COMM+ On-line prompt card

This card is designed to provide you with a handy guide to the major functions of COMM+. It contains a brief summary of the most commonly-used commands for on-line use. Those new to communications are advised to read the Cookbook beginning on page 49 of the manual. This card assumes that you are using the standard version of COMM+ (without a JCF front-end) in help level 2. The format of the card is as follows:

To do this ...	Press these keys	See page
Eg. Capture an online session in memory	<ESC> <G>	58

Words and letters enclosed in <> angle brackets tell you which key to press. Thus <ESC> means 'press the key marked ESC', and <G> means 'press the key marked G'. You do **not** type the angle brackets. Note that the <RETURN> key on your computer may be marked <ENTER>, <CR> or <Carriage-return>.

So, to record your session in a disk file you would press the ESCape key and followed by the G key. If you wanted full details, you would refer to page 58 of the User Guide.

Getting started

Boot your system disk in the usual way. Insert the **COMM+** disk in drive A. Type **COMM** and press the <RETURN> key. When you get the copyright notice, press <RETURN> twice.

COMM+ is now ready for you to connect to an online service. Once connected, you can display a help menu at any time by pressing <ESC> <H>. To select any function from this menu, press <ESC> followed by the command letter shown.

Sending and receiving disk files

To select the logged disk drive	<ESC> <V> <Y> drive	63
To clear memory for new text	<ESC> <C> <Y>	60
To receive text by capturing it in memory	<ESC> <G>	58
To switch memory off after receipt of text	<ESC> <T>	
To save the captured text to disk	<ESC> <N> filename <RETURN> <ESC> <S>	
To prepare text offline for later transmission	<ESC> <B>	60
Then, when you have finished text entry	<ESC> <T>	
Note: you can use the editor to do this		16
Then, to transmit the text in memory	<ESC> <M>	61
If you want a copy saved on disk also	<ESC> <N> filename <RETURN> <ESC> <S> <Y>	59
To transmit a disk file to an online service	<ESC> <N> filename <RETURN> <ESC> <F> <Y> <T>	56
If you need to pause the transmission	<SPACE> or any other key	
Then, to continue transmission	<SPACE> or any key except <ESC>	
OR, to abandon transmission	<ESC>	

Quitting COMM+

To exit back to the operating system	<ESC> <Q> <Y>	62

Using COMM+ with Prestel

To select viewdata mode (as used by Prestel) Note: use underline (_) instead of hash (#), eg. *1_	<ESC> <W> <1>	119

Most modems will need to be reset to 1200/75-baud operation, although 'intelligent' modems will do this automatically, and you may also need to change speed in COMM+: see page 106 of the manual for details.

The editor

To enter or edit text in memory OR	<ESC> <E> <Y> <E> <N>	16
To edit a file from disk	<ESC> <E> <Y> <C> <SPACE> <N> filename <RETURN> <I> <N> <SPACE> <E> <N>	41

The following functions are accessed using the <CTRL> key, sometimes called <CONTROL> or <ALT>. To perform these functions, press the <CTRL> key and **hold it down** while you press the second key.

Cursor-left		<CTRL> + <S>	80
Cursor-right		<CTRL> + <D>	
Cursor-up		<CTRL> + <E>	
Cursor-down		<CTRL> + <X>	
Scroll up	- one 'page' (screenful)	<CTRL> + <R>	
Scroll down	- one 'page' (screenful)	<CTRL> + <C>	
Delete	- left	<DEL>	
	- right	<CTRL> + <G>	
	- rest of line (right of cursor)	<CTRL> + <Y>	
Insert	- character	<CTRL> + <V>	81
	- line	<CTRL> + <N>	
Help		<ESC> <H>	

Exit edit mode and return to online mode Note: Stores edited file in memory. Press <ESC> <N> filename <RETURN> <ESC> <S> <Y> to save it to disk also.	<ESC> <E> <RETURN>	81, 18

For further details of any COMM+ facility, please refer to the manual.

[ESC] C = restart suspended JCF
[STOP] or [ALT] C restarts aborted JCF

This version of COMM+ comes with a ready-to-run front end designed to enable you to automate most simply modem communications simply and easily, and to integrate any special applications you may have written in a common menu structure.

It is written for a Hayes-compatible modem and consists of the four job control files COMMJCF, CONNECTJCF, COMPOSEJCF and CREATEJCF, together with a SERVICE.DAT text file containing a list of options you have added to the system, and a file with the extension .SDF for any of these options for which COMM+ will dial and connect to automatically.

When loading COMM+, you will see a menu on your disk offering you a number of options, selected by pressing the relevant numbers by the side of the option followed by return. The first option adds entries to the list, while the second deletes entries. The third option runs COMM manually (ends the front-end JCF) and the fourth option quits back to the operating system. The first four options cannot be deleted : any you add yourself begin with number 5.

Options you add are stored in a file called SERVICE.DAT and can be of two types. The simplest type of added option picks up information from an associated SDF file concerning the telephone number, log-on codes and other parameters associated with a service, and automatically uses these to dial up and connect for you. However, you can also place the names of other job control files (JCFS) that you may have either written or obtained and add these to the list, and they can then be run automatically simply by selecting the relevant number : when such JCFs finish, they will automatically return to the menu.

SERVICE DATA FILES

If you select an option you have just added for which there is no associated file of any type on disk, you are asked whether you would like to create a Service Data File (SDF) for it. If you answer Y (for yes) CREATEJCF will automatically be run, generating a screen enabling such service data files to be defined. You are able to define a number of options as follows :

EMULATION can be any available in COMM+.
PARITY/DATABITS can be either 7 bits even parity or 8 bits no parity : this field should be left empty if COMM+ is being used on a system for which parity and word length cannot be set from within the programme (such a Concurrent DOS implementation).
SPEED can be from 300 to 9600 baud, and should set at the speed you will talk to the modem at, not the speed that the modem be talking on-line : thus when using a modem with V23 speed conversion, set the speed to 1200 baud and NOT 1200/75. As with parity and word length, the speed field should be left empty if COMM+ is being used on a system for which parity and word length cannot be set from within the programme (such a Concurrent DOS implementation).
MODEM SETUP can be either ignored or can be set to any string needed to set your modem up in a particular mode (for instance, the setup string AT&E0 disables error checking on a Quattro modem).
PHONE NUMBER cannot be left blank : prefix the number with T for tone dialling or P for pulse dialling if your modem responds to this convention.

Cancel dialling : [Enter]
Volgorde lijst nummers in Easycomm : Service.dat editen

The remaining (optional) strings are used to define a log-on sequence. Unlike defining the modem setup string and the phone number, if you want to send a carriage return via a send string you must explicitly embed a return in the sequence : any control codes can be embedded by pressing control-P first (thus control-P return embeds a carriage return). The tilde character (~) causes a delay in the send string of about a second (this can also be used in a modem setup or dial string if needed). For instance, a Telecom Gold sequence is as follows :

1. The first Look for item is blank.
2. The first Send out item is tilde control-P return (return to end). This starts by sending out a carriage return.
3. The second Look for item is PAD>
4. The second Send out item is CALL nn tilde control-P return (return to end) where nn is a Dialcom system number.
5. The third Look for item is Please sign on.
6. The third Send out item is ID followed by your system ID followed by tilde control-P return (return to end).
7. The fourth Look for item is PASSWORD.
8. The final Send out item is your password followed by tilde control-P return (return to end).

You are able to chain other JCFs from within this sequence. Simply enclose the name of the JCF inside {curly} brackets as part of a send string. If no JCF of that name exists, then COMM will create a source (SRC) file for the JCF in learn mode using the COMPOSE.JCF file at the right time : you should type the commands in manually and COMM will learn the correct sequence of keys and responses, and save them for you when you type control-Z to tell it to stop. You'll have to edit the resulting SRC file before condensing it into a JCF later on (select the option to run COMM manually from the menu and follow the instructions in the manual on page L-38 : you ought to read the whole of the language section of the manual for optimum results.

You have the option to either save or abandon the service data file details once they are complete. Thereafter, selecting the option relating to this SDF from the menu will enable the programme to pick up the details and run the CONNECT.JCF to log you on. You also have to option to amend any service data files before dialling up. Once the programme detects the NO CARRIER message on a line by itself, it knows that the call is finished and returns you to the menu. However, in between the time that any defined log-on sequence ends and the time that the modem disconnects, you have full access to all facilities on COMM+ (press ESC H for a list of them) including saving text to memory, file uploading and downloading and so on.

The EASYCOMM 2 front end can be disabled : simply rename the COMM.JCF file on disk to something else (such as MENU.JCF). Once this is done, COMM will run without any front-ends. You may find for some applications that this way of running the programme is preferable : be sure that you have saved a version of the programme (via option P from the off-line menu) with the correct speed, emulation and any other settings and user key definitions you need if you want to make optimum use of the programme.

1. The maximum memory version of TSR COMM is set up with the /R/G152/64 parameters, which occupies 220K more than a minimum version : the 152K of graphics memory is only useful if running with VGA 640 x 480 high resolution graphics screens. Currently a minimum memory TSR version loaded via /R/F/4/G4/E and without a viewdata GRAPHICS overlay on disk takes up 59K with expanded memory available and 63K if EMS is not available.

2. The FLAG variable is reset by the KEY VARn command : if the key typed was alphanumeric lower-case, FLAG is set to 1, but is set to 0 for all other keys. Since the KEY VARn command always returns upper case values, a simple flag test will enable a lower-case key to be regenerated via SET VARn=VARn+32 (since lower-case values are always 32 more than equivalent upper-case values).

3. In addition to the QKEY command, which sets FLAG to the number of bytes in the keyboard buffer, the undocumented QPORT command will set FLAG to the number of bytes in the receive character buffer (maximum FLAG value is 127). This command works whether HOLDPORT is on or off.

4. If a JCF turns the serial port off with HOLDPORT, it is automatically turned on again if the JCF either finishes or is aborted.

5. In addition to the special ONSCREEN codes 0 to 4, a code 5 will clear from the cursor to end of line : thus the command ONSCREEN 13,5 will do a carriage return and clear to end of line, which blanks the line on the screen.

6. Both DO strings and ONSCREEN strings may contain lower-case characters, which are not converted to upper-case when the code is either condensed or executed : the statement to the contrary has now been superceded.

7. The statement that directory displays always set FLAG to 1 is of course incorrect. The alternative version where it is stated that directory displays set FLAG according to whether the directory search found any files or not, has the correct usage.

8. An additional JCF command is undocumented : PLACE puts characters into memory in the same way that XTRACT takes them out, using the same memory pointer. There are three forms of the command :

a) PLACE $ puts a string $ into memory at the current position of the JCF memory pointer and increments it by one for each byte in the string. The format of the string is identical to the ONSCREEN and DO strings, and may contain intermixed quote-delimited ASCII text, decimal codes or variables in any order.
b) PLACE + (plus) simply increments the JCF memory pointer.
c) PLACE - (minus) decrements the JCF memory pointer.

If the JCF memory pointer is the same as the pointer to the end of memory then both pointers are incremented for each byte PLACEd until memory is full, at which point the BUFFUL flag is set and no more incrementation occurs. Decrementing the pointer with PLACE - has no effect if it already points at the beginning of memory.

