M 10 PORTABLE COMPUTER

Operations Guide

olivetti

M 10 PORTABLE COMPUTER

Operations Guide

olivetti

PUBLICATION ISSUED BY:

Ing. C. Olivetti & C., S.p.A.
Direzione Documentazione
77, Via Jervis - 10015 IVREA (Italy)

PREFACE

The Olivetti M10 is a small, portable, personal computer of such versatility that it is ideally adapted for use in the office, the home or in the field. It comes in two versions - the M10 MODEM, equipped with an integrated modem and a model without this feature. Where differences occur between versions, these are clearly specified in the text.

Among the many outstanding features of the M10 are ease of operation, an eight-line liquid crystal display, practical keyboard, compatibility with many different peripherals, a range of word-processing functions, a powerful programming capability in BASIC and direct or remote connection to other computers.

All of these, and the other aspects of the M10 are described in this manual which, for the user's convenience, is written in two quite distinct parts.

Part 1, the "M10 User's Guide", deals with the essentials of operation, giving an overall view of the M10 and peripherals, descriptions of its various features and applications and details of how to make use of the five built-in application programs with which the M10 comes equipped. In other words, it introduces the user to the M10 and provides the necessary information for all aspects of its operation.

Part 2, on the other hand, deals exclusively with BASIC, (Beginners All-purpose Symbolic Instruction Code), the high-level programming language of the M10. This part is entitled "BASIC Language Reference Guide" and it is exactly that; it does not purport to teach you how to program in BASIC but is rather a recapitulation of BASIC structure, syntax and commands to which the user can refer when in doubt about the use of this language vis à vis the M10. In particular, it comprises an alphabetic directory of all BASIC Commands, Statements and Functions, summarising their application and format and giving examples of their use.

PRE-REQUISITE PUBLICATIONS: none

RELATED PUBLICATIONS: none

DISTRIBUTION: General (G)

FIRST EDITION: August, 1983

CONTENTS

PART I
M10 User's Guide

1. INTRODUCTION

NOTATION

The following convention has been used throughout this manual to avoid confusion:

1. Words typed in upper case letters represent commands or instructions to the computer and should be entered in the format indicated although not necessarily in upper case since the M10 converts all letters to upper case except in TEXT e.g:

   ```
   PRINT TIME$   may be entered as
   print time$   or
   PrinT tIMe$   but not
   PRINT TIME $  where the spacing has been changed.
   ```

   Likewise, symbols have a precise function and must be entered as shown.

   A list of all symbols used in BASIC programming language appears at the beginning of Part 2 of this manual.

2. Words which appear in lower case print are variables and the user must supply the actual value e.g.

   ```
   PRINT filename
   ```

   In this example, PRINT represents an instruction to be entered as it is, while the user must type the actual name of the file to be printed.

3. The following list shows the meaning of certain groups of words and symbols.

   `A|B|C` choose one of A, B or C

   `[ ]` items enclosed in square brackets are optional

   `...` three leader dots indicate that more values can be included in the string e.g.

   ```
   DATA constant-1 ,constant-2 ...,constant-n
   ```

   `<KEY>` the key whose name appears in angle brackets should be pushed e.g.

   <ENTER> means "press ENTER"

<CTRL> + x means "press x while CONTROL is being held down"
<GRPH + SHIFT> means "press GRPH and SHIFT together"

4. Text appearing exactly as it on the screen is shown in boldface e.g.

File to edit?

GENERAL DESCRIPTION

The M10 is a self-contained, easy-to-use ,personal computer which is fully portable. It is depicted in Figure 1-1 below. It comes in two models, one with integrated modem (M10 MODEM) and one without. In this manual, wherever a reference is made to the M10, it can be assumed that this applies equally to both models. Where differences occur, these are specifically mentioned.

Fig. 1-1 The M10 Portable Computer

The M10 is ideally adapted to many applications. It offers facilities for programming, word processing, telecommunications, maintaining an electronic address book and schedule of activities.

The M10 can be operated either from its own battery supply (four 1.5V dry cells) or from the AC mains supply, using an AC adaptor. It weighs only

1900 grammes (4 lb 3 oz.), with overall dimensions of 30 x 22 x 6 cm. (11 3/4 x 8 1/2 x 2 3/8 in.) and fits easily into a normal briefcase, making it ideal for use during travel or in the field, as well as in the office or at home. The M10 comes in its own attractive soft case, protecting it from damage and dust.

A prominent feature of the M10 is its keyboard and screen. One of the most frequently expressed criticisms of portable computers is that the keyboard is not easy to operate or that the display is difficult to read. This is not a criticism we expect to hear levelled at the M10 with its solid, full-size keyboard and liquid crystal display of 8 rows of 40 characters.

TECHNICAL DESCRIPTION

The M10 is equipped with 32k of ROM (read-only memory) which accommodates the five built-in facilities. These are:

- BASIC interpreter which allows you to create, store and run programs.
- TEXT for composing text and word processing.
- TELCOM for telecommunications.
- ADDRSS to set up and maintain an electronic address and telephone book.
- SCHEDL to keep an easily accessed schedule of your appointments and other activities.

The other main features are:

- 8k of random access memory (RAM), expandable in 8k steps up to 32k.
- OKI 80C85 CMOS microprocessor, compatible with the Intel 8085.
- Keyboard giving 188 alphanumeric and graphic characters; the M10 MODEM has a standard USA-ASCII keyboard while Italian, French, German and UK standard keyboards are available for the model without integrated modem.
- Additional function keys for special applications.
- Tiltable liquid crystal display (LCD), with adjustable screen contrast.
- Interfaces for peripherals.

The M10 has three directories in RAM. Filenames are suffixed according to the directory in which they are located.

Files suffixed .BA are BASIC language programs stored in binary form.

Files suffixed .CO are machine language programs.

Files suffixed .DO are those created under the TEXT application program or BASIC programs stored in ASCII format or basic data files (this manual deals only with .DO files created under TEXT).

INTERFACES AND PERIPHERALS

The M10 has interfaces for connecting a variety of peripherals. Note that the M10 MODEM can be connected directly to a telephone for telecommunications applications whereas the other model requires an external modem/acoustic coupler for this. Figure 1-2 shows a typical configuration of the M10 with peripherals attached. Special cables are available for all interfaces.

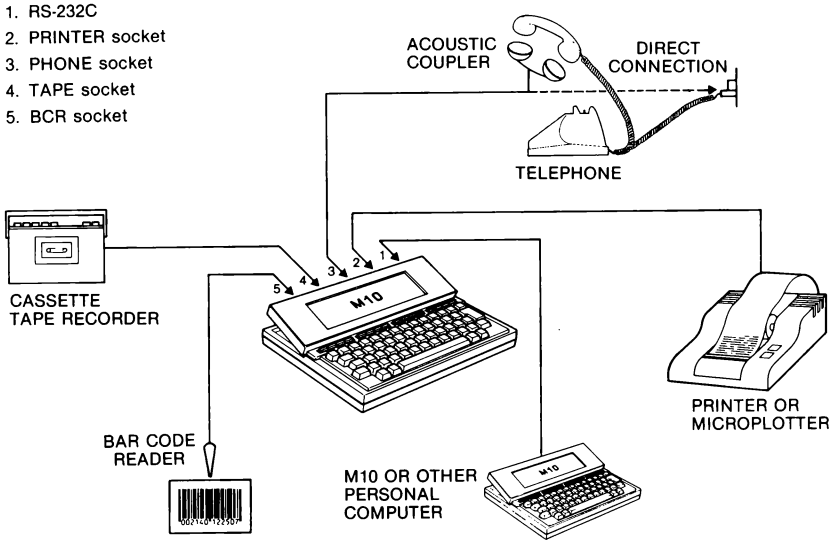

Fig. 1-2 Typical System Configuration

The principal peripherals are:

Cassette Tape Recorder

The M10 can be connected to any model of cassette tape recorder provided it has, in addition to the standard controls, an input jack (MIC), an output jack (EAR), a remote control jack (REM) and a tape counter. See Chapter 9 for further details.

Printer

Any printer or microplotter conforming to the industry standard for parallel interfaces can be connected to the M10 by means of the Olivetti microplotter cable.

Telephone

The M10 MODEM is equipped with an internal modem which allows it to be connected either directly or via an acoustic coupler to a telephone line. The other model requires an external modem/acoustic coupler.

Bar Code Reader

The M10 has an interface for connecting a Hewlett-Packard HEDS-3050 or HEDS-3000 bar code reader for stock control applications.

The M10 can be connected through the RS-232C interface to another computer either directly in local connection or through a modem for remote connection to a host computer system.

THE KEYBOARD AND SCREEN

The M10 MODEM has a USA-ASCII standard keyboard (see Figure 1-3). The model without integrated modem is equipped with one of four national standard keyboards - Italian, French, German or UK (see Figure 1-4). All of these give the standard alphanumeric character set.

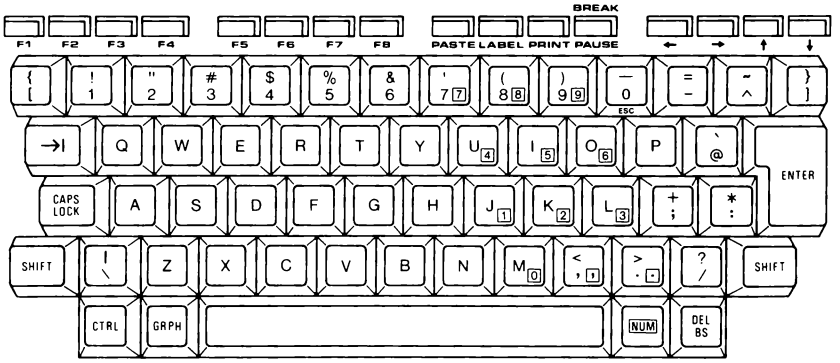

Fig. 1-3 USA-ASCII Keyboard

In addition, by pressing the key <GRPH>, another 47 characters are available, as shown in Figure 1-5. Pressing <GRPH + SHIFT> gives a further 47 characters, as shown in Figure 1-6. The character sets given by <GRPH>

and <GRPH + SHIFT> are identical for all models.

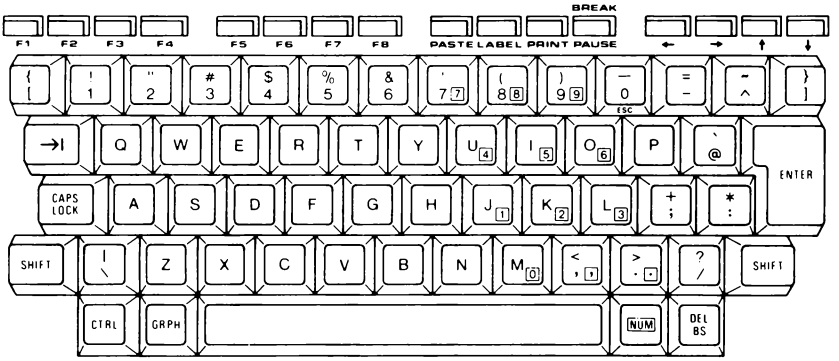

Fig. 1-4 UK Keyboard

Note that all character keys have a repetitive function when held down, as does DEL/BS, the tabulation key (→I) and the cursor movement keys.

The SHIFT key has the same role as on a normal typewriter - it changes letters to upper case and gives the upper alternative on number and symbol keys.

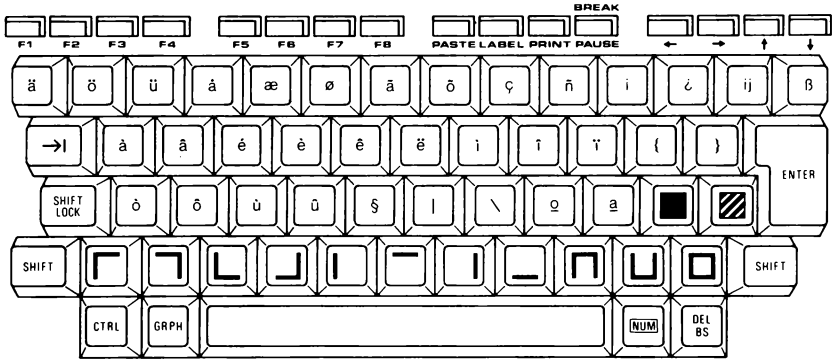

Fig. 1-5 Keyboard When <GRPH> is Pushed

There is a functional difference between CAPS LOCK on the US keyboard and SHIFT LOCK on the other models. CAPS LOCK acts only on the letter keys, leaving the number and symbol keys unchanged. SHIFT LOCK gives the upper case alternative on all keys. Both CAPS LOCK and SHIFT LOCK remain in position until released by pressing again.

The tabulation key is labelled →| and moves the cursor from one tab to the next. These tabs are set every eight character spaces and can not be altered.

CTRL is a key for communicating special control characters to the computer. It is also used in conjunction with the keyboard for certain functions. These are dealt with as they occur.

GRPH gives access to another character set, as already described.

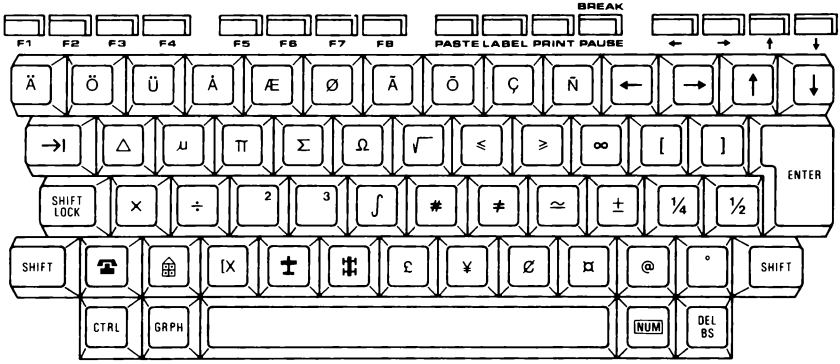

Fig. 1-6 Keyboard When <GRPH + SHIFT> is Pushed

At the right side of the keyboard are 12 keys which, in addition to their normal characters, carry the digits 0 - 9, a comma and a full stop, inscribed in a small square as shown in Figure 1-7. This number pad is accessed by pressing NUM. All other keys are disabled when this is pressed and remain so until NUM is pressed again. If your keyboard apparently blocks, producing no letters but only numbers, you will find that the NUM key has been pressed.

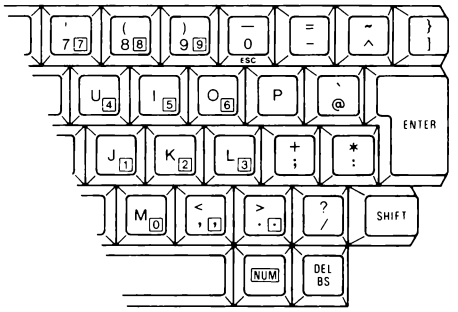

Fig. 1-7 Number Pad on the M10 Keyboard

DEL/BS is the delete/backspace key. Pressing this key deletes the character preceding the cursor and moves the cursor back one space. In TEXT mode, <DEL/BS + SHIFT> deletes the character over which the cursor is positioned and moves subsequent text back one space.

The ENTER key is used to select a program or file from the main menu once the cursor has been positioned, or to select an option inside one of the five built-in application programs. It also serves to indicate the end of a data record in a file and as the carriage return of the display.

Located just below the display are four groups of four keys, labelled in blue.

The first two groups, labelled F1-F8 are function keys which have different functions, according to the operation being carried out. These functions are described in subsequent chapters.

The third group comprises the command keys PASTE, LABEL, PRINT and PAUSE/BREAK respectively. The function of these keys does not vary from one program to another.

- PASTE is used in 'Cut and Paste' and 'Copy' operations. This is a text-editing function and has its most obvious application in TEXT mode. A detailed description of how to use this facility is given in Chapter 5.

- LABEL is used to display at the foot of the screen the function of F1-F8 for whichever of the five application programs is currently in use. Pressing LABEL a second time removes these definitions from the screen.

- PRINT is used when a printer is connected to the M10. Pressing this key causes the current LCD display to be printed. <SHIFT + PRINT>

prints the entire file and allows the user to specify the number of characters per line to be printed.

- PAUSE/BREAK has two functions. In BASIC, the lower case alternative, PAUSE, causes the execution of a program to be suspended. Operation is resumed when this key is pressed again. The upper case alternative, <BREAK> i.e. <SHIFT> + <PAUSE/BREAK>, allows the user to halt a program or to restart the normal functioning if the system has "frozen", for example, when a PRINT command is given with no printer connected to the M10. In TEXT mode, pushing <BREAK> cancels a 'Select', 'Find', 'Load', 'Save' or PRINT operation.

The final four keys, labelled with arrows, are the cursor movement keys. These are used to position the cursor on the screen in TEXT mode and their functions are summarised in Figure 1-8.

KEY	CURSOR MOVEMENT
→	One space to the right
<SHIFT> + →	To beginning of next word
<CTRL> + →	To end of current line
←	One space to the left
<SHIFT> + ←	To beginning of last word (or current word)
<CTRL> + ←	To beginning of current line
↑	One line up
<SHIFT> + ↑	To top line of screen
<CTRL> + ↑	To beginning of file
↓	One line down
<SHIFT> + ↓	To bottom line of screen
<CTRL> + ↓	To end of file

Fig. 1-8 Cursor Movement Keys

In addition to the functions noted in the table, the cursor movement keys have a repetitive action when held down. Holding down the Right Arrow key causes the cursor to scan from left to right and from line to line down the file. Holding down the Left Arrow key has the inverse effect. By holding the Down Arrow key depressed, the cursor can be made to scroll line by line down the file while an upward scroll can be effected in the same way using the Up Arrow key.

The screen, on which the characters appear as they are typed, is a liquid crystal display (LCD). It is made up of 8 lines each 40 character spaces wide and each character is formed in a 6 x 8 dot matrix. The right hand column is automatically left blank in TEXT mode to preserve the spacing between words on successive lines.

The screen can be tilted manually from the horizontal position to an angle of about 30 degrees for optimum viewing. Screen contrast can be adjusted by means of the circular control on the right hand side panel.

Up to 8 lines of text can be displayed on the screen at any one time. When viewing in TEXT mode, the screen can be scrolled up or down by using the cursor movement keys. There is automatic word-wrap in this mode so that if a word over-runs the end of a line, it is automatically carried over to the next line to avoid splitting.

Although the M10 screen shows only 40 characters on a line, it is possible, when using a printer, to have up to 132 characters per line on the printed output.

REAR PANEL CONNECTIONS

On the rear panel of the M10 are a number of connectors, as shown in Figure 1-9.

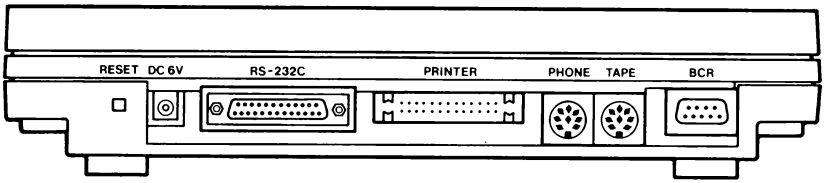

Fig. 1-9 Rear Panel Connections

From left to right these are:

RESET

This recessed button is, as its name implies, for resetting the computer when it has "frozen" and all functions are blocked. This is unlikely to happen when using one of the built-in application programs but it could occur when operating a program of your own devising.

DC 6V

This jack takes the plug on the end of the AC adaptor when the M10 is connected to the mains supply.

RS-232C

This is an interface conforming to the EIA standard with the exception of the data carrier detect (DCD) line which is not connected. Typical applications of the RS-232C are:

- to connect the M10 to an external modem or modem/acoustic coupler.

- to connect the M10 to equipment that will act as a data terminal such as the Olivetti ET351 electronic typewriter or the Olivetti PR430 serial printer.

- to connect the M10 to another M10 or other personal computer.

PRINTER

This is the parallel interface output from the M10 to a printer or micro-plotter.

PHONE

This connection exists only on the M10 MODEM. It provides the link between the M10 and a telephone line, either for data communication with a host computer or for automatic dialling of a number (see Chapter 8).

TAPE

When a cassette tape recorder is being used to store files from the M10 (as is recommended) the recorder is connected to this socket. More is said of this in Chapter 9.

BCR

This socket provides the connection point between the M10 and a bar code reader. It is specifically designed to connect to the Hewlett-Packard HEDS-3050 or HEDS-3000 bar code reader. This has an application in warehousing and general stock control.

UNDERSIDE PANEL

Underneath, the M10 appears as shown in Figure 1-10.

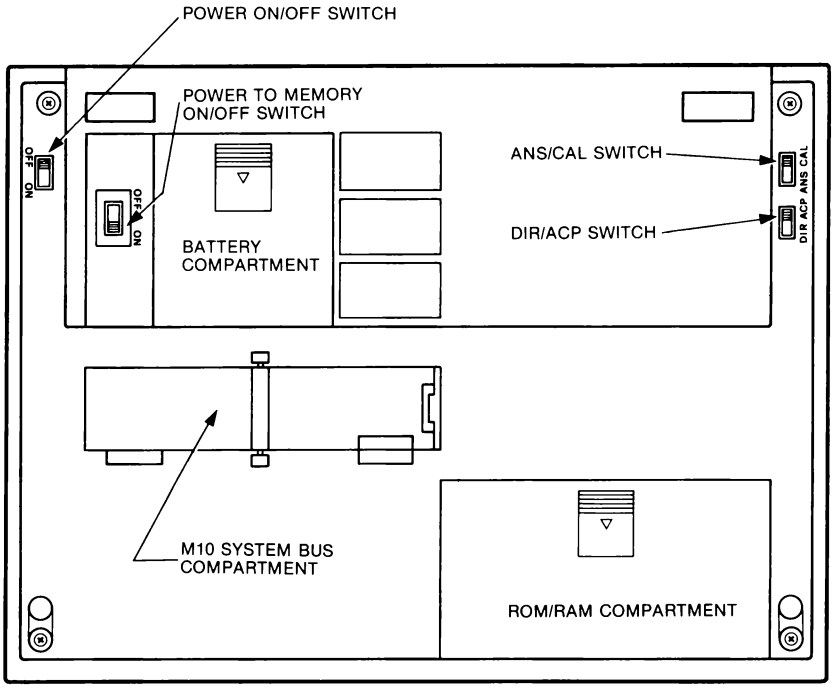

Fig. 1-10 Underside Panel

The features shown in the figure are:

1. Power ON/OFF Switch.

This switches the computer on and off.

2. Power to Memory ON/OFF Switch

This switch keeps power supplied to the memory even when the computer is switched off. When the M10 is delivered, this switch will be turned to OFF. During the start-up procedure, it will be turned to ON and will remain so unless the computer is not to be used for an extended period.

It is important to realise that when this switch is turned off, files in storage in the RAM will be lost. All necessary files should be saved on tape before turning the memory power off. The computer will not operate if this switch is off. It is located in a recessed compartment to ensure that it is not turned off by accident.

Depending on the amount of RAM installed, the Ni-Cd rechargeable battery supplying the power to the memory will keep the RAM contents intact from between 8 to 30 days when the M10 is switched off. It is good practice to make sure the computer is used regularly every few days to keep the battery charged.

3. Battery Compartment.

The M10 is powered by four 1.5V AA alkaline dry cells, installed in this compartment. They should be inserted as shown in Figure 2-1. These cells have an operational life of approximately 20 hours regardless of the memory capacity installed. When the battery is running low, the LED on the front panel marked 'Battery Low' lights up. You then have about 20 minutes of operation left before the battery runs out. Switch off and replace the battery.

4. ANS/CAL Switch (M10 MODEM only).

This switch has an application in data communications and is used in the TELCOM program. It is dealt with in Chapter 8.

5. DIR/ACP Switch (M10 MODEM only).

The same remarks apply as to the ANS/CAL switch.

6. ROM/RAM Compartment.

This compartment contains space for an extension to both the ROM and the RAM. The M10 has 32k bytes of ROM as a standard and this can be increased to 64k with the addition of a single extra block. The standard RAM capacity is 8k bytes which can be extended to 32k in blocks of 8k.

7. M10 System Bus Compartment.

This compartment is closed by a hinged cover and contains a 40-pin connection to the M10 system bus.

2. SWITCHING ON THE M10

The M10 operates either on a 6V DC supply derived from a battery or from the mains supply, using an AC adaptor. The latter is an optional accessory which can be obtained from your Olivetti dealer. Several models are available, all with a 6V DC output but operating from 240V, 220V, or 120V AC supply (for U.K., Europe and USA respectively).

INSTALLING THE BATTERY

Being portable, the M10 is designed to operate from a battery of four 1.5V DC alkaline dry cells, size AA. The battery compartment is located on the underside of the chassis as shown in Figure 1-10. Figure 2-1 shows the insertion and polarity of the cells.

The cells have an average useful life of about 20 hours, regardless of the amount of memory installed, after which the 'Battery Low' lamp on the front panel will light. When this happens, there remain only about 20 minutes of operating time left before the battery dies, so the cells should be changed immediately.

In addition to the 6V battery, there is an internal Ni-Cd cell which supplies power to the memory even when the M10 is switched off. This cell recharges when the computer is operational and, when fully charged, can maintain power to the memory for about eight to thirty days, according to the amount of memory installed. It has a useful life of around two years, after which time you should have it changed by your local Olivetti dealer.

