will be reset.

Skip if Continuous Interrupt Switch Interrupt - SKIC;

This will skip the next instruction in sequence where an internal interrupt is called after each step. If the skip is taken, the interrupt will be reset.

I/O Reset - RIO;

This instruction clears the flags of all Input/Output devices connected to the computer.

Skip if 7th Level Interrupt - SK7L;

This will skip the next instruction in sequence when an internal interrupt is called if a certain one-bit memory register has been set by entry of the 7th level stack. If the skip is taken, the interrupt will be reset.

Skip if Greater Than 15 Indirects Interrupt - SK15;

This will skip the next instruction in sequence when an internal interrupt is called by more than fifteen indirects being 'chained'. If the skip is taken, the interrupt will be reset.

Skip if Timer Interrupt - SKTI;

Skip if Illegal Function Interrupt - SKIF;

This will skip the next instruction in sequence when an internal interrupt is called by a 'Privileged' instruction being used by program while the Addressing Mode Memory is set to zero. These instructions cannot be allowed to be used by a User program running under the Operating System, as they are likely to interfere with the running of other programs. For the purposes of this specification, an 'Illegal Function' is defined as :-

a) All Input/Output Instructions without exception

b) All Register and Control Instructions in Mode 0 with the exception of the 7 instructions mentioned on Page 3:17.

If the skip is taken, the interrupt will be reset.

Skip if Extracode Interrupt - SKEX;

This will skip the next instruction in sequence when an interrupt is called by the 'EXC' instruction (see page 3:18). If the skip is taken, the interrupt will be reset.

Mode 1

The Mode 1 instructions include 21 basic instructions which can manipulate the contents of Accumulators A or B and the Carry flag. These instructions are micro-programmable; that is, they can be combined with other instructions also in Mode 1 to perform specialised operations (see also Page 3:24). There now follows a selection table for Mode 1.

	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Octal
Mnemonics	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	Representation
CLC;	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	002400
LSA;	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	003300
LSB;	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	002300
RSA;	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	003100
RSB;	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	002100
LRA;	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	003240
RRA;	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	003040
LRAC;	0	0	0	0	0	0	1	1	0	1	0	1	1	0	0	0	0	003260
RRAC;	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	003060
LRB;	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	002240
RRB;	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	002040
LRBC;	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	002260
RRBC;	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	002060
DECA;	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	003010
DECB;	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	002010
INCA;	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	003004
INCB;	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	002004
AMSB;	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	003002
BMSB;	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	002002
ALSB;	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	003001
BLSB;	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	002001

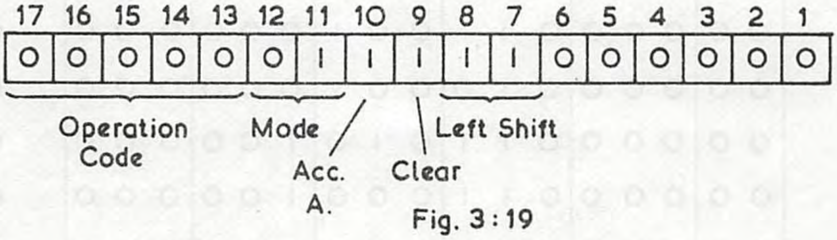
Fig.3:18

Micro-programming

If, for instance, it is desired to clear Carry and Left Shift Accumulator A, to perform this task the program could include the following instructions (given in both mnemonic and octal form).

```
CLC;   002400
LSA;   003300
```

However, when the Mode 1 instruction format is analysed, the following can be seen :-



17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0

Fig. 3:19

Since the CLC and LSA instructions occupy separate bit positions they may be used in the same instruction, thus combining the two operations into one instruction. This instruction would be written in Usercode as CLC,LSA; which is 003700 in octal. In this manner, many more Register instructions, separated by a comma, may be combined to make the execution of the program more efficient.

The operation of each individual instruction specified by Bits 10 to 1 is described below :-

Clear Carry - CLC;

This instruction causes the Carry flag to be cleared.

Left Shift A or B - LSA; or LSB;

Bits 16 to 1 of the specified accumulator will be shifted one place to the left. Bit 17 (the sign bit) of the accumulator to be shifted remains stationary.

If there is a '1' bit in position 16 of the accumulator, the Carry flag will be set after a left shift.

If there is a '0' bit in position 16 of the accumulator, the Carry flag, should it be set already, will not be overwritten after a Left Shift.

Right Shift A or B - RSA; or RSB;

Bits 16 to 1 of the specified accumulator will be shifted one place to the right. Bit 17 of the accumulator to be shifted remains stationary.

If there is a '1' bit in position 1 of the accumulator, the Carry flag will be set after a right shift.

If there is a '0' bit in position 1 of the accumulator, the Carry flag, should it be set already, will not be overwritten after a Right Shift.

Left Rotate A or B - LRA; or LRB;

This instruction rotates Bits 1 to 16 of the specified accumulator one place to the left.

i.e. One left rotate will perform as for one left shift, except that the figure which was in Bit 16 will re-enter the Accumulator at Bit 1. The Carry flag is not affected. In other words, it treats Bits 1 to 16 of the specified accumulator as a closed loop, and performs what is commonly called a circular shift, meaning that any bit rotated off the left end will re-appear at the right end.

```
e.g.     one left rotate of  0110  1100  0110  0101
                      gives  1101  1000  1100  1010
```

Right Rotate A or B - RRA; or RRB;

This instruction rotates Bits 1 to 16 of the specified accumulator one place to the right.

i.e. One right rotate will perform as for one right shift, except that the figure which was in Bit 1 will re-enter the Accumulator at Bit 16. The Carry flag is not affected. In other words, it treats Bits 1 to 16 of the specified accumulator as a closed loop, and performs what is commonly called a circular shift, meaning that any bit rotated off the right end will re-appear at the left end.

```
e.g.     one right rotate of  0110  1100  0110  0101
                       gives  1011  0110  0011  0010
```

Left Rotate A or B with Carry - LRAC; or LRBC;

This instruction causes any previous Carry flag to be introduced into the gap left by the rotate, and the bit rotated off the left end will replace the Carry flag. In other words, it treats Bits 1 to 16 plus the Carry flag as a closed loop, and performs a circular shift as previously described.

```
e.g.            0110  1100  0110  0100  Carry Flag is 1
```

after one left rotate with Carry would read :-

```
                1101  1000  1100  1001  Carry Flag is Ø
```

Rotate with Carry is used for multiple store shifting.

Right Rotate A or B with Carry - RRAC; or RRBC;

This instruction causes any previous Carry flag to be introduced into the gap left by the rotate, and the bit rotated off the right end will replace the Carry flag. In other words it treats Bits 1 to 16 plus the Carry flag as a closed loop, and performs a circular shift as previously described.

```
e.g.        0110  1100  0110  0100   Carry  is 1
                                     flag
```

after one right rotate with Carry would read :-

```
            1011  0110  0011  0010   Carry  is Ø
                                     Flag
```

Rotate with Carry is used for multiple store Shifting.

Decrement A or B - DECA; or DECB;

This instruction will decrement the contents of the specified Accumulator by 1.

Increment A or B - INCA; or INCB;

This instruction will increment the contents of the specified accumulator by 1.

Skip if Bit 16 of A or B equals zero - AMSB; or BMSB;

This instruction causes the program to skip the next step if Bit 16 of the specified accumulator is zero.

Skip if Bit 1 of A or B equals zero - ALSB; or BLSB;

This instruction causes the program to skip the next step if Bit 1 of the specified accumulator is zero, or, in other words, if there is an even number in the accumulator.

Mode 2

This Mode includes 15 basic instructions which can clear and complement the contents of Accumulators A and B and the Carry flag. These instructions are micro-programmable, as previously explained. There now follows a selection Table for Mode 2 :-

Mnemonics	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Octal Representation
	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
CLA;	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	005400
CLB;	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	004400
COMPA;	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	005200
COMPB;	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	004200
CLC;	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	004100
COMPC;	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	004040
SKIP;	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	004020
SWAPA;	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	005010
SWAPB;	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	004010
CLSA;	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	005004
CLSB;	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	004004
COMPSA;	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	005002
COMPSB;	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	004002
ESWRA;	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	005001
ESWRB;	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	004001

Fig. 3:20

The operation of each individual instruction specified by Bits 10 to 1 is described below :-

Clear A or B - CLA; or CLB;

This instruction sets the specified accumulator (all 17 bits) to zeros.

Complement A or B - COMPA; or COMPB;

This instruction causes the specified accumulator (all 17 bits) to be set to the one's complement of its original value; that is, all ones become zeros, and all zeros become ones.

e.g.	Before one's complement	-	01000 1100 1110 1111
	After one's complement	-	10111 0011 0001 0000

Clear Carry - CLC;

This instruction causes the Carry flag to be cleared.

Complement Carry - COMPC;

This instruction causes the state of the Carry flag to be complemented (i.e. reversed).

Unconditional Skip - SKIP;

This instruction causes the program to skip the next step, unconditionally.

Swap A or B - SWAPA; or SWAPB;

This instruction causes the contents of the top half (Bits 16 to 9) of the specified accumulator to be swapped with the contents of the bottom half (Bits 8 to 1) of the said accumulator. The sign bit is not affected.

Clear Sign of A or B - CLSA; or CLSB;

This instruction causes the sign bit (Bit 17) of the specified accumulator to be cleared (set to zero).

Complement Sign of A or B - COMPSA; or COMPSB;

This instruction causes the state of the sign bit (Bit 17) of the specified accumulator to be complemented (i.e. reversed).

Enter Switch Register into A or B - ESWRA; or ESWRB;

This instruction causes whatever is set on the Data Switches of the Control Panel to be loaded into the specified accumulator.

Mode 3 - contains 18 basic instructions which enable the programmer to perform tests on the Accumulator, Carry flag and Greater Than flag and to skip the next instruction depending on the results of the test. The group is sub-divided into two sections, the instructions of which cannot be mixed together, although six instructions (CLC, CLGT, CLA, CLB, COMPA and COMPB) appear in both sections. The Selection Tables for the two sub-sections now follow :-

SECTION 1

Mnemonics	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Octal Representation
	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
ANEG;	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	007600
BNEG;	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	006600
ANØ;	0	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	007500
BNØ;	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	006500
SKC;	0	0	0	0	0	1	1	0	1	0	0	1	0	0	0	0	0	006440
CLC;	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	006020
SKGT;	0	0	0	0	0	1	1	0	1	0	0	0	0	1	0	0	0	006410
CLGT;	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	006004
CLA;	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	0	007002
CLB;	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	006002
COMPA;	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	007001
COMPB;	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	006001

SECTION 2

	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
APOS;	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	007200
BPOS;	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	006200
A Ø;	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0	007100
B Ø;	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	006100
SKNC;	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	006040
CLC;	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	006020
SKNGT;	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	006010
CLGT;	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	006004
CLA;	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	0	007002
CLB;	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	006002
COMPA;	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	007001
COMPB;	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	006001

Fig. 3:21

The operation of each individual instruction specified by Bits 10 to 1 is described below, under the two Section headings :-

SECTION 1

Skip if A or B is negative - ANEG; or BNEG;

This instruction causes the next instruction to be skipped if the contents of the specified accumulator are negative (i.e. if the sign bit is set).

Skip if A or B is not zero - ANØ; or BNØ;

This instruction causes the next instruction to be skipped if the contents of the specified accumulator are not zero (i.e. if any of Bits 1 - 17 is set).

Skip if Carry - SKC;

This instruction causes the next instruction to be skipped if the Carry flag is set.

Clear Carry - CLC;

This instruction causes the Carry flag to be reset (cleared).

Skip if Greater Than - SKGT;

This instruction causes the next instruction to be skipped if the Greater Than flag is set.

Clear Greater Than - CLGT;

This instruction causes the Greater Than flag to be cleared.

Clear A or B - CLA; or CLB;

This instruction sets the specified accumulator (all 17 bits) to zeros.

Complement A or B - COMPA; or COMPB;

This instruction causes the specified accumulator (all 17 bits) to be set to the one's complement of its original value; that is, all ones become zeros, and all zeros become ones.

SECTION 2

Skip if A or B is positive - APOS; or BPOS;

This instruction causes the next instruction to be skipped if the contents of the specified accumulator are positive (i.e. if the sign bit is not set).

Skip if A or B is zero - AØ; or BØ;

This instruction causes the next instruction to be skipped if the contents of the specified accumulator (all 17 bits) are zero.

Skip if Not Carry - SKNC;

This instruction causes the next instruction to be skipped if the Carry flag is not set.

Clear Carry - CLC;

This instruction causes the Carry flag to be reset (cleared).

Skip if Not Greater Than - SKNGT;

The next instruction will be skipped if the Greater Than flag is not set.

Clear Greater Than - CLGT;

This instruction causes the Greater Than flag to be cleared.

Clear A or B - CLA; or CLB;

This instruction sets the specified accumulator (all 17 bits) to zeros.

Complement A or B - COMPA; or COMPB;

This instruction causes the specified accumulator (all 17 bits) to be set to the one's complement of its original value; that is, all ones become zeros, and all zeros become ones.

c) LITERAL (I/O, SHIFTS, etc.) INSTRUCTIONS

As all Input/Output instructions are deemed Privileged, this section describes first the Literal functions, which deal with multiple short shifts and rotates. These Literal functions have the following format in Machine Language :-

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Ø	Ø	Ø	Ø	Ø	Ø	Ø	A/B	1	L/R	S/R	Ø	A/L				
OP. CODE (all zeros)					MODE								No. of Shifts or Rotates (O to 15 Dec.)			

Fig. 3:22

The instruction word uses Bits 17-13 to specify the operation code (being all zeros for Literal functions), Bits 12 and 11 to specify the Mode (again both zeros), Bit 10 to specify the relevant accumulator, Bit 9 on to specify Special Shift, Bit 8 to specify Left or Right, Bit 7 to specify Shift or Rotate, Bit 6 being always zero, Bit 5 to specify Arithmetic or Logical and Bits 4 to 1 to indicate the number of shifts or rotates required (Ø to 15). These instructions are not micro-programmable.

Fig. 3:23 below shows the binary structure of each Literal instruction, but without showing the number of shifts or rotates required. (The number (Ø to 17 Octal) should be added in Octal.)

Mnemonics	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Octal Representation
	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	
SLAA△Ø;	0	0	0	0	0	0	0	1	1	1	1	0	1	NO.OF SHIFTS OR ROTATES				001720
SLAB△Ø;	0	0	0	0	0	0	0	0	1	1	1	0	1					000720
SRAA△Ø;	0	0	0	0	0	0	0	1	1	0	1	0	1					001520
SRAB△Ø;	0	0	0	0	0	0	0	0	1	0	1	0	1					000520
SLLA△Ø;	0	0	0	0	0	0	0	1	1	1	1	0	0					001700
SLLB△Ø;	0	0	0	0	0	0	0	0	1	1	1	0	0					000700
SRLA△Ø;	0	0	0	0	0	0	0	1	1	0	1	0	0					001500
SRLB△Ø;	0	0	0	0	0	0	0	0	1	0	1	0	0					000500
RLMA△Ø;	0	0	0	0	0	0	0	1	1	1	0	0	0					001600
RLMB△Ø;	0	0	0	0	0	0	0	0	1	1	0	0	0					000600
RRMA△Ø;	0	0	0	0	0	0	0	1	1	0	0	0	0					001400
RRMB△Ø;	0	0	0	0	0	0	0	0	1	0	0	0	0					000400

Fig. 3:23

The operation of each individual instruction is briefly described below :-

Shift Left Arithmetic A or B - SLAA or SLAB

Bits 16 to 1 of the specified accumulator will be shifted left the specified number of places. The sign bit (Bit 17) remains unchanged.

Any bit shifted off the top of the register sets the Carry flag but is otherwise lost. A zero will be inserted in Bit 1 on each left shift.

```
e.g.   SLAAΔ4; of  0  0110  1100  0110  0101
            gives  0  1100  0110  0101  0000  (with Carry Flag set)
```

Shift Right Arithmetic A or B - SRAA or SRAB

Bits 16 to 1 of the specified accumulator will be shifted right the specified number of places. The sign bit (Bit 17) remains unchanged, but is propagated into the Bit 16 position on each right shift.

Any bit shifted off the bottom of the register sets the Carry flag but is otherwise lost.

```
e.g.  SRAAΔ4; of  1  0110  1100  0110  0101
           gives  1  1111  0110  1100  0110   (With Carry flag set)
```

Shift Left Logical A or B - SLLA or SLLB

Bits 16 to 1 of the specified accumulator will be shifted left the specified number of places. The sign bit (Bit 17) remains unchanged.

Any bit shifted off the top of the register sets the Carry flag but is otherwise lost. A zero will be inserted in Bit 1 on each left shift.

```
e.g.  SLLBΔ2; of  0  0110  1100  0110  0101
           gives  0  1011  0001  1001  0100   (With Carry flag set)
```

Shift Right Logical A or B - SRLA or SRLB

Bits 16 to 1 of the specified accumulator will be shifted right the specified number of places. The sign bit (Bit 17) remains unchanged.

Any bit shifted off the bottom of the register sets the Carry flag but is otherwise lost. A zero will be inserted in Bit 16 on each right shift.

```
e.g.  SRLAΔ3; of  0  0110  1100  0110  0101
           gives  0  0000  1101  1000  1100    (With Carry flag set)
```

Rotate Left Multiple A or B - RLMA or RLMB

Bits 16 to 1 of the specified accumulator will be rotated left the specified number of places, with the sign bit (Bit 17) remaining unchanged and the Carry flag being unaffected. Bits 15 to 1 will be left shifted once on each rotate, with the figure which was in Bit 16 re-entering the Accumulator at Bit 1. In other words, Bits 1 to 16 of the specified accumulator are treated as a closed loop, and what is commonly called a circular shift is performed for each rotate, in that any bit rotated off the left end will re-appear at the right end.

```
e.g.   RLMAΔ2; of    0  0110  1100  0110  0101
            gives    0  1011  0001  1001  0101
```

Rotate Right Multiple A or B - RRMA or RRMB

Bits 16 to 1 of the specified accumulator will be rotated right the specified number of places, with the sign bit (Bit 17) remaining unchanged and the Carry flag being unaffected. Bits 16 to 2 will be right shifted once on each rotate, with the figure which was in Bit 1 re-entering the Accumulator at Bit 16. In other words, Bits 1 to 16 of the specified accumulator are treated as a closed loop, and what is commonly called a circular shift is performed for each rotate, in that any bit rotated off the right end will re-appear at the left end.

```
e.g.   RRMBΔ3; of    0  0110  1100 0110  0101
            gives    0  1010  1101 1000  1100
```

INPUT/OUTPUT INSTRUCTIONS

These are a set of program instructions which, like the majority of the Mode 0 Register Instructions, are 'Privileged', that is not used in general programming but only by the system software. There now follows a description of the Input/Output instructions, which control all transfers of data to and from the peripherals and also perform various operations within the processor. They provide the following general capabilities :-

a) Fix the state of the Busy and Done flags
b) Test the state of the Busy and Done flags
c) Enter data from a specified device into the A or B Registers
d) Output data to a specified device from the A or B Registers.