9. WY60 Emulation: An extra emulation for the Wyse 60 terminal has been added as option F on the emulation menu. While largely compatible in most respects to the Wyse 30 the following additional features should be noted, which upgrade the Wyse 60 emulation to approximately the same capability as the Ampex 232.

a) The attributes on the Wyse 30 emulation are active from the position set to the end of the page : on the Wyse 60 emulation (like the Ampex and ADM emulations) attributes set are serially active for all characters subsequently placed on the screen until a different attribute is set.

b) The control-d # code for switching on transparent printing (ending with a control-T code) is supported in the same way as other transparent print codes for the TV925 and Ampex emulations.

c) The ESC c E code for using alternate character set and the ESC c D code for reverting to the normal character set function in the same way as the similar ESC $ and ESC % codes for the Ampex. Note that these select an alternate character (usually available only on IBM PC clones) rather than the line graphics characters selected via the ESC H codes, which carry on being supported.

d) The ESC z) and ESC z (codes function identically to the Ampex/Televideo/Hazeltine ESC F and ESC f codes for displaying information on the 25th line of the screen. The ESC ? code for returning the cursor address (offset by 20H) is supported also.

10. Additional support is now available for using string variables in job control files.

a) The 16 user keys can be treated as string variables in job control files in the same way as the 120 character variables. The command affected are those that can use strings : DO, GET, ONSCREEN, EXECUTE, LOAD, PLACE and IF WORD. In any of these commands, USERn can be inserted in the same way as VARn, and takes the contents of the userkey at the time of execution. Empty userkeys are ignored (GET USERn immediately returns true if USERn is empty). Since each userkey can hold up to 20 characters, the effective length of any string containing userkeys is reduced by twenty for each userkey it contains.

b) Note that though the commands DO USERKEY0 and DO 27,"0" appear equivalent, the former need not be followed by a BACK before another DO and can be used within commands such as naming : thus DO 27,"N",USER0 will name a file with the name contained in userkey 0 : but DO 27,"N",27,"U0",13 will try to name name a file ESC-U-0 and isn't at all useful.

c) To support the extra utility of userkeys within job control files, a small change has been made to the operation of the define userkey command. When defined on-line in the usual way via DO 27 "U", the contents of a user key are echoed to the screen. However, beginning with version 01.6-4, userkeys defined via DO 27,"EU" (via the off-line menu) will not be echoed to the screen. The situation is the opposite of the directory display command, where DO 27,"D" does not display on screen, whereas DO 27,"ED" does display on screen.

11. The KERMIT file transfer protocol now simulates initialization packets before doing a fetch.

12. PCDOS versions of COMM now have the option to print directly to any of the parallel ports LPT1: - LPT3: or any of the serial ports COM1: or COM2 as well as using the default operating systems list device. This is particularly useful for the TSR version, when a background COMM+ is now able to print at the same time as a foreground process simply by being configured to use a different printer, but the option is available on all PCDOS versions via typewriter mode : if the FIRST key pressed after option T from the off-line menu is ESC, nothing is printed, but a one-line sub-menu giving both the current printer selected and the available alternative appears. The options are D (for default printer) P (for a parallel printer) or S (for a serial printer) with the last two options being followed by a number (1-3) giving the device selected (thus S2 = COM2 and P2 =LPT2). This display is repeated until the current printer selection is confirmed with a CR. This option should be used with caution if the PCDOS version is being run on a machine which is not a full IBM clone : a selection of P1 on such systems may be unreliable.

13. The TSR version of COMM now uses any ANSI.SYS (or other console device driver) when operating with no emulations in foreground mode rather than using a teletype display. However, when in background, a teletype emulator rather than any console device driver is still used if no emulation has been selected. It is not unusual to find that because of this, characters which have been received when in background display in a different colour to characters received in foreground (depending of course on the way the system is set up). This feature is sometimes useful : but selecting one of the emulations (usually one of the ANSI ones) will display all text in identical colours if preferred.

14. When running as a TSR, COMM now replaces the DOS critical error handler with one of its own, which operates without keyboard intervention. However, the normal DOS critical error handler is still used for all foreground tasks.

E-mail: margolis@cix.compulink.co.uk

COMM+ Manual

The software and the documentation accompanying are provided in good faith to registered users, and every effort will be made to ensure that both the software and documentation perform according to reasonable requirements.

However, neither the authors nor any suppliers can offer any guarantee that this package is suitable for any particular combination of computer hardware, software or telecommunication equipment.

All warranties and conditions, whether statutory, expressed, implied or otherwise, relating to quality, description or performance of this package are hereby excluded.

The Copyright and other property rights to all ideas, writings, inventions, systems, programmes or programme descriptions, designs, formulae, algorithms and other intellectual property contained in this publication and the accompanying software are solely the property of the author, who reserves the right to revise both this publication and the accompanying software and to make changes without obligation to notify any person of such revision or changes.

COMM+ is an extremely powerful communications package suitable for electronic mail applications, intelligent terminal use, viewdata access, and error free file transfer from one computer to another. In order to become familiar with the structure of COMM+ in as short a time as possible, load COMM+ into your computer now, and work through the Off-line Tutorial at the keyboard. You should then, with the aid of the Cookbook, be able to start on-line communications immediately.

The power of the programme will become plain if you read through the Supplementary Reference Guide and Language Manual, but do not be misled by the complexities that those parts of the manual reveal - if you don't want to read through them then you don't have to. Any occasional references you may need to make (via the comprehensive index) do not compel you to work through and understand the whole. Like most extremely powerful programmes, most COMM+ users don't make use of all the facilities that their programme offers - but everything is there if you do need it. COMM+ runs on a wider range of systems than any other programme, outperforms most of them in all areas, and provides unparalleled facilities. We are confident that for general-purpose communications, it is the only package you will ever need.

COMM+ is written with the standard Digital Research ASM and ASM86 assemblers. Both the programme source code and the manual text were edited using Wordstar (by Micropro). The manual was printed and indexed using MagicBind (by Computer EdiType) on an Olivetti PG101 printer with the no. 1 font cartridge. Thanks are due to those who helped with information and hardware for programme implementations, and also to the users whose comments and suggestions help to improve the product - all these are too numerous to mention. Finally, a credit is due to Terry Kelly, who wrote the original version of the Tutorial, and all those connected with Lion Micro Systems Ltd. who gave help and support to the programme.

This manual is divided into five main sections:

1. **THE COMM+ OFF-LINE TUTORIAL**, which begins with a list of each function as it appears on the menus, with a brief description of each. It continues into more detailed operating procedures for the major off-line functions, together with key sequences and options. Pages in this section are prefixed with T.

2. **THE COOKBOOK**, a step by step guide to the main functions of COMM+. It is designed to be used as a quick reference and consists of a series of 'recipe' sheets which will enable you to use COMM+ without needing much understanding of how the programme works in detail. Pages in this section are prefixed with C.

3. **THE COMM+ SUPPLEMENTARY REFERENCE GUIDE** contains more detail on those aspects of the programme which are not self-explanatory but are unsuitable for covering fully in the off-line tutorial. It also contains information on features specific to particular operating systems (such as the PCDOS and Concurrent DOS implementations of COMM+) together with information on the optional TSR version of COMM+ for PCDOS systems. Pages in this section are prefixed with R.

4. **THE COMM+ LANGUAGE GUIDE** is the part which deals with the integral language in COMM+ which is used to automate and customize the programme for specific uses. Pages in this section are prefixed with L.

5. **APPENDIX**. This contains various details regarding operating systems, cables, and modems which you may need to know about to run COMM+ properly. Since these details aren't really to do with the programme itself, they are placed in an appendix. The appendix also contains various machine-specific notes. Since COMM+ runs on a variety of machines and operating systems, where certain features differ on an implementation, these are detailed in the appendix too. Pages in the appendix are prefixed with A.

Additionally, you will find a **GLOSSARY** of terms used in communications after the Appendix. Pages in the Glossary are prefixed with G. The manual ends with a comprehensive **INDEX** for all the parts of the manual.

COMM+ Off-line Tutorial

(this is a blank page)

INTRODUCTION

The aim of this off-line tutorial is to enable you to get used to the way that COMM works without having to connect up to another computer. Sections two and three also serve as the main reference guide you may need to use in on-line operation where the cookbook does not offer enough options directly.

We have tried in the tutorial to avoid unnecessary detail - we believe that there is enough coverage of the features COMM offers you in the tutorial to ensure that you will be able to use the program effectively without having to make yourself into an expert on the subject of communications.

Any functions which might need more elaboration are covered further in the Supplementary Reference Guide. We suggest that you make yourself familiar with its contents in case you need to read it in more detail. In addition, some advanced topics are not discussed at all in the tutorial because we concentrate here on the simpler uses of COMM.

However, if you go through the tutorial section while off-line and keep the cookbook handy while on-line, you should have no difficulty whatsoever in becoming a proficient communicator in the minimum amount of time.

TABLE OF CONTENTS

1. PRELIMINARY INSTRUCTIONS

When a properly installed version of COMM is loaded, the first thing you see on the screen is a copyright message in a box, together with the command:

Press any Key to begin

(The "press any key" message will appear frequently, and it is a good idea to get into the habit of pressing the same key each time. We suggest the space-bar, because it is relatively harmless: remember that on most keyboards, a key pressed down for any length of time will begin to repeat itself automatically).

Having duly reflected on the sins of infringing copyright, pressing any key will result in the screen clearing, your disk drives will briefly come on, and you will either be placed at the "start point", or if your copy of COMM+ came with a ready-to-run "front end" Job Control File (JCF - see the Language Manual for more details) you will automatically begin to run the specific application that this front-end is designed to deal with.

(Most Job Control Files will be documented separately : in the rest of this tutorial, we assume that no such front end will be running, and to use the tutorial you should, after any installation menus that the front-end might take you through, select an option to operate COMM manually from the JCF menu you see on screen : this will place you at the OFF-LINE MENU, where you can continue with section 2 of the tutorial.)

If COMM is improperly installed, your computer is liable to start doing peculiar things - lights may come on the disk drives, funny characters may appear on the screen, and so on. If this happens, OPEN THE DISK DRIVES and then SWITCH OFF OR RESET YOUR COMPUTER. Contact your supplier for advice.

The first time you run COMM, you will probably see a message saying

Found graphics ... loading
Found driver ... loading

This can be ignored. However, if you see a message saying that the driver cannot be found, you are missing a vital part of the program - contact your supplier to ask where it is. (The driver is that part of COMM which is different on each type of machine - you have to have a properly configured driver to run the program - normally this will load automatically from your program disk, or will be built in to the program file.)

Inside the "start point" box is the reminder "ensure all connections are in order". This is important: one thing COMM does from the time it is loaded on many systems is to look all the time to see if anyone is trying to communicate with you. (Technically, this procedure is called 'polling'.) On some machines, if nothing at all is connected to the computer, it will look as if there is something sending a continual stream of characters, which have no value, but still appear to be sent - this can cause problems. If you have made sure that your connections are in order, even if there is nothing at the other end (you do NOT have to dial up, only plug in) this problem will never arise even if your hardware is susceptible to it.

It is also true that some machines don't like connections being made whilst they are on - this is especially true of equipment which involves input voltages, such as printers and modems.

The first instructions proper appear as an option:

<RETURN> to go on-line or
<ESC> for off-line operation

(the words and letters enclosed in <> angle brackets denote keys to press throughout the manual - you don't need to type the brackets themselves).

Press the <ESC> key, and the screen will clear again, and display the OFF-LINE MENU. You will see a printout of this screen below.

(If you have pressed the <RETURN> key by mistake, and ended up in on-line mode, type <ESC> again followed by <E> followed by <Y>.)