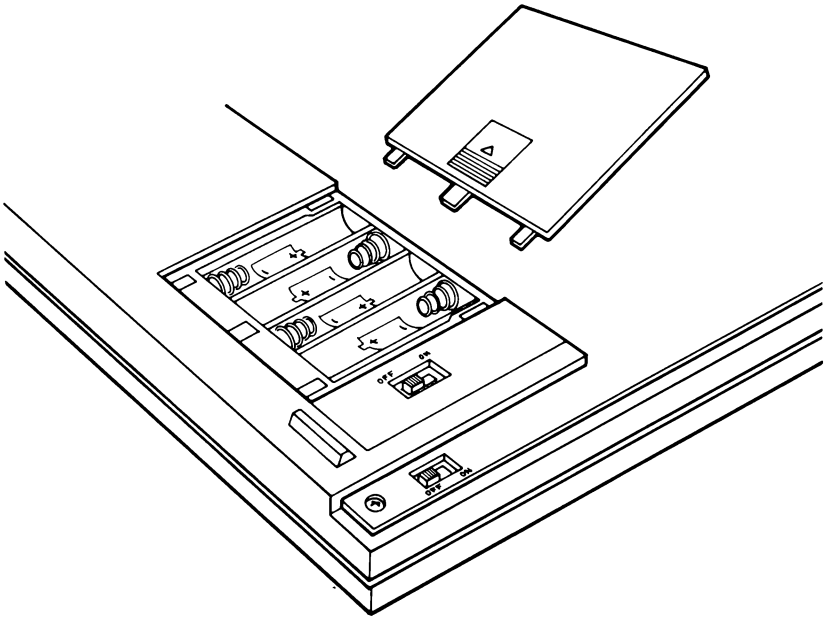

Fig. 2-1 Installing the Battery

SWITCHING ON FOR THE FIRST TIME

1. Remove the cover from the battery compartment on the underside of the M10 chassis.

2. Insert four 1.5V, size AA dry cells as shown in Figure 2-1.
   Alternatively, connect the end of the cable on the AC adaptor to the jack marked DC6V on the rear panel of the M10; plug the adaptor into the mains supply.
   It is important to remember that the M10 and all peripherals must be turned off before connecting or disconnecting the AC adaptor to the M10.

3. Turn the memory ON/OFF switch to ON (on the underside, see Fig.1-10).

4. Turn the power ON/OFF switch to ON (on the underside, see Fig. 1-10).

It is important not to forget this last step as the computer will not otherwise function.

To switch off the M10, first turn off any peripherals that may be connected (e.g. a printer) then turn the power switch of the M10 to OFF. Do not turn off the power to the memory as this will cause all the contents of the RAM to be lost.

Once the initial switch-on procedure has been carried out, the M10 is switched on and off using only the power ON/OFF switch on the underside panel.

Note that the M10 will switch itself off after about 10 minutes if no keys are pushed. This is to preserve the battery. To return to normal operation, switch off then on again. This automatic switch-off feature can be disabled by means of the BASIC command POWER CONT or, alternatively, the time after which the M10 switches off can be modified by the BASIC command POWER. Details of these commands are given in the BASIC Language Reference Guide in Part 2 of this manual.

3. THE MAIN MENU

When the M10 is first switched on the screen will appear as in Fig. 3-1. This is the main menu of the computer.

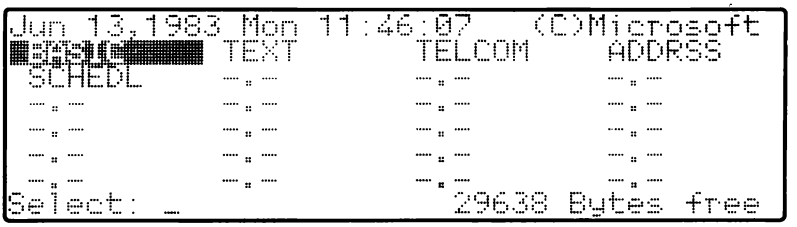

Fig. 3-1 The Main Menu

It can be seen that the LCD carries 8 lines of text. The dark rectangle positioned over BASIC is the cursor, which can be moved using the cursor movement keys, described in Chapter 1. The word on which the cursor is located appears "in negative". The screen contrast can be altered using the control on the right hand side panel.

The first line of the display is taken by the date and time. On first starting up this will read:

Jan 01, 1900 Sun 00:00:00 (C) Microsoft

The next 6 lines are reserved for a list of the programs on the computer. To begin with, there are only the five built-in application programs:

- BASIC
- TEXT
- TELCOM
- ADDRSS
- SCHEDL

There is room for a further 19 files to appear on the menu.

The bottom line shows:

Select : _ 29638 Bytes free

The option **Select** allows you to type in and access the program or file of your choice. This can also be done by positioning the cursor over the file name on the menu and pressing <ENTER>.

The entry **29638 Bytes free** refers to the available memory at your disposal. The actual value of this parameter on start-up depends on the amount of RAM installed. The different values are shown below:

Memory Installed	Bytes free
32k	29638
24k	21446
16k	13254
8k	5062

These values will, of course, diminish when you start to use the computer memory.

On start-up, the cursor is set over the BASIC program. It can be moved to any other position by means of the cursor movement keys, indicated by arrows. To begin with, however, only BASIC, TEXT and TELCOM are available. The procedures for setting up ADDRSS and SCHEDL are dealt with in Chapters 6 and 7 respectively.

SETTING THE TIME

As already noted, a read-out of the date, day and time appears on the first line of the main menu. Upon first switching on the computer this indicates:

Jan 01, 1900 Sun 00:00:00

and it will start to record time from this datum forward.

The procedure for setting this to current values is described below.

TO RESET THE TIME:

Position the cursor over BASIC and press <ENTER>. The cursor will appear as a flashing dark rectangle below the BASIC prompt, **Ok** .

Type on the screen :

TIME$="hour:minute:second"

and press <ENTER> where

hour is a two-digit number from 00 to 23

minute is a two-digit number from 00 to 59

second is a two-digit number from 00 to 59

in a form identical to the time on a digital watch.

The computer clock will register this time from the moment that <ENTER> is pushed.

If you do not enter the data correctly an error message will be displayed on the screen.

To check the value you have entered, type:

PRINT TIME$

and press <ENTER>.

The time you entered will be displayed on the screen.

TO RESET THE DATE:

Type the date on the screen in the following form:

DATE$="month/day/year" if you have the M10 MODEM

DATE$="day/month/year" if you have a European model

and press <ENTER> where

month is a two-digit number from 01 to 12

day is a two-digit number from 01 to 31

year is a two-digit number from 00 to 99

For example, the instruction

DATE$="06/14/83" or DATE$="14/06/83", according to the model, enters the date as June 14, 1983.

The command PRINT DATE$ will cause the date you entered to be displayed on the screen.

TO RESET THE DAY:

Type the day on the screen as follows:

DAY$="day"

and press <ENTER> where 'day' is one of the following three-letter abbreviations:

Monday	- Mon
Tuesday	- Tue
Wednesday	- Wed
Thursday	- Thu
Friday	- Fri
Saturday	- Sat
Sunday	- Sun

Again the value entered can be checked by typing PRINT DAY$ and pressing <ENTER>.

To return to the main menu, press function key <F8>. The day, date and time which you entered now appears on the first line of the menu.

4. THE BASIC FACILITY

The BASIC facility is one of five built-in application programs with which the M10 comes equipped. Most of the information required to exploit this facility to the full is contained in Part 2 of this manual - the BASIC Language Reference Guide. For all questions relating to the programming, syntax and structure of BASIC, the user should refer to this Reference Guide which also contains a comprehensive list of all BASIC commands with notes on their function and application. However, since BASIC is the very heart of the M10, the present chapter outlines some of the main points for operational completeness.

BASIC stands for "Beginners All-purpose Symbolic Instruction Code" and it is the high-level programming language of the M10. Using this facility it is possible to write, store and run your own programs, written in BASIC.

THE SCREEN AND OPERATING MODES IN BASIC

In order to select BASIC from the main menu, position the cursor over the name BASIC and press <ENTER>. The screen appears as shown in Figure 4-1. The cursor appears as a flashing rectangle immediately below the BASIC prompt, **Ok** .

Alternatively, an existing BASIC program may be loaded and executed by placing the menu cursor over the BASIC file name and pressing <ENTER>.

Fig. 4-1 Screen on Entering BASIC

There are three operating modes in BASIC:

- Direct
- Execute
- Text or Edit

There is no need for the user to select the first two operating modes; this is done automatically by the computer, according to the operation being carried out.

DIRECT MODE

This is the mode in force when the BASIC prompt **Ok** appears on the screen. Direct Mode is used to enter programs or immediate lines (see the explanation of the concept of lines below). When <ENTER> is pressed to run the program, the M10 passes to Execute Mode, reverting to Direct when the operation is complete.

EXECUTE MODE

This mode is operative when the M10 is actually running a program or an immediate line is being executed. When the operation is complete, the computer returns to Direct mode and the **Ok** prompt returns to the screen.

TEXT or EDIT MODE

The command EDIT <ENTER> invokes this mode when the BASIC prompt **Ok** is on the screen. It is used in order to edit a program. When Text Mode is operative, the cursor movement keys function as described in Chapter 5. To quit Text mode, press <F8>.

The concept of lines is the following:

- A logical line is a line of commands or instructions, terminated by a carriage return (i.e. by pressing <ENTER>) and may be up to 255 characters long.
- A physical line is a text line on the screen and can therefore not exceed 40 characters in length.
- An immediate line is a command or series of commands which are executed by the computer immediately on pressing <ENTER> e.g.

```
PRINT TIME$
```

FUNCTION KEYS IN BASIC

The tables shown below summarise the use of the function keys, F1-F8 and PASTE, LABEL and PAUSE/BREAK when using the BASIC program.

KEY NAME	FUNCTION
F1 File	Lists on the screen all the files on the menu
F2 Load	Used to load a file from an external device (e.g. a cassette tape recorder) or from RAM
F3 Save	Saves the current program to an external device such as a cassette tape recorder or to RAM
F4 Run	Runs the current program
F5 List	Lists the current program on the screen
F6	Not used
F7	Not used
F8 Menu	Returns to the main menu
PASTE	Inserts the contents of the PASTE buffer at the location of the cursor
LABEL	Displays the functions of F1-F8 on the screen
PAUSE	Halts the program momentarily; program can be resumed by pressing PAUSE again
BREAK (SHIFT+PAUSE)	Stops execution of the program

Fig. 4-2 Function Keys in Direct Mode

It is also possible to program the function keys to have a particular function as defined by the user. Details of the procedure to follow are given in the second part of this manual which deals exclusively with BASIC.

KEY NAME	FUNCTION
F1 Find	Allows you to specify a string to be located in the file under edit
F2 Load	Loads a program from a cassette tape recorder
F3 Save	Saves a program file to a cassette tape recorder
F4	Not used
F5 Copy	Stores a selected string in the PASTE buffer to be copied at the location specified by the cursor
F6 Cut	Stores a selected string in the PASTE buffer and removes the original from the file
F7 Sel	Allows you to select a string for cut, copy and paste operations
F8 Exit	Exits from the Text Mode
PASTE	Inserts the contents of the PASTE buffer at the location of the cursor
LABEL	Displays the functions of F1-F8 on the screen

Fig. 4-3 Function Keys in Text Mode

5. THE TEXT APPLICATION PROGRAM

The second of the five built-in programs listed on the menu is TEXT. This is a text-editing facility, specifically designed for creating, editing and storing files of text. It offers a good range of word-processing facilities - search function, insertion and deletion of text, word-wrap, cut and paste operations etc. Probably the first occasion the user will have to use the TEXT program will be to create the files ADRS.DO and NOTE.DO needed for the ADDRSS and SCHEDL applications (as described in Chapters 6 and 7).

To access TEXT, position the cursor over the name TEXT in the main menu and press <ENTER>. Alternatively, you may give the command:

TEXT<ENTER>

via the keyboard. This will appear at the foot of the screen immediately to the right of **Select:** . On pressing <ENTER>, the screen appears as in Figure 5-1.

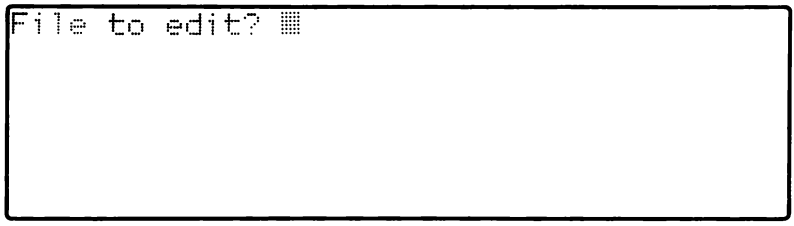

Fig. 5-1 The Screen for the TEXT Program

FILE NAMES IN TEXT

In response to the prompt:

File to edit?

the user should enter the name of the file to be edited or created and press <ENTER>. There are certain facts to be borne in mind when choosing a name for an M10 file. The name may be up to 6 characters in length, with the following restrictions:

- A file name must not begin with a number.

- The following symbols are not accepted as the first character in a file name:

  ! " # $ % & ' *

  ( ) + * : , . /

- In addition, the full stop (.) and the colon (:) are forbidden anywhere in a file name. The only exception to this is when the full stop occurs right at the end of the name, in which case it is incorporated into the suffix (see the example below).

- The M10 converts all letters to upper case, so you can not use capitals as a means of distinguishing between otherwise identical file names.

Having typed in the selected file name, press <ENTER>. If the file name is rejected for violation of any of the quoted restrictions, the M10 emits a warning "beep" and the prompt **File to edit?** is repeated on the next line. If accepted, the screen is cleared and the cursor appears in the top left hand corner enclosing a left-pointing arrow to mark the start of the text.

Files created under TEXT are all suffixed .DO by the computer. There is no need for the user to do this. If, for example, you enter the file name RED in response to the prompt, the file will be listed RED.DO in the main menu. The following example gives some instances of legal and illegal file names.

FILE NAME ENTERED	NAME ALLOCATED BY TEXT
FILE	FILE.DO
file	FILE.DO
fILe	FILE.DO
file.	FILE.DO
F I L E	F I L E.DO
#FILE	Not Accepted
FILE#	FILE#.DO
FI.LE	Not Accepted
"FILE"	Not Accepted
FILE.DO	FILE.DO

Up to 19 files can be stored in RAM at any one time and they will appear on the menu in addition to the five applications programs. If you need more files than this, some will have to be stored on an external device such as a cassette tape recorder (see Chapter 9).

To delete a file from RAM, access the BASIC program from the main menu and give the following command:

KILL "filename" <ENTER>

It is important to remember the double quotation marks round the file name. It is equally important to type the full file name, including the suffix.

TYPING INTO A TEXT FILE

Once you have attributed a valid file name, you can begin to type in text via the keyboard. It appears on the screen as it is typed. As mentioned, the screen will take up to 40 characters per line. There is no need to press <ENTER> for a carriage return at the end of each line; the cursor proceeds automatically from line to line, word-wrapping where necessary. To correct typing errors, place the cursor one character space to the right of the letter to be deleted and press the <DEL/BS> key. If <SHIFT> + <DEL/BS> is pressed, the character over which the cursor is positioned is deleted. To insert a character, or a block of text, simply position the cursor in the desired location and type in the insertion. The subsequent text will move one space to the right for each character inserted. The M10 operates permanently in 'Insert' mode. To add to the text already in a file, move the cursor to the end of the current text and type in the addition.

The position of the cursor is controlled by the cursor movement keys, a group of four keys labelled with arrows on the top right hand side of the keyboard, just below the screen. They have already been described in Chapter 1 but for the sake of convenience the information is repeated here. Their use is detailed in Figure 5-2.

Like the majority of keys, the cursor movement keys are repetitive when held down. By holding down the Right Arrow key, the cursor is made to scan from left to right, moving from the end of one line to the beginning of the next. When it reaches the bottom of the screen it will continue to scan, causing the next line to appear. The Left Arrow key executes a reverse scan, right to left and bottom to top, in exactly the same way. When held down, the Up Arrow moves up the screen, causing the preceding text to appear until it reaches the beginning of the file. The Down Arrow key has an identical action in the oppposite directon until the end of the file is reached. These keys can be said to have a scrolling function.

KEY	CURSOR MOVEMENT
→	One space to the right
<SHIFT> + →	To beginning of next word
<CTRL> + →	To end of current line
←	One space to the left
<SHIFT> + ←	To beginning of last word (or current word)
<CTRL> + ←	To beginning of current line
↑	One line up
<SHIFT> + ↑	To top line of screen
<CTRL> + ↑	To beginning of file
↓	One line down
<SHIFT> + ↓	To bottom line of screen
<CTRL> + ↓	To end of file

Fig. 5-2 Cursor Movement Keys

FUNCTION AND COMMAND KEYS IN TEXT

As shown in Figure 5-3 below,the keys F1-F8 control special functions in TEXT mode.

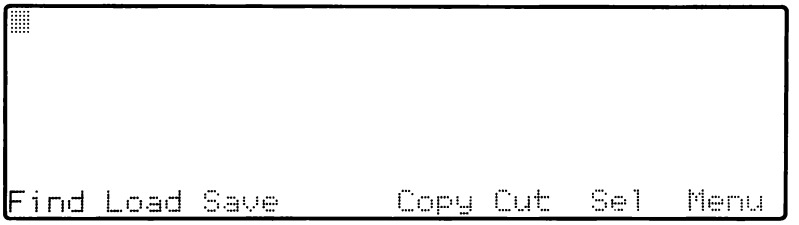

Fig. 5-3 Screen with the Function Keys Labelled

To display these functions on the screen, press <LABEL>. These functions, which offer the user a wide range of word-processing options, are dealt with in the next few sections. A summary of the function and

command keys in TEXT is given in Figure 5-6.

FINDING A CHARACTER STRING IN TEXT

The 'Find' function is controlled by the function key F1. In order to select this option, simply press this key. The screen will then display the message

String:

at the bottom of the screen with the cursor immediately to the right. Type in the character string you are looking for and press <ENTER>. Typically, this would be a word or phrase. The cursor will move to indicate the start of the string where it first occurs in the text and the string prompt disappears from the screen. On pressing <F1> again the prompt and string reappear. Pressing <ENTER> causes the computer to search for further occurrences of the same character string. If the string entered is not found in the text, the computer returns the message

No match

There are certain important points to be remembered when using this option.

1. The character string must be no longer than 24 characters, including spaces. If you try to exceed this number, the computer gives a warning "beep" and the cursor will not move any further to the right.

2. Spacing is of vital importance. If the spacing is wrong, the string will not be found.

3. The search is initiated from the current cursor location. If you wish to search the whole text for a particular string, make sure the cursor is at the beginning of the file before selecting <F1>. There is no way of searching backwards through the file.

4. In this application, as in others, the M10 makes no distinction between upper and lower case letters and therefore ignores such differences when matching the target string to the text.

When changing from one string to another, press <F1> and begin typing the new characters to be searched for. This eliminates the preceding string.

THE 'LOAD' FUNCTION IN TEXT

This function, invoked by pressing <F2>, provides a means of loading information from a cassette tape recorder into a TEXT file. To do this carry out the following procedure:

1. Ensure that the M10 is connected to a suitable cassette recorder (see Chapter 9 for details of this).

2. Access the TEXT file into which the data are to be loaded (if necessary create a new file for this).

3. Press <F2>. This brings up the prompt

Load from:

4. In response to this prompt, type the file name and press <ENTER>. While searching for the specified file the computer emits a high-pitched sound. When it has been located, the following prompt appears on the screen:

FOUND: filename

where 'filename' represents the name entered in the previous step. If there is more than one file on the cassette, each time the M10 encounters another during its search, it will display the message

SKIP: filename

If you specify that the information from the cassette tape is to be loaded into a file already containing text, the contents of the cassette file will be appended to the TEXT file you have accessed.

More detailed information on the use of a cassette tape recorder in conjunction with the M10 appears in Chapter 9.

THE 'SAVE' FUNCTION IN TEXT

When you have created or edited a TEXT file and wish to store the contents in a file on a cassette tape, you must use the 'Save' function (F3). To do this proceed as follows:

1. Ensure that a suitable cassette recorder is connected to the M10 (see Chapter 9).

2. Access the file you wish to save and press <F3>. This brings up the prompt

Save to:

at the bottom of the screen.

3. Select a file name not exceeding six characters for the cassette file in which the contents of the TEXT file are to be saved.

4. Type in the chosen file name and press <ENTER>.

When the prompt disappears from the screen, the information has been saved under the chosen file name.

For more detailed information on how to save files to tape, see Chapter 9.

THE 'SELECT' FUNCTION

The 'Select' function (F7) is used to define a block of text which is to be copied ('Copy' - F5), deleted ('Cut' - F6) or moved (Cut and Paste). Such a block can vary in length from a single character to the entirety of the text in the file.

You are, however, limited by the amount of memory you have at your disposal. If the chosen block occupies more memory than indicated on the main menu under **Bytes free** you will receive the message

Memory full

Likewise, if you receive this message when you press PASTE, it means that you have exceeded the RAM capacity. Care should be taken when selecting long blocks of text.

To define a text block, position the cursor at the beginning of the text to be manipulated and press <F7>. The cursor movement keys can then be used to define the block, either character by character with the right arrow key or line by line, using the down arrow key. Characters included in the selected block appear "in negative" on a black background, as shown in Figure 5-4 below.

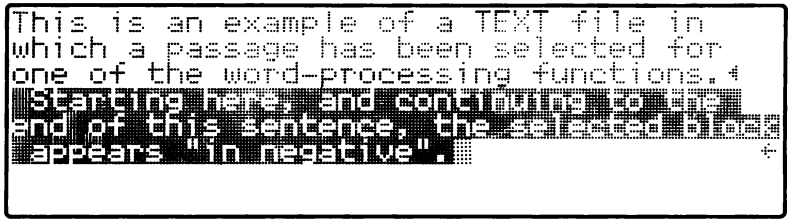

Fig. 5-4 A Selected Block of Text

In fact, the text can be defined from the cursor position onwards, or from the same position backwards through the file. This is summarised in Figure 5-5.

TEXT BLOCK TO BE DEFINED	KEY SEQUENCE
Next character to the right	<F7> then <Right Arrow>
Next character to the left	<F7> then <Left Arrow>
Word following cursor	<F7> then <SHIFT> + <Right Arrow>
Word preceding cursor	<F7> then <SHIFT> + <Left Arrow>
From the cursor to end of current line	<F7> then <CTRL> + <Right Arrow>
From the cursor to beginning of current line	<F7> then <CTRL> + <Left Arrow>
From cursor to bottom of screen	<F7> then <SHIFT> + <Down Arrow>
From cursor to top of screen	<F7> then <SHIFT> + <Up Arrow>
All text from cursor to end of file	<F7> then <CTRL> + <Down Arrow>
All text from cursor to beginning of file	<F7> then <CTRL> + <Up Arrow>

Fig. 5-5 Defining a Text Block

Another method of selecting a block of text involves using the 'Find' function. This allows you to select text from the current cursor position up to a particular word or phrase. To do this, proceed as follows:

1. Press <F7> when you have moved the cursor to the starting point.

2. Now press <F1> and enter the string or word at which the selected block is to end.

3. Press <ENTER>. All text from the original cursor position up to but not including the first character in the string is now selected and appears "in negative" on the screen.

If more text is selected than required, moving the cursor back to the chosen end point removes the extra text from the block. To cancel a 'Select' operation, press <BREAK> i.e. <SHIFT> + <PAUSE> and the selected block will disappear from the screen.

USE OF THE 'COPY' FUNCTION

The function key F5 offers the possibility of duplicating a block of text (e.g. a recurring word or phrase) in another part of the file. The two functions 'Copy' and 'Cut' involve using not only the respective function keys (F5 and F6) but also the 'Select' function and the PASTE command. To carry out a 'Copy' operation, proceed as follows:

1. Press <F7> and define the block of text to be copied, as explained in the preceding section. The selected block appears "in negative" on a black background.

2. Press <F5>. The dark background disappears, showing that the selected block has been stored in the PASTE buffer.

3. Move the cursor to the position where the copy is to be inserted and press <PASTE>.

Note that the insertion will be made in such a way that the last character of the copy will occupy the space preceding the cursor position, unless the last character in the selected block is a carriage return (when <ENTER> is pressed) in which case it will occupy the preceding line. If the same text is to be copied in more than one location in the file, only step 3 need be repeated since the selected text remains in the PASTE buffer until superseded by another 'Select' operation.

THE 'CUT AND PASTE' FUNCTION

This function is used to delete a block of text or to move it from one position to another in the file. The function key F6 is used in conjunction with 'Select' (F7) for simple deletion, and with 'Select' and <PASTE> in a 'Cut and Paste' operation. The procedure is as follows:

1. Press <F7> and define the text to be deleted as explained previously. The selected block appears against a dark background.

2. Press <F6>. The selected text is deleted from the file and stored in the PASTE buffer. If the operation in question is a simple deletion, it is now complete. The deleted text can be recovered from the PASTE buffer until superseded by another 'Select' operation.

3. To place the text elsewhere in the file, move the cursor to the new position and press <PASTE>. The text deleted from one position will now be inserted in the new location.

NOTE: Both the 'Copy' and 'Cut and Paste' operations can be used to transfer data from one file to another. The procedure up to the last step is identical to that described. Instead of positioning the cursor for insertion in the current file, however, access the file to which it is to be transferred, position the cursor in the appropriate location in that file and press <PASTE>.

PRINTING A TEXT FILE

A TEXT file can be printed using the special command key PRINT. In order to use this command, of course, the M10 must be connected to a printer by means of the Olivetti printer cable. The connection is made to the PRINTER parallel interface on the rear panel.

- Pressing <PRINT> prints what is currently displayed on the screen.