I/O instructions are recognised by the computer when the four most significant bits of the instruction word are 0000 and Bit 13 is a 1 (Octal Code 01). Bits 6 to 1 select the device that is to respond to the instruction; the format thus allows for 64 codes. In all I/O instructions Bits 11 to 7 specify the complete function to be performed, bits 10 and 11 either controlling or sensing Busy and Done, as shown under 'Funct.' in the Selection Chart (Fig.3:24) on the next page. If Bits 7 to 9 are all set (Mode 111) there is no transfer, and Bits 11 and 10 then specify a skip condition. Bit 12, where relevant, specifies the A or B register (A=1, B=Ø).

	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		
Mnemonics	OP. CODE					A/B	FUNCT.		MODE			DEVICE CODE							
STT	0	0	0	0	1		0	1										Set Busy. Clear Done	
STOP	0	0	0	0	1		1	0										Clear Busy. Clear Done	
IOP	0	0	0	0	1		1	1										Input/Output Pulse	
	0	0	0	0	1				0	0	0							Allows Functions above	
DIP1A	0	0	0	0	1	1			0	0	1							DATI 1A	Can be used with any of above functions
DIP1B	0	0	0	0	1				0	0	1							DATI 1B	
DIP2A	0	0	0	0	1	1			0	1	0							DATI 2A	
DIP2B	0	0	0	0	1				0	1	0							DATI 2B	
DIP3A	0	0	0	0	1	1			0	1	1							DATI 3A	
DIP3B	0	0	0	0	1				0	1	1							DATI 3B	
DOP1A	0	0	0	0	1	1			1	0	0							DATO 1A	
DOP1B	0	0	0	0	1				1	0	0							DATO 1B	
DOP2A	0	0	0	0	1	1			1	0	1							DATO 2A	
DOP2B	0	0	0	0	1				1	0	1							DATO 2B	
DOP3A	0	0	0	0	1	1			1	1	0							DATO 3A	
DOP3B	0	0	0	0	1				1	1	0							DATO 3B	
SKB	0	0	0	0	1		0	0	1	1	1							Skip if Busy	
SKNB	0	0	0	0	1		0	1	1	1	1							Skip if not Busy	
SKD	0	0	0	0	1		1	0	1	1	1							Skip if Done	
SKND	0	0	0	0	1		1	1	1	1	1							Skip if not Done	

Fig. 3:24

Operation Code

Input/Output instructions are recognised by the computer when Bits 17 to 13 are set to the Octal Code 01.

Accumulator Indicator - Bit 12

Denotes to which accumulator the instruction refers

Accumulator A = 1
Accumulator B = Ø

Function - Bits 11 and 10

When any Mode except 111 is set, the functions are as follows :-

00	=	No operation	
01	=	Set Busy, Clear Done	(to start the device)
10	=	Clear Busy, Clear Done	(to idle the device)
11	=	Input/Output Pulse	(the effect, if any, depends on the device).

The functions as interpreted when Mode 111 (Skip Mode) is set, are shown with this Mode.

MODES - Bits 9-7

Mode 000

This has no input/output transfer, but just enables any of the functions shown previously.

Mode 001 - DIP1A or DIP1B

Enables input from Register 1 of the specified device (each peripheral can have up to three registers or buffers) into Accumulator A or B, as specified, and can also enable any Function shown previously.

Mode 010 - DIP2A or DIP2B

Enables input from Register 2 of the specified device; used to inspect Status codes, and can also enable any Function shown previously.

Mode 011 - DIP3A or DIP3B

Enables input from Register 3 of the specified device into Accumulator A or B, as specified, and can also enable any Function shown previously.

Mode 100 - DOP1A or DOP1B

Enables output to Register 1 of the specified device from Accumulator A or B, as specified, and can also enable any Function shown previously.

Mode 101 - DOP2A or DOP2B

Enables output to Register 2 of the specified device from Accumulator A or B, as specified, and can also enable any Function shown previously.

Mode 110 - DOP3A or DOP3B

Enables output to Register 3 of the specified device from Accumulator A or B, as specified, and can also enable any Function shown previously.

Mode 111 - Skip Mode

If this Mode is set there is no input/output transfer and the Function bits (11 and 10) then specify a skip condition.

e.g.

Function 00	=	Skip if Busy
Function 01	=	Skip if Not Busy
Function 10	=	Skip if Done
Function 11	=	Skip if Not Done

Device Code

There can be 64 different device codes (00-77 in Octal) of which 63 are used to address devices, the code 00 not being used. A table in Figure 3:25 on the next page lists all devices for which codes have been assigned. The addresses have been grouped into fixed blocks to cover all machine variations, to prevent inadvertent misdirection of data between devices in test programs or board changing. Devices must all carry serial numbers on program descriptions and listing, and all programmers must notify the Factory in writing as soon as possible which devices have which codes, and also the order of priorities as required.

Alpha/Numeric Displays have their own addresses, ranging normally from 60 to 67. When required, they are paired with Keyboards as follows :-

Keyboard 20 with A/N Display 60
Keyboard 21 with A/N Display 61

etc.

3:38 I/O DEVICE ADDRESSES

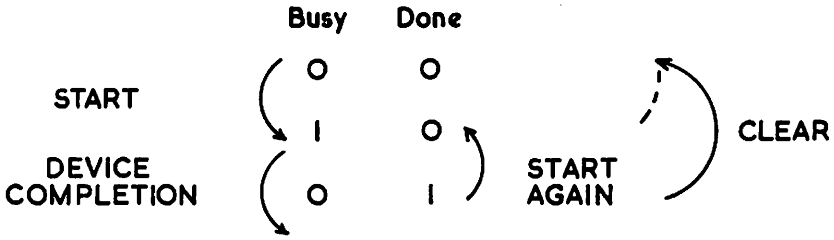
00	Not Used				
01	Mag. Stripe Reader 1	Front Feed			
02		Rear Feed			
03	Mag. Stripe Reader 2	Front Feed			
04		Rear Feed			
05	Mag. Stripe Reader 3	Front Feed			
06		Rear Feed			
07	Spare				
10	PE Reader 2		or	Mark Senser 1	
11	PE Reader 1		or	Mark Senser 2	
12				Mark Senser 3	
13				Mark Senser 4	
14	Single Shot/Hopper Reader(80 col)		or	"	5
15	" " " "	e.p.c.	or	"	6
16	" " " "	"	or	"	7
17	" " " "	"	or	"	8
20	Alpha-Numeric K/B	1		10 Key K/B	1
21	" " "	2		"	2
22	" " "	3		"	3
23	" " "	4		"	4
24	" " "	5		"	5
25	" " "	6		"	6
26	" " "	7		"	7
27	" " "	8		"	8
30	Line Printer	1			
31	Line Printer	2			
32	Tape Punch	2			
33	Tape Punch	1			
34	High Speed Serial Printer (Dot)	1			
35	High Speed Serial Printer (Dot)	2			
36	----------------Spare----------------				
37	----------------Spare----------------				

40	Serialiser (Modem)TX Cathode Ray Display	1	
41	" " "	2	
42	" " "	3	
43	" " "	4	
44	" " "	5	
45	" " "	6	
46	" " "	7	
47	" " "	8	
50	Serialiser (Modem)RX Cathode Ray Keyboard	1	
51	" " "	2	
52	" " "	3	
53	" " "	4	
54	" " "	5	
55	" " "	6	
56	" " "	7	
57	" " "	8	
60	I/O Writer	4 I/P	A/N Display or Tally Roll Printers
61	"	4 O/P	
62	"	3 I/P	"
63	"	3 O/P	"
64	"	2 I/P	"
65	"	2 O/P	"
66	"	1 I/P	"
67	"	1 O/P	"
70	Disc	1	
71	Disc	2	
72	----------Spare------------		
73	----------Spare------------		
74	Mag Tape Handler 1		
75	Mag Tape Handler 2		
76	Drum/Disc (Fast Access) 1		
77	Drum/Disc (Fast Access) 2		

Fig. 3:25

Input/Output (continued)

Every peripheral device has up to three buffer registers, an Interrupt Disable Flag, Busy and Done Flags and a 6-bit device selection network. This selection network decodes Bits 1-6 of the Input/Output Instruction (which contain the device address), thus ensuring that only the correct device responds to signals sent by the processor over the in-out bus. The Busy and Done Flags together denote the basic state of the device. When both are clear, the device is idle. The overall sequence of Busy and Done states is determined by both the program and the internal operation of the device :-



The Data-in or data-out instruction that the program gives in response to the setting of Done can also restart the device. When all data has been transferred, the program generally clears Done so that the device neither requests further interrupts nor appears to be in use. (Busy and Done both set is a meaningless situation).

Any device whose Interrupt Disable flag is set, cannot cause an interrupt to start and is therefore regarded by the program as being of low priority. The Interrupt Disable flags are used in setting up a priority structure which enables higher priority devices to interrupt an interrupt already in progress. This priority is determined by the use of a mask which controls the states of the Interrupt Disable flags in the different devices (as previously described under Mask Out, see Page 3:19).

Each peripheral device can have up to three buffer registers, as explained on previous page. DATI 2 Input (DIP 2A or DIP 2B) inspects the Status Codes of all devices except disc. Status codes are codes allocated to indicate conditions of the device being addressed, and appear in Bits 8 - 1 as shown below, reading from left to right in descending order. Parity false condition is reset by the Input/Output Pulse in all cases except the Trend Reader.

STATUS CODES

Device	Status	Code
Card Reader	No Card	00000001
Card Reader	Parity Odd	00001000
Displays	Power Monitor	00000001
Fast Serial Printer	Power Monitor	00000001
IBM (see note below)	Parity Fail	10000000
Keyboards	Parity Error	00000010
Keyboards	Finger Trouble	00000100
Line Printer	Off Line	00000001
Line Printer	Buffer not Ready	00000010
Line Printer	Print cycle in progress	00000100
Line Printer	Paper feeding	00001000
Line Printer	Data Channel Mode	00010000
Mag. Stripe Card Handler	Last Line	00000001
Mag. Stripe Card Handler	Card jammed	00000010
Mag. Stripe Card Handler	Parity fail (Read only)	00000100
Mag. Stripe Card Handler	Data late (read or write)	00001000
Mag. Stripe Card Handler	Card not in Hopper	00010000
Modems	Do not transmit	00000001
Modems	Do not transmit or receive	00000010
Modems	Data link not connected	00000100
Modems	Parity Odd	00001000
Modems	Receiver Idling	00010000
Paper Tape Reader	Latch Open	00000001
Paper Tape Reader	No Tape	00000010
Paper Tape Reader	Parity Odd	00001000
Punch	Card position	00000001
Punch	Out of Tape	00000100
Punch	Check-back-failure	00000010
Single Shot	No Card	00000001
Single Shot	Parity Odd	00001000
Tally Roll Printer	Power Monitor	00000001
VDU	Parity Error	00001000

Note :- Any parity failure on Input or Output will be detected and shown as a status failure in the Status Register of the IBM Input only.

The Status Codes referring to 'No Tape' in the Trend Reader and 'Out of Tape' in the Punch will be available only when decided by the Technical design/development/production departments. Please refer to Pre-production Engineers for news of latest positions on these Status Codes.

For Disc Status Codes, please see Page 4:37.

SECTION 4 - Peripherals

List of Contents

PERIPHERALS

The input/output hardware allows a program to address up to sixty-three peripherals. This Section lists many of the peripheral devices available for the Molecular 18 (enhanced version), together with operating instructions where necessary.

Periodically, new peripherals are added to the Molecular 18 Product Line, and details of these additions will be issued and added to this Manual as soon as they are available.

There now follow brief descriptions of some of the peripherals available with the Molecular 18 :-

PAPER TAPE READER

The high speed paper tape reader has a maximum reading speed of 500 ch/sec. The reader employs photo-electric sensing, which permits high operating speeds and also reduces failures arising from mechanical wear.

The sensing system automatically adjusts for changes in tape parity, lateral tape registration or tape colour, and can read 5 to 8 channel tape. The reader uses a braking system which allows the tape to stop reliably on a single character from maximum speed.

The reader has no Input buffer, but the DATI 1 instruction gives input in the normal manner, character-by-character.

How to Load the Paper Tape Reader

The paper tape reader is normally housed in the same cabinet as the Paper Tape Punch. The tape that is to be read should previously have been wound on a plastic reel, 2" in diameter, specially formulated to fit the dispenser hub of the reader (see diagram below). The first 3ft. of tape must contain sprocket holes only, thus allowing enough manoeuvrability for the tape to be placed on the spooling system and yet still allow the reader to sense the first character of data to be read.

At the other end of the tape there should be at least 6 inches containing sprocket holes only, in order to ensure that the reader operates efficiently.

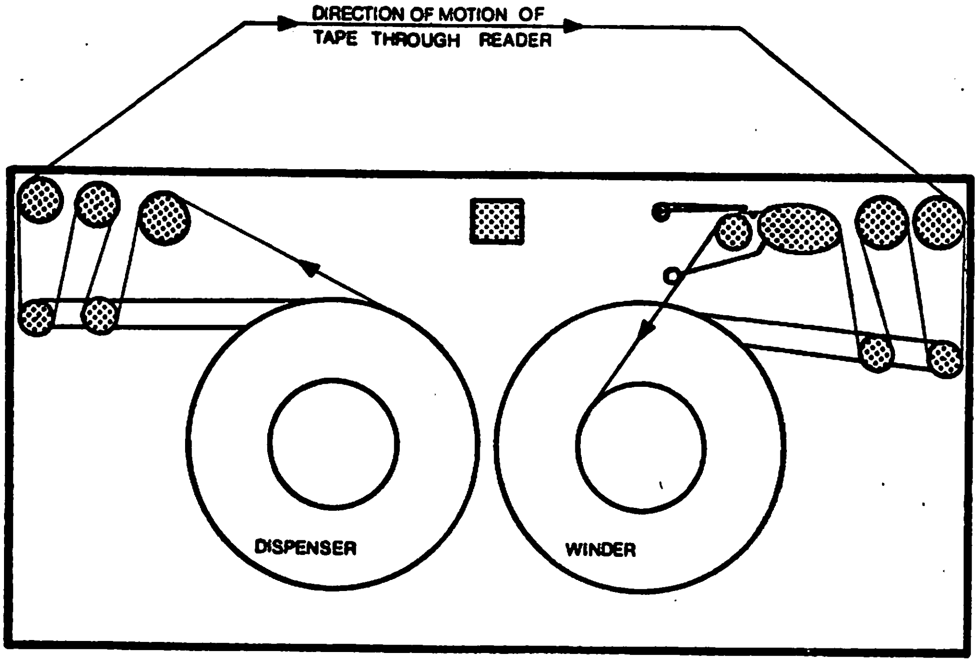


Fig. 4:1

Instructions for Loading the Tape Reader

1) Ensure that the power switch for the spooler is 'off'.

2) Holding the reel of tape so that the sprocket holes (or guide edge) are furthest away, place it on the pliable hub of the dispenser. The tape should now leave the reel in an anti-clockwise direction ; if it does not, the reel will have to be rewound before use.

3) Tighten the hub in a clockwise direction.

4) Pass the tape through the pulleys on the take-up arm on the left-hand side, as shown in the diagram.

5) Lift the reader gate, and pass the tape through the reader. (The guide edge should now be towards the back of the reader. If it is not, the tape has been wound incorrectly, and all data will be misread if the tape is not first rewound).

6) Pass the tape through the pulleys on the take-up arm on the right-hand side exactly as in the diagram, so that the tape passes over the Tape Sensing Arm and then between the flap and roller on the Tape Width Adjustment mechanism.

7) Roll the tape round the winder hub (which should have a plastic reel on it) in the direction shown.

8) Ensure that the hub of the winder has been tightened in a clockwise direction.

9) Manually turn the winder in an anti-clockwise direction several times, to make sure that the tape is secure.

10) Lower the gate on the reader.

11) Put the power switch for the spooling system 'on'.

PAPER TAPE PUNCH

The Paper Tape Punch is normally housed in the same cabinet as the High Speed Reader (see Page 4: 2). It is capable of punching paper, oiled paper or syntosil tapes. No adjustment is necessary to cater for different widths of standard tape, and the design features simple tape insertion. Appendix 14 shows the Standard Tape Dimensions (1" tape).

The punching speed is approx. 70 ch/sec continuous for the fast paper-tape only version, but should it be necessary to punch cards as well as tape there is a slower version which operates at approx. 40 ch/sec continuous.

The punch can be hard-wired for automatic insertion of the parity bit, with a choice of odd or even parity or of no parity. If no parity, then any code can be punched.

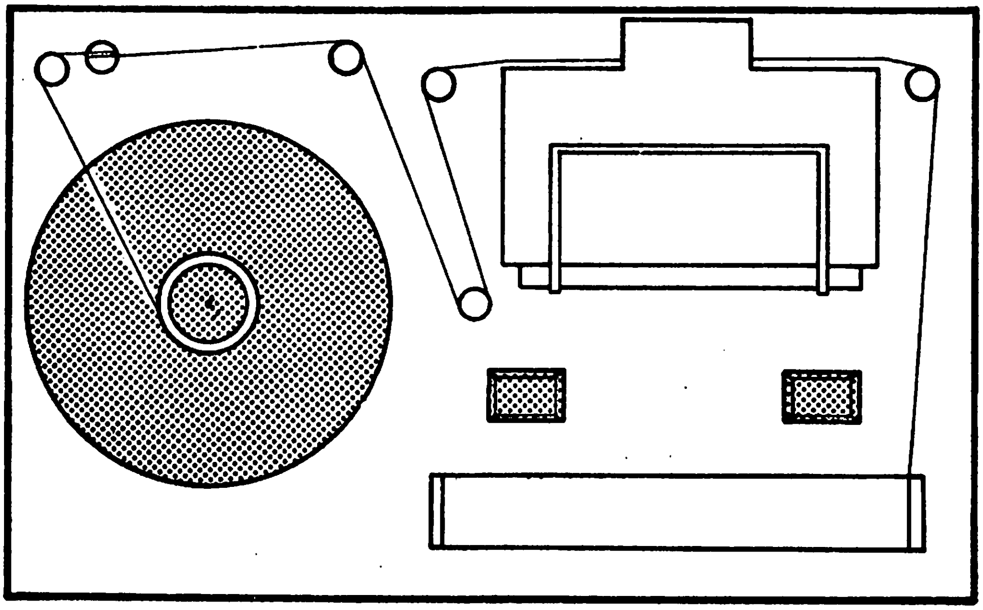


Fig. 4:2

Instructions for Loading the Paper Tape Punch

1) Ensure that the power switch for the punch is in the 'off' position.

2) Pull out the drawer and open the front flap.

3) Place a fresh spool of tape on the turntable so that it is running in a clockwise direction.

4) Pass the end of the tape first round the left-hand tension arm, then over the right-hand pulley and over the end of the drawer.

5) Close the front flap of the drawer (it is held by a magnetic catch), holding the tape carefully so that it still runs freely over the pulley, and push the drawer in.

6) Bring the end of the tape up so that it passes over the right-hand pulley (see diagram) and into the punch, pressing the Tape Insertion button. If there are no sprocket holes present on the tape, the Feed Control lever should be in the '+' position. If there are already sprocket holes, the Feed Control lever should be in the '-' position.

7) Pass the tape over the remaining pulleys and through the tape-lock, as shown in the diagram, Fig. 4. 2, and wind onto the spool, which should already have in place a plastic reel, 2" in diameter, on which to wind the tape.

8) Make sure the tape is secure, tighten the hub and place the power switch for the punch in the 'on' position.

Note

To empty the chad box, just pull it away from the punch and lift it clear.