The OFF-LINE MENU is divided into three sections. We suggest that you read through the next section of the manual carefully to understand all the functions on this menu. (You will find that it is not nearly as complicated as it may appear at first.)

```
O F F - L I N E   M E N U      File =  A:TEST      .TXT

-------------------------------------------------------
                      DISK COMMANDS
 S= Save File        |N= Name File   | D= Directory
 F= Send File        |Z= Rename File | V= Select Drive
 A= Append to File   |K= Delete File | R= Reset Disks
          J= Change area being used on Disk
-------------------------------------------------------
                         MEMORY
 M= Send Memory      |E= Edit and/or | C= Clear Memory
 O= Get old Memory   | format Memory | I= Input File
-------------------------------------------------------
                      UTILITIES
 U= Define Userkeys |X= Duplex      | H= Help level
 L= UpLoad-DownLoad |Y= Handshake   | W= DOS GateWay
 P= Programme Save  |B= Baud + word |-----------------
 T= Typewriter Mode |G= Get JCF     | Q= EXIT COMM+
-------------------------------------------------------

        ** Press RETURN for operation On-line **

                          Select:
```


2. FUNCTIONS IN THE OFF-LINE MENU

1) DISK COMMANDS -

S= Save file (the contents of the memory are saved under the name of the current file)

F= Send File (the contents of the currently named file will be sent where nominated)

A= Append to File (the memory contents will be added to the end of the current file)

N= Name File (here you name the current file for disk operations)

Z= Rename File (change the name of the currently named file on disk)

K= Delete File (deletes the currently named file)

D= Directory (displays directory of files on disk)

V= Select Drive (selects one of your drives for all disk operations)

R= Reset Disks (resets the disk system after changing disks)

J= Change directory path or area being used on disk
(you can change the logical area of the disk from which you save and read files - not available on MSDOS/PCDOS systems with DOS versions earlier than 2.0)

2) **MEMORY** –

M= Send Memory	(the memory contents will be sent where nominated)
O= Get old Memory	(allows you to recover the memory contents after clearing them)
E= Edit Memory	(allows you to edit, format and alter the memory contents on the screen)
C= Clear Memory	(clears out the contents of memory)
I= Input File	(inputs the contents of the currently named file to the memory)

3) **UTILITIES** –

U= Define userkeys	(allows you to memorize a string of characters or commands in a "user" key)
P= Program save	(allows you to save the version of the program you are using back to disk, including user keys and parameter settings)
T= Typewriter mode	(allows you to type directly to the printer)
X= Duplex	(you can select the correct duplex mode)
Y= Handshake	(you can configure the correct handshaking protocol)
B= Baud + word	(allows you to define the correct speed rate, word length and parity within COMM – on some machines this can only be done through systems software. Consult appendix if in doubt.)

L= Upload–Download (enables you to use the block transfer options to send or receive any type of file to or from a system running COMM or a compatible programme – this is not covered in the tutorial, but is the subject of a section in the Supplementary Reference Guide)

G= Get JCF (you can write programmes to run within COMM itself, automating communications handlings and laying out your own menus and prompts – the COMM language and how to use it is the subject of a separate part of this manual, and is not covered in this tutorial. You shouldn't attempt to write JCFs in any case until you fully understand the manual operation of COMM).

W= DOS GateWay (MSDOS/PCDOS systems version 2.0 or later only : you can temporarily exit from COMM to run secondary DOS tasks such as formatting disks, using your favourite word processor and so on : COMM stays in the memory of the machine until you have finished, when it'll carry on running).

In addition to the three main sections we have just covered, there are two other entries on the off–line menu –

H= Help level	(displays the current level of prompting)
Q= EXIT COMM+	(releases you from COMM and returns you to the operating system level from which COMM was initially loaded and run)

and at the bottom of the menu are the two instruction lines,

**** Press RETURN for operation On–line ****
Select:

3. FUNCTIONS IN THE ON-LINE MENU

The on-line menu contains another set of options, this time divided into eight sections. To call each function you have to preface the single letter on the menu with the command key (as displayed at the bottom of the menu) - this is initially <ESC>. The Supplementary Reference Guide contains details on changing the command key should you need to do so. Most users will find <ESC> to be quite satisfactory.

1) ENTRY COMMANDS -

T= On-line - memory off (*)
G= On-line - memory on ( )
B= Type directly to memory ( )

The above allow you to select which on-line mode you enter. The brackets and star signify which option is currently active.

(*) is ON
( ) is OFF

The above convention will be used relatively frequently in the on-line part of the program.

2) TO EXIT -

E= to off-line menu (takes you back to the off-line menu)
Q= to system (releases you from COMM and returns you to the operating system)

3) HELP -

J= Set help level (2) (allows you to change the prompt level in the bracket)
H= This help menu (self explanatory)

If you press the <RETURN> key, the screen will clear and a message will appear on the screen:

On-line Mode - Memory OFF
Type <ESC> H for Help

(This is the place you would have reached directly if you had pressed <RETURN> at the start point).

Press the <ESC> key followed by <H>.

The screen will clear, and the on-line menu will display. A printout of this screen follows:

```
O N - L I N E   M E N U     File =  A:TEST     .TXT
-----------------------------------------------------
 ENTRY COMMANDS                    |          TO EXIT
T= On-line  -  Memory off ( * ) |  E= Off-line Menu
G= On-line  -  Memory on  (   ) |  Q= to System
B= Type direct to Memory  (   ) |------------------
-----------------------------------  ON/OFF TOGGLES
 HELP                    |        L= Line feed   (   )
J= Help level   ( 2 )    |        P= Printer echo(   )
H= This On-line Menu     |        I= Handshake   (   )
-------------------------|        W= Emulation   (   )
 PROTOCOL                |----------------------------
Y= Set up Handshake      | U= User keys  SPECIAL KEYS
X= Duplex  ( F )         | K= Change Command key
-------------------------| .= UpLoad-Download /= Break
 DISK                    ----------------------------
N= Name File  Z= Rename File       |      MEMORY
F= Send File  V= Select Drive      |M=  Send Memory
S= Save File  A= Append to File    |C= Clear Memory
D= Directory  R= Reset Disk Drives |O=   Old Memory
-----------------------------------------------------
 ( * )= on (   )= off: Command key is now <ESC>
```

4) **ON/OFF TOGGLES** –

L= Line feed	( )	(adds a line feed to each carriage return transmitted)
P= Printer Echo	( )	(any data will be printed on the printer as well as being displayed on the screen)
I= Handshake	( )	(invokes the handshake selected from the handshake sub-menu)

The brackets signify which "toggles" are switched on,

(*) is **ON** and **( )** is **OFF**

W= Emulation	( )	(enables terminal emulation – your computer can behave like a completely different type of terminal with this feature enabled. A large number of emulations are available, and if emulation is enabled, the selected emulation is displayed in the brackets)

5) **HANDSHAKE** –

Y= set up handshake
X= Duplex (F)

This gives you an opportunity to alter the duplex setting or configure a handshake 'on the fly' prior to transmission or reception.

6) **SPECIAL KEYS** –

U= User keys	(allows you to redefine the user keys – these are invoked by <ESC><0> to <ESC><9>, and can hold up to twenty characters each)
K= Change command key	(you can change the command key from <ESC> to any other)
.= Upload-Download	(full stop – block transfer enables you to transfer any type of file of to a system running COMM or a compatible programme just as in the L option from the off-line menu)
/= Output break	(send a break signal to the host. This may not be implemented on some versions of COMM, including any which do not allow for setting up the Baud rate within the programme – consult the appendix for your particular system).

7) **DISK** –

N= Name file	S= Save file
Z= Rename file	A= Append to file
F= Send file	D= Directory
V= Select drive	R= Reset disk drives

8) **MEMORY** -

M= Send memory

C= Clear memory

O= Old memory

These disk and memory functions are the same as in the off-line menu, but are selected by prefacing your choice with <ESC> or another command key. Re-read the off-line menu section if you have forgotten what these do.

The following line appears at the bottom of the on-line menu:

(*) = on ( ) = off : Command key is now <ESC>

Any of the options in the on-line menu can be selected by pressing the <ESC> key, (provided you have not redefined the command key) followed by the relevant letter. All other keys pressed are interpreted not as commands to COMM, but as data to be sent somewhere : when at the off-line menu, by contrast, all keys pressed are interpreted as commands to COMM.

Now return to the off-line menu, by typing <ESC> followed by <E>.

You will be asked -

Exit to Off-line menu (Y/N) ?

COMM is asking whether you really do want to leave the on-line mode, and is giving you the chance to reconsider. If you wished to remain in on-line mode but had pressed <ESC> E by mistake, type <N> for no. If you try it, you'll see that the exit command will be cancelled, and control will be returned once again in on-line state, when any keys you press are considered to be data rather than commands.

Now repeat the <ESC> <E> keystrokes, and press <Y> in response to the prompt. This time, the screen will clear and display the off-line menu.

We shall now explain how to use some of the functions on the off-line menu in more detail - the introductory tour is now finished.

4. EXTENDED TUTORIAL WITH THE OFF-LINE MENU

1) Introductory notes.

All of the options in the off-line menu are executed by a single keystroke. Each option is assigned a different letter of the alphabet, and pressing that letter selects the appropriate option. Pressing any other key (including space-bar) may cause the computer to beep. Every time you hear a beep it means that you have given an invalid response. Giving invalid responses will not cause the system to crash. COMM is friendly and patient. It will, if necessary, wait all day for a valid response. If your computer doesn't have a beep (some don't) then you won't be warned about an invalid response. Simply, nothing will happen. (Some invalid responses are treated slightly differently - you may be told that the command is 'not implemented' instead of being beeped at. This is simply because we have allowed for expansion and customization of COMM, so have not excluded all the keys which don't appear on the menu.)

2) File naming.

In the top right hand corner of the screen, above the menu, you will see:

File = A: .

(The **A:** indicates the drive you are using - if you loaded COMM off a different drive such as drive B, then **B:** will display instead).

This is the area which will display the name of the file when it has been named. The file name can consist of up to eight characters, followed by a further three to denote the file type. The full stop separates the file name from the file type thus:

FILENAME.TYP

The name you give to a file will subsequently be used to recall the file so it helps if you try to make sure that the names you pick are meaningful. A file named ALC900Q.P5L may give little indication of what it contains. When you name or rename a file, COMM will

automatically change all lower case letters to upper case, and change illegal characters to X, and ignore full stops except as a name.typ separator.

Let us create a file we can use to practise on COMM and call it TEST.TXT.

Type N (you can use upper or lower case in all the menus).

At the bottom of the screen the message will appear:

Enter file name :

Type in '**TEST.TXT**', and press return (to enter the name).

The off-line menu will now be displayed with the file name at the top.

In order to give the file some content, we have to type in some text - because we'll need some text to save under the file name and we can't do that till we've typed it in. This is the creative bit - what you type for your first COMM message is completely up to you. Think about it a while.

Then we will enter on-line mode to place text into memory.

3) Entering text directly into memory, saving it to disk, and clearing memory.

You should still be at the off-line menu.

Press return and <ESC> H to go to the on-line menu.

At the top of the on-line menu are three entry command options, on-line with memory off, on-line with memory on and Type direct to memory. The * in the 'On-line - memory off' box indicates the currently active mode.

In order to change the mode to B (type directly to memory) type <ESC> and then <B>

(the <ESC> key is the current command key, which must be pressed before any other key from the menu).