- Pressing <SHIFT + PRINT> prints out the entire TEXT file. This command brings up the prompt

  Width?

  and displays the current width setting. If you wish to change the line width of the printed file, type in a value from 10 to 132 and press <ENTER>. This fixes the number of character spaces in a line of printed text.

  If you press <ENTER> without entering a value for the line width, the last value entered will be effective. The default line width is 80 character spaces.

'MENU' FUNCTION

The function key F8 is labelled 'Menu' and has two simultaneous functions. Pressing this key closes the file and returns the user to the main menu. The file will then be stored in RAM under its file name and the menu displayed on the screen.

SUMMARY OF FUNCTION AND COMMAND KEYS

Figure 5-6 below summarises the use of the various function and command keys when operating in TEXT mode.

KEY	FUNCTION
F1 Find	Locates a given character string in the text.
F2 Load	Loads a file from a cassette tape into the M10.
F3 Save	Saves a file to an external cassette tape recorder.
F4	Not used in TEXT.
F5 Copy	Used in conjunction with PASTE to copy a block of text to another location in the file or to another file.
F6 Cut	Deletes a block of text or, used with PASTE to effect a 'Cut and Paste' operation.
F7 Select	Defines a block of text for subsequent manipulation.
F8 Menu	Closes the file and returns the user to the main menu.
PASTE	Inserts the contents of the PASTE buffer into the file at the position given by the cursor. Used in 'Copy' and 'Cut and Paste' operations.
LABEL	Displays labelled functions on the screen.
PRINT	Prints the contents of the screen or the file on an external printer.
PAUSE/BREAK	Only BREAK is used in TEXT - to cancel a 'Select', 'Find', 'Load' or 'Save' operation.

Fig. 5-6 Function and Command Keys in TEXT

EQUIVALENCE OF CONTROL CHARACTERS

All the function and cursor movement keys have an equivalent control key sequence. Of course, it is not necessary to know these sequences but it could be useful and a table of equivalents is given in Figure 5-7.

CONTROL KEY SEQUENCE <CTRL> +	EQUIVALENT TO	FUNCTION
A	<SHIFT> + <Left Arrow>	Cursor to start of preceding word
B	<SHIFT> + <Down Arrow>	Cursor to bottom of screen
C	<BREAK>	Cancels 'Select', PRINT, 'Save', 'Load' and 'Find'
D	<Right Arrow>	Cursor one space to right
E	<Up Arrow>	Cursor up one line
F	<SHIFT> + <Right Arrow>	Cursor to start of next word
G	<F3>	'Save' function
H	<DEL/BS>	Deletes preceding character
I	< →| >	TAB skip
L	<F7>	'Select' function

Fig. 5-7 Control Key Equivalents of Function Keys (cont.)

CONTROL KEY SEQUENCE <CTRL> +	EQUIVALENT TO	FUNCTION
M	<ENTER>	Enter a command or carriage return
N	<F1>	'Find' function
Q	<CTRL> + <Left Arrow>	Cursor to start of current line
R	<CTRL> + <Right Arrow>	Cursor to end of current line
S	<Left Arrow >	Cursor one space to left
T	<SHIFT> + <Up Arrow>	Cursor to top of screen
U	<F6>	'Cut' function
V	<F2>	'Load' function
W	<CTRL> + <Up Arrow>	Cursor to beginning of file
X	<Down Arrow>	Cursor down one line
Y	<PRINT>	Print operation
Z	<CTRL> + <Down Arrow>	Cursor to end of file

Fig. 5-7 Control Key Equivalents of Function Keys

6. THE ADDRSS APPLICATION PROGRAM

This program is designed to provide the user with a means of having all the information in a personal address and telephone book readily accessible in a variety of different forms. As explained in this chapter, the user can adapt the different features of the ADDRSS program to suit his own needs and to list the information in the personal address book by name, number, category, profession or geographical area.

In the case of the M10 MODEM, ADDRSS can be used in conjunction with the TELCOM application program to take advantage of the automatic dialling facility.

The first step, however, in using this program is to input to the computer the necessary information, in this case the list of names, addresses and telephone numbers to be used.

CREATING THE 'ADRS.DO' FILE

Access TEXT by positioning the cursor over the entry TEXT in the main menu and pressing <ENTER>. In response to the prompt, enter the file name. For this particular application, there is no choice of file name - the name ADRS.DO is mandatory.

If you call the ADDRSS program before creating the ADRS.DO file, the computer will display the message:

ADRS.DO not found
Press space bar for MENU

Once you have created the ADRS.DO file, enter the names, addresses and telephone numbers you wish to figure on your list. If you have the M10 MODEM and you want to make use of the autodialling facility, the list should be arranged in a particular order as shown below:

Name :telephone number: address

This format is to ensure that the autodial option of TELCOM can access the necessary information. In particular, the telephone number must be preceded and terminated by a colon to inform the computer of the position of the number in the record. If the number comprises a regional or district code, and a pause is required between this code and the rest of the number, insert the symbol = to create a two-second pause at that point in the auto-dial procedure. For example, the number 123=4567 will have a two-second pause between the 3 and the 4 during automatic dialling. The number 123==456 would have a four-second pause and so on.

Regardless of whether you are going to use the auto-dial option, it is still recommended to adopt a fixed and orderly format, to facilitate the

use of the 'Find' function and to have a readily understood address list when you look at it on the screen or on a print-out.

It is absolutely essential to terminate each entry by pressing <ENTER>. This is the only means whereby the computer can distinguish between one record and the next. The <ENTER> symbol, a dark, left-pointing arrow-head, should appear nowhere else but at the end of each record.

Remember that this is a TEXT file and therefore all the editing options described in Chapter 5 are equally valid for entering and editing text in the ADRS.DO file. Do not become confused between the related but quite distinct ADRS.DO and ADDRSS. ADRS.DO is the file containing the address book; ADDRSS is the program which sorts and organises this information according to the user's instructions.

When the list of names, addresses and telephone numbers is complete, close the file and return to the menu by pressing <F8>. The file appears on the menu as ADRS.DO.

USING THE ADDRSS PROGRAM

The ADDRSS program can now be called by setting the cursor to ADDRSS and pressing <ENTER>.

The screen will appear as shown in Figure 6-1.

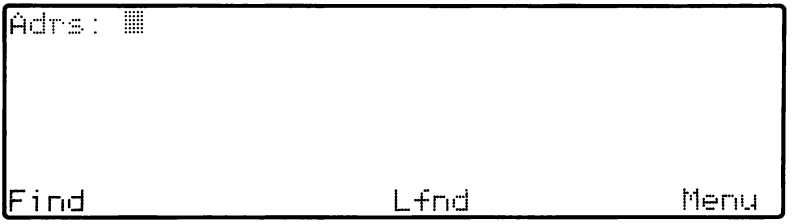

Fig. 6-1 Screen for ADDRSS Program

As can be seen, only three of the function keys are in use in this program viz. F1, F5 and F8.

- F1 invokes the 'Find' function.
- F5 is labelled 'Lfnd' and has the same function as F1 except that the result is output on the printer.
- F8, as in the other applications, returns the user to the menu.

The 'Find' function has an identical role to that which it plays in the

TEXT program. At the top of the screen appears the prompt:

Adrs:

In order to search for a particular entry in the ADRS.DO file, you need only press <F1> and type in any string of characters after the word 'Find' which appears following the prompt. Such a string can be of any length from a single character to the whole entry.

All entries containing the string entered will be displayed in full on the screen.

If <F5> rather than <F1> is pressed the result will appear at the output of the printer. Pressing <F5> when no printer is connected has the effect of blocking the computer. Press <SHIFT + PAUSE/BREAK> to clear it.

If you wish to view the contents of ADRS.DO in the order in which they were entered, access ADDRSS, press <F1> then <ENTER>. This will display the first six lines of the file on the screen. The functions F3 and F4 are labelled 'More' and 'Quit' respectively. These designations can be removed by pressing <LABEL>. To see the next six lines press <F3> and so on to the end of the file. Pressing <F4> terminates the viewing mode and displays the ADDRSS prompt on the screen.

The following example will serve to illustrate what has already been explained.

Suppose that the ADRS.DO file has been set up to read as follows:

1. W. M. Archer : 786=987469 : 34 Buckingham Street, Little Sutton.

2. Steven Y. Bennett : 316=99472 : 62 Sinclair Drive, Kingsley.

3. Bob Charles : 64 39 97 : 112 Laburnum Lane, West Brenton.

4. R.B. Johnston : 36182 : 56 Charles St., Landsdowne.

5. Peter King : 321=4961 : 85 Tavistock Ave., Fernbury.

Now suppose that the ADDRSS program is called and the 'Find' option selected.

If the first character string to be found is "Archer" then Entry 1 will appear in full on the screen.

If the selected string is "Charles" then Entries 3 and 4, in which it occurs, will both be displayed.

Now change the required string to "King"; obviously this will display Entry 5 but it will also bring up Entries 1 and 2 in which the string "king" occurs in "Kingsley" and "Buckingham" (remember that there is no distinction made between upper and lower case letters). In this instance, if the string is changed to " king " (i.e. with a space before and after the letters), only Entry 5 will appear.

If you do not wish to have spurious occurrences of the target string you must ensure that the string chosen is unique to the single entry you are seeking.

Of course, this feature can be turned to the user's advantage. By appending an unlikely sequence of characters to each entry you can establish categories and the ADDRSS program will give you a list by category when you enter the sequence. For example, all men could be coded XXX, all women YYY, all Personnel Officers PXP, all bank managers BXB etc.

The uses of the program are adaptable to the user's requirements. By imaginative use of such codes it is possible to have the information in ADRS.DO listed in many different forms according to category, profession, geographical location etc.

7. THE SCHEDL APPLICATION PROGRAM

This facility is provided to give the user a day-to-day schedule in which information can be coded in different ways according to the user's needs. As with ADDRSS, SCHEDL is a handling program for data contained in another file, in this case the file NOTE.DO. Thus SCHEDL can be used to select and organise the information in NOTE.DO in such a way that the latter acts as an appointment book, record of expenses, diary, general notebook and other applications of your own choosing.

SETTING UP THE 'NOTE.DO' FILE

Access the TEXT program by setting the cursor over TEXT in the main menu and pressing <ENTER>. In response to the prompt **File to edit?** type NOTE and press <ENTER> (as always the suffix .DO will be appended automatically). The screen will clear, leaving the cursor at the beginning of the file. If you call SCHEDL before creating NOTE.DO the following message appears on the screen:

NOTE.DO not found
Press space bar for MENU

The information entered in the file NOTE.DO is what would normally be contained in an appointment book, diary etc. The contents are very much a matter of personal choice. Most people will enter a list of forthcoming engagements, events and dates which have to be remembered, professional expenses and the like. However, regardless of what you choose to record in this file, the following guidelines are useful:

- Arrange the information in a logical, orderly fashion, bearing in mind how it will appear when you recall it using SCHEDL.

- Each separate entry or record should be terminated by pressing <ENTER>. These records should be kept to a reasonable size so that when they are recalled to the screen, the desired information is immediately available. At the same time, the entries must be long enough to contain all the requisite information.

- The use of symbols as prefixes for the type of entry in the file makes the task of tracing these much easier. Thus expenses could be prefixed $ or £, appointments #, items requiring action ! and so on.

Once you have completed the entries in NOTE.DO press <F8> to close the file and return to the menu.

USING THE SCHEDL PROGRAM

SCHEDL is called by positioning the cursor over SCHEDL on the main menu and pressing <ENTER>. The screen will then appear as in Figure 7-1 below.

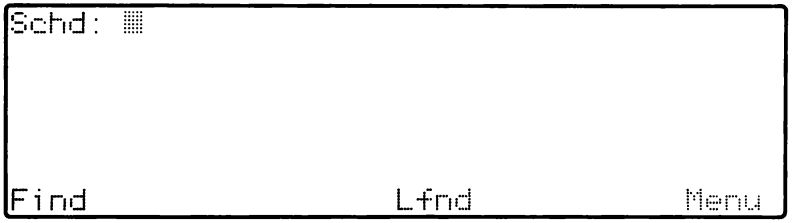

Fig. 7-1 The Screen for SCHEDL

At the top of the screen appears the prompt

Schd:

There are only three special function keys operative with this program viz. F1 - Find, F5 - Lfnd and F8 - Menu and these are shown at the bottom of the screen. To remove the labels from the screen, press <LABEL>.

- F1 (Find) has an identical function as in the ADDRSS program. Press <F1> to bring up the prompt **Find** , type in the character string you want the program to look for in NOTE.DO then press <ENTER>. Any entry in which the string occurs will be displayed in full on the screen.

- F5 (Lfnd) acts in exactly the same way as 'Find', except that the result appears not on the display but at the output of a printer connected to the M10.

  Note that if you select this option when no printer is connected, the computer will block. Press <SHIFT + PAUSE/BREAK> to clear it.

- As in the other applications, pressing <F8> returns the user to the main menu.

To examine the contents of the NOTE.DO file, call SCHEDL, press <F1> then <ENTER>. The first six lines of NOTE.DO will then be displayed on the screen. The function F3 is labelled 'More' while F4 is labelled 'Quit'. Press <F3> to view the next six lines and so on to the end of the file. Pressing <F4> stops the viewing mode and brings back the SCHEDL prompt.

The use of SCHEDL is best illustrated by an example. Suppose the entries below represent an extract from the NOTE.DO file.

```
11/07 $  Expenses in Geneva
         Taxi from airport SF 42.70
         Lunch             SF 25.00
         Dinner            SF 67.90
         Taxi to airport   SF 45.00

12/07 #  10.30   Appointment with H.B. Francis

12/07 #  14.30   Departmental Meeting

13/07 #  12.30   Lunch with Mr. Lombard

14/07 !  Settle telephone A/C

15/07 *  15.00   Flight to London  HK 489
         Reservation at Royal Oak Hotel

16/07 !  Telephone Mr. Henry in Manchester
```

This varied information for the dates 11/07 to 16/07 can be examined selectively using the SCHEDL program. In this particular example, the entries have been coded as follows:

$ represents professional expenses

represents appointments

! represents items requiring action

* represents travel

Now here are a selection of ways in which this information can be sorted and presented:

First call SCHEDL and press <F1>.

- Choosing the character string

  14/07 !

  will display all items requiring action that day (in this case payment of a telephone bill).

- Likewise if the string is one of the single symbols #,$ or *, you can list appointments, expenses or travel arrangements.

- By selecting a date, you get a list of all entries for that particular day.

- Selecting a name (e.g. Lombard) brings up the entry concerning that person.

Obviously, this idea of coding can be greatly extended to meet the user's requirements. For example, the double code *$ could be used for entries concerning both travel and expenses etc.

Note that the symbols used in this example are only to demonstrate the principle; the user may use any code that suits his purpose. In this context, it is worth pointing out that the M10 keyboard provides special symbols such as:

an aircraft	<SHIFT> + <GRPH> + c	
a telephone	<SHIFT> + <GRPH> + \	on the US and UK keyboards
	<SHIFT> + <GRPH> + <	on the Italian, French and German keyboards
a car	<SHIFT> + <GRPH> + v	

The NOTE.DO file can be updated by deleting outdated entries from time to time. Alternatively, if you wish to keep a permanent record of, say, your expenses or travels then a more subtle method is to add a letter to the code which shows it immediately to be a past record. Suppose the letter chosen is x. Then asking SCHEDL to find *x will provide a list of all past travels since you started to keep records.

Again the possibilities of the system are limited only by the needs and inventiveness of the user. Your NOTE.DO file can be as simple or as complex as you care to make it.

8. THE TELCOM PROGRAM

The TELCOM program is designed to allow the M10 to communicate with other computers, either by transmitting and receiving data over a telephone line or by direct connection. In this program, more than any other, there are major differences between the M10 MODEM and the models without this facility. All the functions and operations described in this chapter are available to the M10 MODEM.

For the user's convenience, the table in Figure 8-1 summarises the differences between the M10 MODEM and the other models for the various connections and operations of the TELCOM program. It also lists the required accessories for all of these options. Thus the user can tell at a glance which operations are available with his model and which accessories are needed.

	M10 MODEM	M10 WITHOUT MODEM
Direct connection to a telephone line	Modem cable	Not possible
Auto-dialling	Modem cable	Not possible
Manual dialling from keyboard	Modem cable	Not possible
Automatic log-on to host computer	Modem cable	Not possible
Data exchange by telephone with a host computer	Modem cable or acoustic coupler	Combined acoustic coupler/modem
Direct connection to another computer	Null modem connecting cable (cross cable) with straight cable for gender change	Null modem connecting cable (cross cable) with straight cable for gender change

Fig. 8-1 Accessories for TELCOM Options

TELCOM allows the user an automatic dialling option (M10 MODEM only), and connection to another computer either directly or via a telephone line. Thus data can be exchanged between the M10 and a host computer or computer service such as Compuserve or Dow Jones in the US. By using the various options at his disposal, the M10 MODEM user can automatically dial and log on to a host computer. All the different options available

to both the M10 MODEM and the other versions of the M10 are described in detail in this chapter.

To access the TELCOM program, position the cursor over the entry TELCOM on the main menu and press <ENTER>. The screen then appears as shown in Figure 8-2 below.

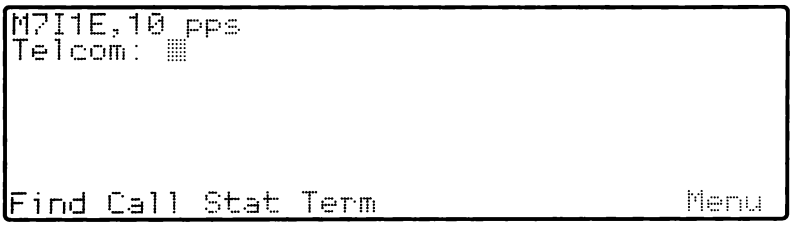

Fig. 8-2 Screen When TELCOM is Called

The code which appears on the first line of the display is a list of the telecommunications parameters which are discussed in some detail later in this chapter. On the second line is the TELCOM prompt. The bottom line displays the definition of the function keys on entering TELCOM.

TELCOM operates in two different modes, Entry and Terminal. On first accessing the program it is automatically in Entry mode. In each of these modes the function keys F1-F8 have different definitions and uses. Entry mode is essentially for using the auto-dialling option while Terminal mode is for data exchange with another computer. The two modes can be used in sequence by the M10 MODEM to give an automatic dial and log-on option. In such a sequence, Terminal mode is entered automatically as part of the procedure; otherwise it is necessary to enter Terminal mode manually by means of the function key F4.

CONNECTING THE M10 TO A TELEPHONE LINE

The M10 can be connected to a telephone line, either directly or through an acoustic coupler. In the M10 MODEM version, both options are open. A special modem cable is used to make the direct connection to a modular telephone line and a simple acoustic coupler is used when linking the computer to the line via the receiver. In all other versions, the connection must be made through a combined acoustic coupler/modem in order to use the data communications facility offered by the TELCOM program.

These three accessories - modem cable, acoustic coupler and combined acoustic coupler/modem are all available from your Olivetti dealer. It is particularly recommended to use the Olivetti MC 10 Modem Coupler, specially designed for data communications using the M10 without integrated modem. A detailed description of this device is given in Appendix C.

All countries have their own legislation concerning the use of devices connected to the telephone network. Before connecting the M10 to the telephone system, make sure you are familiar with the regulations of your national PTT Administration. Your Olivetti dealer will be able to help you with this.

In the US, before connecting the M10 MODEM to the telephone network, you must provide the local telephone company with certain information such as FCC ID, FCC Registration number and Ringer Equivalence Number. All the required information appears on the identity tag on the underside of the M10 MODEM.

Figure 8-3 depicts the connection between the M10 MODEM and a modular telephone line, using a modem cable.

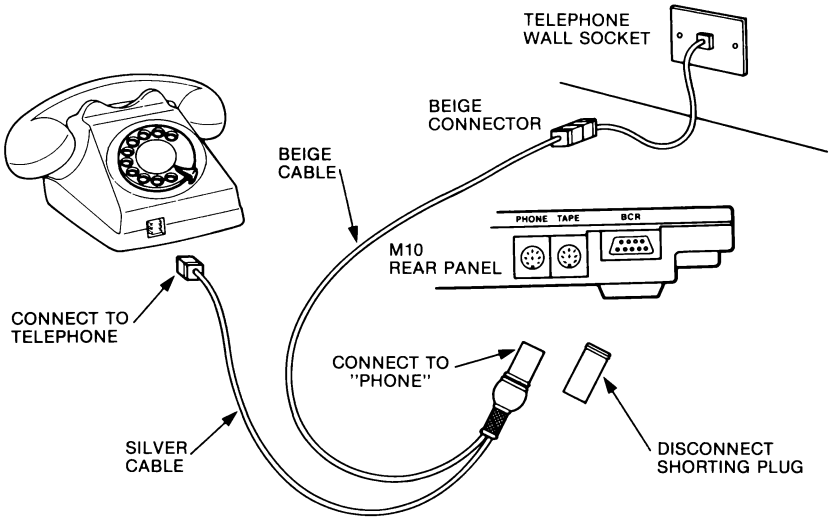

Fig. 8-3 Direct Connection to a Telephone Line

The modem cable has two branches, a silver cable and a beige cable. Both carry male connectors so you must be careful when making the connection between M10 MODEM and telephone line.

1. Disconnect the telephone line from the back of the handset but leave the cable connected to the wall socket.

2. Connect the silver branch of the modem cable to the socket on the back of the telephone handset from which you have just removed the

wall connection.

3. Plug the connector you have just removed from the handset (and which goes to the wall socket) into the beige box at the end of the beige cable.

4. Remove the shorting plug from the other end of the modem cable and insert the round connector into the socket marked PHONE on the rear panel of the M10.

5. Set the DIR/ACP switch on the underside of the M10 MODEM to DIR.

6. The M10 MODEM is now connected for data exchange by telephone with another computer.

If you do not need the telephone handset, simply connect the beige cable directly to the telephone wall socket. If you wish to use the telephone normally without undoing all the connections, remove the modem cable from the PHONE socket on the M10 MODEM and attach the shorting plug to the connector.

Sometimes it is not possible to detach the line from the back of the telephone handset (in hotel rooms, for example). In this case you must use the acoustic coupler.

1. Connect the end of the coupler to the PHONE connector on the rear panel of the M10 MODEM.

2. Fit the cups of the acoustic coupler to the receiver so that the coupler speaker is over the telephone microphone and the coupler microphone over the telephone speaker.

3. Set the DIR/ACP switch to ACP.

If you have the choice, always use the direct connection for a better and more reliable exchange of data.

Note that the auto-dialling option is not available when the acoustic coupler is used.

In all models of the M10 without integrated modem, it is necessary to use a combined acoustic coupler/modem in order to exploit TELCOM to the full. The Olivetti MC 10 Modem Coupler is the ideal equipment for this application. In this case, set up the the M10 for remote communication using the MC 10 as follows:

1. Connect the MC 10 cable to the RS-232C connector on the rear panel of the the M10.

2. Ensure that the CAL/ANS switch on the underside of the MC 10 is set to CAL.

3. Turn the power ON/OFF switch on the MC 10 to the ON position. The indicator lamp will show a green flashing light.

The M10/MC 10 combination is now ready for connection to a telephone.

ENTRY MODE

When you access TELCOM from the menu, you are always in Entry mode. It is a mode of operation principally, but not exclusively, designed to provide the user of the M10 MODEM with an automatic dialling option. In addition, this mode controls entry to Terminal mode, the return to the main menu and modification of the communications parameters.

THE FUNCTION KEYS IN ENTRY MODE

In Entry mode the function keys, as can be seen in Figure 8-2, control the following operations:

F1 (Find) - Pressing this key enables the TELCOM program to find a name and telephone number already listed in the ADRS.DO file and to display it on the screen. This function is available only on the MODEM model.

F2 (Call) - This key enables the M10 to call a specified number automatically when it is connected to a telephone line. Again this function exists only on the M10 MODEM.

F3 (Stat) - This lists the current communications parameters and allows the user to modify them. It is used for data communication with another computer and is dealt with in detail later in this chapter.

F4 (Term) - This key changes the mode to Terminal.

F5, F6 and F7 are not used.

F8 (Menu) - Returns the user to the main menu.

AUTOMATIC DIALLING

This option is available only to the M10 MODEM for which the TELCOM program has access to the ADRS.DO file set up in connection with the ADDRSS program (see Chapter 6). To find a name and telephone number listed in this file, proceed as follows:

1. Press <F1>. This brings up the prompt **Find** .

2. Enter the name of the person (or other identifying string) whose number is required, exactly as you would in a 'Find' operation in the ADDRSS program. The target string may be taken from any part of the entry in the ADRS.DO file i.e. name, address or telephone number. Press <ENTER>.

3. The bottom line of the screen changes to read F2 - Call, F3 - More and F4 - Quit. Meanwhile the first occurrence of the target string is indicated on the screen. What is displayed is, in fact, the corresponding record in the ADRS.DO file up to the second colon. If you have followed the recommended format for entering records in ADRS.DO, this will be:

Name : telephone number

(see Chapter 6 for details of how records are entered in ADRS.DO).

4. You now have a choice; pressing 'More' causes the computer to look for a further occurrence of the string pressing 'Quit' returns the TELCOM prompt; pressing 'Call' instructs the computer to dial the number displayed on the screen.

Of course, in order to dial the number successfully, the M10 must be directly connected to a modular telephone line in the way described in the preceding section of this chapter.

When 'Call' is pressed, the message

Calling .....

appears on the screen. The dots represent the name of the person listed in ADRS.DO. As the number is dialled, the M10 emits faint but audible pulses and each digit is displayed on the screen as it is dialled. You must lift the receiver before the number is complete. This entire operation from 'Find' through to completion of the number is the auto-dial option.