IBM 735 I/O WRITER

The typewriter is the IBM 735, with 88 printing symbols plus Tab, space, back space, carriage return, vertical space and case shift. The normal electric typewriter functions are present, such as left and right margins, line space adjuster, clutch release, tab set, tab clear, platen adjuster and impression adjuster. The maximum printing speed is 15½ ch/sec.

The IBM can use Netherlands correspondence Golf-balls and other special heads in addition to the normal BCL 'Sterling' head.

An important point to note is that the Tab and Carriage Return Keys on the typewriter are not mechanically connected to the mechanism, therefore if these Keys are depressed during an Input no physical action will be observed - this is a hardware function. Also, when the IBM is in upper case, operation of the Tab or Carriage Return Keys has no effect whatsoever. All other Keys, apart from Function Keys, print simultaneously when depressed. The entire character set for the IBM 735 is listed in the table in Appendix 2. The Operating System ensures that the IBM internal code is converted automatically to and from the ECMA 6 character-code.

The typewriter separates its Input and Output functions and should be regarded as two different devices. Each has its own Device Code, its own Busy, Done and Interrupt Disable flags and its own interrupt priority mask-bit assignment. The IBM Input device includes the Keyboard Lock which has no operational indicators or status. The Keyboard of the typewriter remains locked until the 'Unlock' is programmed.

Unlock is a DATO 2 command to the Input Device, with Bit 2 only set in the specified accumulator.

Lock is a DATO 2 command to the Input Device, with Bit 1 only set in the specified accumulator.

Any parity failure on Input or Output will be shown as a Status Failure in the Status Register - DATI 2 - of the IBM Input device only.

I.B.M. 735,—KEYBOARD

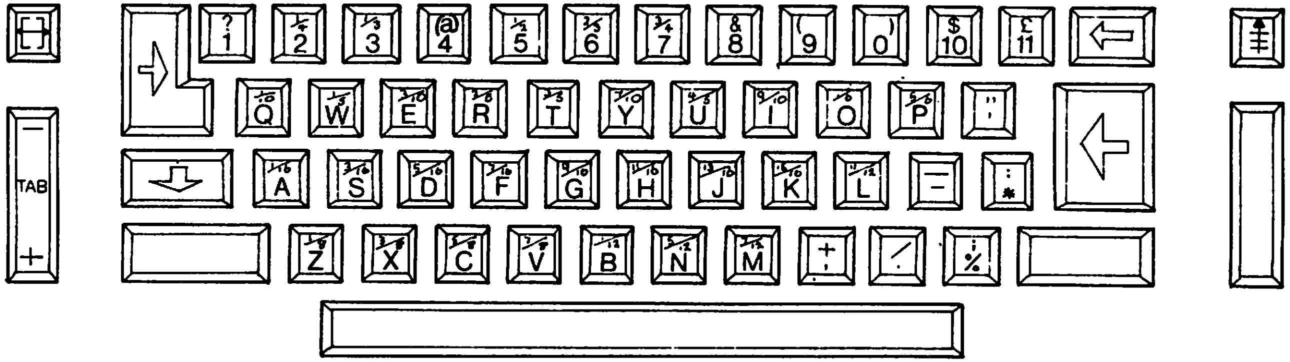


Fig. 4 :3

TALLY ROLL PRINTER

Each character is composed of a number of dots - these dots are selected from a 5 x 7 dot matrix, namely 5 columns of 7 dots each. The printing speed is 60 lines per minute, which gives a character rate of 50 ch/sec (20 char/sec. per line). Hard copy paper-roll print-out of data is given and the maximum number of character positions per line can be fixed to give from 18 to 20. There are 64 printing characters available, including Letters A to Z, Numerals Ø to 9 and also various other symbols as shown in the Code Set in Appendix 5. The character size is a maximum of 2.5 mm high and 1.8 mm wide with a minimum of 0.7 mm between one character and the next, giving 2.5 mm pitch. This gives approximately 10 characters to the inch.

The printer accommodates action paper-roll, the maximum width being 60 mm $\pm^{0}_{1}$ (2 5/16th" ± 1/32nd"). The action paper roll is readily replaceable by using a hinged lid at the front and there is a paper-feed push button at the rear of the cover.

Further developement is envisaged to give two action paper print-out, and also a VT command which will index the paper 2", at a rate of 4 lines per second.

The printer has a single character buffer, each character being printed with the print head moving continuously in a serial entry mode. Should the section of data being output include a Carriage Return (C/R) a new line will be initiated, as also occurs after the output of 20 characters. The paper transport is automatic during carrier return after each line of print.

START and DATO 2 initiate the printer cycle, and when the Printer is ready to accept data Done and Interrupt will be set. DATO 1 and START plus data in the normal manner perform the print operations for the current line, and when all characters for that line have been transferred START and DATO 2 are given if another line is to follow. After the print cycle has been initiated, the print head continues to move whether printing or not, and undue delays (i.e. more than 2-3 ms) must be avoided in the output øf data.

The 'Bel' Code (which produces a half-second audible alarm) may be output only as the last character of a line :-

i.e. a) Immediately before a C/R

b) As the 20th character of a line

c) As the last character of the whole output

If this code is inserted in any other position the output will cease abruptly and jam the mechanism of the Printer.

TALLY ROLL PRINTER

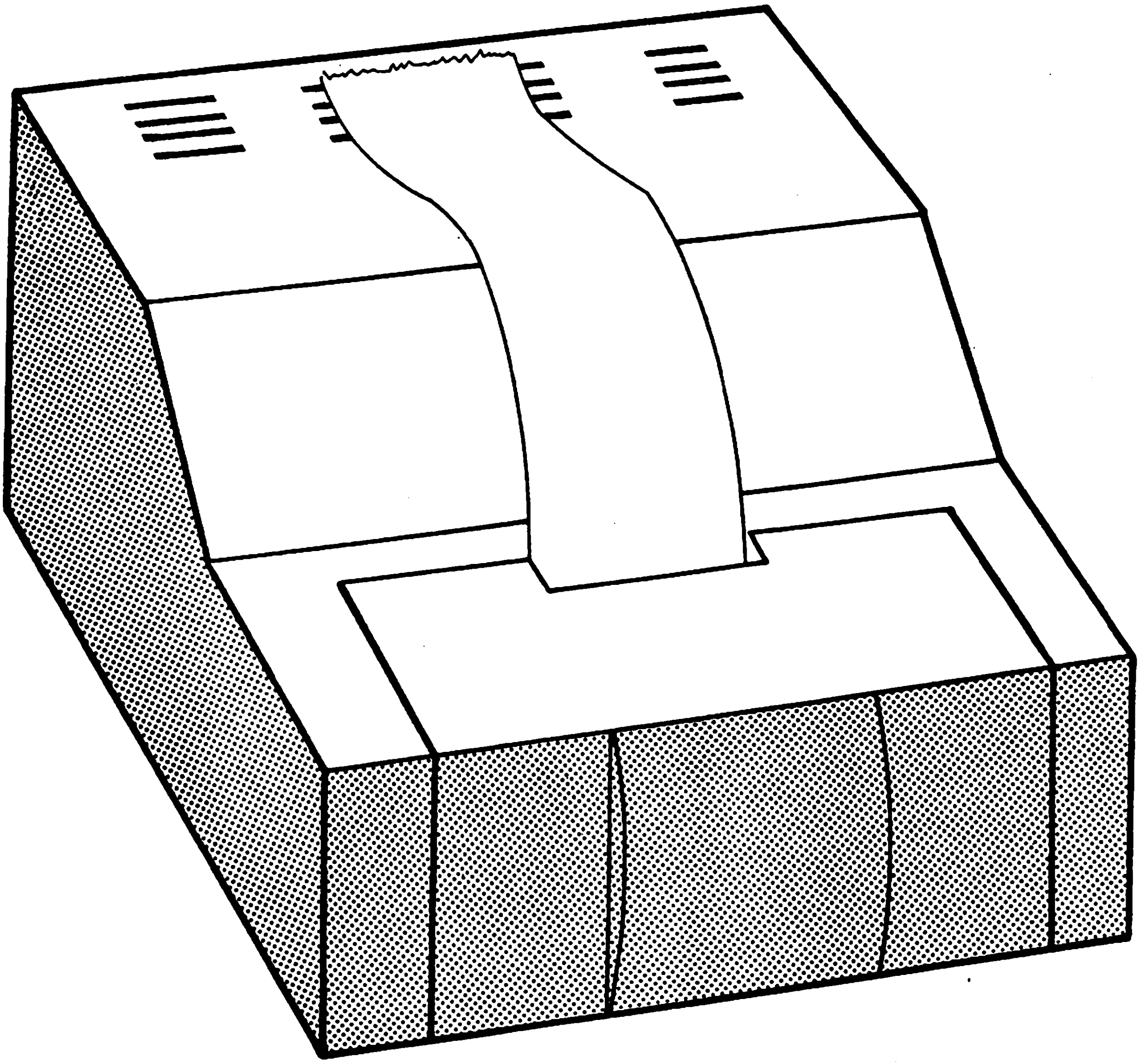


Fig. 4:4

B.C.L. 7 x 5 DOT MATRIX SERIAL PRINTER (PO75/248)

This printer is of the mosaic type, each character being formed by a series of discontinuous points. The mosaic is a matrix of 5 x 7 points, 5 in the vertical plane, 7 in the horizontal plane. The speed of printing is 75 characters per second with synchronous or asynchronous modes of data transfer. The printer handles two sets of continuous stationery and outputs hard copy of 249 characters per line, the printing head tabulating to any position from 1 to 248 both in the forward and reverse direction.

There are 64 printing characters available, including Letters A to Z, Numerals Ø to 9 and also various other symbols as shown in the Code Set in Appendix 3. The character size is 2.54 mm. high and 1.83 mm. wide, with 2.56 mm. pitch. Line spacing is 6 lines to the inch.

The printer accommodates up to six copies of 49 gsm (maximum) using carbon interleaving, or 7 or more if special paper is used. The maximum width of paper overall, either two sets side by side or a single set, is 28", and the minimum width of paper which can be used is 4". Standard perforated (sprocket holed) continuous fanfold stationery (to BS4623-1970) should be used.

The output from the processor to the printer is controlled by the DATO 1 START command, DATO 2 START being used instead whenever a 'Tab' is to be performed. The printer will tab backwards or forwards to the specified position at a speed of 75 ch/sec before printing the next character. Since a TAB is executed at printing speed only, there is no time advantage in using the facility, only a saving in core storage. Spaces may, of course, be used instead of tabs where required, but there is no backspace available. Carriage Returns and Line Feeds may be called where necessary from program in any sequence. The C/R command on its own causes the printer to carriage return at an approximate speed of 220 ch/sec, but does not move the paper up. Should a C/R command be issued whilst the carriage is at the left-hand margin there will be no movement of the carriage. If, through a program error, the PO75 is asked to print more than 249 characters, eventually an automatic CR will be called to prevent the printing head locking up. This facility is only there to protect the hardware and must not be relied upon. LF1 normally refers to the top left-hand tractor and LF2 to the bottom or right-hand tractor.

Controls

There are five push-buttons located on the right-hand side of the cabinet, as shown below :-

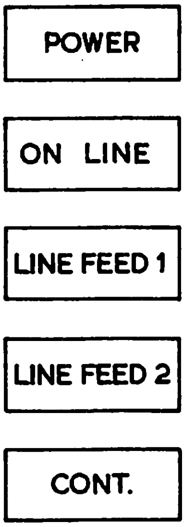


Fig. 4:5

1) Power

This is a red pushbutton control which, when depressed, applies power and also illuminates the switch. Depressing it once again turns the power off, and the illumination is also turned off. When power is first switched on, the PO75 automatically performs a C/R.

2) On Line

This is a green pushbutton control which is illuminated when on. Pressing this pushbutton places the printer 'On Line'; that is, causes the printer to respond only to program instructions, disabling the manual controls.

Should the machine be switched 'off-line' while in use, any instruction already in hand will be completed before going 'off-line', when the machine will no longer respond to program instructions.

3) Line Feed 1

This is a yellow pushbutton with no illumination. In the 'off-line' mode, each depression of 'Line Feed 1' will index the top (left-hand) tractor by one line.

4) Line Feed 2

This is a yellow pushbutton with no illumination. In the 'off-line' mode, each depression of 'Line Feed 2' will index the bottom (right-hand) tractor by one line.

5) Cont.

This is a white pushbutton with no illumination. In the 'off-line' mode, if this 'Continuous' button is pressed in conjunction with either 'Line Feed 1' or 'Line Feed 2', the paper will feed continuously until either button is released.

On the top right-hand side of the cabinet there are two forms control Knobs, by means of which the print may be adjusted to correspond to the lines on the paper. These Knobs may be used when the machine is switched off, and the arrangement is such that the range of movement from these two controls is unlimited.

LINE PRINTER

The Line Printer available for use with the Molecular 18 computer is the LP 3000, with a printing speed of 135 lines/min (300 ch/sec.), plus or minus 2%. It outputs hard copy composed of 132 characters per line. 64 printing characters are available, including Letters A to Z, Numerals Ø to 9 and various other symbols as shown in the Code Set in Appendix 4. Each character is .070" wide and .1" high and there are 10 characters per inch. The printer accommodates up to three copies of paper stock from 4" to 14⅞" wide, each form being 11" maximum in length. Spacing is six lines to the inch.

Each printer has a hardware buffer which holds up to 132 characters. To print a line, the program must first load this buffer character-by-character ; once the buffer is full the line of characters is printed automatically. However, when it is not desired to print a full line the program need only send characters, including spaces, as far as the right-most nonspace character ; the giving of a Print command (C/R) at this point initiates a print cycle, with only the filled portion of the buffer producing a print-out. In every print cycle, printing starts automatically in the correct position at the left edge of the paper ; it is not necessary to send a LF instruction unless double spacing is required.

Output to the Line Printer can be controlled in either of two ways, Character-by-Character mode or Data Channel Mode.

In character-by-character mode it is possible to output several lines in one output.

e.g. If an output is made of "Name C/R Address C/R Address C/R" the Line Printer will print out three lines.

It must be emphasised that the printer will automatically output a line *either* as soon as the buffer (132 chars.) is filled *or* when a C/R command is encountered, and not under any other circumstances.

The Line Printer may also be used in Data Channel Mode in which case the instruction DATO 2 sets the mode. Then DATO 1 START (with the relevant Accumulator containing the appropriate core buffer address) initiates the transfer to the internal buffer of the Line Printer. Done is set and printing occurs *either* when this buffer is filled *or* when it receives a C/R, therefore only *one* line at a time may be output when in Data Channel Mode. The Line Printer then returns to character-by-character mode.

e.g. If an output is made of "Name, C/R Address, C/R Address, C/R", only the first line, i.e. Name, will be printed if in Data Channel Mode.

The disadvantage of using character-by-character mode is that it uses much more processor time than when in Data Channel mode.

Note :- Any "Line Feed" or "Top of Form" characters contained in a line to be output do not go into the internal buffer of the Line Printer, and are executed before the line is actually printed. (This applies to both modes of output).

e.g. Output of A B L/F CD L/F C/R results in L/F L/F ABCD.

Therefore, as L/F and 'Top of Form' characters do not go into the internal buffer, they do not count towards the 132 characters required to to fill the buffer.

Operating Controls and Indicators for Line Printer

Prior to operating the Line Printer, ensure that :-

a) The paper is loaded

b) The ribbon is installed

c) The Forms Thickness control is set to the required forms thickness

d) The power is switched on.

There are six controls situated on the top right-hand side of the cabinet, as shown in Fig. 4. 6, below :-

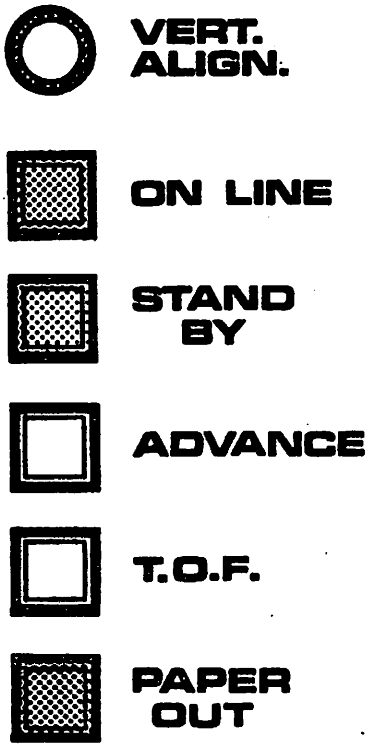


Fig. 4:6

1) VERT. ALIGN. This is a rotary control knob, adjustment of which changes the relationship of the printed line to the paper form. Turning the knob clockwise raises the printed line on the paper form. Turning the knob counter-clockwise lowers the printed line on the paper form. A total adjustment of four lines can be made.

2) ON LINE. This is a green pushbutton control which is illuminated when on. Pressing this pushbutton places the Printer 'On Line'; that is, causes the Printer to operate only under program control. When this pushbutton is illuminated, the Standby pushbutton illumination is turned off.

3) STANDBY. This is a yellow pushbutton control which is illuminated when on. Pressing this pushbutton places the Printer in a Standby position ; it is also activated when power is first applied to the Printer. When in a Standby condition, the On Line pushbutton illumination is turned off, and the Advance and T.O.F. buttons may then be activated for manual control of paper movement.

N.B. When the Printer is to be shut down, the Standby pushbutton should be depressed before switching power off.

4) ADVANCE. This is a white pushbutton with no illumination. Pressing this pushbutton causes the paper to advance one line if the Printer is in Standby mode.

5) T.O.F. This is a white pushbutton with no illumination. Pressing this pushbutton causes the paper to advance to the top of the form (the start of a new printing page form), if the Printer is in Standby mode.

6) PAPER OUT. This is a red pushbutton which is illuminated if the paper runs out. This condition will occur when the forms in the machine are used up, when the Printer automatically goes into Standby mode.

i.e. The Standby lamp is illuminated and the On Line lamp turned off.

At this time there will be approximately six inches of unused form left in the machine. If it is desired to complete the printing on this form, depression of the On Line pushbutton momentarily, will cause one line to be printed. This may be repeated until all lines have been filled, remembering that the Printer will continue to print after the bottom of the form is past if the On Line pushbutton continues to be pressed.

Another use for the Paper Out button, either when On Line or in Standby Mode, is that depression of it rotates the ribbon, thereby enabling a piece of torn or holed ribbon to be by-passed.

Format Tape.

A vertical format unit for either 12 channel or 4 channel IBM compatible format is available for operation with the Printer on an optional basis.

NUMERIC KEYBOARD & DISPLAY

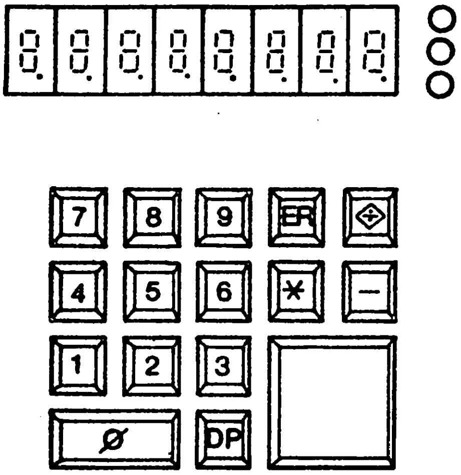


Fig. 4:7

NUMERIC KEYBOARD WITH DISPLAY

The design of the Numeric Keyboard is based upon a 10-key adding machine design, with a raised dot on the '5' key to encourage touch operation. The actual lay-out of the 16 key Keyboard is shown on the opposite page, see Fig. 4. 7.