The message

Type directly into Memory

appears on the screen, and you are ready to proceed. What you have just done is enter one part of the programme which enables you to type text directly into COMM's memory buffer - you can't edit text in this mode, but you are simulating what happens if you are on-line to another computer with memory switched on to capture any information coming in from the communications line.

(Normally, you would capture text whilst on-line with <ESC> <G> rather than <ESC> <B> - but since <ESC> <G> only captures text coming in from the communications port, and this is an off-line tutorial, we can't really capture on-line text yet. So we are using the <ESC> <B> function to place text into memory from the keyboard instead.)

Take some time to type in a couple of sentences.

Then press <ESC> then <E>, followed by <Y>, to return to the off-line menu.

We now have a file name, and some text, which can be saved as follows:

Press <S> (to save file). The messages:

Saving A: TEST .TXT
File created and opened
File closed.

will appear in succession.

COMM has created, opened and saved a file called TEST.TXT which contains whatever you typed.

Had a file already existed with the same name, you would have been asked if you wanted the contents of the file deleted, and replaced with your new memory contents. If you wanted to overwrite the old file, you would have typed Y for yes.

If you didn't want to overwrite, there would still have been several options open - you could have renamed the file on disk (with <Z>), or chosen a different name to save under (with <N>).

In the event that there had not been enough room on the disk, COMM would have closed the file, and told you that the disk was full, giving you an opportunity to change disks and save the file on a different one (after using <R> to reset the drives following the change if your operating system required it).

You've now placed text in memory and saved in to disk in more or less the same way as you would have done had you been on-line with memory switched on. Congratulations on your first successful session with COMM.

We can look at the text we've saved in a number of different ways, and we'll try a couple of them out next.

4) **Sending Memory, clearing memory, sending files, restoring memory.**

The send file and send memory functions are two of the 'transmission' options, which are one of the main reasons that COMM was written.

Make sure you are at the off-line menu (press ESC <E> to get there) and press <M> (send memory).

The following message appears -

Send to <S>creen <P>rinter <T>ransmit :

You now have three choices, which we will go through in turn.

i) Screen. The contents of the memory will be sent to the screen, which will (if the file is a long one) fill with characters and scroll up. Send the current memory to the screen, you will see your text travel up the screen, and begin to disappear off the top. At the end, the " press any key " prompt will appear at the bottom of the screen, and the space-bar will return you to the off-line menu.

Send the memory to the screen again, but this time, press the space bar when your text begins to appear on the screen. The scrolling will stop. You have suspended transmission. (Pressing any key will suspend transmission but we recommended you use the space-bar). The interrupt facility allows you to scan the file quickly. Having read through your text up to the point it stopped, pressing space-bar again will resume transmission to the screen.

Let the complete text finish displaying a second time, and get back to the off-line menu by pressing space-bar (when the press any key prompt appears). Now send the memory to the screen once again, and interrupt transmission as before by pressing the space bar. This time, don't press space-bar again to resume transmission, but press <ESC> . The familiar message to press any key appears immediately, without finishing the display to the end.

Once transmission has begun, any key will suspend it: but once suspended, only <ESC> will abort. All other keys resume transmission.

ii) Printer The contents of the memory will be sent to the printer if you have one.

iii) Transmit Whatever you have in memory will be transmitted via your communications port - this is unsuitable for experimenting with off-line.

The interrupt, suspend and escape commands will work in transmission mode for all of the above options. Play around with sending memory to the screen, suspending the display and aborting it. If you have a printer attached, try sending to the printer as well. Then press any key when asked to return to the off-line menu.

Now press <C> to clear memory. The message

Memory Cleared

will appear.

Now try sending memory to the screen again. You will find that all you get are messages saying

Memory Clear - aborted
... press any key

This is hardly surprising - we did clear the memory after all, and so there is nothing present. However, if you've been paying attention, you'll remember that we have our text saved in the file on disk.

So from the off-line menu again, this time, press <F> (send file).

The same message appears as when you sent memory -

Send to <S>creen <P>rinter <T>ransmit :

If you have a printer attached amd switched on, you could press P to print the text you saved. Otherwise, simply press S and you'll see you disk drive light come on, and your text will be typed on the screen just as you had typed in into memory, followed by the message

... press any key

Press the space bar and the off-line menu will be redisplayed.

Although you cleared memory, the file was saved on disk, and you saw the disk drive light come on when the text was read from disk. You can, if you like, try sending memory again just to make sure that memory is still clear.

Let's use one more useful feature of COMM and get back the memory that we just cleared out.

In reality what happened when we cleared memory was that the internal pointers, which mark the beginning and end of the memory contents, had been reset. The memory contents actually remained unchanged – unless written over by new data. This enables us, if we wish, to restore the memory we had before we cleared it, provided we haven't altered memory meanwhile by doing something like entering the editor, or switching memory on whilst on-line, or inputting a file into memory. The O function often works even if you have quit COMM with a Q command and then reloaded it by typing COMM, or even if you've reset your machine on some computers (though this isn't a recommended practise).

So now press <O> to restore the old memory.

The message

Memory Buffer Restored

will appear, indicating that the old memory has now been recovered.

So press press <M> to send memory, followed by <S> to send to the screen, you'll see your text just as it appeared when you sent it from the file on disk, followed by the

... press any key

prompt you should now be used to.

The O (Get old Memory) function reverses the action of C (Clear Memory). COMM gets rid of its existing end of file marker and it

looks through the rest of memory for the old one. It marks this as the top of the memory buffer, so that the contents of the memory buffer (which haven't been overwritten) can be accessed from this marker down to the bottom of the memory and displayed or saved to disk.

A repetition of O would cause COMM to reach higher up in the memory to find and mark the next EOF and so on, until eventually it will tell you that the entire memory available to you is now restored. This, incidentally, is usually 64K on 16-bit systems and between 25,000 and 30,000 characters on 8-bit systems – sometimes a little more, sometimes less, depending on the size of the operating system on your particular computer. We do not advise trying to restore memory like this – the Get Old Memory feature was only designed to reverse the action of Clear memory. If you do get a message telling you that the entire memory contents are available, you should regard it as a danger signal – in normal use, this message never appears.

You should now be at the off-line menu again, but this time you should have some text in memory. We'll now edit it with the in-built editor in COMM.

Press the space bar and the off-line menu will be redisplayed.

Although you cleared memory, the file was saved on disk, and you saw the disk drive light come on when the text was read from disk. You can, if you like, try sending memory again just to make sure that memory is still clear.

Let's use one more useful feature of COMM and get back the memory that we just cleared out.

In reality what happened when we cleared memory was that the internal pointers, which mark the beginning and end of the memory contents, had been reset. The memory contents actually remained unchanged - unless written over by new data. This enables us, if we wish, to restore the memory we had before we cleared it, provided we haven't altered memory meanwhile by doing something like entering the editor, or switching memory on whilst on-line, or inputting a file into memory. The O function often works even if you have quit COMM with a Q command and then reloaded it by typing COMM, or even if you've reset your machine on some computers (though this isn't a recommended practise).

So now press <O> to restore the old memory.

The message

Memory Buffer Restored

will appear, indicating that the old memory has now been recovered.

So press press <M> to send memory, followed by <S> to send to the screen, you'll see your text just as it appeared when you sent it from the file on disk, followed by the

... press any key

prompt you should now be used to.

The O (Get old Memory) function reverses the action of C (Clear Memory). COMM gets rid of its existing end of file marker and it

looks through the rest of memory for the old one. It marks this as the top of the memory buffer, so that the contents of the memory buffer (which haven't been overwritten) can be accessed from this marker down to the bottom of the memory and displayed or saved to disk.

A repetition of O would cause COMM to reach higher up in the memory to find and mark the next EOF and so on, until eventually it will tell you that the entire memory available to you is now restored. This, incidentally, is usually 64K on 16-bit systems and between 25,000 and 30,000 characters on 8-bit systems - sometimes a little more, sometimes less, depending on the size of the operating system on your particular computer. We do not advise trying to restore memory like this - the Get Old Memory feature was only designed to reverse the action of Clear memory. If you do get a message telling you that the entire memory contents are available, you should regard it as a danger signal - in normal use, this message never appears.

You should now be at the off-line menu again, but this time you should have some text in memory. We'll now edit it with the in-built editor in COMM.

Screen Display - Edit Help Menu

```
        *  E D I T   H E L P   M E N U  *
-------------------------------------------------------
 HELP = <ESC> H or ^O      |    Redisplay  page= ^B
---------------------------|---------------------------
 Cursor Control            |    Screen Scroll
UP    = ^E or ^K or <ESC> A|  Line  UP      =^W
DOWN  = ^X or ^J or <ESC> B|  Page  UP      =^R
LEFT  = ^S or ^H or <ESC> D|  Line  DOWN    =^Z
RIGHT= ^D or ^L or <ESC> C |  Page  DOWN    =^C
-------------------------------------------------------
 Delete                    |  Insert
Character LEFT = <DEL>     | Character= ^V or <ESC> @
Rest of Line   = ^Y or     | Line     = ^N or <ESC> L
    <ESC> J or <ESC> M     |--------------------------
Character RIGHT = ^T or    |     FIND   |  Exit Edit
        ^G or <ESC> N      |  = <ESC> F |  = <ESC> E
-------------------------------------------------------
 User sequences       .....  to embed any Control
DEFINE = ^U or <ESC> P     | Characters, prefix with
INVOKE = <ESC> 1 - 9       |         ^P or <ESC> R
-------------------------------------------------------
 NOTE: ^Q can be used instead of <ESC>.

... Press any key
```


5) Introduction to editing, entering and formatting text with the editor.

You should still be at the off-line menu.

Press <E> (edit memory) followed by <N> in response to the prompt:

Format text (Y/N)

The screen will clear, and your text will be displayed on the screen. You can now edit it as you wish. At any time, pressing <ESC> and then <H> will display a list of the possible commands for moving the cursor around the screen, inserting and deleting characters or lines, scrolling the screen up and down, and so on. The editor is described in more detail in the supplementary reference guide, and we shan't go into its rather basic features here. You can't go a lot wrong simply be experimenting with moving around. You will notice that the editor operates in overtype rather than in an insert mode - in this it is different to most word processors. You can insert spaces and lines if you want to however. Look at the help screen with <ESC> <H> to see how. A printout of this screen is on the preceding page.

After you have finished playing with the editor, press <ESC> and then <E> to exit the editor. You will find yourself back at the off-line menu. You can try sending memory to the screen if you like, to review what you have done.

Let's edit the text again, but this time, when you see the prompt

Format text (Y/N)

press <Y> for yes.

You are now faced with four options:

(N)arrow (37 col Viewdata) (W)ide (80 col)
(T)lx (68 col, Caps) (M)edium width (68 col)

File exists, Delete Y/N ?

You are being reminded that you already have a file containing text stored under that name. Press <N>. You will see the message

File Delete denied,create aborted.

Now, from the off-line menu, press <N> (Name File).

When the message

Enter File Name :

appears, type in **TELEX.TXT** and press return to enter the new name.

By pressing <S> we can save the text of TEST.TXT which has been formatted in telex form under the new file name TELEX.TXT, and transmit it at our leisure.

6) **Setting up user keys and inputting files into memory.**

COMM has a number of user-definable keys which are designed to make it easy for you to type commonly-used text strings and commands with just one keystroke.