To illustrate this explanation, let us take an entry from the ADRS.DO file already used as an example in Chapter 6, e.g.

Peter King : 321=4961 : 85 Tavistock Avenue, Fernbury

1. Press <F1>, and, in response to the prompt **Find** type in "Peter" and press <ENTER>.

2. Immediately, the screen displays

   Peter King : 321=4961

   i.e. the record up to the second colon. The result would be identical if the target string was "Tavistock" or "321".

3. The functions 'Call', 'More' and 'Quit', corresponding to F2, F3 and F4, appear at the bottom of the screen. Press <F2> to initiate the dialling operation.

4. The screen displays the following message:

   Calling Peter King : .....

   As dialling proceeds, the digits appear one by one in the position indicated by the dots. At the same time, you will hear the faint sound of the pulse train for each digit. Note the two-second pause between the 1 and the 4.

5. You must pick up the receiver to take the call before the last digit appears on the screen.

Note that this example can be tried without connecting the M10 MODEM to

the telephone line, although needless to say no actual call can be made. In all other respects, however, the results are identical.

MANUAL DIALLING

Telephone numbers can also be dialled manually from the M10 MODEM keyboard by pressing <F2>, typing the number after the prompt **Call** and pressing <ENTER>. Again the audible pulses are emitted and the digits displayed on the screen. The receiver must be lifted before the last digit is dialled. You can verify this procedure without connecting the M10 MODEM to the telephone line. Even with the M10 MODEM connected directly to the telephone line, a number can be dialled in the classical way from the telephone itself.

TERMINAL MODE

The object of this mode of operation is to allow the M10 to communicate, either by telephone or directly, with other computers or information services - in effect to act as a terminal on a host computer. There are two ways of entering Terminal mode - automatically or manually.

The former occurs when you follow the procedure for auto-dialling of a host computer or information service. This procedure, described later in this chapter, is available only to the M10 MODEM.

Manually, Terminal mode is entered by pressing <F4> after the TELCOM program has been called from the main menu. The screen then appears as shown in Figure 8-4.

Fig. 8-4 Screen on Entering Terminal Mode

THE FUNCTION KEYS IN TERMINAL MODE

As can be seen, the function keys F1-F8 now control different functions from those effective in Entry mode. Unlike the functions available in Entry mode, those in Terminal mode are identical for all models of the M10. They are:

F1 (Prev) - Pressing this key enables you to view the previous eight lines of text on the screen when in Terminal mode. To return the display to its original position, press <F1> again.

F2 (Down) - The purpose of this key is to save or 'download' incoming data from a host computer to a file for subsequent viewing. When you press <F2>, the following prompt appears on the screen:

File to Download?

Type in the name of the file to which the data are to be saved and press <ENTER>. To stop downloading press <F2> again.

F3 (Up) - The purpose of this key is to allow you to send a file of data which you have prepared in advance for transmission to the host computer. When you press <F3>, the prompt:

File to upload?

will appear on the screen. Type in the name of the previously prepared file that you wish to send and press <ENTER>. This brings up the prompt:

Width?

The response to this prompt determines the maximum number of characters the M10 will send before inserting a carriage return and can have any value between 10 and 132.

F4 (Full/Half) - This key switches the transmission between Full Duplex and Half Duplex. In Full Duplex mode, required by most host computer systems, the characters are transmitted to the host before appearing on the screen. In this way, you know that the character has been accepted by the host as soon as it appears on the screen. In Half Duplex, the characters appear on the screen and are transmitted simultaneously so there is no guarantee that what you read on the screen is in fact what the host computer has received. A noisy telephone line, for example, can distort the transmission.

F5 (Echo) - By connecting a printer to the M10 and pressing <F5>, you will obtain a hard copy of what is displayed on the screen. Thus you can retain a printed record of the

exchange with the host system. F5 is a switch key i.e. when it is pressed once, the function operates; pressing again disables the function and removes the label 'Echo' from the screen.

F6 and F7 are not used.

F8 (Bye) - This key allows you to leave Terminal mode.
When you press <F8>, the prompt **Disconnect?** appears on the screen. In response, type <Y> (Yes) or <N> (No) and press <ENTER>. To disengage the M10 from the telephone system, you must enter 'Y'.
This returns you to Entry mode.

THE DATA COMMUNICATIONS PARAMETERS

Reference was made earlier to the communications parameters which appear on the top line of the screen when the TELCOM program is first accessed. This represents the current values of the communications parameters. To communicate with another computer these parameters must match those of the host and you can set the appropriate values on the M10 within the limits quoted in Figure 8-5.

PARAMETER	POSSIBLE VALUES	MEANING
Baud Rate	M	Modem (300 baud)
	1	75 baud
	2	110 baud
	3	300 baud
	4	600 baud
	5	1200 baud
	6	2400 baud
	7	4800 baud
	8	9600 baud
	9	19200 baud
Word Length	6	6 bits
	7	7 bits
	8	8 bits
Parity	O	Odd
	E	Even
	N	No Parity
	I	Ignore Parity
Stop Bit	1	1 Stop bit
	2	2 Stop bits
Line Status	E	Enable
	D	Disable
Dial Pulse Rate	10	10 pps
	20	20 pps

Fig. 8-5 Data Communications Parameters

In Figure 8-2, the parameters are set to **M7I1E, 10 pps** which are the initial values for the M10 MODEM. Other models have initial values of **37I1E**. From the table it can be seen that this should be interpreted as:

Baud Rate - M (Modem); this value is numerically 300 baud but it should always be set to M for the M10 MODEM, otherwise the modem is disabled. For all other models, the starting value is 3 (300 baud).

Word Length - Initially set to 7 bits

Parity - I for Ignore Parity

Stop Bit - 1 stop bit

Line Status - E for Enable

Dial Pulse Rate - 10 pps; this parameter does not exist in models without integrated modem, since it determines the pulse rate used in auto-dialling, a facility available only to the M10 MODEM.

It is not imperative for the user to be conversant with the significance of all these parameters in order to be able to operate in Terminal mode with a host computer. The purpose of including them is to allow you to match the M10 to the specifications provided by the host.

To change the communications parameters, access Entry mode and press <F3> (Stat). The prompt **Stat** appears on the display. Type in the communications protocol (i.e. the string of parameters in the given format, as in M711E,10 pps) and press <ENTER>; the TELCOM prompt appears again. Now verify that the new values have been registered by the M10 by pressing <F3> again. When the **Stat** prompt appears, simply press <ENTER> and the new values will be displayed on the next line. Note that the communications parameters are accessed and modified in Entry mode although their application is entirely in Terminal mode.

The current values of the communications parameters can be checked at any time, using <F3> and <ENTER>.

DATA EXCHANGE WITH A HOST COMPUTER

Terminal mode is designed to allow the M10 to communicate with other computers, either by direct connection or via a telephone line.

In the case of the M10 MODEM, the connection to the telephone may be direct, as explained earlier in this chapter or, when this is not feasible, by means of an acoustic coupler.

For the M10 models without integrated modem, the connection must be via a combined acoustic coupler/modem.

Before using the M10 in Terminal mode, it must be linked into the telephone system. Terminal mode can be accessed automatically or manually; automatic access to Terminal mode is available only to the M10 MODEM. In addition, this model offers the possibility of automatic log-on procedure to a host computer or information service, greatly facilitating access to a host system.

ACCESSING TERMINAL MODE MANUALLY

Manual access to Terminal mode is necessary in all models without an integrated modem and may be chosen, if one so wishes, with the M10 MODEM. To access Terminal mode on the M10 MODEM, proceed as follows:

1. Make sure that the M10 MODEM is connected to the telephone line as shown in Figure 8-3 or by means of an acoustic coupler. In the first case, set the DIR/ACP switch to DIR, in the second set it to ACP.

2. Now set the CAL/ANS switch (on the underside of the M10 MODEM) to CAL and access TELCOM.

3. Lift the telephone receiver and dial the number in the normal way.

4. When the host system replies to your call, you will hear a continuous, high-pitched tone. Press <F4> to enter Terminal mode.

5. If the connection between the M10 and the telephone is direct, you can at this point simply replace the receiver. If you are using an acoustic coupler, reconnect the receiver to the coupler when you hear the continuous tone.

When the M10 enters Terminal mode, it emits a high-pitched sound and the new designations of the function keys F1-F8 appear at the foot of the screen (see Figure 8-4). The M10 MODEM is now ready to log on to a host computer system.

With all other models, it is necessary to use a combined acoustic coupler/modem to set up the M10 for remote communication with a host system. When using the M10 with the MC 10 Modem Coupler, carry out the following procedure:

1. Ensure that the MC 10 is connected to the RS-232C interface connector on the rear panel of the M10 and that the ANS/CAL switch on the underside of the MC 10 is set to CAL.

2. Turn the power ON/OFF switch on the top of the MC 10 to ON. The indicator lamp will show a flashing green light.

3. Lift the telephone receiver and dial the number of the host system in the normal way.

   If the host system has automatic data transmission connection, you will hear a high-pitched, continuous tone when connection has been established.

   If the host system is manual, ask the operator to make the data transmission connection. When you hear the continuous tone, the connection has been established.

4. Once communication has been established, press <F4> to enter Terminal mode.

5. Fit the telephone receiver to the MC 10 cradle, first connecting the ear-piece then inserting the mouth-piece into the cup marked CORD.

   After a moment, the indicator lamp will show a continuous green light.

When Terminal mode is entered, the M10 emits a high-pitched sound and the new designations of the function keys F1-F8 appear at the bottom of the screen, as shown in Figure 8-4.

The M10/MC 10 combination is now ready for log-on to a host computer.

AUTOMATIC ENTRY TO TERMINAL MODE AND AUTO LOG-ON PROCEDURES

This option, available only to the M10 MODEM, greatly facilitates the whole procedure of calling a host system and logging on to it. It may take you a little while to set it up but once you have done so your access to a host computer or computer service will be effected by a single command.

Entering Terminal Mode Automatically

To do this you must start with the ADRS.DO file, created when you first used the ADDRSS program (see Chapter 6). Let us suppose that you are a subscriber to an information service called International Data Services (IDS), whose telephone number is 492 77095. This information has to be recorded in the ADRS.DO file. Access ADRS.DO and make the following entry:

IDS : 492=77095 < >

This follows the standard format for entries in ADRS.DO with the exception of the angle brackets whose significance will become clear later in this section. Now quit ADDRSS and access TELCOM. The procedure for automatic entry to Terminal mode is the following:

1. Ensure that the M10 is directly connected to the telephone system as described earlier and as shown in Figure 8-3.

2. Set the DIR/ACP switch to DIR, if you have not already done so.

3. Set the CAL/ANS switch to CAL.

4. Press <F1>, then, in response to the prompt **Find** type in IDS and press <ENTER>.
   This will cause the entry you have made in ADRS.DO to be displayed on the screen as shown below.

   IDS : 492=77095 < >

   while the bottom line of the display shows 'Call', 'More' and 'Quit' corresponding to F2, F3 and F4.

5. Press <F2> to initiate the auto-dial option. This brings up the message

   Calling IDS : 492=77095 < >

   the digits appearing one by one as they are dialled.

There is no need to lift the receiver when auto-dialling a host system since the communication is not verbal. On completion of the number, the M10 MODEM emits a high-pitched tone to indicate that Terminal mode has been entered. At the same time, the screen changes to show the designation of the function keys, F1-F8, in Terminal mode. In most cases, the M10 will echo what it "hears" at the other end of the line. If the line is free, you will hear the ringing, if it is engaged, you will hear the

engaged signal and so on. Automatic entry into Terminal mode is now complete and the M10 MODEM is ready to log on to the host computer. The M10 MODEM is equipped with an automatic log-on protocol to further simplify your task.

Automatic Log-On

The combination of automatic dialling and automatic entry into Terminal mode allows one further extension of the system to give automatic log-on to a host computer or information service. Log-on procedures for computers vary only in the details, the basic features, such as entering user ID and password being common to them all. The M10 MODEM provides a series of key commands which allows you to take advantage of this fact and devise a protocol to execute an automatic log-on with a single command.

When you become a subscriber to an information service or are given access to a host computer, you will be issued with instructions for the log-on procedure. What the the M10 MODEM offers is the possibility of encoding the procedure in a string of commands and storing this in the ADRS.DO file so that the entire sequence of calling the host and logging-on can be effected automatically.

The four special command keys shown in the table below enable you to encode the log-on sequence and store it in the ADRS.DO file.

KEY	FUNCTION
?	Wait for specified character
=	Pause for two seconds
!	Send the next character as it stands
^	Send the next character as a control character

? This character tells the M10 to wait for a specific instruction or prompt before proceeding with the log-on sequence. If you know from the host's instructions that you must wait for the prompt **User ID:** before entering your identity, then the coded instruction to the M10 would be '?' followed by any character unique to that prompt (say 'U'). ?U is a coded message to the M10 saying "Do not proceed until you have received a prompt containing the character U".

= This symbol, which you have already encountered in the description of the ADRS.DO file, imposes a delay of 2 seconds before continuing the log-on procedure.

! The exclamation mark tells the M10 MODEM to transmit the next character as it stands. It provides the computer with a means of distinguishing between key commands and characters to be transmitted. Thus if the combination != occurs anywhere in the sequence, the M10 MODEM will send the character = instead of pausing for 2 seconds as it normally would on encountering this character.

^ The circumflex causes the next character to be transmitted as a control character. Hence ^C tells the M10 MODEM to send Control-C.

The entire log-on sequence is stored in the ADRS.DO file under the entry for the host's name and telephone number. It must be enclosed in the angle brackets < > immediately after the number.

The best way to illustrate setting up a coded log-on protocol is by returning to the example used earlier. Suppose that IDS have assigned you the user ID 5138Q and the password 'Redwing' and that they have issued the following instructions for log-on:

1. Notify the system that you are ready to log-on by sending Control-C.

2. Wait for the prompt **User ID:** then enter the identity number you have been assigned.

3. Wait for the prompt **Password:** then enter your password.

4. Wait for at least 3 seconds then send the question OK?.

5. The system will send you the message **Log-on successfully completed .**

Taking these instructions one by one you can now establish a coded sequence for the M10 MODEM.

1. It is good practice to leave a short delay between calling the number and starting the log-on to allow time for good communications to be established, so begin the sequence with =. The first instruction requires a Control-C which is written ^C. So, after complying with the first instruction, the sequence looks like this:

   =^C

2. Now you must inform the M10 to wait for the prompt **User ID:** then send the ID number assigned to you. There are only two prompts in the log-on procedure so choose any letter that is not common to both e.g. U, I or D. In this procedure, unlike many others, the M10 distinguishes between upper and lower case characters. Thus U is valid as a unique character in the prompt, whereas u is not. Note also that you must select a single character since the key command ? applies only to the character immediately following it. The instruction ?U tells the M10 to wait for a prompt containing the character U. Now your ID is required, so continue with 5138Q. This has to be entered which you normally do by pressing <ENTER>. The keyboard equivalent of ENTER on the M10 is Control-M. Hence the instruction is terminated with ^M. The sequence for the second instruction is then:

?U5138Q^M

3. In the same way the coded sequence for entering the password is:

?PRedwing^M

where ?P tells the M10 to wait for a prompt containing the character P before transmitting the password.

4. This instruction requires a pause of at least 3 seconds; the characters == give a 4-second pause. Now you are required to enter the question OK?. You must tell the M10 MODEM to transmit the question mark and not interpret it as a key command to wait, so this becomes OK!?^M. The final instruction gives rise to the sequence:

==OK!?^M

Putting these all together in a single sequence gives the final protocol which has to be enclosed in angle brackets thus:

<= ^C ?U 5138Q ^M ?P Redwing ^M == OK!? ^M>

Here the individual commands have been spaced out so that the user can readily identify them. When entering them in the ADRS.DO file they should appear as a single, long string as shown below:

IDS : 492=77095 <=^C?U5138Q^M?PRedwing^M==OK!?^M> :

To initiate the automatic log-on sequence, using the example just quoted as an illustration, the procedure is as follows:

1. Ensure that the M10 MODEM is connected directly to a modular telephone line.

2. Set the DIR/ACP switch to DIR and the CAL/ANS switch to CAL.

3. Access TELCOM and verify that the communications parameters correspond to the specifications of the information service IDS. If not, change them to the required values.

4. Still in Entry mode, press <F1> and, in response to the prompt **Find** type IDS and press <ENTER>.

5. The entry in the ADRS.DO file for IDS will be displayed on the screen, up to the second colon.
   Note that the log-on sequence you have just entered between the angle brackets is not displayed although the brackets themselves are. This is to protect the secrecy of your User ID and password.
   Press <F2> 'Call', then <ENTER> to initiate automatic dialling.

6. The message **Calling IDS :49277095** is displayed on the screen, the digits appearing one at a time as they are dialled.
   When the number has been dialled, you will hear the ringing tone. A high-pitched tone is emitted by the M10 to indicate that Terminal mode has been entered.

Up to this point the procedure is that of auto-dialling.

7. At this point the automatic log-on sequence commences. The first indication you have of this is when the prompt **User ID:** appears on the screen. The M10 MODEM responds to this automatically, according to the coded instructions you have given, and your user ID will appear in response to the prompt.

8. Next the prompt **Password:** appears on the screen. Again the M10 responds automatically but your password is not displayed.

9. After a 4-second pause, the question OK? will appear, followed by the message **Log-on successfully completed .** The M10 MODEM is now logged on to the IDS system.

A word of caution here - if you attempt to simulate this procedure without connecting the M10 MODEM to a telephone line, the computer will block as soon as the simulated dialling is over because it can proceed no further. This blocking in TELCOM can even inhibit the automatic switch-off facility. If this happens, press <SHIFT> + <BREAK/PAUSE> to release the M10.

MANUAL LOG-ON TO A HOST COMPUTER

If you do not have the M10 MODEM but another version of the M10, then the automatic log-on sequence is not available to you. In this case, you must dial the host's number manually on the telephone and follow the instructions for logging-on. By way of illustration, we shall take the example used in the last section. Since that section refers exclusively to the M10 MODEM, you may not have read it, so we shall repeat the essentials of the example here. Let us suppose that you are a subscriber to an information service called International Data Services (IDS), whose telephone number is 492 77095. Suppose, moreover, that IDS have assigned you the user ID 5138Q and the password 'Redwing' and that they have issued the following instructions for log-on:

1. Notify the system that you are ready to log-on by sending Control-C.

2. Wait for the prompt **User ID:** then enter the identity number you have been assigned.

3. Wait for the prompt **Password:** then enter your password.

4. Wait for at least 3 seconds then send the question OK?.

5. The system will send you the message **Log-on successfully completed .**

The procedure is as follows:

1. Connect the MC 10 Modem Coupler to the RS-232C output on the rear panel of the M10, set the ANS/CAL switch to CAL, and turn the power ON/OFF switch to ON, as explained earlier in the present chapter.
   If you are using another modem coupler, follow the operating instructions provided by the manufacturer, to ensure that it is ready for calling a host computer.

2. Access TELCOM from the main menu.
   Check that the communications parameters displayed on the first line of the screen correspond to the IDS specifications. If not, press <F3> and enter the appropriate values. Remember that with the MC 10, the maximum data signalling rate (baud rate) is 300 baud.

3. Lift the telephone receiver and dial the IDS number - 492 77095. As soon as communication has been esatablished, place the receiver in the cradle of the Modem Coupler as described in the section on manual access to Terminal mode.

4. Press <F4> (Term) to enter Terminal mode. A high-pitched tone is emitted by the M10 to indicate that Terminal mode has been entered. At the same time, the designations of F1-F8 for Terminal mode are displayed at the foot of the screen.

5. Now follow the IDS instructions for the log-on procedure.

6. Having allowed a short delay to ensure that good communications have been established with the host, press <CTRL> + c to indicate to the IDS system that you are ready to log-on.

7. Wait until the prompt **User ID:** appears on the screen then type in 5138Q and press <ENTER>.

8. In response to the prompt **Password:** which next appears on the screen, type 'Redwing' and press <ENTER>. Your password will not actually be displayed but it is transmitted to the host system.

9. Allow a short delay of at least 3 seconds then type in the question OK?. If everything has proceeded normally, you will receive the message **Log-on successfully completed** and the M10 will be logged on to the IDS system.

USING THE UPLOAD AND DOWNLOAD FACILITIES

In the introduction to Terminal mode of operation earlier in this chapter, reference was made to these two facilities when describing the function keys. The user requires some more details to make use of these options.

The Upload Facility

This option is provided to enable the user to prepare a file of information in advance for subsequent transmission to another computer. To do this proceed as follows:

1. Using the method explained in Chapter 5, create a TEXT file containing the data to be transferred to another computer. Save this file to RAM.

2. Once you have established communication with the other computer via the TELCOM program and initiated the sequence on the host to prepare it to accept data, press <F3> (Up).

This brings up the prompt:

File to upload?

3. Type in the name of the file you have previously created for this purpose and press <ENTER>.

4. In response to the prompt **Width?** which now appears, select a value between 10 and 132 and press <ENTER>. This determines the line width of the file to be uploaded. The physical line width of an M10 file is 40 characters (the width of the screen). However, because the word-wrapping feature makes carriage returns redundant for continuous text, a text line can be many times that width. If the M10 does not encounter a carriage return for a number of characters equal to the value entered as a response to the **Width?** prompt, it will impose one. All carriage returns encountered in the file are transmitted. If <ENTER> is pressed in response to the prompt, no extra carriage returns will be embedded.

The Download Facility

This option allows the user to "download" incoming data from a host computer i.e. to save it to a file for future reference. This is particularly useful when the data are coming in streams that require subsequent interpretation. To set this option in operation, do the following:

1. Since you are already in communication with a host computer, the M10 will be in the TELCOM program and in Terminal mode.
   Type the command to the host computer to initiate the data stream BUT DO NOT YET PRESS <ENTER>.
   Press <F2> (Down). This brings up the prompt:

File to download?

2. Type in the name of the file in which the information is to be stored and press <ENTER>. You need not have created the file in advance; if you select a name not exceeding 6 characters in length, observing the rules for the nomenclature of TEXT files, it will be created and stored in RAM.

3. Press <ENTER> to complete the host command and start the downloading process.

4. While downloading is in progress, the label 'Down' at the foot of the screen appears "in negative" to indicate that the incoming data are being downloaded.

5. When you have downloaded the data you want, terminate the Download operation by pressing <F2> again.

To read and edit the downloaded data, press <F8> (Bye) to terminate communication with the host computer. Press <Y> then <ENTER> in response to the prompt **Disconnect?** to disengage the M10 from the telephone network and return to Entry mode. Return to the main menu by pressing <F8>

(Menu) and select the file containing the downloaded data. You can access and edit this as you would any TEXT file. Some editing may be required if the host system has inserted control characters in the transmitted information.

If the file you have downloaded is a BASIC program file, you must enter BASIC and load the file. This will convert it to binary form and allow you to run and save it as a .BA file. If you save it as a BASIC file, the .DO file may be deleted.

9. USING A CASSETTE TAPE RECORDER WITH THE M10

It has already been mentioned, on several occasions, that files created on the M10 can be stored on cassette tapes. In fact this practice is recommended to provide back-up for your files and to clear the RAM for storing new files.

The M10 applications programs BASIC and TEXT provide the necessary commands for saving a file to tape and loading a file from tape. In view of this, the use of a cassette tape recorder can be regarded as a standard operational facility of the M10. The present chapter is devoted entirely to these operations.

CONNECTING THE M10 TO A CASSETTE RECORDER

Connecting the M10 to a cassette tape recorder enables you either to transfer files from the M10 to the tape (Save) or to transfer files from the tape to the M10 (Load). Before making the connection, insert the cassette tape and rewind it to the position where the recording is to start, using the counter to fix the position. Always take note of the counter readings when putting files on tape. This greatly simplifies the task of refinding them. Any standard cassette tape recorder can be used, provided it is equipped with the following features, in addition to the normal facilities such as RECORD, PLAY, STOP/EJECT, WIND etc.:

- Input microphone jack MIC
- Output jack EAR
- Remote control jack REM
- Tape counter

The connections between the M10 and the cassette recorder are depicted in Figure 9-1. The interconnecting cable can be supplied by your Olivetti dealer. This cable has three plugs for connection to the cassette recorder jacks on one end, and a round eight-pin connector for connection to the M10 at the other. The three plugs are red, white and black. The black plug has a perceptibly smaller connecting pin than the others.

1. Connect the round 8-pin connector on one end of the cable to the TAPE socket on the rear panel of the M10.
2. Connect the white plug to the output jack EAR.
3. Connect the black plug to the remote control jack REM.
4. Connect the red plug to the input jack MIC.

5. Switch on the cassette recorder and the M10.

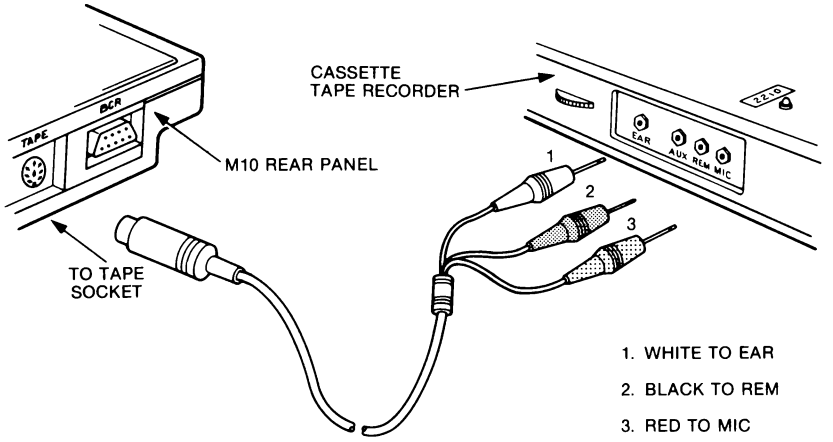

Fig. 9-1 Connecting the M10 to a Cassette Recorder

When the recorder is connected in this way, two control signals can be sent from the M10:

MOTOR ON - When this signal is sent it returns the recorder to local control and the recorder buttons once again control its operation.