The unmarked key at the bottom right-hand side is the "Accept" (or End Entry) key. This key produces the "ETB" Code, and should be programmed to give an End Entry function.

Each key produces a Code, as shown in the Code Set in Appendix 7.

The ER key produces the "Can" Code and will normally be programmed to provide "Erase", "Error" or "Cancel"Facilities, and will clear the Display.

The ◈ key can be programmed to initiate a sub-total Function (either on an invoice, or to display a sub-total of entries made so far), or perhaps to act as a "Divide" key in a straightforward calculator context.

The * key can be programmed to initiate a "Total" Function (either on an invoice, or to display a total of the entries made to date), or perhaps to act as a "Multiply" key in a straightforward calculator context.

The - key produces the "-" Code and will normally be programmed to indicate negative figures. It can also be programmed to alter the arithmetic meaning of fractional digits entered .

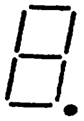
i.e. 12.05 = 12 1/20th ordinary decimal entry
12-05 = 12 5/16ths programmed fractional entry of pounds and ounces.

The DP key produces the "period" code, and will normally be programmed for use as a decimal point.

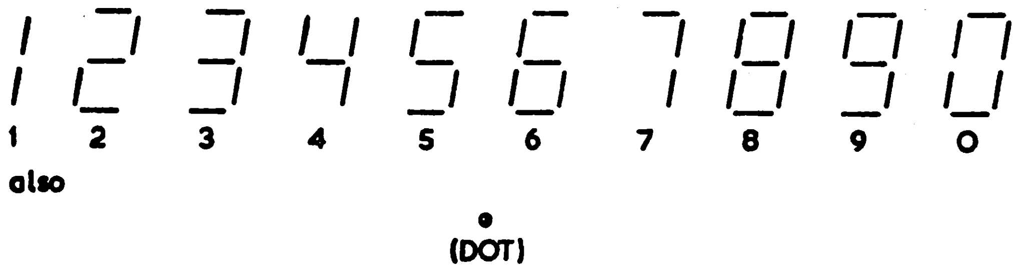
The Accept key, as mentioned previously, produces the "ETB" Code, and should be programmed to give an "End Entry" function.

It should be mentioned that the afore-mentioned keys are not tied by hardware to any particular part or function and, where necessary, one of the keys may be programmed as a "Control" (or Change Meaning) key, so that the combination of this key and a numeric key can give Control Functions to suit particular requirements. This is an easy method of providing Control facilities with very little complication in operation or program.

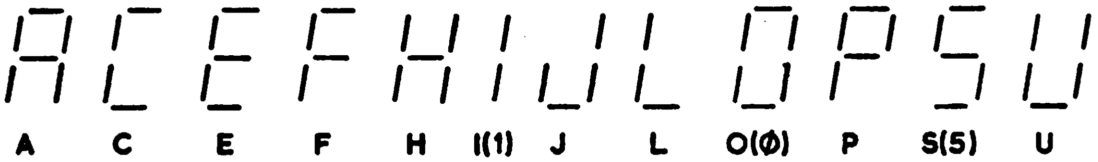
The Keyboard has a single character Input buffer. Associated with the Keyboard is the eight-positional Display, which uses a 7-bar display system. The 7-bar matrix is as follows :-



All numerals can be displayed from the Keyboard or the processor thus :-



The numeric display can also display certain alphabetic characters called from the memory by program :-

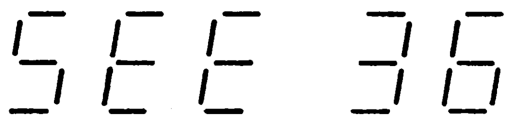


Characters displayed from memory are called up by the appropriate Codes as follows (and as shown in Appendix 7) :-

	Bits		Bits
1	65...1	E	7...3.1
2	65..2.	F	7...32.
3	65..21	H	7..4...
4	65.3..	I (1)	.65...1
5	65.3.1	J	7..4.2.
6	65.32.	L	7..43..
7	65.321	O(Ø)	.65....
8	654...	P	7.5....
9	654..1	S(5)	.65.3.1
Ø	65....	U	7.5.3.1
A	7....1	.	.6.432.
C	7...21		

This Facility can be used to convey simple messages to the operator. i.e. error indications, type of entry required next, etc.

Where a large number of messages have to be displayed for the operator's attention, one of the best is :-



meaning "refer to Message 36" - this allows as many messages as necessary. Obviously, many other messages can be displayed - some of these are listed on opposite page.

A
ALL
ALLOC
ASSESS
CALL
CEASE
CHANGE
CHOICE
CHOOSE
CUS
EACH
FAIL
FALSE
FILE
FILL
HI
IF
IS
LESS
LIEU
LO
LOOP
LOSS
LULL
OF
OFF
PASS
PLEASE
PLUS
POS
SAFE
SALE
SCALE
SPACE
UP

The display output from the processor is a programmed DATO 1 command. The program can clear the Display or send a message at any time (the maximum speed of output of messages sent in this way is at the rate of 1 character per 100 µ).

The Display works on the shift register principle, each character entering at the right and all shifting left one place. When programmed to do so :-

It shows a decimal point when the DP key is used
a blank frame when the '-' key is used
a blank frame when the ✱, ◈ or Accept keys are used.

Operation of the 'ER' key clears the Display.

The Keyboard is also equipped with a series of operator warning lights, which are entirely automatic. When the Keyboard is 'ready', and the CPU able to accept data, the Green light is on, telling the operator to enter a character. Immediately an entry is made the Green light goes off, the Amber light comes on and a buzzer sounds, starting to make a half-second 'bleep' noise, signalling that no more data will be accepted. Immediately the character is processed and the Keyboard is made 'ready' again, the buzzer stops, the Amber light goes off and the Green light comes on again. If the character is accepted straight away, this process is normally so fast that the Amber flicker and bleep will not be noticed by the operator. However, if the Keyboard is not made 'ready' again quickly (i.e. because the Keyboard is Off Line or awaiting access), the amber light stays on and the half-second bleep will be heard. Should the operator press another key while the Keyboard is in this state, the Red light will come on and Status Bit 3 be set (finger trouble). The Red light also comes on for a Parity error, when Status Bit 2 is set. (The Red light is cleared by an I/O Pulse from Software.)

The half-second bleep can also be produced by programming the Code "BEL" (Bits 1, 2 and 3). The Code "CAN" (Bits 4 and 5) clears the Display.

ALPHA/NUMERIC KEYBOARD WITH DISPLAY

The design of this Keyboard is based upon an accepted typewriter (ECMA) configuration with a numeric Keyboard joined to the right-hand side. The actual lay-out is shown on the opposite page, Fig. 4.8.

The design of the Numeric Keyboard is based upon a 10-key adding machine design, with a raised dot on the '5' key to encourage touch operation.

The unmarked key at the bottom right-hand side is the "Accept" (or End Entry) key. This key produces the "ETB" Code, and may be programmed to give an End Entry function.

Each key produces a Code, as shown in the Code Set in Appendix 7. :-

The ER key produces the "CAN" Code, and would normally be programmed to provide Erase, Error, or Cancel facilities, and will clear the Display.

The⟠key can be programmed to initiate a sub-total function, or perhaps to act as a 'Divide' key in a straightforward calculator context.

The ✱ key can be programmed to initiate a total function, or perhaps to act as a 'Multiply' key in a straightforward calculator context.

The - key produces the '-' code and will normally be programmed to indicate negative figures.

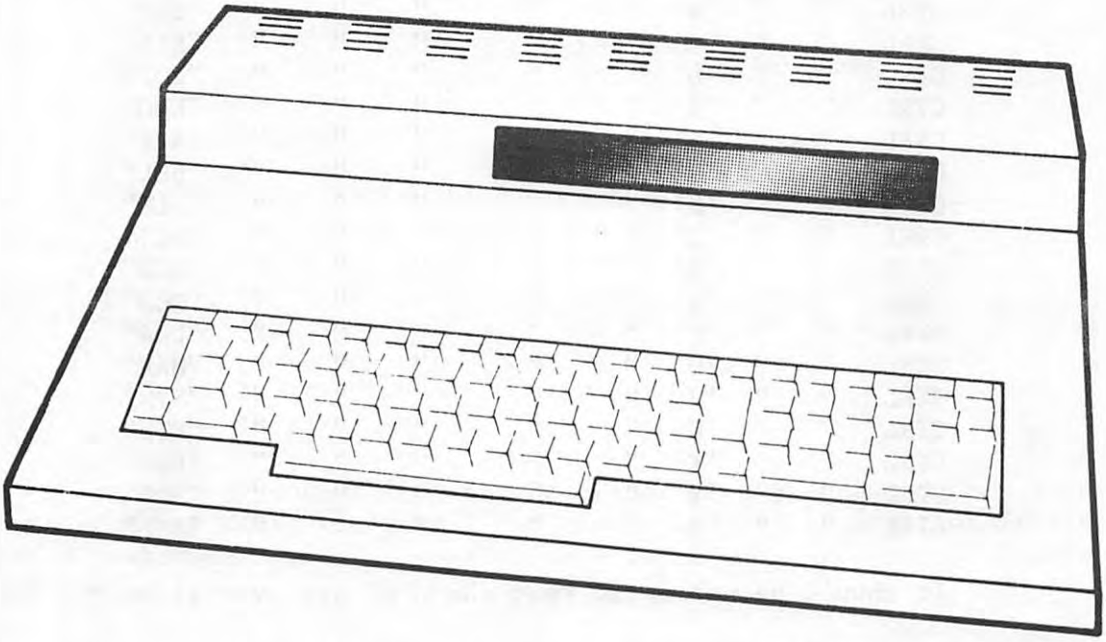
The DP key produces the'period' code, and will normally be programmed for use as a decimal point.

The Accept key, as mentioned previously, produces the 'ETB' Code, and could be programmed to give an 'End Entry' function.

The 'typewriter' portion of the Keyboard produces 68 characters, as well as being equipped with the normal Tab, Carriage Return, Space bar, Shift, Backspace, Line Feed and Vertical Tab. The Control key (CTRL) used on its own has no function, but used in conjunction with certain other keys can produce the ECMA Control Codes. The blank key between the zero of the numeric keyboard and the right-hand shift key is inoperative. The % (per cent) and ‰ (per thousand) keys set next to the numeric portion of the Keyboard may be used to modify calculations only if programmed in this way.

All available Upper Case characters are as depicted on the lay-out on the opposite page. The key with the down-pointing arrow on the left-hand side just above the shift key is the shift lock ; while this is in use, a red light will show in the bottom left-hand corner of the Keyboard. Either shift key can, of course, be used independently of this shift lock.

The BCL 7-bit coded character set for the Keyboard is shown in Appendix 7. Parity is even - Bit 8 is added where necessary.



ALPHA—NUMERIC KEYBOARD & DISPLAY.

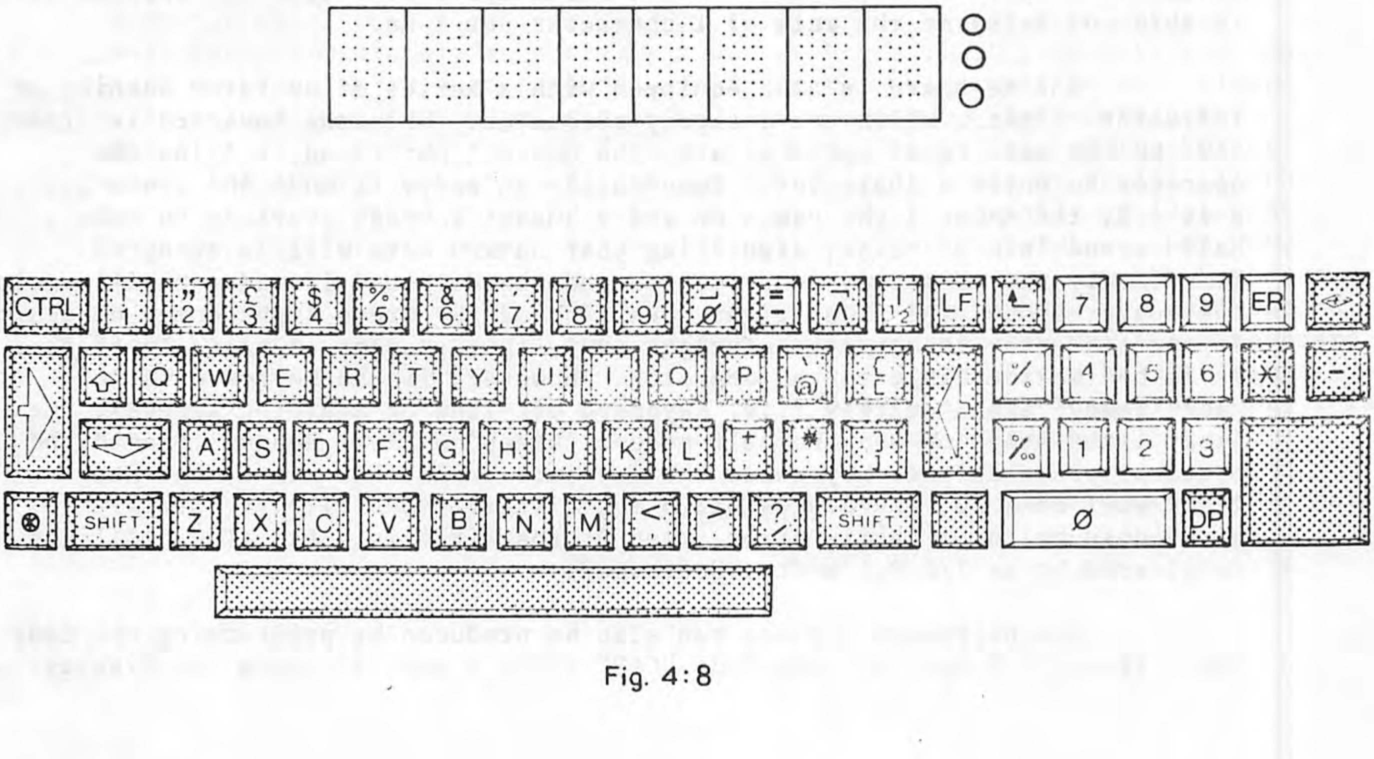


Fig. 4:8

CTRL plus the '@' key produces the ASCII Code "NUL"
CTRL plus the 'A' key produces the ASCII Code "SOH"
CTRL " " 'B' " " " " " "STX"
CTRL " " 'C' " " " " " "ETX"
CTRL " " 'D' " " " " " "EOT"
CTRL " " 'E' " " " " " "ENQ"
CTRL " " 'F' " " " " " "ACK"
CTRL " " 'G' " " " " " "BEL"
CTRL " " 'P' " " " " " "DLE"
CTRL " " 'Q' " " " " " "DC1"
CTRL " " 'R' " " " " " "DC2"
CTRL " " 'S' " " " " " "DC3"
CTRL " " 'T' " " " " " "DC4"
CTRL " " 'U' " " " " " "NAK"
CTRL " " 'V' " " " " " "SYN"
CTRL " " 'W' " " " " " "ETB"
CTRL " " 'X' " " " " " "CAN"
CTRL " " 'Y' " " " " " " EM "
CTRL " " 'Z' " " " " " "SUB"

It should be mentioned that the CTRL key over-rides the Shift Key.

Associated with the Alpha/Numeric Keyboard is the eight-positional Display, which uses a 5 x 7 Matrix Dot System.

The Display works on the shift register principle, each character entering at the right and all shifting left one place (i.e. 'The cat sat on the mat' displays 'THE MAT' at the end of the text).

Any 'displayable' character called from memory by program, using the appropriate code, can be shown on the Display. This facility can be used to convey many types of messages to the operator. The Display Output from the processor is a programmed DATO 1 command. The program can clear the Display or send a message at any time, - the maximum speed of output of messages sent in this way being at the rate of 1 character per 1 ms.

The Keyboard is also equipped with a series of operator warning or indicator lights, which are entirely automatic. When the Keyboard is 'ready', and the CPU able to accept the data, the Green light is on, telling the operator to enter a character. Immediately an entry is made the Green light goes off, the Amber light comes on and a buzzer sounds, starting to make a half-second 'bleep' noise, signalling that no more data will be accepted. Immediately the character is processed and the Keyboard is made 'ready' again, the buzzer stops, the Amber light goes off and the Green light comes on again. If the character is accepted straight away, this process is normally so fast as to be imperceptible to the operator. However, if the Keyboard is not made 'ready' again quickly (i.e. Keyboard off-line or awaiting access), the Amber light stays on and the half-second 'bleep' will be heard. Should the operator press another key while the Keyboard is in this state, the Red light will come on, and Status Bit 3 set (finger trouble). The Red light also comes on for a Parity error, when Status Bit 2 is set. (The Red light is cleared by an I/O Pulse from Software).

The half-second bleep can also be produced by programming the Code "BEL" (Bits 1, 2 and 3). The Code "CAN" (Bits 4 and 5) clears the Display.

THE C.R.T. VISUAL DISPLAY UNIT TERMINAL

The C.R.T. Visual Display Unit Terminal can be used as an on-site device, or as a stand-alone remote terminal communicating with the CPU in a conversational mode, either over a telephone line or by a cable up to 50 ft. long. The current unit is type 92423-10, which is called a 'Block Mode Conversational Display Terminal'.

Three methods of display and conversation with remote devices are possible :-

a) Half Duplex (Conversational Mode)

b) Full Duplex (Echoplex/check)

c) Block Mode

In each case, data coming into the Terminal will take preference over local operations, and the Keyboard will be disabled whenever data transfer is taking place. Received data will display at and continue from the last cursor position, unless programmed to do otherwise.

The 2-position Duplex switch (HALF and FULL), which determines the manner in which Keyboard-entered data is transmitted to the communication system, is on the back of the display terminal, and is normally preset for a particular installation.

Half Duplex setting causes each Keyboard data entry to transmit character-by-character to remote equipment and also to display immediately as a result of key depression.

Full Duplex setting causes each Keyboard data entry to transmit character-by-character to remote equipment, and each entry is not displayed on the view screen unless returned (echoed) from the remote equipment (e.g. the CPU).

Block Mode The Block Mode switch is located above the numeric cluster on the Keyboard. This two-position pushbutton switch activates BLOCK MODE operation only when in the depressed position. Block Mode setting causes each Keyboard data entry to display immediately as a result of key depression. This enables data to be composed on the display before actual transmission. Transmission can be in stages using the 'STX' and 'EOT' keys, or the whole page can be transmitted at once. By exercising control of the cursor on a full page and selective use of 'EOT', the page may be transmitted in separate blocks.

The speed of transmission is determined by the BAUD RATE switch on the back of the display terminal, which is selectable. However, the baud rate is normally preset for the system on which it is used, as required by the Modem or Telephone line when transmitting data. The Modem Coupler to which it is connected in the CPU must be of the correct type for the speed selected.

The complete VDU unit consists of :-

a Keyboard
a C.R.T. internal control memory
modem interface CCITT-V24

The Keyboard

The Keyboard can be describedas having three-level or tri-state operations :-

Shift, Unshift or Control.