From the off-line menu, you can define a user key by typing <U>. The prompt

Change which User key ?

appears. You can enter any number from 0 to 9. Type <1> to define user key 1. The prompt

New User key 1 :

appears on the screen.

At this point, type in **PLEASE REPLY TO:** (no return yet) and then press <ESC>, then <2> and then press return to finish loading the

These options are as simple as they seem. (N)arrow = 37 columns wide - this is designed for Viewdata pages. Though these are actually forty columns wide, the last line on a frame is usually 3 characters shorter than all the others. (M)edium = 68 columns wide - this is normal A4 text. (T)lx = telex = 68 columns wide - this is the same as the medium width, but formats all text in memory to capitals. Finally (W)ide = 80 colums wide - this makes maximum use of most CRT screen displays. To carry on into the editor without formatting at all, simply press any other key - the effect is the same as if you had typed N when asked if you wanted to format the text, and the width will default to the last one you used (or medium if you haven't selected a width at all). If you press the <ESC> key, you will be returned to the off-line menu and the edit command will have been cancelled.

As described above, the T option in formatting will display the text in telex format, 68 columns wide, in capitals. It is worth a special note, for once formatted in capitals for telex transmission, although you can reformat the width of the colums, the text will remain in capitals. You may have already discovered this by accident if you've been experimenting. Bear this in mind when in the editor for if you now exit and save the file in its reformatted upper case form accidentally, you will have to re-type it if you want to send it in upper and lower case form as a conventional document.

Let us assume that your text was in fact a telex. Press <T> for telex format at the prompt. The editor screen will appear again with your text on it. You'll find that your text is wrapped around to 68 column lines and that all lower case characters are turned into upper case. Of course, if you had originally typed all upper-case letters with double-spaced lines no longer than 68 characters you won't notice much difference.

Press <ESC> and then <E> to leave the editor again. We'll suppose that you now wish to save it for transmission later.

From the off-line menu press <S> (save file) to elicit the following response:

Saving A: TEST .TXT

user key. We have just defined user key 1 to generate the text "PLEASE REPLY TO:" and then to chain up user key 2.

Now repeat the above procedure with the difference that you should now press <2> instead of <1> to define user key 2, and type in your name followed by a return as the text string to load into user key 2. Though you could type your name in lower-case, use capitals, since we are are going to put your name on a telex.

We now have two user keys defined. We can use these either in the editor or when we are on-line. We could, if we had wished, entered commands as well as text into the user keys (such as <ESC> and <S> to save when on line) in the same way as we have chained user key two to user key one. But since this is an off-line tutorial, we'll use these key sequences off-line in the editor.

Remember the test file we just created, and re-formatted as a telex? Make sure that it is named as the current file on your off-line menu. If you've been following this tutorial in sequence, the file should still be in memory. However, let's suppose that you switched off your computer and went to have a well-deserved lunch (or sleep, if you began this in the afternoon).

Make sure that you are at the off-line menu, and that the TELEX.TXT file has been named (with <N>). Also, make sure that memory is clear with <C> from the off-line menu.

Now press <I> (for Input File). You will see a message -

Continuation read (Y/N) ?

COMM is asking whether you want to read the file in from the beginning, or from the last point at which it was read. Since we would only need do a continuation read if the file was too big to fit into memory completely, we don't want a continuation read.

So press <N>. You will see the message

Reading Input File
File now in Memory

What you have just done is to read the TELEX.TXT file from disk into COMM's memory buffer so that you can edit it - this works in exactly the same way as a word processor accesses files on your disk.

Now press <E> to enter the editor, followed by <N> for no formatting since the text is already formatted.

Now scroll down in the editor (with <^Z> or <^C>) to the end of the file if your original was bigger than a screenful, and position the cursor at the beginning of first blank line after the last line of the message. Let's use two of the user keys to sign off the telex we composed.

Now invoke the user keys with <ESC> <1>. The sign-off message will be placed on the screen with the first character at the cursor position when you typed <ESC> <1>. You'll now see your original telex, together with the sign off message.

You can alter or edit the text some more if you wish, and then save it in its new form by returning to the off-line menu with <ESC> and then <E>, and then pressing <S>. You will be told that a file exists, and asked whether you want it deleted (and the current memory contents saved under that file name) just as happened the first time we tried to save an existing file

<Y> will save your new telex in place of the original version, and <N> will abort the saving operation and return you to the off-line menu, but with the new version still in memory ready for an alternative file name.

Try aborting the save command with <N> for No, and then press <N> for Name File to respecify the file name as TELEX2.TXT.

Then press <S> to save under the new name, and then pressing any key returns you to the off-line menu.

not be displayed). All input and output in COMM is buffered: this means that information is held in temporary store if it comes in at the wrong time. There is room in the temporary receive buffer for approximately 4000 characters.

Once again, return to the off-line menu when asked to press any key, and press <A> for Append to File. The messages

Saving A: TELEX2.TXT
File Closed

appear, and your telex with the sign off addition has been appended to (stuck to the end of) the new file. Pressing any key returns you to the off-line menu. Send the file to the screen with <F> and then <S> - you will see the whole message appear on the screen twice. The first time is from when you saved it, and the second time is from when you appended to same text to the original file.

Obviously, there isn't much point in appending an already saved message. All we've done is to show you how to place text from memory at the end of an already existing disk file instead of saving it under a new name. So, if you regularly use an electronic mail service, you could have one large file called MAIL.IN, and capture all incoming messages in memory each time you log on, and then append then to MAIL.IN. The file would then contain your incoming mail in chronological order. And for a separate version of just one session's mail, you could print out the contents of memory.

COMM is flexible enough to handle many different methods of filing text and messages.

Meanwhile, you are left with a file on disk called TELEX2.TXT which contains two copies of the same message. You'll see from a directory that there are three files which we have created, TEST.TXT, TELEX.TXT, and TELEX2.TXT. The file TEST.TXT was our original effort. TELEX.TXT is the same thing in telex format, and TELEX2.TXT has a double copy of TELEX.TXT with an added sign-off message. We will now delete the silly TELEX2.TXT file.

7) File Directories, Appending files and Deleting files

You should still be at the off-line menu.

Press <D> (for directory). You will be asked

Specify files:

Type *.* and the directory of all the files on disk will be displayed. You will see at least the following files displayed in this format -

TEST .TXT : TELEX .TXT : TELEX2 .TXT : COMM .COM

followed by the familiar

... press any key

message on the screen.

In case you are unfamiliar with the wildcard * character in a directory scan, it means simply match any name.

Now press any key when asked, and try the same thing again from the off-line menu, but instead of *.* try typing ***.TXT**. This time you'll see only the following:

TEST .TXT : TELEX .TXT : TELEX2 .TXT :

... press any key

on the screen - only the file with the .TXT extension have been displayed.

You can access the directory and the off-line menu (or any of the menus) at any time, without losing any information. Only on-line mode with memory on, using the editor, or inputting a disk file can affect memory in the computer. Further, any transmissions sent to you will be (assuming you are set up correctly) saved in a special receive buffer even if you are looking at a menu (so characters may

Before we delete any unwanted file it has to be specified as the current file. The delete file command deletes the file named as the current file, and although you are given a prompt, it is obviously important to check that the file you wish to delete has been named correctly. In this case, there is no problem, since the named file really is TELEX2.TXT

From the off-line menu, press K to delete (Kill) a file.

You are prompted

Delete file (Y/N) ?

If you had changed your mind, and didn't wish to delete the file, pressing <N> would return you to the off-line menu. However, <Y> will delete the file, and give you the prompt

File Deleted.

You can then press <S> to resave a single copy of TELEX2.TXT.

Remember, when deleting files, it is important to check that the file to be deleted has been named as the current file.

5. SUMMARY

We have been through as much as it is possible to cover in off-line operation. If you have followed the off-line tutorial carefully, you will be reasonably familiar with the way that COMM works and at least some of the range of functions it offers.

You will now, of course, want to connect up to another computer and go on-line. Nearly all of the functions we have covered in the tutorial are available directly from the on-line menu by prefixing them with the command key (default is <ESC>). You will probably find that the experience you have gained in off-line operation, together with the relevant sheets in the cookbook, will enable you to manage successfully on-line too.

However, there are certain areas which cannot be properly covered in a generalized manual which may affect the success of your usage of COMM. The program can be run on many different computers and operating systems: and more importantly, it can be used to connect your microcomputer to any number of host systems.

Because it is impossible for us, as the authors of this package, to predict the exact nature of the host you will connect up to, we have made many of the features of COMM in its on-line operation as flexible as possible. We suggest that if you run into problems the first time you connect up to something and send text automatically, you now read the section in the Supplementary Reference guide on **Handshaking** carefully, and also look at the **Appendix** for any notes on your particular computer system.

If you do experience any problems when you connect up on-line, you will then know what to do about it.

You will probably also want to read the sections in the Supplementary Reference Guide on the **user keys**, and also on **saving a customized version of the program**. The **editor** is also discussed in more detail in the Reference Guide, and to get the most out of its use, you may well want to read that section and experiment further with its operation. One additional utility which we have not covered here but which is both simple and useful is the **typewriter** mode.

If you want to use COMM to transfer files between two microcomputers via the upload-download options, read the section on **Block Transfer** in the Supplementary Reference guide.

And if you want to write your own programmes to run within COMM, automating communications or designing your own menus and prompts, read the **Language Guide** which will tell you all about COMM JCFs.

Finally, please do not hesitate to write to us with suggestions for improvement of either the program or the manual.

COMM+ Cookbook

(this is a blank page)

1. INTRODUCTION

a. This cookbook is NOT a full user manual. It is designed for people who want to use COMM for relatively simple electronic mail type applications with a little bit of file transfer and have neither the time nor the patience to read manuals. People who do have the time and patience ought to work through the Off-line Tutorial, and then read the Supplementary Reference Guide.

b. The Cookbook consists of a series of pages designed to lead you through various parts of the program. No attempt is made to explain much about what the program is actually doing, or how it works.

c. The Cookbook is designed to be used as a short-cut. Should the user find a situation in running the program not covered by it, we advise referring to the fuller guides in the Off-line Tutorial and the Supplementary Reference sections of the manual.

d. We have attempted to make each page as self-contained as possible, but at the same time the pages are in a logical sequence. Especially at the beginning of the cookbook, it is important to read the pages in the correct order. Further on it is not so important: cross-references are given where necessary.

e. Refer to the Glossary for explanations of terms. Refer to the troubleshooting guide if problems arise. It may suggest immediate remedies. Also refer to the Glossary if you come across unfamiliar terms.

f. Refer to your computer manuals or to your electronic mail/host computer manuals for items outside the scope of this set of manuals. It is not feasible to document fully all possible uses for COMM. We suggest you make your own, internal, office additions to the cookbook where necessary for your system and installation.

TABLE OF CONTENTS

2. EQUIPMENT CHECKLIST

a. A computer with at least one floppy disk drive, a keyboard, a screen, a communications interface, at least 56 KBytes of memory, running under any one of the CP/M-80, CP/M-86, CCP/M-86, CDOS, MSDOS or PCDOS operating systems (or derivatives thereof).

b. A copy of the COMM communications processsor on disk, properly configured for the computer and operating system it is to run on.