MOTOR OFF - This signal disables the recorder buttons and puts the recorder fully under the control of the M10. The motor will start and stop automatically when files are being transferred.

These are both BASIC commands (see also in the Directory of BASIC Commands) The command MOTOR OFF should be given before a 'Load' or 'Save' operation.

SAVING A FILE TO CASSETTE TAPE

This can be done for either a .DO file using the TEXT function keys or for a .BA file using the BASIC function keys. In both cases, start off by setting the tape to where you want the saved file to be stored. Connect the recorder as described in the preceding section, switch it on and press the RECORD and PLAY buttons simultaneously.

SAVING A TEXT FILE TO TAPE

1. Access the TEXT file you want to save by placing the cursor over it on the main menu and pressing <ENTER>.

2. Press the function key <F3> (Save); this brings up the prompt:

   Save to:

3. Select a filename for the tape file to which it is going to be saved. This must follow the prescriptions laid down for M10 file nomenclature (see Chapter 5).

   Now type in the selected filename and press <ENTER>.

4. The recorder will start to turn automatically. When the file is saved, the recorder will stop and the prompt will disappear from the screen.

The file is now saved on the cassette under its tape filename.

SAVING A BASIC FILE TO TAPE

The procedure for BASIC files is slightly different.

1. Access the BASIC facility from the main menu.

2. Press function key <F2> (Load) and in response to the prompt **Load "** type in the RAM file name of the file to be saved and press <ENTER>.

3. Now press <F3> (Save) and in response to the prompt **Save "** which then appears, enter the command:

   CAS:tape filename <ENTER>

   where 'tape filename' is the name selected for the cassette file in which it is to be saved. This must be chosen according to the rules of nomenclature for M10 files.

The recorder will turn until the 'Save' operation is complete, then stop. The file has now been saved to tape under the given cassette filename.

LOADING A FILE FROM CASSETTE TAPE

Again the procedure is slightly different according to whether it is a TEXT or a BASIC file that is to be loaded.

LOADING A TEXT FILE FROM TAPE

1. Rewind the cassette tape to the counter position corresponding to the beginning of the file to be loaded to the M10. If you do not know this, set it to the start of the tape; the M10 will search the entire cassette for the file requested.

2. Connect the recorder to the M10 as described earlier. Switch on the recorder and press the PLAY button. The recorder will not operate until it receives the signal from the M10.

3. You may create a special TEXT file for receiving the cassette file from the recorder or you may choose an existing file to which the cassette file is to be appended.
   Press function key <F2>. This will bring up the prompt:

   Load from:

4. Type in the tape filename under which the file was originally saved and press <ENTER>.

5. The M10 will now begin to search through the cassette for the file requested, making a high-pitched sound as it does so. Each time it encounters a file which is not the one requested, it displays the message:

   SKIP: tape filename

   where 'tape filename' is the name of the file encountered.

   When the correct file has been located, the M10 displays the message:

   FOUND: tape filename

   where 'tape filename' is the name of the file requested.

When this message appears, the tape file is loaded into the TEXT file from which you called. In a 'Load' operation, the tape file does not over-write the contents of the TEXT file but is appended at the end.

LOADING A BASIC FILE FROM TAPE

1. Rewind the cassette tape to the correct position for the file to be loaded, connect the recorder and press the PLAY button, as explained in steps 1 and 2 of the preceding section.

2. From the main menu, access BASIC and press <F2>. This brings up the prompt:

   Load "

3. Type in the following command:

   CAS:tape filename <ENTER>

   where 'tape filename' is the name under which the file was originally saved on tape.

4. Again you will hear a high-pitched sound as the M10 looks for the desired file on the cassette. The message

   SKIP: tape filename

   is displayed each time it encounters an undesired file, and

   FOUND: tape filename

   is displayed when the correct file is located.

Note that in the 'Load' and 'Save' operations in BASIC, it is necessary, in response to the prompts, to specify not only the filename but also the device name, in this case CAS. This is because the BASIC program has access to other external devices and can also load from and save to RAM. The 'Load' and 'Save' operations in BASIC can also be effected by direct command, using the BASIC commands CLOAD and CSAVE. You will find further reference to these in the Directory of BASIC Commands in Part 2 of this manual.

MAINTENANCE

When using a cassette recorder regularly with the M10, it is important to follow the maker's recommendations for maintenance and cleaning. In particular, be sure that the recording heads are kept clean and demagnetized to avoid damage or "noise" on your computer files.

PART 2
M10 BASIC Language Reference Guide

10. INTRODUCTION TO BASIC

The next few chapters are entirely devoted to BASIC, Beginners All-purpose Symbolic Instruction Code, which is the high-level programming language of the M10. Different versions of BASIC exist but all descriptions and references in this manual apply to the M10 version. In Chapter 4, a short description of the BASIC Application Program was given so the user will already be familiar with certain aspects of the M10 BASIC facility such as accessing BASIC, the three modes of operation, the concept of lines, the role of the function keys etc. The following chapters deal with writing and running programs, data and arithmetic, relational and logical operators and finally gives a Directory of BASIC Commands for reference.

NOTATION

The convention given in Chapter 1 is maintained throughout the manual. However, for the purposes of BASIC syntax, it has to be somewhat expanded for this part of the manual.

The following symbols occur in BASIC syntax and should be entered as they stand.

Double quotation marks	"
Full stop	.
Comma	,
Colon	:
Semi-colon	;
Addition symbol	+
Subtraction symbol	-
Multiplication symbol	*
Division symbol	/
Exponentiation symbol	^
Back slash	\
Equals	=
Greater than	>

Less than	<
Question mark	?
Percentage sign	%
Exclamation mark	!
Number sign	#
Dollar sign	$
Parentheses	( )

11. BASIC PROGRAMS

ORGANIZATION

BASIC programs for the M10 are stored in memory. Up to 19 file names can be displayed on the menu and this determines the maximum number of programs which can be stored in memory. Programs can also be stored on cassette tapes or other peripheral units. It is good practice to keep two separately stored copies of each program; one for normal use and one as a back-up.

DOCUMENTING A PROGRAM

It is most important that programs are created in a logical way and that they are well documented. A procedure which seemed obvious at the time of writing may well be incomprehensible at a later date, unless appropriate notes are added. This can be done with the REM statement, or an apostrophe, ', can be used as an abbreviation.

REM

Remarks may use a complete line or they may be on the same line as other statements. For the latter case the remark must be the last statement on the line.

CREATING A PROGRAM

The user creates a BASIC program on the M10 by entering the instructions via the keyboard; listing the instructions to confirm that they appear correct; and saving the program on tape so that it can be used repetitively.

ENTERING A PROGRAM

Each program line must start with a line number which is typed on the keyboard.

Program lines may be entered in any order and the system re-arranges them into numerical order when the program is listed. Listing can be done at any time and is described below. The program can also be saved at any stage in its development; saving is described below. All the commands, statements and functions which can be used on the M10 are detailed in Chapter 15.

LISTING A PROGRAM

To list a program on the display the command LIST is available. The command LLIST lists the program on the printer.

LIST can be used to display one or more program lines on the screen as follows:

LIST n -m <ENTER>

where n is the first line to be listed and m is the last line to be displayed. If n is not specified all lines up to m are listed; and if m is not specified, only line n is listed. If neither n nor m are specified the whole program is listed. LIST . will list the last line referenced by BASIC.

A program which has been listed is stored in memory but the syntax has not yet been checked.

The command LLIST works in exactly the same way but the program lines are listed on the printer as well as the display.

Details of LIST and LLIST are given in Chapter 15.

SAVING A PROGRAM

A program stored in memory can be saved on a cassette or any other storage device connected to the M10. The SAVE command has a number of options available for different peripherals and these are outlined in Chapter 15.

To store the current program on cassette the 'CSAVE tape filename' command can be used, where tape filename is a name (of up to six characters) identifying the file to contain the program on the cassette. It is possible to check that the program has been transferred correctly by using the command 'CLOAD? tape filename' which compares the program on cassette with that in the memory. If there are any discrepancies the message **Verify failed** appears on the screen, and if the transfer is satisfactory the **Ok** prompt is displayed. To stop the program saving activity press <SHIFT + BREAK>. Details of the use of CSAVE and CLOAD? are given in Chapter 15.

USING A PROGRAM

To use a program it must first be loaded into memory, then executed. If it runs satisfactorily, results can be printed. If there are problems with the program which prevent its execution, e.g. syntax errors, they will be flagged by the system. The procedure for de-bugging is described below.

LOADING A PROGRAM

The LOAD command has several options so that programs stored on different peripherals can be loaded into memory. These are detailed in Chapter 15. Programs stored on cassette using CSAVE can be loaded into memory using the command 'CLOAD tape filename' where 'tape filename' is the name of the file containing the program required. If the filename is omitted, the first program on the tape is loaded into memory. To stop the loading press <SHIFT + BREAK> and use the MOTOR OFF command to stop the cassette drive.

It is possible to load and run a program from a cassette with one command and this feature is described below.

EXECUTING A PROGRAM

To execute a program use the RUN command. Typing the command

RUN <ENTER>

will attempt to execute the program last loaded into memory. If the program is not in memory and is stored on a cassette it can be loaded and then run by using

RUN "CAS:tape filename" <ENTER>

where tape filename is the name of the file containing the required program.

If there is any problem with the program an error will be indicated by the system: the procedure for de-bugging is described below.

If the program requires information from the user in the form of data, a prompt ? is displayed. Normally the prompt is preceded by a message explaining the data required. Refer to the INPUT statement in Chapter 15 for details of how this is written into a program. Type the required data and send it to the system by pressing <ENTER>.

PRINTING

Program results can be displayed by using the statement PRINT, or LPRINT may be used to output to the line printer or microplotter.

It is possible to print data in a particular format by specifying it in a PRINT USING or LPRINT USING statement. The latter is useful for printing files or the results of programs in a form different from that in which they appear on the screen. If LPRINT is used the exact form shown on the display is reproduced on the line printer.

To output data to a serial printer (connected to the RS-232C interface) the data must be written to a file. This is explained in Chapter 15.

DE-BUGGING A PROGRAM

Errors made in the writing of a program, e.g. incorrect syntax, are detected by the system when the user attempts to run the program. They are identified by line number so that the errors can be corrected.

ERRORS

A complete list of error codes is given in Appendix B. Each code has a two-character name which is displayed, and a number. The number is an integer in the range 1 to 255; many of the numbers are not allocated and they are available for system expansion in BASIC or for the user to define his own errors. In addition to the errors which can be made while writing a program there are a number of errors which may occur at other times, e.g. IO error meaning that an input or output fault occurred when trying to read or write data.

The user can define an error by using the statement ERROR n, where n is the specified error code integer. Details of this statement are given in Chapter 15.

ERROR TRAPPING

Error handling subroutines can be written into programs and the statement ON ERROR GOTO n is used to specify the first line, n, of such a routine. At any time after the ON ERROR GOTO statement any error will result in a jump (trap) to the subroutine. Error trapping will be cancelled after an ON ERROR GOTO Ø statement.

MODIFYING A PROGRAM

A program can only be modified when it is stored in the memory; if it is not currently in memory it must first be loaded, as described above.

CHANGING A PROGRAM

Program lines can be altered by retyping the complete line and pressing <ENTER>. The new line then replaces the old one in memory. A line can be deleted by typing the line number and pressing <ENTER>.

EDITING A PROGRAM

For several small changes to the text of a program it is simpler to use the editing facility. The command EDIT transfers the system to the Text Mode where editing is done. Details of the format of the command are given in Chapter 15.

Text Mode provides a screen editor which is used for creating and editing files. It has a number of commands which simplify the creation or modification of files, and, with the exception of the 'word-wrap' feature, they are available for modifying programs. The full description of this mode is given in Chapter 4.

MERGING TWO PROGRAMS

A program on cassette can be merged with another in memory by using the command

```
MERGE "CAS:filename
```

With this command the lines from the program on cassette are merged in numerical order with those of the program in memory. If a line in the cassette program has the same number as a line in memory the latter is replaced and the original line is lost.

12. DATA AND ARITHMETIC

DATA IN BASIC PROGRAMS

Within programs data items can be constant or variable. A constant has the same value throughout program execution, whereas the value of a variable may be altered. Variables are, therefore, given names which are used in the writing of the program, e.g. the formula

C + 273

gives the temperature in degrees kelvin where C is a variable representing the temperature in degrees celsius. A program using this expression can be used many times by using different values for C.

Variable names can be longer than two characters, but only the first two are recognized and they must, therefore, be unique. A constant may also be given a name, e.g. it is easier to enter a short name than a long constant. A reserved word may not be used as a name or as part of a name. Therefore it is recommended that variable names are restricted to one or two characters, so that the only reserved words which must be avoided are IF, ON, OR and TO.

Both constants and variables can be purely numeric or 'strings'. A string is a sequence of any characters except the double quotation marks ("), as these are used to define or delimit the string.

An 'expression' is a general name for an item, or a combination of items, of data. It can contain constants or variables or both, but it cannot contain a mixture of numeric and string data. The formula above is a numeric expression and the numeric variable C must be assigned a value before the expression can be evaluated.

STRING DATA

String data are of variable length and BASIC allocates memory as it is required. The length of the string can by any number of printable characters from 0 to 255. If the string contains no characters i.e. two double quotes without a space between them, it is called a 'null' string.

NUMERIC DATA

BASIC can handle numeric data of three different types depending on the precision required in calculations, viz. integer, single precision, or double precision data. All numeric data are entered in a decimal form (i.e. to the base 10).

Integers

These are most efficient in terms of the amount of memory used and the speed of computation. However, the range of values is limited to integers from -32768 to 32767.

Single Precision Numbers

These are used for general purpose applications in BASIC. The range of values is from $\pm 10^{-64}$ to $\pm 10^{62}$ and up to seven significant digits are used in computations. Up to six digits can be displayed, the least significant being rounded up if the seventh digit is greater than or equal to 5. More memory is required to store single precision numbers and calculations take longer than for integers.

Double Precision Numbers

These are the most precise form for data in BASIC. The range of values is from $\pm 10^{-64}$ to $\pm 10^{62}$ and up to 16 significant digits are used in computations. Up to 15 digits can be displayed, the least significant being rounded up if the 16th is greater than or equal to 5. Double precision numbers use the most memory and calculations are the slowest.

CLASSIFICATION OF DATA

Normally a string is recognized by the enclosing double quotes. While data is being input, or in an instruction using the DATA statement, the characters need not normally be enclosed in double quotes.

Numeric data constants are classified by the minimum precision category which is appropriate, i.e:

- if a number has more than seven digits it is a double precision number
- if a number is a whole number in the range -32768 to 32767 it is an integer
- if a number falls into neither of these categories it is a single precision number.

Numeric variables are initially classified as double precision.

CHANGING CLASSIFICATION

Both constants and variables can be given different classifications. The type of a constant can only be changed by adding a type declaration tag at the end of the data item. The four tags available are listed in Fig. 12-1.

Variables can be defined at the start of a program as a particular type by using one of the following statements:

- DEFINT ident specifies integer variables
- DEFSNG ident specifies single precision variables
- DEFDBL ident specifies double precision variables
- DEFSTR ident specifies string variables

where, in each case, ident gives the letters which will identify a particular type of variable when used as the first letter of a variable name, e.g:

```
10 DEFINT A-C, M, X-Z
```

states that all program variables starting with the letters A, B, C, M, X, Y and Z will be integers. The format of the DEF statements is detailed in Chapter 15.

It is possible to override these definitions using the type declaration tags listed in Fig. 12-1.

TAG	MEANING
%	Integer
!	Single Precision
#	Double Precision
$	String

Fig. 12-1 Type Declaration Tags

Unless the numeric variables and constants are specified as integer or single precision using DEF statements or type declaration tags, all calculations are carried out in double precision.

CONVERSIONS

Sometimes it is necessary to assign one type of constant to a different type of variable, or to convert one type of variable to another. This can be done using the form of a = b and some examples are illustrated below.

Single or double precision to integer, and double precision to single precision involve truncation or rounding, and numbers outside the range of the converted variable will produce overflow errors. Integer to single precision produces no errors.

Example 1

Convert the single precision variable A (= 23.56) to an integer. Follow the BASIC command sequence:

```
A% = 23.56 <ENTER>
Ok
?A% <ENTER>
23
Ok
```

Note that the number is truncated and not rounded.

Example 2

Convert the double precision variable A (= 5.ØØ69999999) to single precision.
Follow the BASIC command sequence:

```
A! = 5.ØØ69999999 <ENTER>
Ok
?A! <ENTER>
5.ØØ7
Ok
```

In these examples the lines containing <ENTER> are commands to be entered by the user; the other lines are the output from the BASIC program.

ARRAYS

An array is a collection of variables of the same type which are grouped together. Each item is identified by the name of the array and a subscript which is different for each element, e.g:

A(0), A(1), A(2)

are the three elements of a one dimension array, A, which has a maximum subscript of 2. This is defined as the upper bound of the dimension.

An array may have any number of dimensions providing there is sufficient memory space available. Similarly each dimension can have any number of elements providing memory space is available. The appropriate amount of memory is allocated by BASIC for an array when it is first specified. The

following are elements of a two dimension array the maximum subscripts of which are n, m.

```
B(0,0) B(0,1) B(0,2) - - - - - - - - B(0,m)

B(1,0) B(1,1) B(1,2) - - - - - - - - B(1,m)

B(2,0) B(2,1) B(2,2) - - - - - - - - B(2,m)

       |                    |

       |                    |

B(n,0) B(n,1) B(n,2) - - - - - - - - B(n,m)
```

This two dimension array has n + 1 rows and m + 1 columns. A single dimension is suitable for holding the contents of a list, and two dimension arrays are typically used for tables.

Defining Arrays

An array can be defined using the DIM statement and the maximum subscripts to be used. If the array is to hold string variables this must be specified by the use of $ when it is defined, e.g.:

```
10 DIM A$(5,5), B(12) <ENTER>
```

establishes a two dimensional string array with sub-scripts from 0,0 to 5,5 and a one dimensional array with sub-scripts from 0 to 12.

If an array element is mentioned in a program and the array has not previously been defined, one will be created with the number of dimensions required by the element and each dimension will have an upper bound of 10.

The DIM statement sets all the elements of the specified arrays to zero or null string. There is no separate command for erasing an array.

BASIC ARITHMETIC

Operations within BASIC fall into three categories, viz. relational, numeric and logical. For each category there are a number of functions available, and each function is identified by a particular symbol which is referred to as an operator.

RELATIONAL OPERATORS

The simplest operation is a comparison between one expression and another. Only like expressions can be compared, i.e. it is not possible to make comparisons between strings and numbers. The following relational operators are available for both strings and numbers. Strings must be the same length or the longer one is truncated to match the shorter one. The 'value' of a string is determined by the ASCII value of each character, with the first character being most significant.

The table in Fig. 12-2 lists the relational operators and their meanings.

OPERATOR	MEANING
=	Equal to
>	Greater than
<	Less than
= > or > =	Greater than or equal to
= < or < =	Less than or equal to
< > or > <	Not equal to

Fig. 12-2 Relational Operators

The result of a relational expression is given as -1 if it is true and Ø if it is false.

NUMERICAL OPERATORS

Numerical operations are possible only for numbers. The + symbol meaning addition can also be used with strings to join them together, but no other numerical operator may be used with strings. The table in Fig. 12-3 lists the available operators and their meanings.

OPERATOR	MEANING
+	Addition
-	Subtraction or Negation
*	Multiplication
/	Division
\	Integer Division
^	Raising to a power

Fig. 12-3 Numerical Operators

The numbers used with an operator are called operands.

For integer division the operands are rounded to the nearest integer and the result of the division is truncated to the integer part. The remainder of such a division can be expressed to the nearest integer using the MOD function.

The order of priorities for numerical operators is as follows:

1. Raising to a power

2. Negation

3. Multiplication and division

4. Integer division

5. Addition and subtraction

For operations with the same priority, the operations are carried out in the order they are entered from left to right. Priorities can be altered by using parentheses to enclose the operation required first. This can be extended by using parentheses within parentheses as illustrated in the following examples:

1. For $\frac{x + y + z}{2}$ enter (X + Y + Z)/2

2. For $2x + 5$ enter 2 * X + 5

3. For $2(x + 5)$ enter 2 * (X + 5)

4. For $(x + y)^2$ enter (X + Y) ^ 2

5. For $\frac{x^2(x + y)^4}{z^4}$ enter X ^ 2 * ((X + Y) / Z) ^ 4

Parentheses can be used even if they are not strictly necessary. This can be useful in helping to clarify an expression.

If an expression contains a variable which has not been assigned a value, the variable is set to zero.

Results of Computations

The type of data in the result of an expression is dependent on the types of operand used. An expression involving several operands can be split into a series of operations each using two operands. For example, in the expression

```
2 * (X + 5)        X + 5 is calculated first, giving Y
                   then 2 * Y is calculated.
```

The result of each individual operation takes the more precise form if the two operands are different, e.g. if one operand is double precision and the other is an integer the result will be double precision.

Operands can be expressed in scientific notation, e.g.

5.26E4 which represents 5.26×10^4

If the value of the result is bigger than the maximum permitted for the data type an overflow (OV) message is displayed. An attempted division by zero will produce the same result with the /Ø error message. This error message also appears for zero being raised to a negative power. If zero is raised to the power zero the result is 1.

If the value of the result is smaller than the minimum permitted it is displayed as zero.

LOGICAL OPERATORS

Logical operations can be performed in BASIC and the following Boolean functions are available; they are listed in their order of priority:

- NOT
- AND
- OR
- Exclusive OR (XOR)
- Imply Y (IMP)
- Equivalence (EQV)

The truth tables for these functions are given in Fig. 12-4.

	A B 0 0	A B 0 1	A B 1 0	A B 1 1
NOT A	1	1	0	0
A AND B	0	0	0	1
A OR B	0	1	1	1
A XOR B	0	1	1	0
A EQV B	1	0	0	1
A IMP B	1	1	0	1

Fig. 12-4 Logical Operators

Logical Operators can be used in conjunction with Relational Operators. Logical operations are performed after numeric and relational operations, on the binary values of the operands in the expression. If the binary values are outside the range -32768 to +32767 an error is indicated. The operation is carried out bit by bit and the result is given as a decimal value in the same range. Negative binary numbers are held in the 2's complement form. The following examples illustrate how logical operators can be used. It is only possible to use two consecutive logical operators if the second is NOT.

Example 1

In BASIC, enter the command:

```
? 4 OR 2 <ENTER>
 6
Ok
```

The first line is the command, the second the result displayed by the M10 and the third the BASIC prompt.

```
In binary, 4 = 100
           2 = 010
and        6 = 110
```

which is the result of the OR operation on each pair of bits.

Example 2

Enter the command:

```
? 7 AND 1Ø <ENTER>
 2
Ok
```

Again the first line is the command, the second the result displayed by the computer and the third the BASIC prompt.

```
In binary,  7 = 0111
           1Ø = 1010
and         2 = 0010
```

which is the result of the bit-by-bit AND operation.

Logical expressions can be used to test a condition in IF... GOTO... ELSE or IF... THEN... ELSE statements.

INPUTTING DATA

Data can be input using the LET, DATA, READ and RESTORE statements as part of the program. It is also possible to enter data when the program is being executed. This is done by using the INPUT or LINE INPUT statements for data entered at the keyboard and INPUT or LINE INPUT for data entered from a file. The format is different in the two cases (see INPUT and LINE INPUT statements in Chapter 15).

13. PROGRAMMING FEATURES

BRANCHES

Within BASIC programs the statement GOTO n causes an unconditional transfer of control (branch) to line n, changing the normal sequence of execution.

For a conditional transfer of control the following statements are available:

```
IF... THEN... ELSE
IF... GOTO... ELSE
ON... GOTO
```

All of these statements can be 'nested' (the process of introducing an activity within an activity) and details of the format of these statements are given in Chapter 15.

LOOPS

Program loops for repetitive operations can be generated using the FOR... NEXT statement. Loops can be nested in the same way as branches, the limitation being the available memory. A NEXT statement may conclude more than one loop if the inner loop terminates at the same point as the outer loop. There is no error condition for a FOR without a matching NEXT statement, but if a NEXT is used without a preceding FOR the error code NF is displayed.

SUBROUTINES

A subroutine may be used to allow the same sequence of instructions to be carried out at any point in a program. The statement GOSUB n causes the program to jump to line n and to continue with execution of the subroutine which starts at line n. The last statement of the subroutine must be RETURN which transfers program control back to the line following GOSUB n.

MACHINE LANGUAGE SUBROUTINES

A machine language subroutine can be used within a BASIC program by using the CALL statement to invoke the particular program required. A few of the subroutiness available cannot be started with a CALL statement; see CALL in the Directory of BASIC commands (Chapter 15).

14. FILES

Files can contain programs or data. BASIC program files are automatically assigned the suffix .BA, whereas TEXT files and BASIC data files have the suffix .DO. Machine language program files have the suffix .CO. All data files are in a sequential form. Normally they are created in memory; they can then be transferred to a cassette tape or another peripheral. It is possible to write a file directly onto cassette but it is not possible to append data onto a cassette file. Anything written to a cassette file overwrites the original. To read or write to a data file it must first be opened using the statement OPEN.