It has three sections :-

a) A silent electronic data entry Keyboard 'styled' like a standard tele-typewriter layout.

b) A numeric 10-key adding machine cluster, in addition to the basic numeral row.

c) Common tele-typewriter function codes available by 'CNTRL + Key' depression and, in addition, frequently used functions available from their own keys.

The Keyboard is the Input device for the Terminal, using Device Code 50 upwards. The actual lay-out for the 92423-10 is shown on the opposite page. The character repertoire includes the alphabet in upper and lower case, arabic numerals Ø to 9, punctuation marks and special characters, all of which are shown in Appendix 6 .

It is possible to disable the 96 character set and select a 64 character upper-case only sub-set (Columns 2, 3, 4 & 5 of Code Set in Appendix 6.).

The following paragraphs describe operation of the Shift, Lock, Space bar and Rub Out keys and the Repeat key.

Shift and Lock Keys

The two Shift keys and the Shift Lock key allow selection of uppercase letters or the uppercase characters on the double-character keys. They have no effect on the Numeric cluster. Depression of the Shift Lock key will lock the Keyboard in uppercase position. The Lock key is released by depression of the adjacent shift key.

Space Bar

The depression of the Space bar causes a character space to display above the cursor. The cursor will move forward one character position in the normal manner.

C.R.T. VISUAL DISPLAY TERMINAL KEYBOARD. (92423-10)

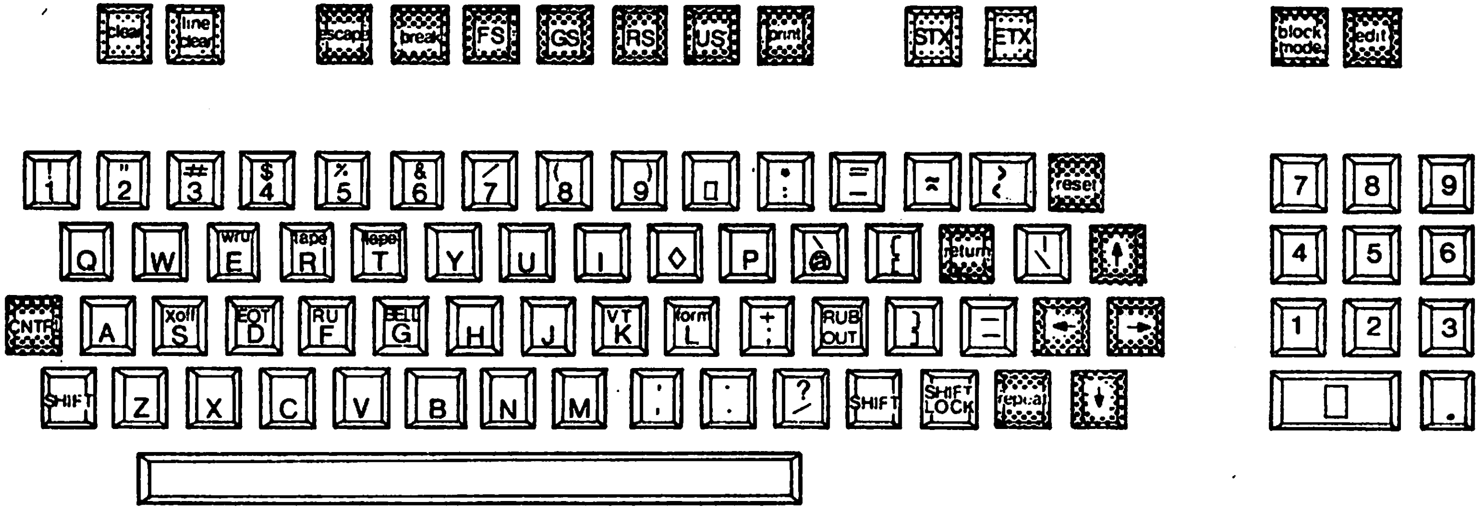


Fig. 4:9

Rub Out

When the terminal is set for Block Mode Duplex operation, depressing this key produces the symbol '▮' above the cursor, the cursor then moving forward one character position in the usual way. In either of the other two Duplex modes, depressing this key generates the Rub Out code for transmission only (LOCAL operating mode, of course, prevents transmission). This same symbol '▮' could appear within a message received from the CPU. It would indicate that a parity error may have occured in the transmission of the character meant for that position.

Repeat

This is an assigned special function key. Depression of this key in conjunction with any other key or combination of keys causes the particular character or control code for that/those key/s to be generated and/or displayed on the screen at approximately 10 times per second. This key is provided purely as an operator convenience for entering characters which have to be repeated. (e.g. ---------------)

Control Function Keys

The terminal Keyboard will generate 32 control function codes commonly used with tele-communications input devices. Appendix 6. shows these codes, and also explains the Keyboard operations required to generate these codes. Note that many often-used communication and edit function codes are available from their own separate keys as well as from the CNTRL + character-key combination, typical of tele-typewriter operations.

Use of the control function keys depends on the communications mode in which the terminal is being operated, and whether it is desired to affect the display only or to transmit control codes to remote receiving devices. Rules governing the use of the control function keys are described in the following paragraphs.

N.B.

All 32 control functions can be generated without using the shift key. However, depression of the Shift key in conjunction with CNTRL + key operation or the separate control function does not interfere with the desired function.

CNTRL + Character Key

Depressing the CNTRL key in conjunction with any data key assigned a control function, generates control codes as follows :-

IF HALF DUPLEX (and not LOCAL) the code is transmitted to the remote device (e.g. CPU) and the display is affected in accordance with the function's purpose.

IF FULL DUPLEX (and not LOCAL) the code is transmitted to the remote device (e.g. CPU) and the display is not affected unless the remote device returns (echoes) a display affecting a control function code or displayable character code to the terminal.

IF BLOCK MODE, the paired lower-case sign on non-operational Control keys is stored (hidden) in the displayed message at the correct cursor position, and appears on the view screen as a blinking character whenever the CNTRL key or the EDIT key is depressed. (This blinking character identifies the control function code stored at that particular point, and will correspond to the appropriate character shown in Columns 4 and 5 of Appendix 6.). The exceptions are :-

1) The SO and SI codes (CNTRL + N and CNTRL + O), which display as their identifying characters (non-blinking❮and❯respectively) and initiate and terminate inverse video display.

2) The ETX code, which always displays a non-blinking '▲'.

Generally, when the terminal is in BLOCK MODE, the CNTRL + key operation should be used only when it is desired to insert a control function in a message to a remote device as part of that message for the receiving device to perform. The separate control function key should be depressed when it is desired either to transmit (STX) or to perform edit functions (CLEAR, RESET, etc.) on the displayed message while your terminal is in Block Mode.

Separate Control Function Key

Depressing separate control function keys (e.g. RESET, RETURN, etc.) generates control codes as follows :-

IF HALF DUPLEX (and not LOCAL) the code is transmitted to the remote receiving device, and also causes display operation as described later under the control function key descriptions.

IF FULL DUPLEX (and not LOCAL) the code is transmitted to the remote receiving device, and the display is not affected unless this remote device returns a display-affecting control function code or a displayable character code to your terminal.

IF BLOCK MODE (or LOCAL) the code causes display operation as described under the control function key description below.

CLEAR Key (or CNTRL + X when not Block Mode)

The CLEAR function removes all data from the display screen and resets the cursor. If the machine is in scroll format mode, the cursor resets to the first character position of the last line. If in page mode the cursor resets to the first character position of the top line.

LINE CLEAR Key (or CNTRL + V when not Block Mode)

When in page mode, the LINE CLEAR Function removes all displayed data from the cursor position to the end of the line. The cursor does not move, and only the line with the cursor is affected.

When in scroll mode a LINE CLEAR cannot be executed, either from Keyboard or by output from program.

RETURN Key (or CNTRL + M when not Block Mode)

The RETURN function causes the cursor to reset to the first character position of the line it is in (Carriage Return). This does not affect data.

RESET Key (or CNTRL + Y when not Block Mode)

The RESET function does not affect displayed data. If in scroll format mode, the cursor will reset to the first character position of the last line. If the machine is in page mode, the cursor will reset to the upper left-hand corner of the display.

The following four keys with arrows give control over the cursor in four directions, for use as a pointer to the next character position which will be affected by input or output :-

BACKSPACE (←) Key (or CNTRL + H when not Block Mode)

The backspace function moves the cursor back one character position without affecting displayed data. If the cursor is in the first character position of a line when the key is depressed, it will move automatically to the last character position in the preceding line. If the cursor is in the first character position of the top line, it moves to the last character position in the bottom line.

SKIP (→) Key (or CNTRL + U when not Block Mode)

The skip function advances the cursor one character position. If the cursor is in the last character position of any line except the last line when the skip function is enabled, it will move automatically to the first character position of the next line. If the cursor is in the last character position of the bottom line when in scroll mode, all data moves up one line position, the top line disappears from the viewing screen and the cursor moves to the first character position of the new blank bottom line. If the cursor is in the last character position of the bottom line when in page mode, the cursor moves up to the first character position in the top line.

Up (↑) Key (or CNTRL + Z when not Block Mode)

The Up function moves the cursor to the same relative position in the next line up, and displayed data is not affected. Should the cursor be in the top line when the up function is enabled, the cursor moves to the same relative position in the bottom line.

Down (↓) Key (or CNTRL + J when not Block Mode)

The Down function (Line Feed) moves the cursor to the same relative position in the next line down. Displayed data is not affected unless scroll mode is active and the cursor is in the last line. In this case all data moves up one line, the top line disappears from view on the viewing screen and the cursor returns to the first character position in the blank bottom line. This operation provides a Carriage Return and Line Feed function when these conditions apply. When the cursor is in the bottom line and page mode is active, the cursor moves to the same relative position in the top line.

Inverse Video (CNTRL + N and CNTRL + O)

The coding CNTRL + N generates inversion of video (black characters on a white back-ground) in certain selected areas of the display. The 'end' inverse video control code is CNTRL + O. The portion of the display between these start and end codes appears as black characters on a white back-ground. There are no limitations as to where, how many, or how much may be inverted. Inverse video fields are bracketed on the display, with the start inverse video symbol '<' and the end inverse video symbol '>'. The start and end codes may also be received from the CPU (see Appendix 6.).

BREAK Key

This is an assigned special function key. Depression of this key when the VDU is in remote operating condition transmits a 'break' signal to the CPU. This is a common tele-communications signal, which drives the Data signal line to "space" condition for 300 millisecs.

If a break condition should occur on the receive data line from the CPU, the display will show two error symbols '▮▮'.

Start of Text (CNTRL + B)

When in HALF or FULL DUPLEX mode, depression of the CNTRL key in conjunction with the B key transmits the Start of Text control function code to the remote equipment. In FULL DUPLEX, if the remote equipment echoes the code, the terminal disregards the code. In HALF DUPLEX, the display portion of the terminal ignores the code.

When the terminal is in BLOCK MODE, depression of the CNTRL key in conjunction with the B key enters the Start of Text control code at the cursor position in the displayed message. Thereafter, whenever the CNTRL key or Edit key is depressed, this control code will appear on the view screen as a blinking 'B'. This function serves to mark the beginning of multiple messages which an operator may wish to compose on the display screen. After the start of a message has been marked with this function, the end may be marked with the End of Text Function (see ETX description).

End of Text (ETX or CNTRL + C)

When in HALF or FULL DUPLEX mode, depression of the CNTRL key in conjunction with the C key or depression of the actual ETX key, transmits the End of Text Code to the remote equipment. The EXT character (▲) is displayed when in HALF DUPLEX mode, and also when in FULL DUPLEX mode if the remote equipment echoes this code back.

When operating in BLOCK MODE, depression of either the ETX key or CNTRL plus C keys enters the End of Text control code at the cursor position in the displayed message. The code appears on the view screen as the ▲ character. This function may be used to mark the end of the text of multiple messages which are being composed. (This code will not stop message transmission ; only the EOT code will stop transmission before the end of the last display line is reached).

End of Transmission (CNTRL + D)

When in HALF or FULL DUPLEX mode, depression of the CNTRL key in conjunction with the D key (EOT) transmits the End of Transmission code, but the display ignores it.

When the terminal is in BLOCK MODE, depression of the CNTRL key in conjunction with the D key enters the End of Transmission (EOT) control code at the cursor position in the displayed message. Thereafter, whenever the CNTRL key or Edit key is depressed, this control code will appear on the view screen as a blinking 'D'. This function must be used if it is desired to end message transmission at some point before the end of the last display line.

Start of Block Transmission (STX Key)

The STX key allows transmission of displayed messages to remote equipment when operating in BLOCK MODE transmission. When in BLOCK MODE, depression of the STX key locks the data entry keyboard, and the terminal then automatically transfers the message from wherever the cursor is set through the first following EOT code or the end of the last display line, whichever occurs first. Any control codes contained in the message will be transmitted along with the displayed characters. If a carriage return control code is transmitted, any spaces between that carriage return and the next displayable character on that line will not be transmitted, but any other codes will be. The first displayable character, or the end of the line, or the end of the block transfer will disable this space suppression mode of operation.

When operating in FULL DUPLEX or HALF DUPLEX mode, depression of the STX key transmits the Start of Text code, but the display ignores it.

EDIT switch

Above the numeric cluster on the Keyboard is the two position EDIT switch/indicator. In the depressed position, all control functions entered previously by CNTRL plus key operation during messages composed in BLOCK MODE appear as their blinking character identifier.

N.B.

If the CNTRL key is depressed to view location of previously entered control codes (e.g. Start of Text indicator, blinking 'B') so that the cursor can be set to the correct position, on no account may any other keys, such as any of the cursor control keys, be used while the CNTRL key is still held depressed. The CNTRL key must always be released before positioning the cursor on the display, as otherwise control codes will inadvertently be entered into the displayed data. The EDIT switch will be found very useful for the purpose of viewing these hidden control function characters.

The remaining functions :-

FS	(or CNTRL +)	-	File Separator
GS	(or CNTRL +)	-	Group Separator
RS	(or CNTRL +)	-	Record Separator
US	(or CNTRL +)	-	Unit Separator
Escape	(or CNTRL +)	-	Escape

are available for special usage dependent upon the program.

There are four controls and indicators associated with recording data on a hardcopy printer which are not used in the Molecular 18 system. These are the PRTR BSY and PRTR ACT indicators (which are situated among the 6 indicator lights situated on the left-hand side of the operator panel beneath the display), the 'PRINT' key and the "PRINTER REQUEST' push button Switch/Indicator.

DISPLAY

The display portion of the C.R.T. Visual Display Unit Terminal consists of a television display module with a viewing area 8" high and 10" wide. The C.R.T. displays 8 lines of 80 characters (which can be expanded to display 16 lines of 80 characters), and so has a memory of 640 characters. Characters appearing on the display screen are dot formed within a 5 x 9 dot matrix, and they appear nominally 3/32" wide and 3/16" high. A cursor (entry marker) appears on the display screen as a blinking underline dash to indicate the position where the next character entry will display on the screen. Although it is an underline symbol, it is NOT possible to underline messages on the display.

As the cursor advances from the 72nd position of any line to the 73rd or when it starts the bottom line a momentary audible alarm sounds. This audible alarm can also be activated from the CPU to notify an incoming message, or for other programmed usage.

POWER CONTROLS AND INDICATORS

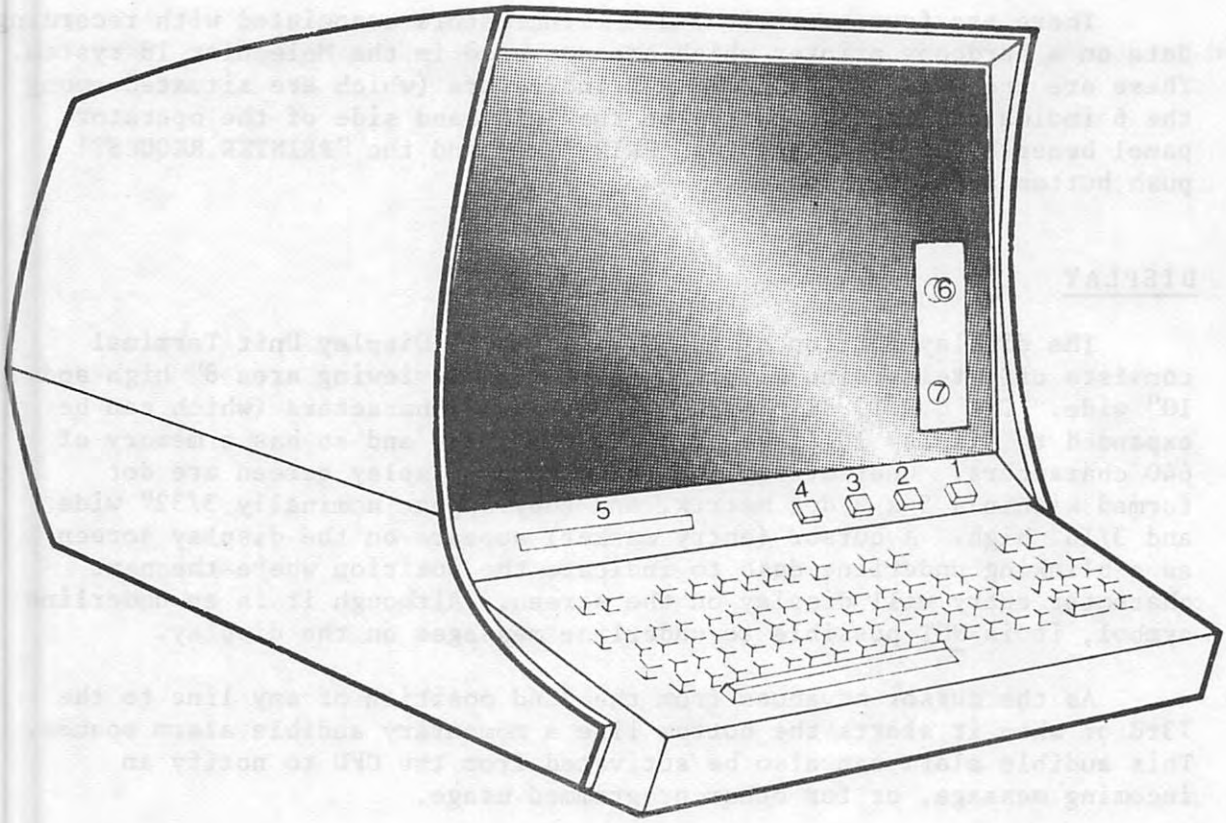
The following section describes the power controls and indicators contained on the C.R.T. as shown in Fig. 4.10.

1) POWER This is a square pushbutton control located on the operator control panel below the display (see diagram on next page). Depressing the switch applies power and also illuminates the switch. Depressing it once again turns the power off and the illumination is also turned off. The cursor should appear on the left-hand side of the display screen within 30 seconds approximately of the power being turned on.