The program is called COMM.COM, and you can find it on the disk by typing DIR. Additionally, on almost all versions a file called DRIVER should be on the disk the first time you run COMM. This is provided on your master diskette, which ought to be copied. (On CP/M-86 versions of COMM, an additional programme called COMM.CMD should be on the disk as well as COMM.COM).

c. A computer to communicate with. If the connection is direct, you will need a suitable cable between the two machines. (Consult Appendix if in doubt about this).

d. If you are going to communicate with a computer over a telephone system you will need a device to connect your computer to the telephone line. Either a directly connected modem or an acoustic coupler will do the job. Again, you may need a cable too.

e. If you are going to use a public computer communications network, such as an electronic mail service, you will also need a subscription and a password.

f. Depending on the above, your telephone should be within reach of your computer.

g. A supply of floppy diskettes for backup purposes.

3. STARTING UP

a. Plug your computer into the mains socket.

b. Connect your acoustic coupler or modem to the communications port on your machine. It is usually a long D shaped socket somewhere on the machine. It may possibly be labelled "SERIAL", "COMMUNICATIONS", "MODEM" or "RS232" but more likely than not is is completely unlabelled. If so, look at the pictures in your manual. Have your telephone handy, but do not leave it next to your diskettes, as magnetic fields from the telephone bell can have a harmful effect.

c. Switch the computer on and load the operating system (CP/M, CP/M-86, MP/M, MPM-86, CCP/M-86, MSDOS or PCDOS are all operating systems). Yours may load up automatically, or you may have to press the RETURN key on your keyboard, depending on your machine and your operating system.

You should get a prompt like this:

```
A>
```

When you do get this prompt, simply type **COMM** and press the return key.

d. Your COMM program will now load up and display its copyright message on the screen.

If you are using a JCF front-end, then you can simply follow the instructions you see on your screen, as your programme will now automatically be configured for whatever tasks the front-end was designed to perform.

The rest of the cookbook relates either to use of the programme without a front end, or to use of a front-end once it has performed its task. For instance, if your JCF automatically dials up and logs on to a computer for you, then you wouldn't need to read section 4 of this Cookbook : but if the JCF did no more than that, you'd certainly need to read from section 5 onwards.

4. HOW TO BECOME A TERMINAL TO A HOST

a. Follow the instructions on the STARTING UP page.

b. When the copyright message appears, press any key.

c. Another screen display will appear telling you to 'ensure all connections in order'. If you have been following the instructions so far, then they will be.

d. Press the <RETURN> key.

e. A message will appear telling you that you are in ON-LINE MODE with memory off. This means that your computer is acting as a terminal to whatever you are connected to via your port (as decribed in 3 on the STARTING UP page) and that any data sent or received will simply be displayed on the screen and not saved in the memory of your computer. The message will also tell you to type <ESC><H> for help: this means that pressing the <ESC> key followed by <H> will display a screen telling you all the things you can do in case you forget and don't have the manuals handy.

f. The next step depends on the type of comunications equipment you may have:

If you are using an acoustic coupler:

> Plug it in and switch it on if necessary.
>
> Pick up your telephone, dial the number of the computer or service you want to access, and when the connection is established, the phone will whistle. Now insert the telephone handset firmly into the cups on your coupler, making sure it is the right way round. If your coupler has a little light labelled "CARRIER", wait for it to come on (otherwise leave it for about 10 seconds). You should now be connected to the other computer - called the HOST machine. Follow the instructions or manuals for the host computer from now on.

If you are using a direct-wired modem:

> Procedures vary for different models of modem. Usually you dial the number on your telephone, and then press the DATA button or the ONLINE switch, or simply replace the handset. Follow the instructions which were supplied with the device you are using.

If you are using a software-driven modem:

> Some modems are software-driven. What this means is that you need not touch the telephone at all, but can tell the modem the number you want to connect up to and it will dial up by itself. For such modems, you simply type the instructions for the modem in at the keyboard. Consult your modem handbook for full details.
>
> For instance, the dial command for the Hayes modem is ATD followed by the number, to be entered after the OK prompt. For the Master Systems modem it is D followed by the number at the : prompt, for the Dacom modem you simply enter the number at the ++ prompt, and for the Racal modem you type LN(number) and then DN at the * prompt. All these modems need the command terminated with a carriage return.

If you are directly connected:

> Simply go ahead and use your computer as if it were any other terminal.

5. HOW TO SEND TEXT FROM YOUR OWN DISKS

a. We assume that you have followed the instructions on the STARTING UP page, and also the instructions on the HOW TO BECOME A TERMINAL TO A HOST page, and that everything is working. We also assume that you are at a point in your session on the host where it makes sense to send text, and that you have already generated suitable text on your computer, either with COMM or with a word-processing package - see the ENTERING TEXT INTO THE MEMORY OF YOUR COMPUTER page.

b. The text you want to send will be stored on your own computer's disk system as a disk file. You must tell COMM the name of the file.

c. If you already know the name of the file, miss out this step. If you do not know the name of the file to send, press the key marked <ESC> followed by the <D> key followed by <RETURN>. A directory will appear on your screen (don't worry, it isn't being sent to the host computer), followed by a message saying 'Control Returned'. (This message appears quite often: all it means is that the computer has finished what it was told to do and is awaiting further instructions.) Make a note of the file you want.

d. Press the key marked <ESC>, followed by the <N> key. You will be asked for the name of the file. Type in the file name, and press the <RETURN> key.

e. Press the <ESC> key again, followed by the <F> key. You will be asked whether you want to send a disk file. You must type <Y> to send the file. You will then be asked if you want to send to the screen, or to a printer (if you have one), or transmit to the host.

f. If you want to check that the file you have named is the right one, type <S> and it will be displayed on the screen first. If you named the wrong file, go back to step 3 above.

g. If you decide at this stage you don't want to send the file after all, press the <ESC> key and you will be back in on-line mode.

h. To transmit the file press <T>. The text in the disk file will now be sent to the host computer just as if you were typing it in yourself at the

keyboard, only much faster and more accurately. Your host will echo what you are sending back to your screen.

i. If you want to stop sending the text for any reason, simply press the space bar. You can resume sending by pressing the space bar again, or break off the transmission by pressing <ESC>. This also applies if you are simply sending to the screen for checking.

j. When the file has been completely sent, the '**Control Returned**' message will appear on your screen telling you so. You may now continue in on-line mode.

6. HOW TO RECEIVE TEXT INTO YOUR OWN COMPUTER

a. We assume that you have followed the instructions on the STARTING UP page, and also the instructions on the HOW TO BECOME A TERMINAL TO A HOST page and that everything is in working order.

b. Text can easily be received into the memory of your computer, but before beginning any reception, you will probably want to make sure that you have nothing already in memory. If you do have something there, anything received will be appended to it. (Of course, this may be what you require).

c. To clear the memory (also called the buffer) press the <ESC> key followed by the <C> key. You will be prompted as to whether you do want to clear the memory - type <Y>. You will then be back in on-line mode, with the **'Memory Cleared'** message appearing on the screen. On some systems, an additional **'Control returned'** message will also appear, depending on whether a status line is used in the software.

d. Now press the <ESC> key followed by the <G> key. A message will appear telling you that you have entered **On-Line Mode - Memory On**. This is the same as the on-line mode you were in before, with the following difference: any information sent out or received will be saved in the memory of your computer.

e. When you have received the text you wanted, press the <ESC> key followed by <T>. You will see a message on the screen that you have returned to ordinary on-line mode, with the memory off.

f. SUMMARY - You can tell your host to type something on your screen, type <ESC><G>, and once it has been sent to you, type <ESC><T> to switch memory off: or you can leave memory on all the time. That way you save everything (but may have to do some editing later on to pull out the bit you want).

7. HOW TO SAVE THE MEMORY OF YOUR COMPUTER

a. The memory of your computer is volatile: this means that it is going to vanish once you switch off the machine. You will want to know how to save the contents of memory, with any text you have received, so that a permanent record will be available whenever you need it.

b. You will also want to save the memory if you are using COMM with a word processing program. If this is the case, text saved from memory can be edited virtually as a normal word processing file.

c. We assume that you are in on-line mode when reading this page. Your memory is to be saved as a file on your floppy disk. You must decide on the name you want to give the file.

d. Press the <ESC> key, followed by the <N> key. You will be asked to enter the name of the file, just as explained on the HOW TO SEND TEXT FROM YOUR OWN DISKS page. Do this, finishing by pressing the <RETURN> key.

e. Press the <ESC> key again, followed by <S>. The computer will then save your memory as a file on disk, and tell you when this has been completed. First you will be asked if you do want to save - type <Y> if you do. Messages will appear telling you that the file has been created, opened, and closed, possibly ending with a **'Control Returned'** message. You will then be back in on-line mode.

8. ENTERING TEXT INTO THE MEMORY OF YOUR COMPUTER

a. If you want to send some text (like an electronic mail message) to a host computer, you can create it with a word processing package, save it on disk as usual with word processors, and send it as a disk file.

b. For short messages it is often simpler and quicker to enter the text directly into the memory of your computer as explained below.

c. We assume you begin in on-line mode, and have followed the instructions on the HOW TO BECOME A TERMINAL TO A HOST page. Press the <ESC> key followed by <B>. A message will appear telling you that you can type directly into the memory buffer. You do not have to be connected to anything yet, as you are only talking to your own computer.

d. You will probably want to clear the memory of anything that is in it already - see the comments on how to do this in HOW TO RECEIVE TEXT INTO YOUR OWN COMPUTER page. To do this, press <ESC> followed by <C>. When prompted for confirmation, type <Y>.

e. You can now treat the computer screen as a blank piece of paper, and type on it from the keyboard. If you fill the screen, it will scroll up to accomodate the extra text you are typing. Press <RETURN> at the end of each line to begin a new one and correct errors with the backspace or delete keys.

f. When you have finished entering text, press <ESC> followed by <T>. This will put you back into on-line mode. You can then send (or save) the text you have entered - see the next page.

9. HOW TO SEND TEXT FROM COMPUTER MEMORY

a. If you have entered text directly into the memory of your computer as described on the previous page, you will probably want to send it.

b. The procedure is similiar to sending text from your own disks, with the difference that instead of typing <ESC> followed by <F> to send a disk file, you press <ESC> followed by <M> to send the contents of memory.

c. We assume, as usual, that you are still in on-line mode. Press the <ESC> key followed by <M>. You will be asked whether you want to send the memory to the screen, or to a printer, or transmit it. Type <T> to transmit. (This is much the same as sending disk files). The text in your memory buffer will be sent to the host computer just as if you were typing it yourself at the keyboard, only much faster and more accurately. Your host will echo what you are sending back to your screen.

d. If you want to stop sending the text for any reason, simply press the space bar. You can resume sending by pressing the space bar again, or break off the transmission by pressing <ESC>. This also applies if you are simply sending to the screen for checking.

e. When the memory has been sent completely, the '**Control Returned**' message will appear on your screen. You will then be back on ordinary on-line mode.

f. You can also save the text you have typed into the memory buffer as a disk file. The procedure is described on the HOW TO SAVE THE MEMORY OF YOUR COMPUTER page.

10. LEAVING COMM

a. When you have finished using COMM, you can leave the program directly from on-line mode by typing <ESC><Q>.

b. You will be prompted as to whether or not you really do want to quit: if you do, type <Y>, and the session will be over. You will be returned to the operating system level.

c. At this point you ought to take backups if you have created or saved any important text. Use the backup or copy utilities described in your system manuals.

d. Remember to remove your disks before switching off. Put them in their jackets, to protect them from harm. Also remember to make sure that you telephone is not still on-line.