OPENING A FILE

The OPEN statement is used to specify the direction of data transferred. To read a file it must be opened in input mode, and to write to a file it must be opened in output mode.

Data files in memory can be opened in the append mode where anything written to the file is added to the existing file and the original is not lost. A file in memory can be transferred to cassette tape using the SAVE statement, and a file on tape can be transferred to memory using LOAD.

A number is allocated to the program or data file as part of the OPEN statement. The number allocated for a particular mode cannot be used for another mode unless the file is first closed.

READING A DATA FILE

Data from a file can be used in a program by first opening the file in input mode, and then using the INPUT statement to extract the items of data and assign them to program variables.

The EOF statement must be used after INPUT to detect the end of the file. If this is not done an EF error will be displayed when the end of the file is reached and the program will terminate in an uncontrolled way.

WRITING A DATA FILE

Data can be written to a file by first opening the file in output mode, and then using the PRINT or PRINT USING statements.

CLOSING A FILE

Some statements and commands will automatically close all open files, e.g. END. To close a file independently of these the CLOSE statement is available.

ABS Function

ABS returns the absolute (positive) value of a numeric expression.

Format: ABS(x)

where: x is an expression.

Example:

```
Ok
PRINT ABS(6*(-3))
 18
Ok
```

ASC Function

ASC returns the ASCII code (in decimal) for the first character of a string expression.

Format: ASC(x$)

where: x$ is a string expression.

If x$ is null an FC (illegal function call) error is displayed. See also function CHR$ to convert an ASCII code to a string.

Example:

```
Ok
10 X$ = "M10"
20 PRINT ASC(X$)
RUN
 77
Ok
```

ATN Function

ATN returns the arctangent of a number.

Format: ATN(x)

where: x is a number or numeric expression of any type.

The result is given in radians in the range - $\pi/2$ to + $\pi/2$.

BEEP Statement

BEEP generates a sound from the buzzer for approximately half a second.

Format: BEEP

Example:

```
IF X = 0 THEN BEEP
```

This statement can also be used in a FOR/NEXT loop to lengthen the sound, or to signal the end of a long program in a GOTO loop.

Example:

```
 99Ø  BEEP
1ØØØ  GOTO 99Ø
```

See also the SOUND statement.

CALL Statement

CALL invokes a machine language subroutine.

Format: CALL address, A ,HL

where: address is the starting address in ROM of the

machine language subroutine

A is an integer to be input to register A

HL is an integer to be input to register HL.

The starting address must be in the range -23758 to 65535. The integers for the registers must be in the range 0 to 255.

Example:

```
CALL 3Ø373
Ok
```

This subroutine sounds the buzzer for half a second (this example is valid for the M10 MODEM only).

CDBL Function

CDBL converts a number into double precision.

Format: CDBL(x)

where: x is an integer or single precision number.

Chapter 12 gives details of how conversions are made. See also CINT and CSNG for converting to integer or single precision numbers.

CHR$ Function

CHR$ returns an ASCII character for an integer in the range 0 to 255.

Format: CHR$(i)

where: i is an integer or integer expression in the range 0 to 255.

Example:

```
Ok
PRINT CHR$(90)
Z
Ok
```

CINT Function

CINT converts a number into an integer by removing all digits after the decimal point.

Format: CINT(x)

where: x is a single or double precision number or expression.

The number is truncated and digits after the decimal point are lost. If the result is not within the integer range of -32768 to 32767 an OV (overflow) error is displayed. Chapter 12 gives details of how conversions are made. See also CDBL and CSNG for converting to double and single precision numbers.

CLEAR Command

CLEAR resets memory space used for data.

Format: CLEAR string space ,highest location

where: string space is the number of bytes to be available for string variables

highest location is the highest memory location available for BASIC. If MAXRAM is specified the whole of the memory is available.

The default value for string space is 256 bytes. CLEAR sets all numeric variables to Ø and all string variables to null. It also closes all open files.

NOTE that the string space is only used by the system if no other space is available for strings, e.g. in a program string space will automatically be allocated to string variables and the memory space set by CLEAR will not be required.

CLOAD and CLOADM are described with the LOAD command. CLOAD? is used to compare a BASIC program on cassette with one in memory.

Format: CLOAD? "filename"

where: filename is the name under which the program was saved.

This command checks that a program has been saved correctly. CLOAD? can be used only for programs which have been saved in binary format.

The program on tape is compared with that in RAM, byte for byte. If there are any differences, the data transfer to tape was not satisfactory and the message **Verify failed** is displayed. In that case the program must be saved again.

CLOSE Statement

CLOSE concludes access to a peripheral device or a file.

Format: CLOSE file number 1 ,file number 2 ...

where:

file number is the number under which the file was opened (see OPEN statement). Any number of files may be closed at the same time.

All peripheral devices in the system will be closed when a CLOSE statement is executed. All files will automatically be closed under the following conditions:

- if an END statement and a NEW command are executed
- if a RUN statement and a LOAD command are executed
- if the program is changed
- if CLOSE is specified without an operand.

CLS Command

CLS clears the screen.

Format: CLS

The complete LCD screen is cleared unless the function key labels are listed on the bottom line. In that case the top seven lines only are cleared.

COM ON/OFF/STOP Statements

COM ON/OFF/STOP enables and disables communication via the RS-232C interface.

Format:
```
COM ON
COM OFF
COM STOP
```

COM ON enables trapping to the communications interface. If the ON COM GOSUB statement is used to refer to a subroutine, BASIC checks the communications buffer at each line in the program. If any new characters have been received from the communications line, the program traps to the subroutine.

COM OFF disables the subroutine.

COM STOP inhibits trapping to the subroutine until a COM ON is encountered. If any characters are received while COM STOP is active, the subroutine is invoked as soon as COM ON is executed.

Refer to ON ... GOSUB for related information.

CONT Command

CONT is used to continue program execution after it has been stopped by the BREAK key or by a STOP statement.

Format: CONT

This command is used mainly as a de-bugging aid. If the program is modified during a halt in execution an attempt to continue will cause a CN (cannot continue) error to be displayed.

COS Function

COS returns the cosine of an angle.

Format: COS(x)

where: x is the angle in radians.

If x is known in degrees multiply by Ø.Ø1745329 to convert to radians.

Example:

```
PRINT COS(60*.Ø1745329)
 .5ØØØØØ13Ø94ØØ4
Ok
```

CSAVE , CSAVEM Commands

See SAVE command.

CSNG Function

CSNG converts a number into single precision.

Format: CSNG(x)

where: x is an integer or double precision number.

Chapter 12 gives details of how conversions are made. See also CDBL and CINT for converting to double precision or integer numbers.

CSRLIN Function

CSRLIN returns the vertical position of the cursor.

Format: CSRLIN

This function can be called for the LCD or for the VDU if one is available. For the LCD the result will be in the range Ø to 7, and for the VDU the result will be in the range Ø to 24, with Ø representing the top line of the display.

DATA Statement

DATA stores the numeric and string constants which are accessed by the READ statement.

Format: DATA constant-1 ,constant-2 ...

where: constant is a numeric or string constant.

Any number of constants may be specified, up to the length of the program line. String constants need not be enclosed in inverted commas unless they contain commas, colons, or significant leading or trailing spaces. See also the READ statement.

Example:

```
Ok
1Ø READ A,B$,C,D$
2Ø PRINT A,B$,C,D$
3Ø DATA 56,AB,-3.55E+20,6abc7
RUN 1Ø
 56            AB
-3.55E+20      6abc7
Ok
```

DATE$ Statement and Function

DATE$ sets the current date on the clock, or returns the current date.

Format: DATE$ = "dd/mm/yy" for the M10
DATE$ = "mm/dd/yy" for the M10 MODEM
DATE$

where: dd is the day, and must be in the range Ø1 to 31.

mm is the month, and must be in the range Ø1 to 12.

yy is the year, and must be in the range Ø1 to 99.

If DATE$ is used as a function (e.g. PRINT DATE$) the current date is returned.

At cold start the system sets 1st Jan 1900 on the menu.

DAY$ Statement and Function

DAY$ sets the current day on the clock, or returns the current day.

Format: DAY$ = "xxx"
DAY$

where: xxx is one of the following: Mon, Tue, Wed, Thu, Fri, Sat, Sun.

If DAY$ is used as a function (e.g. PRINT DAY$) the current day is returned.

At cold start the system sets Sun on the menu.

DIM Statement

DIM is used to specify the number of dimensions and the number of elements in each dimension of an array.

Format: DIM x(dim1 ,dim2 ...,dimn) ,y(dim1 ,dim2 ...,dimn) ,z(dim1 ,dim2 ...,dimn)

where: x,y,z are numeric or string variables.

dim1 dim2 etc. are integers specifying the number of elements in each dimension

Note that the element count starts from 0, so an array A(1) is a one-dimension array of two elements etc. Arrays may have any number of dimensions and each dimension may have any number of elements providing there is sufficient memory space available. If reference is made to an array without a DIM statement the number of elements in each dimension is assumed to be 10.

The DIM statement sets all the elements of the specified arrays to an initial value of zero or null string.

Example:

```
10 DIM A(5),B(2,3),C$(6,7)
```

This sets up a one-dimension array of 6 elements, a two-dimension array of 3 and 4 elements, and a two-dimension string array of 7 and 8 elements.

EDIT Command

EDIT changes the M10 from BASIC to TEXT Mode so that the current program can be edited. To return to BASIC press <F8>.

Format: EDIT n -m

where: n is the first line number to be edited

m is the last line number to be edited.

If no line numbers are specified the whole program is available for editing. All the features of the TEXT program are available via EDIT except that the 'word-wrap' feature is not active. TEXT is described in Chapter 5 of this manual.

END terminates program execution, closes all open files and returns the M10 to Direct Mode.

Format: END

This statement does not cause a BREAK message to be displayed; it may be placed anywhere in a program to terminate execution at a point other than at the last instruction. An END statement as the last instruction in a program is optional.

Example:

```
10 INPUT A,B
20 GOSUB 100
.
.
90 END
100 C = SQR(A*A + B*B)
110 IF A>B GOTO 120 ELSE GOTO 140
120 D = SQR(A*A - B*B)
130 RETURN
140 D = SQR(B*B - A*A)
150 RETURN
```

EOF Function

EOF tests for the end of a file.

Format: EOF(n)

where: file number specified in the OPEN statement.

The EOF function returns -1 if the end of the specified file has been reached, and 0 if not. It should be used to test for the end of a file to avoid the system displaying EF, end of file error. It can be used for any sequential file or for data transferred via the modem or RS-232C interfaces.

Example:

```
10 OPEN"RAM:DATA" FOR INPUT AS 1
20 IF EOF(1) GOTO 100
30 INPUT  1 A$,B
 .
 .
90 GOTO 20
100 CLOSE 1
```

ERL/ERR Functions

ERL and ERR give the line number and error number of an error in an error handling sub-routine.

Format: IF ERL = line-number THEN ...
IF ERR = error-number THEN ...

where: line-number is the line in which the last error occurred.

error-number is the number of the last error; see Appendix B for a complete list of errors with their codes and numbers.

These functions are normally used in IF ... THEN statements to direct program flow in an error trapping routine. For an error in an immediate line ERL will contain the number 65535: to test for this use

IF 65535 = ERL THEN ...

For further information on error handling see ERROR.

ERROR Statement

ERROR is used to simulate an error condition or to define an error.

Format: ERROR n

where: n is an integer between Ø and 255.

If n is the number of an existing error an error message will be displayed.

Example:

```
ERROR 5
?FC ERROR
Ok
```

Error handling subroutines are called using the ON ERROR GOTO n statement where n is the first line of such a routine. To cancel error trapping use ON ERROR GOTO Ø.

To define an error code, select a number which is not currently in use. It is advisable to use numbers at the high end of the range to allow for expansion in the error codes allocated to BASIC. If there is no error

trap routine for the ERROR statement, or if no message has been defined the 'unprintable error' message, UE, is displayed.

Example:

```
40 ON ERROR GOTO 1000
 .
 .
90 INPUT A
100 IF A < 0 THEN ERROR 250
 .
 .
1000 IF ERR = 250 THEN PRINT "Must be positive"
 .
```

EXP Function

EXP returns an exponential function.

Format: EXP(x)

where: x is a number which must be in the range ±87.3365. If this condition is not met an OV (overflow) message is displayed.

FILES Command

FILES displays the program and data files currently stored in RAM.

Format: FILES

Each file name displayed has a suffix which classifies it as follows:

- .BA is a BASIC program file
- .CO is a machine language program file
- .DO is a data file created using TEXT or a BASIC program stored in ASCII format.

The files in RAM are also displayed by pressing the function key <F1>.

FIX Function

FIX returns an integer, truncating the part of a number after the decimal point.

Format: FIX(x)

where: x is any number in the permitted range.

Example:

```
Ok
FIX(3.9)
 3
Ok
FIX(-3.9)
-3
Ok
```

Note that INT(-3.9) or INT(-3.2) would return -4.

FOR...NEXT Statements

FOR and NEXT statements allow a series of instructions to be performed in a loop a given number of times.

Format:

```
FOR a = n TO p  STEP s
  .
  .
NEXT  z,y  ...,  a
```

where: a,y,z are variables

n,p,s are numeric expressions.

The variable a is a counter which has an initial value of n and it is incremented by s for each cycle of the loop. The program lines between the FOR and NEXT statements are executed repetitively until a = p + s. The program continues with the line following the NEXT statement. This means that the loop is performed p times. If s is not specified it is assumed to be 1. Negative steps may be used providing the value of n is greater than the value of p. If an inconsistent set of values is chosen, e.g. if s is greater than the difference between n and p, the loop will be performed once. If s is set to Ø the loop will run continuously until it is interrupted.

FOR...NEXT loops may be nested one inside another. The only limitation on the number of nested loops is the amount of memory available. Each

loop has to be completed with a NEXT and the innermost loop must be terminated first. If the loops end at the same point it is possible to combine the different variables in one NEXT statement with the variable from the innermost loop being written first. If a NEXT statement is found without a variable it is assumed to apply to the last mentioned FOR statement. If a NEXT statement is encountered before a FOR statement an NF (next without for) error is displayed. Therefore a branch into a FOR NEXT loop will cause an error but a branch out of the loop will not.

Example:

```
10 INPUT J
20 FOR I = 1 TO J
30 PRINT I;
40 FOR K = 1 TO 3
50 PRINT "a";
60 NEXT K ,I
80 GOTO 10
RUN
? 4
 1 aaa 2 aaa 3 aaa 4 aaa?
```

FRE Function

FRE returns the number of bytes of memory available for the user.

Format: FRE(x)
FRE(x$)

where: x is a dummy numeric argument; normally 0 is used.

x$ is a dummy string argument; normally a null string, "", is used.

If FRE is used with a numeric argument BASIC returns the number of bytes in memory which are available for a BASIC program, a text file, a machine language program file, etc. If FRE is used with a string argument it returns the amount of memory available for strings. String space is set by the previous CLEAR statement. The space allocated is not always required as programs frequently allocate space for string variables.

Example:

```
Ok
PRINT FRE("")
 200
Ok
```

In this example available string space is 200 bytes.

GOSUB and RETURN Statements

GOSUB and RETURN statements are used to transfer program control to a subroutine and, on completion of the subroutine to transfer control back to the main program.

Format:
```
GOSUB n
  .
RETURN
```

where: n is the line number which starts the subroutine being called.

GOSUB causes the program to branch to the specified subroutine and RETURN transfers program control to the line immediately after the GOSUB statement. A subroutine may contain more than one RETURN if required.

A subroutine should be preceded by a STOP, END or GOTO statement to prevent unprogrammed entry to it, other than by a GOSUB statement. If a RETURN statement is encountered without a previous GOSUB an RG (RETURN without GOSUB) error is displayed. Branching into a subroutine will cause an error but branching out will not.

Sub-routines may be nested, one within another. Nesting is only limited by the amount of available memory.

Example:
```
100 GOSUB 200
110 PRINT "Back from outer subroutine"
 .
190 END
200 PRINT "Executing outer subroutine"
 .
220 GOSUB 300
230 PRINT "Back from inner subroutine"
 .
250 RETURN
 .
300 PRINT "Executing inner subroutine"
 .
320 RETURN
RUN
Executing outer subroutine
Executing inner subroutine
Back from inner subroutine
Back from outer subroutine
Ok
```

See also ON ... GOSUB

GOTO is used to cause an unconditional branch to another line, out of the normal program sequence.

Format: GOTO n

where: n is a line number.

This statement can be used in an immediate line to start a program at a particular line. Normally it is used in programs to transfer program control to another line.

Example:

```
35Ø GOTO 1Ø
```

will make the program jump from line 350 to line 10. For conditional branches refer to IF ... THEN ... ELSE, and ON ... GOTO statements.

HIMEM Function

HIMEM returns the maximum address of memory available to the user.

Format: HIMEM

The value of HIMEM is set by the latest CLEAR statement. If nothing has previously been set, 62960 (MAXRAM) is returned.

Example:

```
Ok
PRINT HIMEM
 40960
Ok
```

IF .. GOTO/THEN .. ELSE Statements

IF .. GOTO .. ELSE or IF .. THEN .. ELSE statements are used to make a conditional branch in a program.

Format:
```
IF x GOTO n ELSE yyy
IF x THEN yyy ELSE zzz
```

where: x is relational expression

yyy and zzz are clauses which may contain statements and line numbers.

If the expression x is true, program control branches to the GOTO or THEN part of the statement. If x is not true, program control branches to the ELSE part of the statement.

Example:

```
.
11Ø IF A > B GOTO 12Ø ELSE GOTO 13Ø
12Ø D = SQR(A*A - B*B)
13Ø D = SQR(B*B - A*A)
.
```

IF .. statements may be nested. The limitation is the length of the line since IF .. GOTO/THEN .. ELSE is one statement and it must be completed on one program line.

Example:
```
5Ø IF A > Ø THEN PRINT "POS" ELSE IF A = Ø
THEN PRINT "EQUAL" ELSE IF A < Ø THEN PRINT "NEG"
```

If there are not the same number of THEN and ELSE items in a statement each ELSE is paired with the closest unmatched THEN. This can result in an incomplete statement.

Example:
```
7Ø IF A = B THEN IF B = C THEN PRINT "A = C" ELSE PRINT "?"
```

will not print ? when A ≠ B, but it will print ? when A = B and B ≠ C.

INKEY$ returns either: a string of one character equivalent to a key from the keyboard; or a null string if no keys are depressed.

Format: INKEY$

Any key will return a character to the program except <CTRL> + c which will terminate the program. This function is often used to stop a program until a key is pressed.

Example:

```
.
11Ø PRINT "Press any key to continue"
12Ø A$ = INKEY$
13Ø IF A$ = "" GOTO 12Ø ELSE GOTO 14Ø
14Ø
```

INP Function

INP returns the byte at a specific port.

Format: INP(i)

where: i is an integer in the range Ø to 255 which identifies the port.

INP is the complementary function of OUT.

INPUT Statement

INPUT is used to input data to a program from the keyboard.

Format: INPUT "prompt"; a ,b ... ,n

where: prompt is displayed to explain to the user the type of data required. It is followed by a question mark from the system: if prompt is not specified the ? is displayed on its own.

a,b,n are variables which may be numeric or string.

The data entered must correspond to the type of variable called for, or a message **Redo from start** is displayed. For an INPUT statement requiring more than one value, each data item entered must be separated from the next by a comma. If insufficient items are entered the M10 displays ?? .

Example:

```
Ok
1Ø INPUT "What is the day and the date"; A$,B
2Ø PRINT "Today is ";A$;B
RUN
What is the day and the date? Sat,1 (data input by user)
Today is Sat 1
Ok
```

INPUT Statement

INPUT is used to input data to a program from a file.

Format: INPUT file-number,a ,b ... ,n

where: file-number is the number under which the file was opened See OPEN.
a,b,n are variables which may be numeric or string.

The data items in the file must correspond to the list of variables. Files may be in RAM, on cassette or from peripherals connected to the communications or modem interfaces

For numeric or string values, leading spaces, carriage return and line feeds are ignored. The first character which is not a space, carriage return or line feed is treated as the start of a number or string. A number is completed on a space, carriage return, line feed or comma. For

strings, if the first character is " the string will consist of all characters between this quotation mark and the next. If the first character is not a quotation mark all characters are part of the string and it will terminate on a carriage return, line feed, comma or after 255 characters.

INPUT$ Function

INPUT$ returns a string of characters from the keyboard or a file.

Format: INPUT$ (n , file-number)

where: n is the number of characters to be read.

file-number is the number under which the file was opened.

The file may be in RAM, on cassette or read from peripherals connected to the communications or modem interfaces. If no file-number is specified the characters are read from the keyboard.

INSTR Function

INSTR returns the position of the first occurrence of a particular string within another string.

Format: INSTR(i, x$,y$)

where: i is an integer up to 255 which represents an offset. If this is specified i characters are skipped before the search is made for the required string.

x$ is the string being searched.

y$ is the string being sought.

INSTR will return a value of Ø for any of the following conditions:

- if i is longer than x$
- if y$ cannot be found

- if x$ is null

- if x$ and y$ are null.

If only y$ is null INSTR returns i or 1. The i characters excluded in the search are included in the count.

Example:

```
Ok
10 A$ = "Olivetti M10 Portable Computer"
20 B$ = "M10"
30 PRINT INSTR(8,A$,B$)
RUN
 10
Ok
```

INT Function

INT returns the largest integer less than or equal to a given number.

Format: INT(x)

where: x is a number of any type.

Example:

```
Ok
?INT(3.9)
 3
Ok
?INT(-3.9)
-4
Ok
```

Note that FIX(-3.9) would return -3.

IPL is used to run a program immediately after switch-on (warm start only).

Format: IPL "file-name"

where: file-name is the complete name of the file.

To cancel a previously set IPL use the same command with no argument.

Example:
```
IPL "BASIC"
IPL "TEST.BA"
```

This causes the M10 to start in BASIC after displaying the menu.

KEY Statement

KEY is used to allocate a string to a particular function key. KEY LIST lists the contents of the function keys and KEY ON/OFF/STOP controls trapping to a sub-routine.

Format:
```
KEY n, xxx
KEY LIST
KEY (n) ON
        OFF
        STOP
```

where: n is a function key number which must be from 1 to 8.

xxx is a string expression up to 15 characters long

The function keys F1 to F8 are set initially to the values shown in the example below. The user can allocate different functions using KEY n, xxx.

Example:
```
KEY LIST
Files               Load "
Save "              Run
List
                    Menu
Ok
```

The listing is in the form:

F1	F2
F3	F4
F5	F6
F7	F8

KEY(n) ON activates trapping to a sub-routine when key n is pressed. The sub-routine must be specified in an ON KEY GOSUB statement (see ON ... GOSUB).

KEY(n) OFF disables the sub-routine.

KEY(n) STOP inhibits trapping to the sub-routine until a KEY(n) ON is encountered. If a key is pressed while KEY(n) STOP is active the sub-routine is invoked as soon as KEY(n) ON is executed.

KILL Command

KILL is used to delete a file from RAM.

Format: KILL "xxxxx.yy"

where: xxxxx is the file name

yy is the file extension, i.e:

- .BA for a BASIC program file
- .CO for a machine language program file
- .DO for a TEXT file.

The full file specification must be given and the current file cannot be deleted.

LCOPY Statement

LCOPY lists the current text image on the screen at the line printer.

Format: LCOPY

The PRINT function key also lists the current screen image at the line printer.

LEFT$ returns a string of characters starting from the left of a given string.

Format: LEFT$(x$,i)

where: x$ is a string

i is an integer from Ø to 255.

The first i characters of the string x$ are returned. If i is longer than x$ the complete string is returned and if i is zero a null string is returned.

LEN Function

LEN returns the length of a string.

Format: LEN(x$)

where: x$ is the string.

The number of characters in x$ is returned, including spaces and unprintable characters.

LET Statement

LET assigns a value to a variable.

Format: LET x = a

where: x is a variable

a is an expression of the same type as x.

A string expression cannot be assigned to a numeric variable and vice versa. The statement LET is optional: the same result is achieved with x = a.

LINE Command

LINE draws a line on the LCD or it can be used to specify a rectangle.

Format: LINE (x1,y1) - (x2,y2) ,c
LINE (x1,y1) - (x2,y2) ,c,B F

where: x1,y1 and x2,y2 are the co-ordinates of the starting and finishing points of the line; x must be in the range Ø (left hand side of LCD) to 239 (right hand side of LCD); and y must be in the range Ø (top) to 63 (bottom).

c is the colour parameter; if c = 1 a line or box is drawn; if c = Ø a line or box is erased.

B draws a box (the sides of which are parallel with the top and sides of the LCD display) with the specified line as a diagonal.

F fills the box.

Any co-ordinates which are out of range will produce an FC (illegal function call) error.

If the first point co-ordinates are not specified the point is assumed to be the last specified by LINE, PRESET or PSET. If there have been no previous points specified the system assumes (0,0).

Note that c is mandatory with the box option, but may be omitted for a line in which case it is assumed to be 1.

LINE INPUT Statement

LINE INPUT is used to input a complete line of up to 254 characters to a string variable without the need for inverted commas as delimiters.

Format: LINE INPUT "prompt string"; a$

where: prompt string is displayed before the string is accepted. No ? is displayed unless it forms part of the prompt string.

a$ is the string variable.

All characters input from the keyboard in response to the prompt string are assigned to the string variable until <ENTER> is pressed to end the

line. If a line feed then carriage return sequence is used, the line feed is part of the string variable, the carriage return is ignored and character entry continues.