2) PAGE MODE This control provides two display format mode selections, scroll and page. The scroll mode is the normal format used, though the page mode is frequently adopted for display terminal communications. In the scroll mode, data enters the bottom line of the display starting at the left-hand side. When the cursor reaches the last character position

cont/ 4:33

C.R.T. VISUAL DISPLAY TERMINAL



1to 5 AS SHOWN BELOW
6 INTENSITY CONTROL
7 KEYLOCK SWITCH

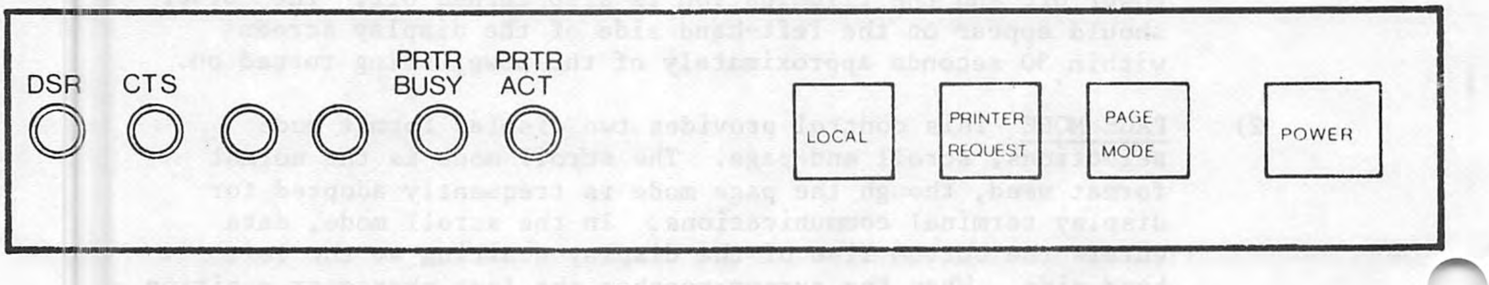


Fig. 4:10

of this line, any additional input of data causes all lines to move up one line space and the cursor to position to the first character position in the bottom line again. The top line disappears from the screen (therefore it should be obvious that one should not use the last character of the last line if 8 lines of information are required to be displayed).

When the PAGE MODE switch is pressed, the indicator is illuminated and scroll format is disabled. The display then operates in the page format mode, meaning that data entry starts at the top left-hand corner of the display and the cursor moves across and down the display screen. When the cursor reaches the last position of the bottom line, any additional input of data positions the cursor to the first character position of the top line, thereafter over-writing the previous data.

3) PRINTER REQUEST Not used.

4) LOCAL This pushbutton switch allows the C.R.T. Visual Display Terminal to be placed off-line from the tele-communications line, and so not transmit any data to the remote party. However, the Terminal will still be able to receive any incoming messages. The switch should be depressed to attain 'Local' condition, when it will be illuminated.

5) There are 6 indicator lights situated on the left-hand side of the operator panel below the display. Only 2 of these are actively used in the Molecular 18 system, to indicate the status of the communications channel between the Terminal and a remote equipment. These are DSR (Data-set Ready) and CTS (Clear to Send).

6) Brightness Control This control should be turned until the required brightness or intensity is reached. Extra high intensity will make any displayed data appear out of focus and will also shorten the life of the display viewing screen.

7) Keylock Switch When the special key is inserted and turned such that the lockout indicator is illuminated, the Keyboard is completely disabled from operator entry. Messages may still be received, but none may be entered by the operator. This control is primarily a protective feature.

There is also a Lockout Indicator which illuminates to indicate that the terminal Keyboard is locked out. This can be accomplished by the Keylock switch and also during data transfer operations.

A Modem Coupler 2100 interface fitted to the Molecular 18 enables data to be transmitted or received serially on one line using the Datel Post Office data transmission service on Public Telephone/Telegraph network or Private circuit (leased lines). The Modem is a free-standing self-contained device connected to a telephone hand-set and the data processing equipment - it is 17" wide x 6.75" high x 14" high. It will be accessed in the normal way, using the same device codes, 50 for Input and 40 for Output. The Modems will all be asynchronous.

THE D1600 & DD1600 DISC

Every disc has 406 circular data tracks on each surface ; a track is divided into 16 Sectors, each of which holds 128 words of data. This gives a total of 12,992 Sectors or 1,662,976 words, which is equivalent to 1600K of core for each disc.

The sectors are numbered Ø to 31277 (Octal). The first 32 sectors (Sectors Ø to 37, the first cylinder) must not be used for permanent data or for vital temporary data, and three cylinders at the end (Sectors 31140 to 31277 inclusive) are reserved for essential routines and contingencies. Sectors 40, 41 and 42 are also reserved, therefore the maximum usable storage available (Sectors 43 to 31137 inclusive) is 12,861 sectors or 1,646, 208 17-bit words. Each word on disc can represent signed or unsigned binary quantities from Ø to 1111111111111111 (i.e. 65,535 decimal), or two character-code Alpha/Numeric characters, or three metacode characters, or four Hexadecimal characters, or an address or a program instruction, etc.

The speed of rotation is 2,400 r.p.m. (25 ms. per revolution). 8 Sectors may be written or read in 12.5 m.secs. once the transfer begins. The average random access time is 49 ms., with the worst being 96.5 ms.

The Disc Controller can control up to 4 disc drives "daisy chained", each able to have 1 fixed and 1 exchangeable disc. Therefore a maximum configuration would give a total of 8 discs, equivalent to a nominal capacity of 12.8 M words on one controller.

There are two boards (2057 and 2058) per controller, mounted in the processor chassis ; the disc drive cabinet contains only the drive, with power packs and air conditioning. The drives incorporate blowers and filters to provide clean air internally as a clean-air environment is very necessary - the heads 'fly' on the boundary layer of air on the disc, and are only loaded when the disc is up to speed.

Bits 15 and 16 in the Status Register specify the Disc Drive (Ø to 3) and, on the DD, Bit 17 is used to specify the Fixed disc. (The DD has the Fixed disc on the same spindle). All discs have the same sector numbering.

Program Functions

Disc transfers occur in Data Channel Mode, therefore the program should address the required disc, specify the 1st buffer core address, specify the Sector Address and whether a Read or Write operation is required.

The Disc System uses several I/O Instructions (see Page 3:35 for correct Assembler mnemonics) :-

DATO 1 (with Device Address)

This loads the Sector address (Ø to 31277 Octal) from Accumulator A or B to Register 1 in the controller and initiates a "seek" - this aligns the read/write heads on the required cylinder. The Sector address will be located, read and checked whether or not START or IOPLS is sent. Register 1 is not cleared at the end of a read or write sequence or when a status error occurs, but will be over-written by a further DATO 1 instruction. Should DATO 1 be sent during a 'seek' it will be ineffective, except that it will set Bit 6 (Address incorrect) in the Status Register. On Auto-Sector Coun (see Page 4:35), the Sector Address in Register 1 will be incremented by the controller, except on the last transfer in each block.

DATO 2 (with Device Address)

This instruction loads the first core address from the specified accumulator into Register 2 in the controller. This address will be incremented automatically after each data-channel transfer by the controller. Note that the core address used must be absolute, rather than the relative address in User Programs. Register 2 is not cleared, but may be over-written by a further DATO 2 instruction.

DATO 3 (with Device Address)

This instruction loads Bits 17 to 10 into Register 3 from the specified accumulator.

Bits 17, 16 and 15 call for the required Drive on a daisy-chain and/or the Fixed disc. These bits are never reset, but may be overwritten by a further DATO 3 instruction.

Bit 17 is "Disc-Select", '1' for fixed or 'Ø' for exchangeable.
Bits 16 & 15 are binary "Drive-Select" (Ø - 3).

Bit 14 is a memory having no operational function, but which may be used as a flag by the program. This bit also is never reset, but may be overwritten by a further DATO 3 instruction.

Bits 13, 12 and 11 A feature has been included which enables the controller to handle multiple sector transfers (up to 8 in sequence, which is equivalent to 1K of words). A counter, in Bits 13, 12 & 11 of Register 3, is loaded on a DATO 3 instruction when multiple sector transfers are required, and will count down to zero as successive sectors are transferred and the Sector Address incremented. On completion, Bits 13, 12 and 11 should be zero and Register 1 should contain the last sector address used. The auto-sector count will be created as follows :-

Bits 13, 12, 11	No. of Sectors
0 0 0	1
0 0 1	2
0 1 0	3
0 1 1	4
1 0 0	5
1 0 1	6
1 1 0	7
1 1 1	8

thus, if multiple sector transfers are not required and Bits 13, 12 and 11 left blank, the controller will transfer a single sector only.

Any status failure will cause an interrupt and the complete transfer should then be retried. Mixed read/write is not possible. Note that it is inadvisable to inspect (DATI 3) during multiple sector transfers, since the counts are changed during the sector sequence.

The count may be cleared by CLEAR or I/O Reset.

Bit 10-Monitor Should be a temporary fault, such as Logic fault, Off Line, Temperature Limit or RTZ be detected (by DATI 3 inspection), DATO 3 + Bit 10 in the relevant accumulator should be sent. A monitor (Bit 10) will then be set in Register 3, only if the delay condition was still present when the DATO 3 + Bit 10 occured. This monitor, if activated, will provide an interrupt when Busy and the source of delay is removed. Monitor may be inspected on a DATI 3 instruction and, if present, the instruction 'START' sent - if Busy is not set at the time of recovery the Interrupt will occur as soon as it is, unless the monitor has been reset previously. Bit 10 may be reset by CLEAR, I/O Reset, or a valid DATO 1 instruction (but not during seek). When a monitor interrupt occurs, the status fail register (Bits 1-8 inclusive) should be clear.

DATI 1 (with Device Address)

This instruction may be used to read back the Sector Address from Register 1 into Accumulator A or B.

DATI 2 (with Device Address)

This instruction may be used to read back the current Core Address from Register 2 into Accumulator A or B.

DATI 3 (with Device Address)

This instruction may be used to read back the Status from Register 3 into Accumulator A or B. This will include advisory status bits, therefore Bit 1, the common fault bit, must be used to test for a status failure.

All bits except 14,15,16 and 17 may be reset by CLEAR, I/O Reset or a valid DATO 1 (not during seek).

START or Set Busy, Clear Done (with Device Address)

This instruction initiates Read from Disc sequence. Read will not occur until a seek has been completed.

IOPLS or Input/Output Pulse (with Device Address)

This instruction initiates Write to Disc sequence. Write will not occur until a seek has been completed.

Note that DATO 1 may occur with START or IOPLS or may follow later. Alternatively DATO 1 may precede the START or IOPLS by not more than 160 μ secs. for Read/Write to occur ; if more than 160 μ secs. elapse, then Read/Write may or may not occur, therefore this situation should be avoided.

Always send the core address first (DATO 2) followed by the DATO 1 etc., as if a core address is not sent any core address sent previously will continue to increment.

CLEAR or Clear Busy, Clear Done (with Device Address)

This instruction clears Busy, Done, interrupt request memories and Status Bits 1 to 10.

I/O Reset

This instruction clears Busy, Done, interrupt request memories and Status Bits 1 to 10, and also restores the heads to the Zero track.

On completion of the sector (or sectors), the Controller sends an Interrupt, and the Status (DATI 3) must then be inspected to ensure that the transfer was valid.

Register 3 bit allocation

Bit 1 = Common 'fault' bit (indicates that a fault exists).

Bit 2 = Logic fault (indicates that the D1600 logic is unsafe, through power supply failure, etc.).

Bit 3 = Off Line (drive not up to speed or not connected).

Bit 4 = Temp. Limit (drive outside temperature limits ; should not be used).

Bit 5 = Checkword Incorrect (cyclic redundancy check failed on read back).

Bit 6 = Address incorrect (address unobtainable, or DATO 1 sent during seek).

Bit 7 = Data late (data channel not available in time).

Bit 8 = Seek error (address called for unobtainable, or heads failed to move correctly).

Bit 9 = Seeking (seek in progress).

Bit 10 = Monitor active.

Bit 11
Bit 12 } = Auto Sector Count.
Bit 13

Bit 14 = Flag.

Bit 15
Bit 16 } = Drive select

Bit 17 = Disc select (fixed or exchangeable).

Status fail bits 1 to 8 inclusive are reset by CLEAR, I/O Reset, valid DATO 1, or Monitor active and source of delay removed.

MAGNETIC STRIPED CARD HANDLER

This device is supplied on a modified IBM 735 I/O Writer to provide both front and rear channel card feed ability, with magnetic data-storage stripes on the front card only.

The unit is completely contained inside a removable assembly mounted above the I/O Writer. The size of the unit is :-

23" (584 mm) Wide
9" (228 mm) High
9" (228 mm) Deep

and the weight is 30 lbs. approximately plus the I/O Writer. The complete assembly is mounted on spring clip pivots so that it can be removed easily for transport or service and swung clear of the typewriter. (It must not be carried by the guide carrier tube as it will bend.)

Three visual indicators coloured Red, Yellow and Green are provided on the right-hand side of the unit for information on Operator (or logic) errors. These indicator lights are under software control, and their function is determined by system details.

The guides on the front feed for the magnetic stripe cards are adjustable in steps of 30 mm.

The Left-Hand Guide has two positions only, A and B :-

Position A is such that character position Ø on the IBM will print 31 mm. in from the left-hand edge of the card.

Position B is 30 mm. to the right of Position A.

The Right-Hand Guide can be positioned at 30 mm. intervals, to cater for 4 card widths.

Removable drop-in guides are provided for the plain card in the rear channel. The Left-Hand Rear Sprocket Feed also has positions A or B only, but the Right-Hand Rear Sprocket Feed has 8 positions at 30 mm. intervals to cater for all plain card widths.
(N.B. Sprocket positioning on the Rear feed only cannot be altered easily by the operator.)

The normal card height is 297 mm. (A4 size), the magnetic stripe card having a minimum width of 210 mm. increasing in 60 mm. steps to a maximum width of 390 mm., and the rear plain card having a minimum width of 180 mm.

cont/4:40

MAGNETIC STRIPED CARD HANDLER

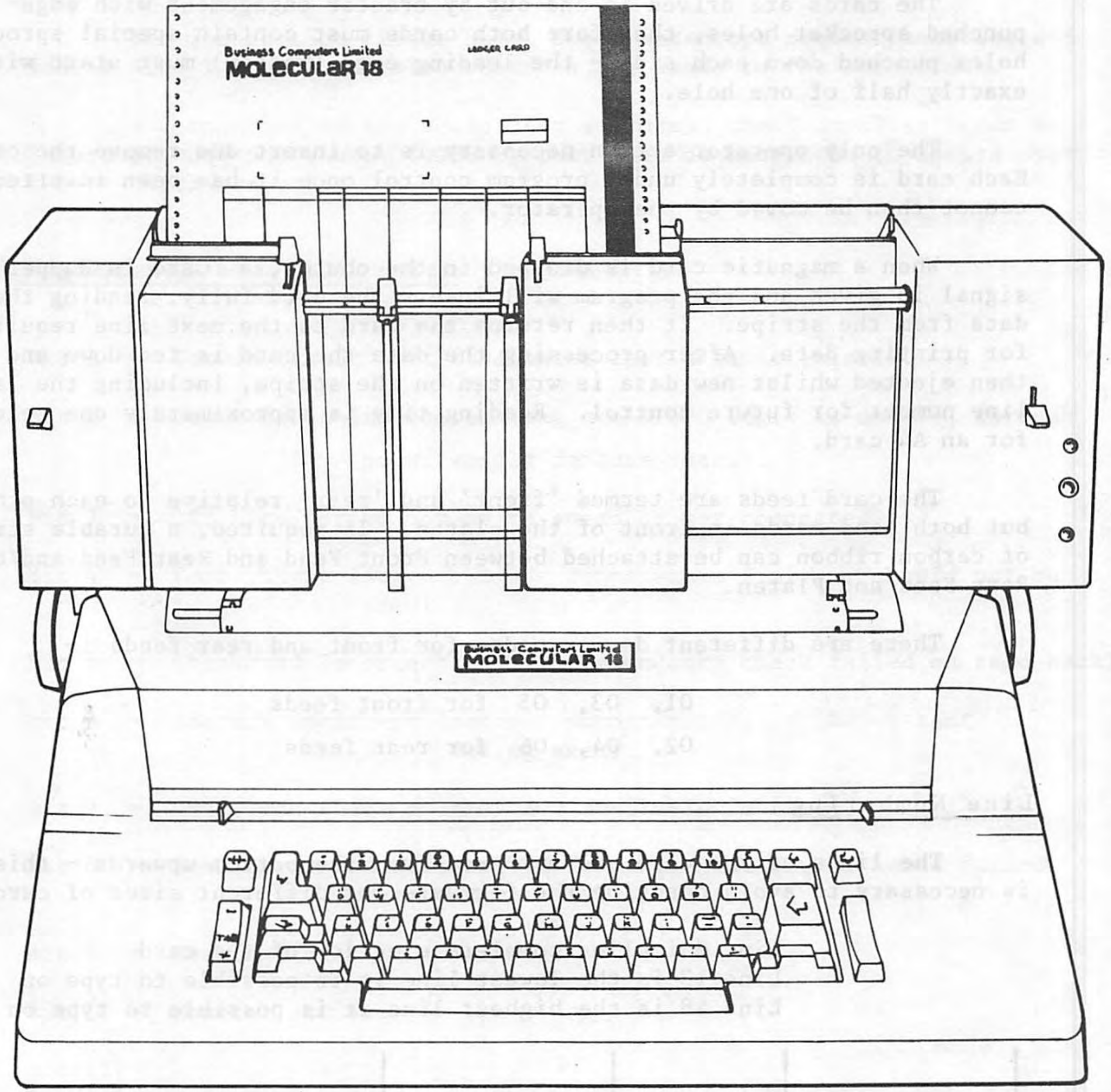


Fig. 4:11

increasing in 30 mm. steps to a maximum width of 390 mm. 31 mm. should be deducted from the width quoted for each magnetic stripe, to obtain the length of the writing line. The stripes must not be written over or printed on. Writing lines are at 6 mm. pitch, therefore the number of lines on an A4 card is 36. The maximum printing speed is 15½ characters per second.

The cards are driven in and out by tractor engagement with edge-punched sprocket holes, therefore both cards must contain special sprocket holes punched down each side - the leading edge (bottom) must start with exactly half of one hole.

The only operator action necessary is to insert and remove the cards. Each card is completely under program control once it has been inserted and cannot then be moved by the operator.

When a magnetic card is dropped in the chutes, a "Card in Hopper" signal is given and the program will insert the card fully, reading the data from the stripe. It then returns the card to the next line required for printing data. After processing the data the card is fed down and then ejected whilst new data is written on the stripe, including the last line number for future control. Reading time is approximately one second for an A4 card.

The card feeds are termed 'front' and 'rear' relative to each other, but both feed cards in front of the platen. If required, a durable strip of carbon ribbon can be attached between Front Feed and Rear Feed and/or Rear Feed and Platen.

There are different device codes for front and rear feeds :-

01,	03,	05	for front feeds
02,	04,	06	for rear feeds

Line Numbering

The lines of the card are numbered from the bottom upwards - this is necessary to avoid changing the firmware for different sizes of card.