11. SELECTING DISKS AND DRIVES

a. It is common for many machines to have dual floppy disk drives. On such systems, most people prefer to use one disk drive for programs and the other for data or text.

b. You can select the disk drive(s) for the files you name, send or save within the on-line modes in COMM by typing <ESC><V>. You will be prompted with a message asking you if you want to select a target disk drive. Type <Y> for yes, then a letter. Usually this will be <B>, for the second drive on a two drive system. You will then be returned to on-line mode. You will now find that all disk operations (directories, sending, saving etc.) will operate from the drive you just selected.

c. An alternative method of selecting a drive is simply to enter the drive letter followed by a colon (:) at the point at which you name a file - as explained on the HOW TO SEND TEXT FROM YOUR OWN DISKS page. Do this, finishing by pressing the <RETURN> key. This will not, however, affect the drive the directories will come from.

12. CHANGING DISKS

a. It is also common to change disks within a program - especially in one like COMM where you are probably going to have to send different files from different disks.

b. Whenever you change a disk, the operating system may need to be told you have done so or it may not allow you to save to the new disk from your program - operating systems vary in their implementations, and this may not be essential on all systems. However, even where unnecessary, it will do no harm. From on-line mode, you can reset the disks by typing <ESC><R>. Follow the prompts, and after the drive has been reset, you will be back in on-line mode (or whatever mode you started in).

13. THE OFF-LINE MENU

a. Until now, all the operations we have been describing have been taking place from within the on-line mode (simple or buffered, with memory on).

b. There are other facilities which COMM offers, and which you may want to use, which cannot be accessed directly whilst you are on-line to another computer. To use them, you have to go to the off-line menu.

c. From the on-line mode, press <ESC> followed by <E>. You will be prompted as to whether or not you want to exit to the off-line the menu. Type <Y>, and the off-line menu will appear.

d. Many of the things you can do whilst on-line can also be done from the off-line menu, with the difference that you don't press <ESC> first - thus to name a file, type <N> rather than <ESC> followed by <N>. If you want to know more about the functions on the off-line menu, refer to the more detailed tutorial section of the user manual.

e. In particular, in this quick cookbook we cannot give any real guide to the workings of the editor in COMM - these do require practice and a proper reading of the manuals.

f. For the remaining pages of this cookbook, we assume that you are at off-line menu rather than in on-line mode. (You can get directly to the off-line menu by pressing <ESC> rather than <RETURN> after you first start up, as described in the HOW TO BECOME A TERMINAL TO A HOST page.

14. INPUTTING A FILE DIRECTLY INTO MEMORY

a. Disk files may be input directly into memory. This means that there are, so far, three ways that text can get into the memory buffer of your computer: first, by receiving it whilst on-line with memory on, second, by typing it in youself, and third, by inputting from disk as we are about to describe below.

b. From the off-line menu, clear memory by typing <C>. Just like the on-line mode, if you don't clear memory, it will still accept input, but the text is tacked on to the end of whatever is there already.

c. From the off-line menu, type <N> to name a file, and enter the file name as described in the HOW TO SAVE THE MEMORY OF YOUR COMPUTER page.

d. Now type <I>. The computer will ask you if you want a continuation read (the alternative is to input from the beginning of the file). Type <N>. The file will be read, and a message saying '**Reading Input File**' will appear. Once the file is read, a second message saying '**File now in Memory**' will display.

e. The file now in memory will behave exactly like all the other contents of memory. (The reason why this option exists is because we can do things with the contents of memory which we cannot do with disk files, like editing and formatting.)

15. EDITING AND FORMATTING THE MEMORY FOR TELEX

a. This page is not a full guide to the editor. You must read the Supplementary Reference Guide for that. All we do here is set out the steps for formatting and slightly changing text.

b. Type <E> from the off-line menu to get to the editor. You will be asked if you want to format text. Let us assume that you want a telex format. Type <Y> to format, then <T> for telex format.

c. The screen will now display your text in telex format - all letters will be upper case and text will be reformed to the 68th column.

d. This might result in some odd-looking text - the formatter wraps lines around at 68 colums unless they end in a full stop or are followed by a blank line. You can edit by simply overtyping.

e. Use the arrow keys on most keyboards to move around the page. Otherwise use the control key with E,S,D,and X. Remember to hold down the control key while typing the letter.

f. Press <ESC> then <H> to get a summary of the edit commands available. The ones you will probably want to use most frequently are <control-N> to insert a new line, <control-G> to delete what is at the cursor, <control-Z> to scroll down through the text, and <control-W> to scroll up.

g. When your text looks good enough to send, press <ESC> then <E> to exit from the editor. Your memory can now be saved as a disk file for later use, or you can send it using on-line mode (press <RETURN> from the off-line menu, after exiting from the editor, to get back on-line.)

16. TROUBLESHOOTING GUIDE FOR COOKBOOKERS

PROBLEM: No directory of disk is possible, and programme doesn't load.

DIAGNOSIS: Faulty drive or incorrect disk format.

SOLUTION: Try looking at the disk from the second drive. Remember: the master disk cannot boot up (see appendix). Check disk format label: possibly the supplied version, though not for your machine, is supplied on a readable format for drive B by using a system utility. If all fails, report problem to your supplier.

PROBLEM: No communication seems to be possible even though everything appears to be plugged in and working.

DIAGNOSIS: Probably you are trying to transmit into the receive pin and receive from the transmit pin over the communications link - see the Appendix. Alternatively, your computer may be running at the wrong speed, or may be configured for the wrong communications port should your machine have more than one available for use.

SOLUTION: For the first case, see if a serial printer will work using this port. If it does, you need a null modem cable (see glossary). This will simply swap over the transmit and receive pins on your serial port. For the second case, you will either need to change the speed or address number of the serial port - see the section on Baud Rates in the Supplementary Reference guide - or else use a special cable. See the appendix for advice on cables.

OTHER POSSIBILITIES: Port not working, Coupler or modem not working or wrong type or improperly installed program.

SOLUTIONS: Contact relevant suppliers.

PROBLEM: Nothing works once program is loaded. Computer may even hang up, needing to be powered down.

DIAGNOSIS: Improperly installed software or badly initialized port.

SOLUTION: Reinstallation of either operating system or COMM needed. Contact your supplier.

PROBLEM: Communication appears to take place, the coupler whistles or modem works, but the screen displays rubbish.

DIAGNOSIS: Baud rate and/or word length and/or parity set up incorrectly.

SOLUTION: Set correct baud rate etc. Some installations will allow COMM to do this internally. Others require you to exit to the operating system and use a utility program called SETUP or CONFIG or something similar to change the speeds. This may require a reload of the operating system (cold boot - press reset) afterwards, followed by restarting COMM - check your computer manual if unsure about this. See the section on Baud Rates in the Supplementary Reference guide.

PROBLEM: On sending files, characters get lost at certain points rather erratically.

DIAGNOSIS: COMM is sending faster than host can receive.

SOLUTION: Set up or alter the WAIT handshake. See Supplementary Reference guide section of user manual for details.

PROBLEM: Cannot save files. Programme crashes out to the operating system with incomprehensible messages.

DIAGNOSIS: Diskette may be write protected - or disks may have been changed in mid-program without resetting.

SOLUTION: If diskette is read only, (write-protected), save files on a normal diskette on other drive. Reset disk system within COMM if you changed disks. If you need to reload COMM, you may be able to use the restore old memory option to rescue you data - look this option up in the index.

PROBLEM: Communications link is sometimes unstable, leading to periods of garbage appearing on the screen.

DIAGNOSIS: The public-switched telephone network is known for occasional bursts of noise on lines, and occasional bad connections.

SOLUTION: Hang up phone, wait, and redial. If running at 1200 baud on a dual speed modem, try 300 baud instead after changing speed in COMM.

PROBLEM: As above, but this time permanent.

DIAGNOSIS: Permanent fault somewhere. (!)

SOLUTION: Check fit of handset in acoustic coupler, check all plugs and cables, call engineer to check out telephone, possible fault on coupler or modem itself (they can vary - contact supplier for service).

COMM+ Supplementary Reference Guide

(this is a blank page)

1. INTRODUCTION

The supplementary reference guide is, as its name implies, supplementary. Many of the functions available within COMM which have been mentioned in the off-line tutorial are straightforward and need no further elaboration. For instance, many of the disk file operations, such as directories, deleting, saving and so on are perfectly straightforward in operation from both the on-line and off-line modes, and they will not be discussed further here.

The topics we shall cover here are those where there are subtleties and details which ought to be mentioned in a manual, but which were unsuitable for inclusion in the tutorial since we wanted to make all the coverage there both suitable for off-line experimentation, and as compact and simple as possible.

We do not think that this supplementary reference guide is suitable for consumption only by experts - it has been written to be as clear as possible. Nevertheless, computer communications, even at the relatively simple asynchronous level at which COMM operates, is still sometimes a complicated business. In trying to allow for these complications, we have of necessity made parts of the programme complicated too. This seems unavoidable. The manual is structured as it is in order to avoid these complications intruding unnecessarily into areas where things are actually quite simple.

TABLE OF CONTENTS

2. THE USER KEYS

The user sequences are programmable keys, which are invoked by typing <ESC> followed by any number from 0 to 9. They are alternatively invoked on IBM PC systems running under PCDOS by pressing one of the function keys. They are available for programming in any of the modes COMM operates in - on-line, off-line and edit. However, they cannot be invoked from the off-line menu. Note that to invoke a user key whilst on-line, you need to prefix it with the current command key.

This is by default (though it need not be) <ESC> - we use the default <ESC> for the command key in this reference guide throughout.

On pressing ^U, or <ESC> P in the editor, U while off-line, or <ESC> U while on-line, you will be prompted to say which of the user keys you want to change. Enter a number from 0 to 9. Suppose you enter 0. The message

New User key 0 :

will appear.

Now type a line of characters (maximum of 20), which will be assigned to the user key. Press return when finished. You may edit the line as you are typing it - use backspace or delete to get rid of the last character entered, use control X or control U to start the line all over again, and use control C to abort the definition and to leave the key as it was before you started. If the first character you type is a return, the user key will be voided - it will generate no characters at all. If you exceed the 20 character limit for any one key, the whole sequence is voided and the user key is left as it was.

Pressing control-R while entering a user key sequence will usually take the next character in the sequence from corresponding place in the last user key defined - however, if you have meanwhile performed certain other operations (such as naming a file) which use the same line input editing conventions, you will find that the sequence being typed in is taken from that operation instead. Using control-R to duplicate line entry is useful in file naming as well as defining user keys.

The user keys are a handy way of continually re-entering a long sequence of words or commands.

Remember that the keys can be used for command sequences as well as characters - or the two may be mixed. One user key can be used to invoke another - for instance, if you want to programme a key to save entering a 30 character sequence, you can enter the first 19 characters into user key 1, and enter <ESC> 2 as the twentieth, followed by the rest of the sequence into user key 2. The whole sequence can thus be invoked with <ESC> 1, which invokes <ESC> 2 as a command. Beware of invoking a key within itself - the programme will go into a loop.

The maximum limit for the number of characters that can be chained together in this way should be regarded as 128.

When defining the user sequence you can embed any character in the line, including control characters. To embed control characters you have to preface them with ^P. Thus, to enter a return into any user sequence, type it as control P control M.