To abort a LINE INPUT press <CTRL> + c or <SHIFT + BREAK>. BASIC then returns to the Direct Mode and **Ok** is displayed. Then type CONT and the program continues from the LINE INPUT statement.

LINE INPUT Statement

LINE INPUT is used to input a complete line of up to 254 characters to a string variable, from a sequential file.

Format: LINE INPUT file-number, a$

where: file-number is the number under which the file was opened using an OPEN statement.

a$ is the string variable.

This statement reads all the characters in the file up to a carriage return. The carriage return/line feed sequence is ignored and the next LINE INPUT statement reads the characters up to the next carriage return.

If a line feed and then a carriage return are present in the sequence of characters they are preserved as part of the character string.

This statement can be used with files in RAM or on cassette, and with sequential data from a buffer which has been received via the modem or communications interfaces.

LIST Command

LIST displays part or all of the current program.

Format: LIST n - m

where: n is the first line number to be listed

m is the last line number to be listed.

If no line numbers are specified the whole of the program is listed. If n is omitted the whole program up to line m is listed. If m is omitted the program is listed from line n to the end. If both n and m are specified the program is listed from line n to line m.

To terminate a listing press <CTRL> + c or <SHIFT + BREAK>.

To interrupt a listing press <CTRL> + s or <PAUSE>; to resume listing press <CTRL> + q or <PAUSE>.

LLIST Command

LLIST displays part or all of the current program on the printer. The format of the command is the same as for LIST.

LOAD , LOADM , CLOAD , CLOADM Commands

LOAD, LOADM, CLOAD and CLOADM are used to load programs into the current program area of memory, but the programs are not registered in the directory. To register a program in the directory it must be saved using the appropriate version of the SAVE command.

Format:
```
CLOAD  "file-name"  ,R
LOAD "CAS: file-name "  ,R
LOAD " RAM: file-name"  ,R
LOAD "COM:cbpsx"  ,R
LOAD "MDM:bpsx"  ,R for M10 MODEM only
CLOADM  "file-name"
LOADM "CAS: file-name "
LOADM " RAM: file-name"
```

where: CLOAD and CLOADM load programs from cassette.

file-name is the name used when the program was saved.

R runs the program after it has been loaded

CAS loads the program file from cassette

RAM loads the program file from RAM

COM loads the program from the communications line; for parameters see below

MDM loads the program received by the internal modem; for parameters see below.

CLOAD and LOAD load BASIC programs into the current program area of memory, deleting any previously held program. CLOADM and LOADM load machine language programs which are stored at the location specified when the program was saved.

Normally a LOAD command closes all files. The R option runs the program after it is loaded and keeps data files open. By using LOAD with the R option it is possible to add one program to another. See also the RUN command.

The CLOAD and LOAD "CAS: commands have the same function. Either loads the specified program file from a cassette. If the file name is omitted the next BASIC program file on the tape is loaded. To stop loading from tape press <SHIFT + BREAK>. Similarly CLOADM and LOADM "CAS: have the same function. Normally it is possible to hear the program data from the cassette but if the SOUND OFF command is used before a command to load a program this will be avoided.

LOAD "RAM: is similar to LOAD "CAS: but RAM: may be omitted.

Programs can be loaded from the modem or communications line but for this the receiving conditions must be specified using the communications parameters. Details of the communications parameters are given in Chapter 8, Fig. 8-5.

LOG Function

LOG returns the natural logarithm of a given number.

Format: LOG(x)

where: x is a numeric expression which must be greater than zero.

Example:

```
PRINT LOG(1Ø)
 2.3Ø2585Ø92994
Ok
```

LPOS Function

LPOS returns the current position of the line printer print head within the printer buffer.

Format: LPOS(x)

where: x is a dummy argument which is normally Ø.

This function does not necessarily return the actual position of the print head.

LPRINT , LPRINT USING Statements

LPRINT and LPRINT USING print data at the line printer. The format is the same as for the PRINT and PRINT USING statements.

If a microplotter is being used instead of a line printer the LPRINT statement is also used in conjunction with CHR$ to change the mode of the plotter. Details of these and other statements available are given under MICROPLOTTER Commands.

MAXFILES Function

MAXFILES is used to specify the maximum number of files which may be opened, or to return the previously set maximum.

Format: MAXFILES = n
MAXFILES

where: n is an integer from 0 to 15.

If MAXFILES = n is not used to set the maximum number of files, the system assumes a default value of 0.

MAXFILES will return the number set by MAXFILES = n.

Example:

```
MAXFILES = 10
Ok
PRINT MAXFILES
 10
Ok
```

MAXRAM returns the highest address of memory available for BASIC.

Format: MAXRAM

This function would normally be used to establish the maximum possible value of the second argument in a CLEAR statement.

MDM ON/OFF/STOP Statements

M10 MODEM ONLY

The MDM ON, OFF and STOP statements are used to enable and disable trapping to a subroutine which is called if characters are received at the internal modem interface.

Format:
```
MDM ON
MDM OFF
MDM STOP
```

MDM ON enables trapping to the modem interface. If the ON MDM GOSUB statement is used to refer to a sub-routine, BASIC checks each instruction; if any new characters are present at the modem interface the program traps to the sub-routine.

MDM OFF disables the sub-routine.

MDM STOP inhibits trapping to the sub-routine until a COM ON is encountered. If any characters are received while MDM STOP is active the subroutine is invoked as soon as MDM ON is executed.

Refer to ON ... GOSUB for related information.

MENU Command

MENU exits BASIC and returns to the menu.

Format: MENU

The function key F8 also causes a return to the menu if it is pressed while the system is in BASIC.

MERGE Command

MERGE merges the lines from a program stored in ASCII format (or from the communications or modem interfaces), with the program currently stored in memory.

Format:
```
MERGE "CAS: file-name "
MERGE " RAM: file-name"
MERGE "COM:cbpsx"
MERGE "MDM:bpsx"   for M10 MODEM only
```

where: file-name is the name under which the file was saved.

CAS merges a program from cassette

RAM merges a program from RAM

COM merges a program from the communications interface.

MDM merges a program from the internal modem interface.

Lines from the external or RAM program are merged with the program lines currently in memory. If any line numbers are duplicated those in the current program are replaced by the corresponding lines from the program being merged.

When the merge is complete the system returns to the Direct Mode and the current program is the result of the merge.

The BASIC statement for printing at the Microplotter is LPRINT. There are some ASCII codes which have been allocated as instructions and these are detailed below. To send them the BASIC function CHR$ must be used.

Example: LPRINT CHR(Ø8)

causes the ASCII code for a backspace (Ø8) to be sent to the Microplotter. If the same form is used for an ASCII code for a printable character, the character will be printed.

Example: LPRINT CHR(9Ø)

when the letter Z (code 90) will be printed.

The Microplotter has two modes of operation - Text Mode (automatically entered at switch-on) and Graphic Mode. The mode can be changed from Text to Graphic by sending code 18; or from Graphic to Text by sending code 17. In Graphic Mode a number of special commands are available which are detailed below. These must be sent as a quoted string.

Example: LPRINT "A"

which returns the Microplotter to Text Mode.

TEXT MODE COMMANDS

CHR$(Ø8) Backspace: this backspaces the pen one character at a time. It is useful for underlining.

CHR$(11) Reverse line feed: this moves the paper back one line at a time. It is useful for superscripts.

CHR$(18) Select Graphic Mode.

GRAPHIC MODE COMMANDS

For these commands if x and y co-ordinates are specified the x-axis is horizontal and the y-axis is vertical. Remember that the distance across the paper is only 480 units; the Microplotter will accept the same range of values for x and y but values above 480 for x are treated as 480.

CHR$(17) Select Text Mode.

A Return to Text Mode: this returns to Text Mode and moves the pen to the left margin.

Cn Change colour: the colour is changed according to n which can be Ø (black), 1 (red), 2 (green), or 3 (blue). The colours specified apply only if the pens have been installed in the recommended positions.

Dx1,y1,x2,y2,x3,y3 ...

Draw: this draws a line from the current pen position to the co-ordinates specified by x1,y1 with respect to the origin (unless otherwise specified by I, the origin is at the left hand edge of the paper). Then the line is drawn from x1,y1 to x2,y2 etc. The values of x and y can be from Ø to 999. The following example draws a small square providing the pen is at position Ø,Ø at the start:
DØ,25,25,25,25,Ø,Ø,Ø

H

Home: this moves the pen to the origin without drawing a line.

I

Initialize: this sets the origin to the current pen position.

Jx1,y1,x2,y2,x3,y3, ...

Draw relative: this acts like D but the co-ordinates are in each case referred to the current pen position, e.g:
JØ,25,25,Ø,Ø,-25,-25,Ø draws a small square. The values of x and y must be from -999 to 999.

Mx,y

Move: this moves the pen without drawing a line to the position specified by the co-ordinates x,y, with respect to the origin.

Pstring

Print: this prints the alphanumeric characters specified in string.

S size

Size: this changes the size of characters printed with P. The value of size can be from Ø (80 characters per line) to 63 (1 character per line). The default value of string is Ø.

Q n

Print direction: this specifies the direction of printing according to n. The value of n can be Ø (left to right), 1 (top to bottom), 2 (right to left upside down), 3 (bottom to top). The default value for n is Ø and at switch-on the Microplotter starts with QØ.

Rx,y

Move relative: this moves the pen without drawing a line to the position x,y with respect to the current position. The values of x and y must be in the range -999 to 999.

Xaxis,step,interval

Draw axis: this draws the x or y axis for a graph. The axis is specified by axis which must be 1 (x) or Ø (y). The distance between measurement marks on the axis is given by step which must be from -999 to 999, and the number of times that step must be repeated is given by interval which must be from 1 to 255.

N

To print an user's pattern with the current position as the home position.

Up1,x1,y1,p2,x2,y2, ... pn,xn,yn,Ø,Ø,Ø ($-7 \leqq x \leqq 7$, $-7 \leqq y \leqq 7$, $p = Ø$ or 1, $n \leqq 15$)

Define user's pattern: this command will make special characters, and style of marks, by step resolution. User's patterns that were memorised using the N command can also be drawn.
p: Ø (move) 1 (draw)
x,y: designated the step by step position of pattern coordinates.

MID$ returns a string from within another string.

Format: MID$(x$,i ,j)

where: x$ is a string expression

i and j are integers from 1 to 255

A string of j characters is returned from the string x$, starting at the ith character. If j is omitted, or if it is larger than the number of characters remaining in the string, then all the characters to the right of the ith will be returned. If i is greater than the number of characters in x$, a null string is returned.

Example:

```
10 A$ = "Olivetti M10"
20 B$ = MID$(A$,7,4)
30 PRINT B$
RUN
ti M
Ok
```

MOD Function

The MOD function returns the remainder of an integer division.

Format: n MOD m

where: n is the dividend and m the divisor, both integers.

For example, 10 MOD 3 returns the value 1.

MOTOR ON/OFF Statements

MOTOR ON and MOTOR OFF control the cassette tape recorder motor from a program.

Format: MOTOR ON
MOTOR OFF

These statements enable or inhibit the action of the cassette recorder switches. They should be used to ensure that the cassette recorder motor only runs when data is being transferred.

NAME Command

NAME is used to rename a file.

Format: NAME "aaa.xx" AS "bbb.xx"

where: aaa is the old file name

bbb is the new file name

xx is the file extension which must be the same for both names.

If the old file name does not exist, or if the new file name is already in use, an FC (illegal function call) error will be displayed. If the file extensions are omitted an FF (file not found) error will be displayed.

NEW Command

NEW deletes the current program from working memory and resets all variables.

Format: NEW

All numeric variables are set to 0 and all string variables are set to null. The string space allocated by a previous CLEAR is not affected.

ON ... GOSUB causes branching to a subroutine.

Format:
```
ON COM GOSUB n
ON MDM GOSUB n      for M10 MODEM only
ON TIME$ = "hh:mm:ss" GOSUB n
ON KEY GOSUB a ,b,c ...
ON x GOSUB a ,b,c ...
```

where: a,b,c,n are program line numbers

COM causes a trap to line n if there is data in the communications buffer

MDM causes a trap to line n if there is data in the internal modem buffer

TIME$ = "hh:mm:ss" sets a time for BASIC to trap to line n

KEY causes a trap to a particular line depending on which of the 8 function keys is pressed

x is a numeric expression, the result of which causes a trap to one of the line numbers listed

The ON COM/MDM GOSUB statements are used in conjunction with COM/MDM ON/OFF/STOP. Providing a COM/MDM ON statement has been executed, BASIC branches to the specified line if data has been received from the communications line or the modem respectively. If COM/MDM OFF has been executed, trapping to the subroutine is disabled. COM/MDM STOP inhibits the subroutine until the next COM/MDM ON. Refer to COM ON/OFF/STOP and MDM ON/OFF/STOP for further information.

The string variable TIME$ sets a time for BASIC to trap to the specified line. When the trap takes place an automatic TIME$ STOP is executed. The RETURN from the subroutine generates an automatic TIME$ ON unless TIME$ OFF is included in the subroutine. The ON TIME$ .. GOSUB statement is only effective if a program is being executed.

For ON KEY GOSUB a maximum of 8 lines can be specified corresponding to the function keys F1 to F8 . When the trap takes place an automatic KEY(n) STOP is executed and the RETURN from the subroutine generates an automatic KEY(n) ON, unless KEY(n) OFF is part of the subroutine.

ON x GOSUB is used to branch to different lines depending on the value of the expression x. If x is 1, BASIC branches to the first line number listed; if x is 2, BASIC branches to the second line number, etc. If x is not an integer, the part after the decimal point is discarded. If x is Ø or greater than the number of items in the list, the program continues with the next statement (providing x < 255). If x is outside the integer range an FC (illegal function call) error will be displayed.

If an error trap occurs from an ON ERROR statement all trapping to

subroutines is disabled.

Example:

```
1Ø INPUT "select 1, 2 or 3";A
2Ø ON A GOSUB 1ØØ,2ØØ,3ØØ
3Ø GOTO 1Ø
1ØØ PRINT "subroutine 1": RETURN
2ØØ PRINT "subroutine 2": RETURN
3ØØ PRINT "subroutine 3": RETURN
RUN
select 1, 2 or 3? 2 (the user enters 2)
subroutine 2
select 1, 2 or 3?  C (the user presses <CTRL> + c)
Break in 1Ø
Ok
```

ON ... GOTO Statement

ON ... GOTO causes branching to another program line.

Format: ON ERROR GOTO n
ON x GOTO a ,b,c ...

where: a,b,c,n are program line numbers

ERROR enables error trapping and specifies the first line of the error handling subroutine.

x is a numeric expression, the result of which causes a trap to one of the line numbers listed.

After an ON ERROR GOTO statement all errors detected, including Direct Mode errors, e.g. SN (syntax error), will cause a jump to the specified line for an error handling sub-routine. Normally an error would cause a break in program execution but an error handling routine allows the program to recover and continue. The error trapping routine can be disabled by executing an ON ERROR GOTO Ø statement. Use the RESUME statement to return to normal program execution at the end of the routine.

ON x GOTO is used to branch to different lines depending on the value of the expression x, in the same way as ON ... GOSUB. If x is 1, BASIC branches to the first line number listed; if x is 2, BASIC jumps to the second line number, etc. If x is not an integer, the part after the decimal point is discarded. If x is Ø or greater than the number of items in the list, the program continues with the next statement (providing x < 255). If x is outside the integer range, an FC (illegal function call) error will be displayed.

OPEN is the statement used for opening files in RAM and on cassette, or for allocating a buffer to a peripheral, treating it as a file.

Format:

```
OPEN "CAS: file-name " FOR mode AS file-number
OPEN " RAM: file-name" FOR mode AS file-number
OPEN "COM:cpbsx" FOR mode AS file-number
OPEN "LCD:" FOR OUTPUT AS   file-number
OPEN "LPT:" FOR OUTPUT AS   file-number
OPEN "MDM:bpsx" FOR mode AS  file-number   (for  M10  MODEM
only)
OPEN "WAND:" FOR INPUT AS   file-number
```

where:

CAS opens the cassette recorder

RAM opens the RAM

COM opens the communications line

LCD opens the LCD screen on the M10

LPT opens the line printer

MDM opens the internal modem

WAND opens the bar code reader

file-name is the name under which the file was saved

file-number is the number allocated to the file

mode is INPUT, OUTPUT or APPEND

An OPEN statement must be executed before any input/output instructions can be carried out using statements or functions which require a file-number, e.g. PRINT, PRINT USING, INPUT, LINE INPUT. INPUT$.

The file-number is used in other input/output statements to refer to the file, while it is open. It must be an integer between 1 and the maximum number specified in a MAXFILES statement.

All three modes INPUT, OUTPUT and APPEND are only available for files in RAM. For cassette, communications or modem files only INPUT and OUTPUT are available.

A file cannot be opened for a particular mode if it is already open for another mode.

When the communications line or modem is opened as a file the communications parameters must be specified (see Chapter 8, Fig.8-5).

OUT Statement

OUT is used to send a byte of data to an output port.

Format: OUT n,i

where: n is the port number
i is the data byte

Both n and i are integers in the range Ø to 255. OUT is complementary to the IN function.

PEEK Function

PEEK returns the byte stored at a specified address.

Format: PEEK(l)

where: l is the memory location which must be in the range -32768 to 65535.

The value returned is in decimal form in the range Ø to 255.

Negative values are subtracted from 65536, e.g. -1 is the same as 65535.

PEEK is often used to access information stored either with the POKE statement or in a machine language subroutine.

POKE Statement

POKE writes a byte of data into a specified memory location.

Format: POKE l,i

where: l is the memory location which must be in the range -32768 to 65535

i is the byte of data which must be an integer in the range Ø to 255 (decimal).

If a negative memory location is given it is subtracted from 65536, e.g. -1 is the same as 65535. Use POKE with care as the system does not check that the memory location specified is unused.

POS returns the current cursor position across the screen.

Format: POS(i)

where: i is a dummy argument which is normally Ø.

The left hand edge of the screen is Ø.

Example:
```
PRINT POS(Ø);POS(Ø);POS(Ø)
 Ø  3  6
Ok
```

POWER Statement

POWER controls the time before automatic power off.

Format:
```
POWER i
POWER CONT
POWER OFF ,RESUME
```

where: i is an integer from 10 (1 minute) to 255 (25.5 minutes)

Power is automatically switched off if no keys are used for the time specified in the POWER i statement. This feature is automatically disabled if a POWER CONT statement is executed.

POWER OFF turns the power off and when the system is restarted by switching the power switch off and on the menu is displayed. If POWER OFF, RESUME is used the system will restart with the same status as before the power was switched off. An automatic power off will also cause the system to revert to the same status as before, when the power switch is switched off and on again.

PRESET Statement

PRESET is used to specify the position of an element of the LCD which is to be displayed in black or white.

Format: PRESET(x,y ,c)

where: x,y are the co-ordinates of the element. x must be in the be in the range Ø to 239 and y must be in the range Ø to 63.

c is the colour parameter: if c = 1 (or if it is omitted) the element is white and if c = Ø the element is black.

If co-ordinates outside the permitted range are specified, an FC (illegal function call) error is displayed.

This statement is similar to PSET but for the latter c = 1 for a black element.

PRINT Statement

PRINT displays data on the screen.

Format: PRINT list
? list
PRINT n, list

where: list is a list of expressions which may be separated by ommas or semicolons.

? is an abbreviation for PRINT.

n defines a starting positon for the printout: for details see below.

The expressions may be numeric or string, but strings must be enclosed in inverted commas. A variable name can be used and the content of the memory location assigned to the variable is printed.

The semi-colon causes one item to be followed immediately by another, except that numbers are always followed by a space. Positive numbers are also preceded by a space where a negative number will have a minus sign. If commas are used to separate the items they are displayed in a tabular form. The tab settings are 14 characters apart; on the LCD two columns are formed, as illustrated in the examples below.

Examples:

```
Ok
PRINT 12;-345;"abc";" def"
12 -345 abc def
Ok
a = 2*4 : b$ = "Jan" : c = 1983
PRINT a;b$;c
 8 Jan 1983
Ok
PRINT a,1Ø,12,2*7
 8             1Ø
 12            14
Ok
```

If a comma or semi-colon is placed at the end of the list in a program line the next PRINT statement will start printing at the position determined by the punctuation mark.

Example:

```
1Ø PRINT 1;2;3;4;
2Ø PRINT 1;2,3,4
RUN
 1  2  3  4  1  2
 3             4
Ok
```

The PRINT statement is used to start printing at a particular position. The value of n must be between Ø and 239: the number represents a character position. The first position, represented by Ø, is at the top left hand corner of the screen. The second and subsequent character positions are counted from left to right on the top line of the screen, then from left to right on the second line, etc. The maximum value of n then represents the position in the bottom right hand corner of the screen. For the LCD the maximum value of n is 239. For the CRT the maximum value of n is 999 for a 40 characters x 25 lines display and 1999 for an 80 characters x 25 lines display.

PRINT , PRINT USING Statements

PRINT and PRINT USING write data to a sequential file.

Format:
```
PRINT  file-number, data
PRINT  file-number, USING "x"; list
```

where: file-number is the number under which the file was opened

data can be a list of items separated by commas or

semicolons, or it can be the contents of a numeric or string variable

x is a string expression which defines the format of the output

list is a list of expressions which may be separated by commas or semi-colons.

These statements have the same format as PRINT and PRINT USING, but the data is transferred to the appropriate file. A file on any peripheral can be written to, using PRINT and PRINT USING, but the file must first be opened in the output mode. See OPEN.

PRINT USING Statement

PRINT USING displays data on the screen using a specified format.

Format: PRINT USING "x"; list

where: x is an expression which defines the format of the output; it must be enclosed in inverted commas.

list is a list of expressions which may be separated by semicolons or commas.

There is no space between printed items unless it is included in the formatting string.

The following formatting commands are available:

COMMAND	DESCRIPTION
! for strings	specifies that only the first character in the given string is to be printed.
\ for strings	the backslashes define the length of the string to be printed, e.g. if there are 2 spaces between the backslashes four characters will be printed.
# for numbers	The hash sign represents each digit position which will be filled. If the number has less digits than are required to be printed it will be preceded by spaces on the display. A decimal point may be used

at any position; if a digit is then specified before the decimal point it is always printed, as Ø if necessary. If numbers have more digits after the decimal point than are required to be printed they will be rounded. If this means that the number is then greater than the specified range a % is printed in front of it. The % is also printed in front of the number if the data entered is greater than the maximum possible with the specified format.

+ for numbers	The plus sign can be positioned before or after the other formatting characters. It causes the sign of the number to be printed before or after the number.
- for numbers	A minus sign after the other formatting characters causes negative numers to be printed with a trailing minus.
** for numbers	Two asterisks at the start of the formatting string causes leading spaces to be filled with asterisks. The asterisks also define digit positions.
$$ for numbers	Two dollar signs before the formatting characters cause a dollar sign to be printed immediately before the number. It is not possible to use the exponential format with this command and negative numbers can only be used with a trailing minus.
**$ for numbers	This combines the effects of two asterisks and two dollar signs: leading spaces will be filled with asterisks and a $ will be printed before the number. This combination defines two more digit positions for the number.
, for numbers	A comma to the left of the decimal point in a formatting string causes a comma to be printed for every three digits to the left of the decimal point. The comma specifies another digit position and has no effect on numbers in exponential format. A comma at the end of the formatting string is printed as a character.
^^^^ for numbers	Four exponent symbols specify that the number must be printed in exponential format. The significant digits

are left registered and a space or minus is displayed in the first digit position unless + or - are included in the formatting characters.

Examples:

```
A$ = "Olivetti" : B$ = "M1Ø"
Ok
PRINT USING "!";A$;" ";B$
O M
Ok

PRINT USING "\ \";A$;B$      (two spaces)
OlivM1Ø
Ok

A=123.45:B=6.789:C=-123.45:D=-6.789
Ok
PRINT USING "##.## ";A;B;C;D
%123.45  6.79 %-123.45 -6.79
Ok

PRINT USING "#.###^^^^ - ";A;B;C;D
1.235E+Ø2  6.789E+ØØ  1.235E+Ø2- 6.789E+ØØ
Ok

PRINT USING "+**$#.### ";A;B;C;D
+$123.45Ø **+$6.789 -$123.45Ø **-$6.789
Ok
```

PSET Statement

PSET is used to specify the position of an element of the LCD which is to be displayed in black or white.

Format: PSET(x,y ,c)

where: x,y are the co-ordinates of the element. x must be in the range Ø to 239 and y must be in the range Ø to 63.

c is the colour parameter: if c = Ø the element is white and if c = 1 (or if it is omitted) the element is black.

If co-ordinates outside the permitted range are specified, an FC (illegal function call) error is displayed.

This statement is similar to PRESET but for the latter c = Ø for a black element.

READ takes values from a DATA statement and assigns them to variables.

Format: READ variable-1 ,variable-2 ... ,variable-n

where: variable is a numeric or string variable

The READ statement can access one or more DATA statements which will be read in order. If there are more variables than available data an OD (out of data) error will be displayed. The variables must match the type of data stored or an SN (syntax error) will be displayed.

If there are fewer variables than data items, subsequent READ statements will start reading at the first unread item. If there are more data items than variables at the end of the READ statements, the remaining data is ignored.

RESTORE is used to return to the first item in the first DATA statement.

Example:

```
Ok
1Ø READ A,B$,C,D$
2Ø PRINT A,B$,C,D$
3Ø DATA 56,AB,-3.55E+2Ø,6abc7
RUN 1Ø
56            AB
-3.55E+2Ø     6abc7
Ok
```

REM Statement

REM allows remarks to be inserted in a program which do not affect execution but do appear in the listing.