Line ∅ implies complete ejection of the card
Line 13 is the lowest line it is possible to type on
Line 48 is the highest line it is possible to type on

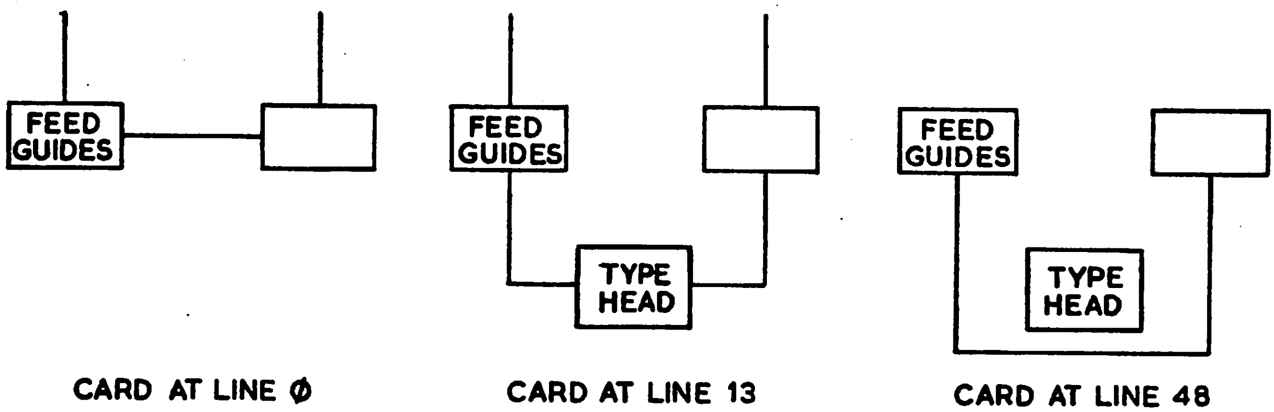


Fig. 4:12

Header and Tail loss :-

Top of card to 1st line	-	72 mm
Bottom of card to last line	-	15 mm

Stripe Formatting

On A4 size (297 mm. high) cards, the stripe format is as follows, from the bottom of the card upwards. (All bytes are 8 bits.)

32 Bytes (alternate "Null" and "All Bits") identifier

1 Byte Preamble (Octal 125)

2 Bytes Hash Total (of Data)

300 Bytes of Data

Only the 300 bytes of data are accessible to the programmer, the other bytes being used for logical and software checks and timing control.

Circuit Description

The BCL - manufactured circuitry is in three parts :-

a) Two small pc boards (2064) which mount inside the unit, replacing the boards number 21314.

b) IO board 2062, which carries control logic for the front (magnetic) card channel, plus circuitry for reading and writing the magnetic stripe.

c) IO board 2063, which carries control logic for two non-magnetic card channels.

FUNCTIONS

The functions available are as follows :-

a)	Insert card to Line "N"	
b)	Eject card to Line "N"	
c)	Line Feed	
d)	Read from Magnetic Stripe	Front Feed only
e)	Write to Magnetic Stripe	
f)	Remove card	
g)	Light Indicator Lamps	

Function (a) - Insert card to Line 'N'

The operation is initiated by DATO 2 + START, with the specified accumulator containing Bit 1 set and also the line number in Bits 9-14 inclusive. Busy is cleared and Done set to signal the completion of the operation.

Function (b) - Eject card to Line 'N'

The operation is initiated by DATO 2 + START, with the specified accumulator containing Bit 2 set and also the line number (which may be zero) in Bits 9-14 inclusive. Busy is cleared and Done set to signal the completion of the operation.

Function (c) - Line Feed

The operation is initiated by DATO 2 + START, with the specified accumulator containing Bit 3 set and Bits 9-14 clear. Busy is cleared and Done set to signal the completion of the operation.

Function (d) - Reading from Mag. Stripe

The Read operation is initiated by the START command, Done being set when the first data word is read into the buffer. DATI 1 then transfers one 8-bit byte from the Read buffer to the specified accumulator, and thus into core.

Function (e) - Writing on Mag. Stripe

Writing is initiated by DATO 1 + START with the first data word in the specified accumulator. START then sets Busy, and when the word has been written Busy is cleared and Done set.

Function (f) - Remove Card

The operation is initiated by DATO 2 + START, with the specified accumulator containing Bit 3 set and Bits 9-14 non-zero. This is a special function designed to give a DONE when the card is removed from its channel or if no card is present. When the card is removed, BUSY is cleared in the usual way.

Function (g) - Indicator lights

The operation of these lights is absolutely independent of all other mag. stripe functions. DATA Bits 1,2, and 3 are used, in conjunction with DATO 3 and the device address. The lights are cleared either when DATO 3 is issued with Bits 1 to 3 of the Accumulator clear, or upon an I/O Reset instruction. No BUSY, DONE or Interrupt is involved.

The indicator lights are controlled only via the Front Feed Device Code (01 etc.).

STATUS BITS (hardware)

Bit 1	Last line
Bit 2	Card jammed
Bit 3	Parity Fail (read only)
Bit 4	Data late (read or write)
Bit 5	Card not in Hopper

Note :-

When inserting/reading a card, it is essential that the IBM Head Carrier is in a position central to the card, in order that the plastic guides do their job.

List of Appendices

THE BASIC INPUT OUTPUT CODING FOR THE MOLECULAR 18

	···	··5	·6·	·65	7··	7·5	76·	765
····	NUL	DLE	SPACE	Ø	@	P	`	p
···1	SOH	DC 1	!	1	A	Q	a	q
··2·	STX	DC2	"	2	B	R	b	r
··21	ETX	DC3	£	3	C	S	c	s
·3··	EOT	DC4	$	4	D	T	d	t
·3·1	ENQ	NAK	%	5	E	U	e	u
·32·	ACK	SYN	&	6	F	V	f	v
·321	BEL	ETB	'	7	G	W	g	w
4···	BS	CAN	(	8	H	X	h	x
4··1	HT	EM	)	9	I	Y	i	y
4·2·	LF	SUB	*	:	J	Z	j	z
4·21	VT	ESC	+	;	K	[	k	{
43··	FF	FS	,	<	L	½	l	|
43·1	CR	GS	–	=	M	]	m	}
432·	SO	RS	.	>	N	∧	n	¬
4321	SI	US	/	?	O	_	o	DEL

TAPE FEED:- FEED HOLE ONLY

PARITY IS EVEN:- ADD BIT 8 WHERE NECESSARY

Appendix 2

CODING CHART FOR IBM (INPUT AND OUTPUT) BCL HEAD

	...	..5	.6.	.65	7..	7.5	76.	765
....	L/C ◢	W $\frac{1}{5}$	B $\frac{1}{12}$		- _		SPACE ◢	* :
...1	' "			11 £		S $\frac{3}{16}$	H $\frac{11}{16}$	
..2.								
..21								
.3..	, +			9 (		I $\frac{9}{16}$	K $\frac{15}{16}$	
.3.1		% ;	E $\frac{3}{10}$		10 $			Z $\frac{1}{8}$
.32.		7 $\frac{3}{4}$	N $\frac{5}{12}$		J $\frac{13}{16}$			0)
.321	Y $\frac{7}{10}$			8 &		4 @	X $\frac{3}{8}$	
4...	BS ◢							
4..1	TAB ◢	O $\frac{1}{6}$	L $\frac{11}{12}$		G $\frac{9}{10}$			6 $\frac{2}{3}$
4.2.	LF ◢							
4.21								
43..		A $\frac{1}{16}$	C $\frac{3}{8}$		P $\frac{5}{8}$			5 $\frac{1}{2}$
43.1	Q $\frac{1}{10}$			3 $\frac{1}{3}$	CR ◢	R $\frac{3}{5}$	D $\frac{5}{16}$	
432.	F $\frac{7}{16}$			2 $\frac{1}{4}$		V $\frac{7}{8}$	U $\frac{4}{5}$	
4321	U/C ◢	M $\frac{7}{12}$	T $\frac{2}{5}$		. /			1 ?

Note :- ◢ only to have parity bit (Bit 8) true added.

Characters on left and right hand side (' ") are Lower Case and Upper Case, respectively.

BCL SERIAL PRINTER CODE SET

	...	..5	.6.	.65	7..	7.5	76.	765
....			Space	Ø	@	P		
...1			!	1	A	Q		
..2.			"	2	B	R		
..21			£	3	C	S		
.3..			$	4	D	T		
.3.1			%	5	E	U		
.32.			&	6	F	V		
.321			'	7	G	W		
4...			(	8	H	X		
4..1			)	9	I	Y		
4.2.	LF1		*	:	J	Z		
4.21	LF2		+	;	K	[		
43..			,	<	L	\		
43.1	C/R		-	=	M	]		
432.			.	>	N	^		
4321			/	?	O	⌴		

LINE PRINTER CODE SET

	···	··5	·6·	·65	7··	7·5	76·	765
····			SPACE	Ø	@	P		
···1			!	1	A	Q		
··2·			"	2	B	R		
··21			#	3	C	S		
·3··			$	4	D	T		
·3·1			%	5	E	U		
·32·			&	6	F	V		
·321			'	7	G	W		
4···			(	8	H	X		
4··1			)	9	I	Y		
4·2·	LF		*	:	J	Z		
4·21			+	;	K	[		
43··	FF		,	<	L	\		
43·1	Print		—	=	M	]		
432·			.	>	N	⎴		
4321			/	?	O	—		

Print cycle occurs automatically if 132 characters are transferred into the buffer store.

LF	Line Feed
FF	Form Feed

TALLY ROLL PRINTER CODE SET

	···	··5	·6·	·65	7··	7·5	76·	765
····			SPACE	Ø	@	P		
···1			!	1	A	Q		
··2·			"	2	B	R		
··21			£	3	C	S		
·3··			$	4	D	T		
·3·1			%	5	E	U		
·32·			&	6	F	V		
·321	Bleep		'	7	G	W		
4···			(	8	H	X		
4··1			)	9	I	Y		
4·2·			*	:	J	Z		
4·21			+	;	K	[		
43··			,	<	L	\		
43·1			–	=	M	]		
432·			.	>	N	^		
4321			/	?	O	_		

CODE SET FOR VISUAL DISPLAY TERMINAL with Keyboard (92423)

Cols.	Ø	1	2	3	4	5	6	7
	···	··5	·6·	·65	7··	7·5	76·	765
····	NUL	DLE	SPACE	Ø	@	P	`	p
···1	SOH	DC1	!	1	A	Q	a	q
··2·	STX	DC2	"	2	B	R	b	r
··21	ETX ⁑	DC3	#	3	C	S	c	s
·3··	EOT	DC4	$	4	D	T	d	t
·3·1	ENQ	Skip *	%	5	E	U	e	u
·32·	ACK	Line Clear *	&	6	F	V	f	v
·321	BEL	ETB	'	7	G	W	g	w
4···	BS *	Clear *	(	8	H	X	h	x
4··1	HT	Reset *	)	9	I	Y	i	y
4·2·	LF *	Cursor Up *	*	:	J	Z	j	z
4·21	VT	ESC	+	;	K	[	k	{
43··	FF	FS	,	<	L	\	l	¦
43·1	CR *	GS	—	=	M	]	m	}
432·	SO ⁑	RS	.	>	N	∧	n	~
4321	SI ⁑	US	/	?	O	_	o	■

Notes

1) All Control Codes in Columns Ø and 1 can be generated by pressing corresponding Keys in Columns 4 and 5 together with the CNTRL Key. In addition certain codes (those under-lined in above chart) may also be produced by specially marked Keys.

2) SO = Inverse Video ON (shows the symbol < on the Display)
SI = Inverse Video OFF (shows the symbol > on the Display)
ETX shows the symbol ▲
Rubout (also Parity) shows the symbol ■
Break Key gives the symbol ■■

3) Codes marked with an * show cursor movement.

4) Codes marked with ⁑ produce special characters (see Note 2 above)

CODE SET FOR ALPHA/NUMERIC KEYBOARD

	···	··5	·6·	··65	7··	7·5	76·	765
····	NUL	DLE	SPACE	Ø	@	P		
···1	SOH	DC1	!	1	A	Q		
··2·	STX	DC2	"	2	B	R		
··21	ETX	DC3	£ or #	3	C	S		
·3··	EOT	DC4	$	4	D	T		
·3·1	ENQ	NAK	°/o	5	E	U		
·32·	ACK	SYN	&	6	F	V		
·321	BEL	ETB (Accept)	'	7	G	W		
4···	BS	CAN (ER)	(	8	H	X		
4··1	TAB	EM	)	9	I	Y		
4·2·	LF	SUB	*	:	J	Z		
4·21	VT	✱	+	;	K	[		{
43··	FF		,	<	L	½		|
43·1	CR	⬄	–	=	M	]		}
432·			.	>	N	Λ		¬
4321	°/oo		/	?	O	⌙		

Parity is even – Bit 8 is added where necessary

CODE LIST FOR ALL IBM CHARACTERS

LOWER CASE	UPPER CASE	BITS 87654321		BITS 87654321
0	)	1110110	L/C	10000000
1	?	1111111	B. SPACE	10001000
2	1/6	0111110	TAB	10001001
3	1/3	0111101	LINE FEED	10001010
4	@	1010111	U/C	10001111
5	1/2	1111100	CR	11001101
6	5/6	1111001	SPACE	11100000
7	2/3	0010110		
8	&	0110111		
9	(	0110100		
10	$	1000101		
11	£	0110001		
A	1/16	0011100		
B	1/12	0100000		
C	3/8	0101100		
D	3/16	1101101		
E	5/16	0100101		
F	7/16	0001110		
G	9/16	1001001		
H	11/16	1100001		
I	5/16	1010100		
J	13/16	1000110		
K	15/16	1100100		
L	11/12	0101001		
M	7/12	0011111		
N	5/12	0100110		
O	1/8	0011001		
P	5/8	1001100		
Q	1/16	0001101		
R	3/8	1011101		
S	3/16	1010001		
T	3/8	0101111		
U	4/5	1101110		
V	7/8	1011110		
W	1/5	0010000		
X	3/8	1100111		
Y	7/16	0000111		
Z	1/8	1110101		
-		1000000		
*	÷	1110000		
'	"	0000001		
,	+	0000100		
%	;	0010101		
.	/	1001111		

Appendix 9

Bit No.	Decimal Value of Bit	Decimal Capacity of all Bits Up to and including Bit Shown
1	1	1
2	2	3
3	4	7
4	8	15
5	16	31
6	32	63
7	64	127
8	128	255
9	256	511
10	512	1,023
11	1,024	2,047
12	2,048	4,095
13	4,096	8,191
14	8,192	16,383
15	16,384	32,767
16	32,768	65,535
17	65,536	131,071
18	131,072	262,143
19	262,144	524,287
20	524,288	1,048,575
21	1,048,576	2,097,151
22	2,097,152	4,194,303
23	4,194,304	8,388,607
24	8,388,608	16,777,215
25	16,777,216	33,554,431
26	33,554,432	67,108,863
27	67,108,864	134,217,727
28	134,217,728	268,435,455
29	268,435,456	536,870,911
30	536,870,912	1,073,741,823
31	1,073,741,824	2,147,483,647
32	2,147,483,648	4,294,967,295
33	4,294,967,296	8,589,934,591
34	8,589,934,592	17,179,869,183
35	17,179,869,184	34,359,738,367
36	34,359,738,368	68,719,476,735
37	68,719,476,736	137,438,953,471
38	137,438,953,472	274,877,906,943
39	274,877,906,944	549,755,813,887
40	549,755,813,888	1,099,511,627,775
41	1,099,511,627,776	2,199,023,255,551
42	2,199,023,255,552	4,398,046,511,103
43	4,398,046,511,104	8,796,093,022,207
44	8,796,093,022,208	17,592,186,044,415
45	17,592,186,044,416	35,184,372,088,831
46	35,184,372,088,832	70,368,744,177,663
47	70,368,744,177,664	140,737,488,355,327
48	140,737,488,355,328	281,474,976,710,655

Appendix 10:1

2-CHARACTER STORAGE

A	0404 101
B	0410 102
C	0414 103
D	0420 104
E	0424 105
F	0430 106
G	0434 107
H	0440 110
I	0444 111
J	0450 112
K	0454 113
L	0460 114
M	0464 115
N	0470 116
O	0474 117
P	0500 120
Q	0504 121
R	0510 122

S	0514 123
T	0520 124
U	0524 125
V	0530 126
W	0534 127
X	0540 130
Y	0544 131
Z	0550 132
Ø	0300 060
1	0304 061
2	0310 062
3	0314 063
4	0320 064
5	0324 065
6	0330 066
7	0334 067
8	0340 070
9	0344 071

½	0560 134
,	0260 054
.	0270 056
SPACE	0200 040
TAB	0044 011
C/R	0064 015
VT	0054 013
L/F	0050 012
BS	0040 010
'	0234 047
(	0240 050
)	0244 051
:	0350 072
;	0354 073
?	0374 077
"	0210 042
!	0204 041
/	0274 057

£	0214 043
$	0220 044
@	0400 100
[	0554 133
]	0564 135
^	0570 136
_	0574 137
%	0224 045
&	0230 046
*	0250 052
+	0254 053
-	0264 055
<	0360 074
>	0370 076
=	0364 075
BEL	0034 007
CAN	0140 030
ETB	0134 027

2-CHARACTER STORAGE (continued)

NUL	0000 000
DEL	0774 177
EOT	0020 004
FF	0060 014
ESC	0154 033
GS	0164 035
SO	0070 016
SI	0074 017
a	0604 141
b	0610 142
c	0614 143
d	0620 144
e	0624 145
f	0630 146

g	0634 147
h	0640 150
i	0644 151
j	0650 152
k	0654 153
l	0660 154
m	0664 155
n	0670 156
o	0674 157
p	0700 160
q	0704 161
r	0710 162
s	0714 163
t	0720 164

u	0724 165
v	0730 166
w	0734 167
x	0740 170
y	0744 171
z	0750 172
`	0600 140
{	0754 173
}	0764 175
|	0760 174
¬	0770 176
SOH	0004 001
STX	0010 002
ETX	0014 003

ENQ	0024 005
ACK	0030 006
DLE	0100 020
DC1	0104 021
DC2	0110 022
DC3	0114 023
DC4	0120 024
NAK	0124 025
SYN	0130 026
EM	0144 031
SUB	0150 032
FS	0160 034
RS	0170 036
US	0174 037

These tables will be found very useful when storing messages, literals, etc. in core, at two characters to a word. It should be used as follows :-

For a character in the top (most significant) half of the store, use the upper four-figure number, and for a character in the bottom (least significant) half of the store add the lower three-figure number in the appropriate position, so that a six-digit octal number is produced.

e.g. Should you wish to store the word'STOP' in 2 words of core, it should be stored as follows :-

```
S  -  Ø514      Ø474      -  O
T  -     124       120       P
```

Ø51524	Ø47520

PERIPHERAL CHARACTER SETS

I.B.M. Code Bit Structure	I.B.M.	Line Printer	Dot Printer	C.R.T. V.D.U.	Alpha Numeric Keyboard	Numeric Keyboard	Tally Roll Printer	ASCII Code Bit Structure
876.....	Space	Space	Space	Space	Space		Space	8.6.....
		!	%	!	!		!	..6....1
.......1 ■	"	"	"	"	"		"	..6...2.
..65...1 ■	£	#	#	#	#		£	8.6...21
.7...3.1 ■	$	$	$	$	$		$	..6..3..
...5.3.1	%	%	%	%	%		%	8.6..3.1
..65.321 ■	&	&	&	&	&		&	8.6..3.
.......1	'	'	'	'	'		'	..6..321
..65.3.. ■	(	(	(	(	(		(	..6.4...
.765.32. ■	)	)	)	)	)		)	8.6.4..1
.765...	*	*	*	*	*		*	8.6.4.2.
.....3.. ■	+	+	+	+	+		+	..6.4.21
.....3..	,	,	,	,	,		,	8.6.43..
.7......	–	–	–	–	–	–	–	..6.43.1
.7..4321	.	.	.	.	.	DP	.	..6.432.
.7..4321 ■	/	/	/	/	/		/	8.6.4321
.765.32.	Ø	Ø	Ø	Ø	Ø	Ø	Ø	..65....
.7654321	1	1	1	1	1	1	1	8.65...1
..65432.	2	2	2	2	2	2	2	8.65..2
..6543.1	3	3	3	3	3	3	3	..65..21
.7.5.321	4	4	4	4	4	4	4	8.65.3..
.76543..	5	5	5	5	5	5	5	..65.3.1
.7654..1	6	6	6	6	6	6	6	..65.32.
...5.32.	7	7	7	7	7	7	7	8.65.321
..65.321	8	8	8	8	8	8	8	8.654...
..65.3..	9	9	9	9	9	9	9	..654..1

Note :- ■ = Upper Case

Cont.

I.B.M. Code Bit Structure	I.B.M.	Line Printer	Dot Printer	C.R.T. V.D.U.	Alpha Numeric Keyboard	Numeric Keyboard	Tally Roll Printer	ASCII Code Bit Structure
.765.... ■	:	:	:	:	:		:	..654.2.
...5.3.1 ■	;	;	;	;	;		;	8.654.21
		<	<	<	<		<	..6543..
		=	=	=	=		=	8.6543.1
		>	>	>	>		>	8.65432.
.7654321 ■	?	?	?	?	?		?	..654321
.7.5.321 ■	@	@	@	@	@		@	87......
...543..	A	A	A	A	A		A	.7.....1
..6.....	B	B	B	B	B		B	.7....2.
..6.43..	C	C	C	C	C		C	87....21
.76.43.1	D	D	D	D	D		D	.7...3..
..6..3.1	E	E	E	E	E		E	87...3.1
....432.	F	F	F	F	F		F	87...32.
.7..4..1	G	G	G	G	G		G	.7...321
.76....1	H	H	H	H	H		H	.7..4...
.7.5.3..	I	I	I	I	I		I	87..4..1
.7...32.	J	J	J	J	J		J	87..4.2.
.76..3..	K	K	K	K	K		K	.7..4.21
..6.4..1	L	L	L	L	L		L	87..43..
...54321	M	M	M	M	M		M	.7..43.1
..6..32.	N	N	N	N	N		N	.7..432.
...54..1	O	O	O	O	O		O	87..4321
.7..43..	P	P	P	P	P		P	.7.5....
....43.1	Q	Q	Q	Q	Q		Q	87.5...1
.7.543.1	R	R	R	R	R		R	87.5..2.
.7.5...1	S	S	S	S	S		S	.7.5..21
..6.4321	T	T	T	T	T		T	87.5.3..
.76.432.	U	U	U	U	U		U	.7.5.3.1
.7.5432.	V	V	V	V	V		V	.7.5.32.
...5....	W	W	W	W	W		W	87.5.321

I.B.M. Code Bit Structure	I.B.M.	Line Printer	Dot Printer	C.R.T. V.D.U.	Alpha Numeric Keyboard	Numeric Keyboard	Tally Roll Printer	ASCII Code Bit Structure
.76..321	X	X	X	X	X		X	87.54...
.....321	Y	Y	Y	Y	Y		Y	.7.54..1
.765.3.1	Z	Z	Z	Z	Z		Z	.7.54.2.
		[	[	[	[		[	87.54.21
.76543..■	½	\	\	\	\		\	.7.543..
		]	]	]	]		]	87.543.1
		⊓	↑	^	^		^	87.5432.
.7......■	–	–	–	–	⌟		–	.7.54321
8...4...	Back Space			Back Space	Back Space			8...4...
8...4..1	TAB			TAB	TAB			....4..1
8...4.2.	LF	LF	LF A	LF	LF			....4.2.
			LF B	VT	VT			8...4.21
87..43.1	CR		CR	CR	CR			8...43.1
				BEL	BEL	BEL	Bleep	8....321
					Error	Error		...54...
					◈	◈		...543.1
					✱	✱		...54.21
					Accept	Accept		...5.321

Note :- ■ = Upper Case

DECIMAL/OCTAL CONVERSION TABLES

The Tables on the next two pages allow easy Decimal/Octal and Octal/Decimal conversion to be made.

The Decimal to Octal Table (on next page) converts up to a maximum of 139,999 decimal (421,337 octal), and the Octal to Decimal Table (Appendix 12:3), converts up to a maximum of 377,777 octal (131,071 decimal).

The tables are used as follows :-

Decimal to Octal (Appendix 12:2)

Split the decimal number into three pairs of digits, and use each pair as the matrix reference to the three sections of the chart, adding together the three figures so referenced.

For instance :- Decimal 129,748

Split into 3 pairs :- 12 97 48

Use most significant pair on top section of the chart	- gives	352300
Use middle pair on middle section of chart	- gives	22744
Use least significant pair on bottom section of chart	- gives	60
	Total	375324

So decimal 129,748 = octal 375324

Note that the addition of the three referenced numbers must be done in octal (6+4=12).

Octal to Decimal (Appendix 12:3)

Split the octal number into three pairs of digits, and use each pair as the matrix reference to the three sections of the chart, adding together the three figures so referenced.

For instance :- Octal 375324

Split into 3 pairs :- 37 53 24

Use most significant pair on top section of chart	- gives	126,976
Use middle pair on middle section of chart	- gives	2,752
Use least significant pair on bottom section of chart	- gives	20
	Total	129,748

So octal 375342 = decimal 129,748

Note that the addition of the three referenced numbers must be done in decimal.

DECIMAL to OCTAL CONVERSION TABLE

0	0	23420	47040	72460	116100	141520	165140	210560	234200	257620
1	303240	326660	352300	375720						
	0	1	2	3	4	5	6	7	8	9

0	0	144	310	454	620	764	1130	1274	1440	1604
1	1750	2114	2260	2424	2570	2734	3100	3244	3410	3554
2	3720	4064	4230	4374	4540	4704	5050	5214	5360	5524
3	5670	6034	6200	6344	6510	6654	7020	7164	7330	7474
4	7640	10004	10150	10314	10460	10624	10770	11134	11300	11444
5	11610	11754	12120	12264	12430	12574	12740	13104	13250	13414
6	13560	13724	14070	14234	14400	14544	14710	15054	15220	15364
7	15530	15674	16040	16204	16350	16514	16660	17024	17170	17334
8	17500	17644	20010	20154	20320	20464	20630	20774	21140	21304
9	21450	21614	21760	22124	22270	22434	22600	22744	23110	23254
	0	1	2	3	4	5	6	7	8	9

0	0	1	2	3	4	5	6	7	10	11
1	12	13	14	15	16	17	20	21	22	23
2	24	25	26	27	30	31	32	33	34	35
3	36	37	40	41	42	43	44	45	46	47
4	50	51	52	53	54	55	56	57	60	61
5	62	63	64	65	66	67	70	71	72	73
6	74	75	76	77	100	101	102	103	104	105
7	106	107	110	111	112	113	114	115	116	117
8	120	121	122	123	124	125	126	127	130	131
9	132	133	134	135	136	137	140	141	142	143
	0	1	2	3	4	5	6	7	8	9

OCTAL to DECIMAL CONVERSION TABLE

0	0	4096	8192	12288	16384	20480	24576	28672
1	32768	36864	40960	45056	49152	53248	57344	61440
2	65536	69632	73728	77824	81920	86016	90112	94208
3	98304	102400	106496	110592	114688	118784	122880	126976
	0	1	2	3	4	5	6	7

0	0	64	128	192	256	320	384	448
1	512	576	640	704	768	832	896	960
2	1024	1088	1152	1216	1280	1344	1408	1472
3	1536	1600	1664	1728	1792	1856	1920	1984
4	2048	2112	2176	2240	2304	2368	2432	2496
5	2560	2624	2688	2752	2816	2880	2944	3008
6	3072	3136	3200	3264	3328	3392	3456	3520
7	3584	3648	3712	3776	3840	3904	3968	4032
	0	1	2	3	4	5	6	7

0	0	1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
2	16	17	18	19	20	21	22	23
3	24	25	26	27	28	29	30	31
4	32	33	34	35	36	37	38	39
5	40	41	42	43	44	45	46	47
6	48	49	50	51	52	53	54	55
7	56	57	58	59	60	61	62	63
	0	1	2	3	4	5	6	7

COMPLETE LIST OF PROGRAM INSTRUCTIONS

Memory Ref. Instructions

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
OPERATION CODE					I/D	Z/C	MEMORY ADDRESS									

0	0	0	1	0	02	JUMP	Jump
0	0	0	1	1	(03)	JSBR	Jump to Subroutine
0	0	1	0	0	(04)	INSZ	Increment, Skip if Zero
0	0	1	0	1	(05)	DESZ	Decrement, Skip if Zero
0	0	1	1	0	(06)	ANDA	'And' to A
0	0	1	1	1	(07)	IORA	'Inclusive OR' to A
0	1	0	0	0	(10)	XORA	'Exclusive OR' to A
0	1	0	0	1	(11)	ADA	Add to A
0	1	0	1	0	(12)	ADB	Add to B
0	1	0	1	1	(13)	SFA	Subtract from A
0	1	1	0	0	(14)	SFB	Subtract from B
0	1	1	0	1	(15)	ADAC	Add to A, with Carry
0	1	1	1	0	(16)	ADBC	Add to B, with Carry
0	1	1	1	1	(17)	SFAC	Subtract from A with Carry
1	0	0	0	0	(20)	SFBC	Subtract from B with Carry
1	0	0	0	1	(21)	LDA	Load A
1	0	0	1	0	(22)	LDB	Load B
1	0	0	1	1	(23)	CMPA	Compare A, Skip if not equal
1	0	1	0	0	(24)	CMPB	Compare B, Skip if not equal
1	0	1	0	1	(25)	STA	Store A
1	0	1	1	0	(26)	STB	Store B
* 1	0	1	1	1	(27)	UNSTK	Unstack

* Input/Output Instructions

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	1	A/B	FUNCT.		MODE			DEVICE CODE					

No Operation.................... 0	0	0	0	0	No I/O Transfer, just Function	
Set Busy, Clear Done (START) 0	1	0	0	1	DATI 1	INPUT
Clear Busy, Clear Done (STOP) 1	0	0	1	0	DATI 2	
Input/Output Pulse.............. 1	1	0	1	1	DATI 3	
		1	0	0	DATO 1	OUTPUT
		1	0	1	DATO 2	
		1	1	0	DATO 3	

Skip if Busy........................ 0	0	1	1	1	Skip Mode
Skip if not Busy................... 0	1				
Skip if Done......................... 1	0				
Skip if not Done................. 1	1				

All instructions marked with an * are 'Privileged' instructions

Literal Instructions

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	0	0	0	A/B	1	L/R	Sh/Rot	0	Ar/Log	No. of Shifts			

Register Instructions

Operation Code					Mode											
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	0	0	1	A/B	CLEAR CARRY	L/R	SHIFT	ROTATE	WITH CARRY	DEC	INC	SKIP BIT 16 = Ø	SKIP BIT 1 = Ø
0	0	0	0	0	1	0	A/B	CLEAR	COMP	CLEAR CARRY	COMP CARRY	SKIP	SWAP	CLEAR SIGN	COMP SIGN	ENTER SW. REG
0	0	0	0	0	1	1	A/B	Tr./Fal	SKIP IF NEG	SKIP IF NOT ZERO	SKIP IF CARRY	CLEAR CARRY	SKIP IF GT	CLEAR GT	CLEAR	COMP.
0	0	0	0	0	0	0	0	1	L/R	Sh/Rot	1	Ar/Log	0	0	0	0

Control Instructions (Octal Code)

000000	No Op
* 000001	Halt
* 000002	Mask Out
* 000003	Ack. Int.
* 000004	Int. On
* 000005	Int. Off
* 000006	SK. Int. On
* 000007	SK. Int. Off
* 000010	SK. Mains Fail
* 000011	SK. Mains Ret.
* 000012	SK. Mem. Parity
* 000013	Reserved
* 000014	SK. Limit
* 000015	SK. MA=SW
* 000016	SK. Cont. Int.
* 000017	I O Reset
* 000020	SK. Overstack
* 000021	SK.>15 Indirects
* 000022	SK. Timer Int.
* 000023	SK. Illegal Op
* 000024	SK. Extra Code
001000	Extracode
001001	Set Greater Than

Interrupt Addresses

000205	Cont. Int.
000206	MA=SW Int.
000207	Mains Return
000210	Extra Code
000211	I/O Int.
000212	Illegal Op
000213	> 15 Indirects
000214	Limit
000215	Mem. Parity
000216	Mains Fail
000217	7th Level Int.

Interrupt Stack

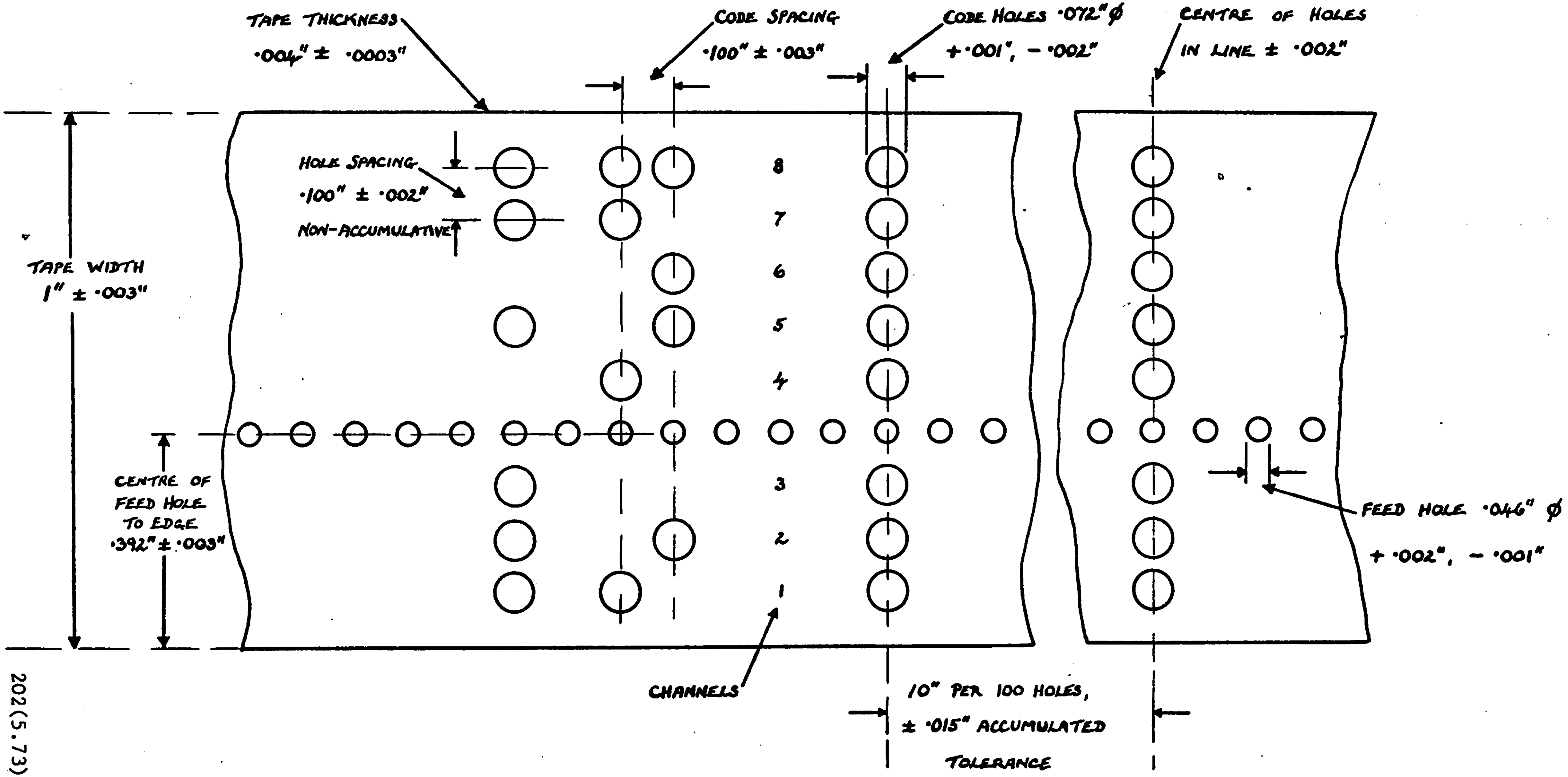
Stack Pointer Address = 000030

1st. Word	Bit 1=Carry, Bit 2=G.Than
2nd Word	PC & Mode Bit 17=1 Absolute Add. Mode
3rd. Word	Base
4th Word	B Register
5th Word	A Register

Mains Return Auto Start Address	000002
Auto Inc. Mem. Addresses	000010 to 000017
Auto Dec. Mem. Addresses	000020 to 000027

Limit Register loaded with 1st. word of program +Base

STANDARD TAPE DIMENSIONS



OCTAL ADDRESS BOUNDARIES OF PAGES Ø to 77 INCLUSIVE

Page Ø	-	000000	to	001777
Page 1	-	002000	to	003777
Page 2	-	004000	to	005777
Page 3	-	006000	to	007777
Page 4	-	010000	to	011777
Page 5	-	012000	to	013777
Page 6	-	014000	to	015777
Page 7	-	016000	to	017777
Page 10	-	020000	to	021777
Page 11	-	022000	to	023777
Page 12	-	024000	to	025777
Page 13	-	026000	to	027777
Page 14	-	030000	to	031777
Page 15	-	032000	to	033777
Page 16	-	034000	to	035777
Page 17	-	036000	to	037777
Page 20	-	040000	to	041777
Page 21	-	042000	to	043777
Page 22	-	044000	to	045777
Page 23	-	046000	to	047777
Page 24	-	050000	to	051777
Page 25	-	052000	to	053777
Page 26	-	054000	to	055777
Page 27	-	056000	to	057777
Page 30	-	060000	to	061777
Page 31	-	062000	to	063777
Page 32	-	064000	to	065777
Page 33	-	066000	to	067777
Page 34	-	070000	to	071777
Page 35	-	072000	to	073777
Page 36	-	074000	to	075777
Page 37	-	076000	to	077777
Page 40	-	100000	to	101777
Page 41	-	102000	to	103777
Page 42	-	104000	to	105777
Page 43	-	106000	to	107777
Page 44	-	110000	to	111777
Page 45	-	112000	to	113777
Page 46	-	114000	to	115777
Page 47	-	116000	to	117777
Page 50	-	120000	to	121777
Page 51	-	122000	to	123777
Page 52	-	124000	to	125777
Page 53	-	126000	to	127777
Page 54	-	130000	to	131777
Page 55	-	132000	to	133777
Page 56	-	134000	to	135777
Page 57	-	136000	to	137777
Page 60	-	140000	to	141777
Page 61	-	142000	to	143777
Page 62	-	144000	to	145777
Page 63	-	146000	to	147777
Page 64	-	150000	to	151777
Page 65	-	152000	to	153777
Page 66	-	154000	to	155777
Page 67	-	156000	to	157777
Page 70	-	160000	to	161777
Page 71	-	162000	to	163777
Page 72	-	164000	to	165777
Page 73	-	166000	to	167777
Page 74	-	170000	to	171777
Page 75	-	172000	to	173777
Page 76	-	174000	to	175777
Page 77	-	176000	to	177777

ALPHABETIC INDEX

(Cont.)

(Cont.)

(Cont.)

(Cont.)