Format: REM

The apostrophe can be used as an abbreviation. All text after REM is considered part of the remark. Therefore, in a multiple statement line the remark must be the last statement.

It is possible to branch to a remark with GOTO or GOSUB and program execution continues with the next executable instruction.

REM must not be used as part of a DATA statement because it will be treated as data.

RESTORE Statement

RESTORE is used to re-read DATA statements.

Format: RESTORE n

where: n is a line number.

After a RESTORE statement the next READ accesses the first item in the DATA statement on the specified line. If n is omitted the READ uses the first DATA statement in the program. If n does not exist a UL (undefined line error) will be displayed.

RESUME Statement

RESUME continues program execution after an error recovery routine has been carried out.

Format: RESUME
RESUME 0
RESUME NEXT
RESUME n

where: n is a line number.

RESUME and RESUME Ø cause execution to continue at the statement where the error occurred. RESUME NEXT causes execution to continue at the statement following the error. RESUME n causes execution to continue at line n.

If a RESUME is encountered which is not in an error trap routine a RW (resume without error) error is displayed.

RIGHT$ returns a string of characters from the right of a given string.

Format: RIGHT$(x$,i)

where: x$ is a string

i is an integer from Ø to 255.

The last i characters of the string x$ are returned. If i is longer than x$ the complete string is returned and if i is zero a null string is returned.

Example:

```
$ = "Olivetti M1Ø"
B$ = RIGHT$(A$,5)
PRINT B$
i M1Ø
Ok
```

RND Function

RND returns a pseudo-random number between Ø and 1.

Format: RND(x)

where: x is a number.

If x>0 then the same series of pseudo-random numbers is generated each time the program is run. If x≤0 then the previously generated number in the series is repeated. This is useful for de-bugging. The sequence of numbers generated may be "seeded" within a program with the TIME$ function. For example:

```
1Ø J=VAL(RIGHT$(TIME$,2))
2Ø FOR I=1 TO J
3Ø RN=RND(1):NEXT
```

will provide 60 different values for RN. This is only one of many ways in which the sequence may be seeded.

RUN , RUNM Commands

RUN and RUNM are used to load programs into the current program area of memory and run them. Programs are not registered in the directory unless the SAVE command has been used.

Format:

```
RUN n  "name.xx"   ,R
RUN "CAS: file-name "  ,R
RUN " RAM: file-name"  ,R
RUN "COM:cbpsx"  ,R
RUN "MDM:bpsx"  ,R   for M10 MODEM only
RUNM "CAS: file-name "
RUNM " RAM: file-name"
```

where:

n is a line number

name.xx is the file-name and extension

file-name is the name used when the program was saved

R keeps data files open

CAS loads and runs a program from cassette

RAM loads and runs a program from RAM

COM loads and runs a program from the communications line.

MDM loads and runs a program from the internal modem interface.

RUN loads and runs BASIC programs into the current program area of memory, deleting any previously held program. RUNM loads and runs machine language programs which are stored at the location specified when the program was saved.

A program from RAM can be run from a particular line, specified by n. If the extension to the file name is not given all .BA files will be searched first and .DO files will be searched if the name is not found among the .BA files.

Normally a RUN closes all open files. The R option keeps the data files open. RUN is therefore the same as LOAD ... ,R.

SAVE and SAVEM are used to store programs.

Format:
```
CSAVE "file name"
SAVE "CAS:file name"  ,A
SAVE " RAM: file name"  ,A
SAVE "COM:cbpsx"
SAVE "MDM:bpsx"    for M10 MODEM only
CSAVEM "file name",a,b ,c
SAVEM "CAS:file name",a,b ,c
SAVEM " RAM: file name",a,b ,c
```

where: CSAVE and CSAVEM store programs onto cassette in binary format

file-name is the name by which the program will be recognized

A is an option which saves the program in ASCII format

CAS saves the current program onto cassette tape

RAM saves the current program into RAM

COM saves the current program to the communications line in ASCII format.

MDM saves the current program to the internal modem interface in ASCII format.

M specifies machine language program

a,b,c are addresses

CSAVE and SAVE "CAS (without the A option) are the same command: they both store BASIC programs in binary format onto cassette. Similarly CSAVEM and SAVEM "CAS are the same, saving machine language programs onto cassette.

SAVE "RAM: is similar to SAVE "CAS: but RAM may be omitted.

SAVE can also be used to list the current program on the display with SAVE "LCD:" for the LCD display or SAVE "CRT:" for the VDU.

Programs can be saved to another machine via the communications line or the modem. Sending and receiving conditions must agree and the remote machine must be set to LOAD with the same communications parameter settings as are used for SAVE.

SCREEN Statement

SCREEN determines which device displays output with PRINT, LIST, etc.

Format: SCREEN d ,k

where: d specifies the display: for d = Ø output is to the to the LCD; for d = 1 output is to the CRT

k controls the display of the contents of the function keys: for k = Ø the display is disabled; for k = 1 the display is enabled.

The default value for both d and k is Ø. If d is set to 1 when there is no CRT connected an FC (illegal function call) error is displayed.

SGN Function

SGN returns the sign of a number.

Format: SGN(x)

This function returns 1 for x > Ø; Ø for x = Ø; and -1 for x < Ø.

SIN Function

SIN returns the sine of an angle.

Format: SIN(x)

where: x is the angle in radians.

Example:

```
Ok
PRINT SIN(0.5)
 .47942553860422
Ok
```

SOUND plays a specified note. SOUND ON/OFF enables or disables the loudspeaker when a program is being loaded from the cassette or if the TELCOM facility is in use.

Format: SOUND note, length
SOUND ON
SOUND OFF

where: note is an integer in the range Ø to 16383; the table below lists the values for particular notes

length is an integer in the range Ø to 255; for an integer i the length is (i+1)*20ms.

SOUND ON enables the loudspeaker and SOUND OFF disables it.

OCTAVE	1	2	3	4	5	6
NOTE						
C		9394	4697	2348	1171	587
C#		8866	4433	2216	1103	554
D		8368	4184	2092	1043	523
D#	15800	7900	3950	1975	987	493
E	14912	7456	3728	1864	932	466
F	14064	7032	3516	1758	879	439
F#	13284	6642	3321	1660	830	415
G	12538	6269	3134	1567	783	
G#	11836	5918	2954	1479	739	
A	11172	5586	2793	1396	693	
A#	10544	5272	2636	1318	659	
B	9952	4968	2484	1244	622	

SPACE$ Function

SPACE$ returns a string of spaces.

Format: SPACE$(x)

where: x is a number from Ø to 255.

If x is not an integer the part after the decimal point is ignored.

Example:

```
Ok
A$ = SPACE$(5)
Ok
PRINT "x";A$;"y";A$;"z"
x     y     z
Ok
```

SQR Function

SQR returns the square root of a number.

Format: SQR(x)

where: x is a positive number.

If x is negative an FC (illegal function call error) will be displayed.

Example:

```
Ok
PRINT SQR(5Ø)
 7.Ø71Ø678118655
Ok
```

STOP terminates program execution and returns to the Direct Mode.

Format: STOP

This statement can be placed anywhere in a program; when it is executed the following message is displayed:

Break in n

where: n is the line number.

Files are left open and the program may be continued by using the CONT command providing the program is not modified while it is stopped.

STRING$ Function

STRING$ returns a string of a specified number of one particular character.

Format: STRING$(a,b)
STRING$(a,x$)

where: a,b are integers in the range Ø to 255

x$ is a string.

The string returned has a characters all of which have the ASCII code b, or are the first character of x$.

Example:

```
Ok
A$ = STRING$(2Ø,47)
Ok
PRINT A$
////////////////////
Ok
```

STR$ Function

STR$ returns a string representation of a number.

Format: STR$(x)

where: x is a number

The string representation of a number is useful in transferring data to peripherals.

Example:

```
Ok
PRINT STR$(34)
 34
Ok
```

The displayed number, 34, is stored as characters, i.e. one byte for each digit.

TAB Function

TAB is used with PRINT and LPRINT to space to a particular position.

Format: TAB(i)

where: i is an integer in the range Ø to 255.

If a value of i is given which is before the current location of the cursor the next item is printed immediately after the cursor (with the appropriate spaces if it is a number).

Ø represents the lefthand edge of the screen. Each digit represents one character position to the right: 4Ø is the left edge of the second line and 255 is the sixteenth character on the seventh line.

Example:

```
Ok
PRINT "a" TAB(5) "b" TAB(3) "c" TAB(5Ø) "d"
a    bc
         d
Ok
```

TAN returns the tangent of an angle.

Format: TAN(x)

where: x is the angle in radians.

Example:

```
Ok
PRINT TAN(Ø.5)
 .5463Ø248984381
Ok
```

TIME$ Statement and Function

TIME$ sets the current time for the clock, or returns the current time. TIME$ ON/OFF/STOP statements are used to enable and disable trapping to a subroutine from an ON TIME$ ... GOSUB statement.

Format:
```
TIME$ = "hh:mm:ss"
TIME$

TIME$ ON
TIME$ OFF
TIME$ STOP
```

where: "hh:mm:ss" is the format used to set the current time on the clock

When setting the time use values of hh between ØØ and 23. NOTE THAT VALUES FROM 24 TO 29 ARE ACCEPTED BY THE SYSTEM BUT THEY ARE REGISTERED AS ØØ, WHEREAS VALUES OF 3Ø OR MORE WILL RESULT IN AN SN (SYNTAX ERROR) BEING DISPLAYED.

TIME$ ON enables trapping to a subroutine and TIME$ OFF disables it. TIME$ STOP inhibits trapping until a TIME$ ON statement is encountered: if the time set in the ON TIME$ ... GOSUB statement has been passed, an immediate trap to the sub-routine takes place.

If TIME$ is used as a function the current time is returned.

Example:

```
Ok
PRINT TIME$
16:13:16
Ok
```

VAL Function

VAL returns the numerical value of a string.

Format: VAL(x$)

where: x$ is a string.

Leading blanks, tabs and line feeds are ignored.

Example:

```
PRINT VAL(" 34")
 34
Ok
```

The function STR$ converts numbers to strings.

VARPTR Function

VARPTR returns the address of the first byte of a variable or a file.

Format: VARPTR(x)
VARPTR(n)

where: x is a variable of any type

n is the number under which a file was opened.

This function is frequently used to obtain the address of a variable or array so that it can be sent to a machine language sub-routine.

Any variable called by VARPTR must have previously been assigned a value or an FC (illegal function call error) will be displayed.

The number returned will be in the range -32768 to 32767. If a negative number is returned, it represents an address from 32768 (-32768)to 65535 (-1).

For arrays, all simple variables must have previously been assigned values and the form VARPTR (array(Ø)) should be used so that the lowest address element is returned.

For a file this function returns the starting address of the file control block.

WIDTH sets the CRT screen width to 40 or 80 characters.

Format: WIDTH n

where: n is 4Ø or 8Ø and sets the screen width accordingly

A. SUMMARY OF TECHNICAL SPECIFICATIONS

Dimensions

30 x 22 x 6 cm (11 3/4 x 8 1/2 x 2 3/8 in.).

Weight

1900 grammes (4lb 3oz).

Power Supply

6V DC either from 4 x 1.5V size AA dry cells or from AC mains supply via AC adaptor.

Microprocessor

80C85 (8 bits CPU).

RAM

8k byte standard, expandable to 32k.

ROM

32k byte standard, expandable to 64k.

B. BASIC ERROR MESSAGES

Errors are indicated on the M10 by the error codes listed below. The error numbers are the integers recognized by BASIC and must be specified when an ERROR statement is used. The errors are also listed in numerical order. A maximum of 255 codes are available, but not all of these are used. Unused error numbers are not listed and some of the error numbers produce an unprintable error code, UE. This applies for all the numbers which may be allocated in the future.

ERROR CODE	ERROR NO.	DESCRIPTION
AO	53	File already open: a sequential output mode OPEN has been issued for a file that is already open; or a KILL has been attempted on an open file.
BS	9	Subscript out of range: an array element has been referenced with a subscript that is outside the dimensions of the array, or the wrong number of subscripts have been given.
BN	51	Bad file number: a statement or command references a file with a file number that is not open or is out of the range of numbers specified by the MAXFILES statement.
CF	58	File not open: the file specified in a PRINT ,'INPUT , or similar statement has not been opened.
CN	17	Cannot continue: an attempt has been made to continue a program that cannot be continued for one of the following reasons: 1. It has stopped due to an error. 2. It has been modified during a break in execution. 3. It does not exist.
DD	10	Redimensioned array: two DIM statements have been given for the same array, or a DIM statement has been given for an

		array after the default dimension has been established.
DS	56	Direct statement in file: a direct statement has been found while loading an ASCII format file and the LOAD is stopped.
EF	54	Input past end: an input statement has been attempted, either after all the data in the file has been input, or for an empty file. Use the EOF function to detect the end of a file to avoid this error.
FC	5	Illegal function call: a parameter that is out of range is passed to a numeric or string function. This error will also be produced as a result of: 1. A subscript which is negative or greater than the maximum permitted. 2. A negative or zero argument with LOG. 3. A negative argument with SQR. 4. An incorrect argument with: INP, INSTR, LEFT$, MID$, ON...GOTO, OUT, PEEK, POKE, RIGHT$, SPACE$ or STRING$.
FF	52	File not found: a LOAD, KILL or OPEN statement references a file that does not exist in memory.
FL	57	Too many files: an attempt has been made to create a new file using SAVE or OPEN when all directory entries are full.
ID	12	Illegal direct: a statement that is illegal in direct mode has been entered as a direct command.
IE	50	Internal error: an internal fault has occurred. Contact your local dealer.
IO	18	Input/Output error: an input/output error has occurred on a cassette, printer or VDU operation. It is a fatal error, i.e. BASIC cannot recover.
LS	15	String too long: an attempt has been made to create a string more than 255 characters long.
MO	22	Missing operand: an expression contains an operator with no operand following it.

NF	1	NEXT without FOR: a variable in a NEXT statement does not correspond to any previously executed, unmatched FOR statement variable.
NM	55	Bad file name: an illegal form has been used for a file name specified with LOAD, SAVE, KILL, NAME, etc.
NR	19	No RESUME: an error trapping routine has been started but it contains no RESUME statement.
OD	4	Out of data: a READ statement has been attempted when there are no DATA statements with unread data remaining in the program.
OM	7	Out of memory: a program is too large; has too many files, FOR loops, GOSUBs, or variables; or it contains expressions that are too complex.
OS	14	Out of string space: string variables have caused BASIC to exceed the amount of free memory remaining. BASIC will allocate string space dynamically until it runs out of memory.
OV	6	Overflow: the result of a calculation is too large to be represented in BASIC's number format.
RG	3	RETURN without GOSUB: a RETURN statement has been encountered for which there is no previous, unmatched GOSUB statement.
RW	20	RESUME without error: a RESUME statement has been found before an error trapping routine has been entered.
SN	2	Syntax error: a line has been found that contains an incorrect sequence of characters, e.g. unmatched parenthesis, misspelt keyword, or incorrect punctuation.
ST	16	String formula too complex: a string expression is too long or complex. It should be split into shorter expressions.
TM	13	Type mismatch: a string variable name has been given a numeric value, or vice versa. Alternatively, a function that requires a numeric argument has been given a string argument, or vice versa.

UE	21	Unprintable error: an error message is not available for the error condition which exists.
UE	23-49	Unprintable error: these codes are undefined and should be reserved for future expansion in BASIC.
UE	59-255	Unprintable error: thes codes are undefined and may be allocated by the user.
UL	8	Undefined line: a line reference in a GOTO, GOSUB, or IF...THEN...ELSE statement is to a nonexistent line.
/0	11	Division by zero: An attempt has been made to divide by zero in an expression, or the operation of involution results in zero being raised to a negative power.

The following table lists the above codes in numerical order.

1	NF	NEXT without FOR.
2	SN	Syntax error.
3	RG	RETURN without GOSUB.
4	OD	Out of data.
5	FC	Illegal function call.
6	OV	Overflow.
7	OM	Out of memory.
8	UL	Undefined line.
9	BS	Sub-script out of range.
10	DD	Redimensioned array.
11	/0	Division by zero.
12	ID	Illegal direct.
13	TM	Type mismatch.
14	OS	Out of string space.
15	LS	String too long.
16	ST	String formula too complex.
17	CN	Cannot continue.
18	IO	Input/Output error.
19	NR	No RESUME.
20	RW	RESUME without error.
21	UE	Unprintable error.
22	MO	Missing operand.
23-49	UE	Unprintable errors.
50	IE	Internal error.
51	BN	Bad file number.
52	FF	File not found.
53	AO	File already open.
54	EF	Input past end.
55	NM	Bad file name.
56	DS	Direct statement in file.
57	FL	Too many files.
58	CF	File not open.
59-255	UE	Unprintable error.

C. THE OLIVETTI MC 10 MODEM COUPLER

GENERAL

The Olivetti MC 10 Modem Coupler is specially designed for use with the M10 personal computer without integrated modem. It consists of a data modem which can be acoustically coupled to a standard switched telephone line. By means of this equipment, the TELCOM option of the M10 can be fully exploited for data communication with a remote computer system.

The specifications of the MC 10 Modem Coupler conform to the recommendations of the CCITT on data transmission in switched telephone networks (Fascicle VIII - Rec. V.21).

The MC 10 is compatible with equipment having either a V.24/V.28 CCITT or RS-232C EIA interface. It operates on the dual-channel full duplex principle. Channel 1 corresponds to the ORIGINATE/CAL mode, Channel 2 to the ANSWER/ANS mode of operation. The first mode is for calling a host computer system or data service, the second for replying to a call from another computer.

The MC 10 is powered by a 12V DC supply either from its internal battery of 8 x 1.5V AM dry cells or from an external power supply connected to the mains.

TECHNICAL SPECIFICATIONS

Operating Mode :	Full duplex, Channel 1 and Channel 2
Data :	Binary, serial or asynchronous.
Data Signalling Rate :	300 baud maximum
Transmission Frequencies :	Channel 1 980 Hz (MARK) 1180 Hz (SPACE) Channel 2 1650 Hz (MARK) 1850 Hz (SPACE)
Receiving Frequencies	Channel 1 1650 Hz (MARK) 1850 Hz (SPACE) Channel 2 980 Hz (MARK) 1180 Hz (SPACE)
Transmission Level :	-18 dBm measured at the telephone output to the line.
Receive Sensitivity :	-43 dBm
Modulation :	FSK (Frequency Shift Keying)
Nominal delay between request to send (C105) and ready for sending (C106) :	from OFF to ON, 30 ms from ON to OFF, 1 ms
Nominal delay of data channel received line signal detector (C 109):	from OFF to ON, 1 s from ON to OFF, 80 ms
Power Supply :	8 x 1.5V AM batteries, type IEC LR6, AA SIZE or UM/SUM-3. Alternatively, an external power supply may be used.
Power Rating :	0.5 W
Temperature :	0° C to 40° C
Relative Humidity :	up to 90%
Data Interface :	According to CCITT Recommendations V.24/V.28 or RS-232C EIA Standard.
Interchange Circuits :	25-pin male connector with the connections shown in Fig. C-1.

PIN	CCITT NO.	EIA CODE	DESIGNATION
7	102	AB	Signal ground
2	103	BA	Transmitted data
3	104	BB	Received data
4	105	CA	Request to send
5	106	CB	Ready for sending
6	107	CC	Data set ready
20	108/2	CD	Data terminal ready
8	109	CF	Data Channel Received Line Signal Detector
9	-	-	+6V Test Voltage
10	-	-	-6V Test Voltage

Fig. C-1 Interchange Circuits

SWITCHES AND LAMPS

The status of the MC 10 is controlled and indicated by two switches and a lamp respectively.

The lamp is located on the top of the unit, between the cups of the coupler, beside the power ON/OFF switch (see Fig. C-2). It gives the following indications:

- Flashing green: this means that the MC 10 is switched on but not yet in communication with another computer.

- Continuous green: this means that the MC 10 is switched on and in communication with another computer.

- Continuous red: this means that the battery voltage has fallen below the necessary threshold for reliable operation and the cells should be replaced at once.

The switch beside the indicator lamp is the power ON/OFF switch.

On the underside of the unit is located the ANS/CAL switch. This switch is the channel selector.

- In position CAL, Channel 1 is selected, in accordance with the CCITT recommendation or the EIA specification for ORIGINATE MODE. This position is selected when the user is calling a host computer system.

- In position ANS, Channel 2 is selected, corresponding to the CCITT recommendation or the EIA specification for ANSWER MODE. This position is selected when a call is received from another computer system.

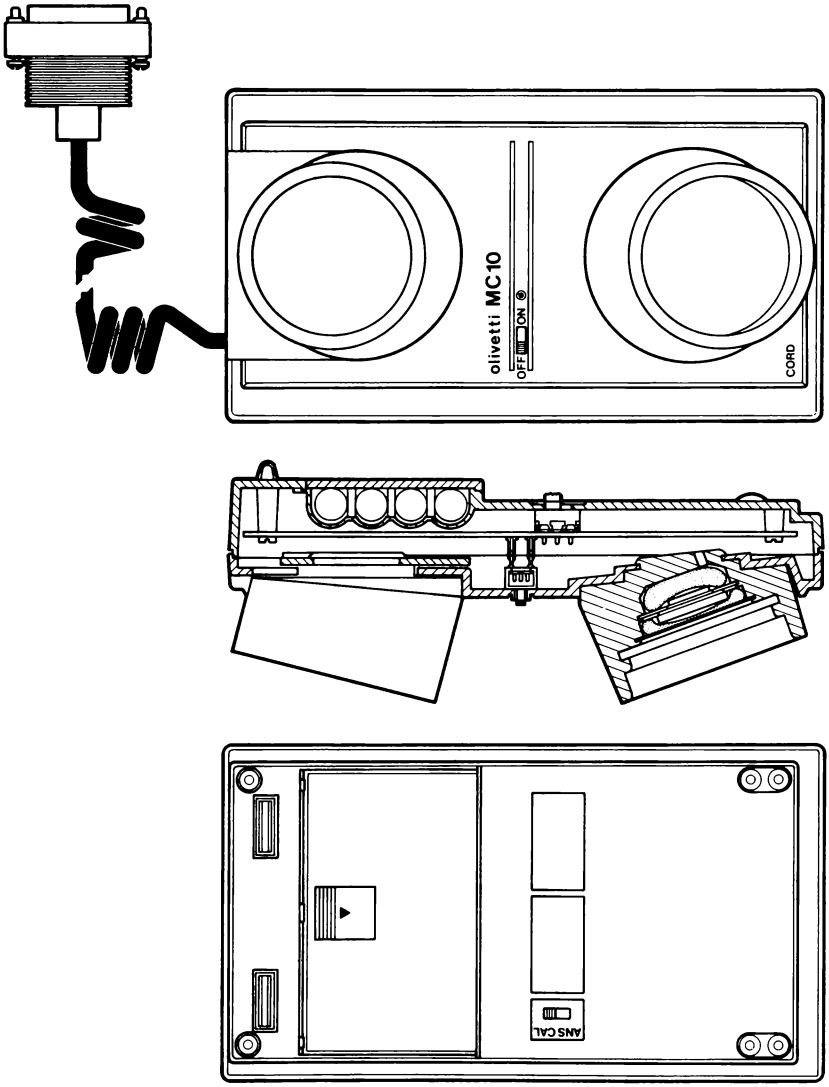

Fig. C-2 The MC 10 Modem Coupler

INSTALLATION

On first installing the MC 10 for operation, it should be set up in the following way:

1. Verify that the power ON/OFF switch on the upper panel is in the position OFF.

2. On the underside, remove the cover of the battery compartment by sliding it in the direction of the arrow engraved on the cover itself. Insert the batteries (supplied with the MC 10) in the compartment, ensuring that the correct polarities are respected. These polarities are marked on the inside of the battery compartment. Replace the cover.

3. Set the ANS/CAL switch to the appropriate position for the operation you are going to carry out (see Chapter 8 for details of data communications operations, using the M10).

4. Attach the MC 10 cable to the RS-232C interface on the M10 computer.

The MC 10 is now connected for data communication operation.

DATA COMMUNICATION OPERATION WITH THE MC 10

This section should be used in conjunction with Chapter 8 of the M10 Operations Guide. If you are preparing for remote data communication with a host computer system, proceed as follows:

1. Having first connected the MC10 cable to the RS-232C interface connector on the rear panel of the M10 and set the ANS/CAL switch to CAL, turn the power ON/OFF switch on the MC 10 to ON. The indicator lamp on the top panel will show a flashing green light.

2. Access the TELCOM application program on the M10.

3. Lift the telephone receiver and dial the number of the host system.

   If the host system is equipped with an automatic connection for data transmission, you will hear a high-pitched continuous tone, indicating that the connection has been established.

   If the host system is manual, you must ask the operator to connect you for data transmission. When you hear the high-pitched continuous tone, the data connection has been established.

4. As soon as the connection has been established, place the telephone handset in the cradle of the MC 10, making sure that the ear-piece is connected first and that the mouth-piece is inserted into the cup marked CORD. Press <F4> on the M10 to access the Terminal mode of the TELCOM program.

   After a moment, the indicator lamp on the top panel will show a continuous green light.

5. The system is now ready for data transmission. Details of data transmission using the M10/MC 10 combination are given in Chapter 8.

When the data transmission operation has been terminated, the receiver should be removed from the modem coupler and replaced on the telephone cradle.

KTX12**A01
Printed in Japan

olivetti