The way the user keys operate is simply to place the sequence as programmed into the end of the internal 128 character keyboard type-ahead buffer which COMM maintains. Suggested uses of the keys are not simply for character strings but also for sequences of often used commands - to send a file, change handshake, enable auto-line feed can be accomplished with just one keystroke.

Note that the user-programmable delay detailed in the handshaking section is permenanently switched on when user keys are sent out from the communications port.

Software driven modems are fully supportable by means of user key programming. Not only can dial commands and telephone numbers be saved within COMM, but software commands to change modem speeds or mode can be combined with COMM commands to change speed within the programme.

3. THE EDITOR

The reason that we have included the editor is to enable easy amendment and correction of messages or other text. As you know, it will reformat text for display on small, medium or large screens, as well as formatting text files for telex transmission.

Word-wrap is included in the features the editor offers. If you type beyond the end of a line, the word you are typing will be moved to the beginning of the line following. The precise point at which this occurs is determined by the width selected on the format option. If you elected to use format, and pressed either W, M, T or N, word wrap will occur as soon as a key is typed in the last column of the selected width - if you have selected 40 columns, a word that extends into column 40 will be wrapped round to the beginning of the next line.

If the file you are editing has been compiled using a word processing program. the text will be displayed with tabs and other control characters in reverse or dim video, bare carriage returns (those without line-feeds) are ignored, and line ends are determined by the line-feed character.

This editor is not an attempt to supplant a word processor. It doesn't incorporate all the finesse and versatility found in dedicated W/P software, but nevertheless it's easy to use and does its job well. You have complete freedom of cursor motion, and can move the cursor through the text or into the empty space between or at the end of words or lines. It also has its own help menu, invoked by <ESC> H, to remind you of the edit facilities available, which are detailed here. A screen display of the edit help menu is on the next page.

Extra user keys

As well as the ten user keys described above, there are also six other key sequences which operate in exactly the same way. They are <ESC> followed by any of the six keys ; : < > = ? . We would recommend that these keys (which are more difficult to remember than simple numbers) be used for chaining user sequences in cases where more that 20 characters need to be entered in a particular sequence.

User key ? has special properties you should be aware of if running in viewdata emulation. Host computer systems (such as Prestel) send an ENQ code as soon as you dial into them. This code (which is ASCII code 5 or control-E) is the same as the Who-are-you code on telex systems, and is used to ask a user for their identity number. If you programme user key ? with you ID code and log-on sequence, then when in viewdata emulation, COMM will automatically log you on to a computer which uses this convention, saving you time and trouble. See the section on emulations for more details on this.

(Please note that this auto-log on feature can be reprogrammed by you simply by redefining user key ?. It will not respond to redefinition by the host computer - Prestel is capable of changing the ID codes on dedicated terminals, but COMM does not accept the commands it uses. You should also be aware that if you have difficulty logging on automatically, you can disable user key ? by simply redefining it with just a return - this makes it void. You may want to do this if your computer is one which cannot implement proper split speeds - COMM buffers incoming characters and can sometimes keep on receiving ENQ codes and responding to them at inappropriate times, especially on noisy lines.)

i) **Cursor control:**

You can use most arrow keys if your keyboard has them, and they have been installed in your driver, or alternatively;

UP	= ^E or <ESC> A
DOWN	= ^X or <ESC> B
LEFT	= ^S or <ESC> D
RIGHT	= ^D or <ESC> C

(^ = control. Hold down the control key whilst pressing the required key. The control key works the same way as a shift key does. Do not make the mistake of assuming that the <ESC> key works the same way – unlike control, <ESC> is a key in its own right.)

ii) **Page Scroll:**

BACK 22 lines	= ^R (or <PgUp> in PCDOS version)
UP one line	= ^W (or <Home> in PCDOS version)
FORWARD 22 lines	= ^C (or <PgDn> in PCDOS version)
DOWN one line	= ^Z (or <End> in PCDOS version)

iii) **Delete:**

CHARACTER LEFT = <DEL>
(deletes character to the left of cursor position)

REST OF LINE = ^Y or <ESC> J or <ESC> M
(deletes the line right of the cursor position, and moves the next line up to start at the cursor position)

CHARACTER RIGHT = ^T or ^G or <ESC> N
(deletes the character at the cursor position, and moves the line one character to the left)

Screen Display – Edit Help Menu

```
        *  E D I T   H E L P   M E N U  *
-----------------------------------------------------
 HELP = <ESC> H or ^O       |    Redisplay  page= ^B
----------------------------|------------------------
 Cursor Control             |    Screen Scroll
UP   = ^E or ^K or <ESC> A  | Line  UP      =^W
DOWN = ^X or ^J or <ESC> B  | Page  UP      =^R
LEFT = ^S or ^H or <ESC> D  | Line  DOWN    =^Z
RIGHT= ^D or ^L or <ESC> C  | Page  DOWN    =^C
-----------------------------------------------------
 Delete                     |  Insert
Character LEFT = <DEL>      | Character= ^V or <ESC> @
Rest of Line   = ^Y or      | Line     = ^N or <ESC> L
    <ESC> J or <ESC> M      |--------------------------
Character RIGHT = ^T or     |     FIND    |  Exit Edit
        ^G or <ESC> N       |  = <ESC> F  |  = <ESC> E
-----------------------------------------------------
 User sequences        .....  to embed any Control
DEFINE = ^U or <ESC> P      | Characters, prefix with
INVOKE = <ESC> 1 - 9        |          ^P or <ESC> R
-----------------------------------------------------
 NOTE: ^Q can be used instead of <ESC>.

... Press any key
```

iv) **Insert:**

CHARACTER = ^V or <ESC> @ (or <Ins> in PCDOS vers)
(inserts a character at the cursor position, and moves the rest of the line one space to the right)

LINE = ^N or <ESC> L
(inserts a line, or part of a line, from the cursor position to the end of the line, and moves the remaining characters down one line)

v) **Help** = <ESC> H

The edit help menu displays, and you now have the opportunity to review the edit facilities.

vi) **Exit** = <ESC> E

Exit Edit by pressing ESC followed by E, which returns you to the off-line menu.

vii) **Redisplay page** = ^B

This redisplays the page in edited form, in case the cursor or display has become confused or misplaced - some computers do occasionally make mistakes when displaying or deleting, and the editor itself may well need pages redisplayed on some systems. COMM itself can become confused if you try and type in text beyond the width you may have selected on entry to the editor.

The redisplay may also be needed if you define user keys within the editor: it enables the system prompts to be removed from on top of the textual display.

viii) **Find** = <ESC> F

The editor also has a simple find function. When you press <ESC> F the prompt

Find?

will display on the screen. You type in the sequence of characters you want to locate (making sure that you use the correct case) and then press return. The editor will begin looking through memory from the END OF THE CURRENT PAGE. In other words, if the characters you wished to find were being displayed before you pressed <ESC> F, that instance will not be found - the next will be located instead.

If the characters you typed occur nowhere at all between the end of the current page and the end of the text, the message

Not Found. Press any key

will display, and when you press a key, the screen will display exactly as it was before you pressed <ESC> F to find. However, if the character sequence you entered is found, the screen will display with that occurence in the middle of the screen, with the cursor at the beginning of the line containing the sequence. Sometimes, the cursor may be a line or two out of position - do not worry about this if it happens. The reason is simply because the editor's line count is thrown out when it encounters lines longer than 80 characters. This will not happen often.

Be patient - finding may take a few seconds on large files. COMM is quick at not finding things - finding takes a little longer as the programme needs to recalculate various internal markers it uses to remember whereabouts in the text it happens to be.

Remember that you can use the buffer editing characters as described in the section on the user keys - pressing control-R will duplicate one character at a time from the last sequence you tried to find.

4. USER AREAS ON DISK

NOTE: This feature is not available on MSDOS or PCDOS versions earlier than 2.0 – the message **'Not implemented'** will appear if you are operating under one of these systems.

This is a file management function, which enables you to divide the disk into different areas, each one of which can be used to send, save and receive files.

Let us assume that you use a separate disk for each of your company's customers, and that you regularly communicate with that customer using COMM, to send to, and receive data from the customers' computer. It is sensible to use one area on the disk to store incoming files, and another area to store outgoing files. Changing to a second user area is very simple – however, the procedure differs slightly under CP/M systems and MSDOS/PCDOS systems.

a) CP/M systems

From the off-line menu only, press J to change area being used on disk. The prompt

User Area (0-9 or A-F) ?

appears, and you can change the area by typing any number from 0 to 9 or a letter from A to F (for user areas 10 to 15). Type 1. Press return to input the character, and space-bar.

You are now in user area 1, and any files subsequently saved will be saved to that area. It is important to remember that any files saved in a particular area will not appear in another user area's directory. Whenever you begin using COMM you will automatically start in the user area you were in before – this is almost always 0. When you exit COMM, you will return to the same area you started from.

ix) User sequences

The bottom of the edit help menu mentions the user sequences. They function in the editor in the same way as they do in both the on-line and off-line modes.

x) Control-Q

Below the Edit Help Menu you will see:

NOTE: ^Q can be used instead of <ESC>

This means that any of the Edit Help functions can be invoked by pressing ^Q as the command key instead of <ESC>. Please note that redefining the command key for on-line use does NOT affect the operation of the <ESC> sequences in the editor – you can use either <ESC> or control-Q irrespective of whether the on-line command key has been changed.

5. DIRECTORY SEARCHING

Pressing D from the off-line menu or <ESC> D from the on-line menu prompts you with the line

Specify Files:

at which point you can enter the same type of file specification you would normally enter from your operating system for a DIR command. Wildcard characters ? and * can be used, and only files matching the specification entered will be displayed on the screen.

If you press <RETURN>, a directory of all files will be displayed. This is equivalent to a search for a *.* specification.

Directory displays, like file displays, can be suspended by pressing any key, and then either aborted by pressing ESC or resumed by pressing any other key. This feature is especially useful if you have a hard disk drive. Additionally, the amount of free disk space in kilobytes is displayed after the requested directory (except on CP/M-80 operating systems earlier than 3.0 and MS/PCDOS operating systems earlier than 2.0).

The method of searching for files on disk allows for access to data files (including JCFs and DRIVER/GRAPHICS files) which are not on either the current directory for Microsoft operating systems, or the current user area for Digital Research operating systems. Such files can be accessed even though they may not appear on a directory that you see on the screen.

a) For Microsoft operating systems (MSDOS and PCDOS), this enables overlays and JCFs to be obtained which are either on directories included in any APPEND path specifications or alternatively are obtained over networks. Where files on the same name are on different directories, the search path order determines which one will actually be read.

b) For Digital Research operating systems (CP/M-86 or Concurrent) this enables overlays and JCFs to be obtained when they are system/read-only files on user area 0, or else are on the default system drive.

b) **MSDOS/PCDOS systems**

From the off-line menu only, press J to change area being used on disk. The prompt

Enter Directory path:

will appear, and you can change the current disk directory by simply typing in a directory path as you would from the operating system level, using a slash to separate directories. If you do not know about the tree-structured directories of MSDOS 2.0, this is not the manual to read - you should read your operating system manual for more information.

If you enter an invalid directory path, the message

Directory path not found.
.... press any key

will appear, and you will be returned to the off-line menu on pressing a key, with the directory the same as it was before.

The operation of this function is identical to the operation of the CD or CHDIR operating system commands. However, neither the MKDIR nor the RMDIR functions for creating and deleting directories are available within COMM - to create a directory use the MD command, and to delete a directory use the RD command.

On exiting COMM, you remain in the directory you last selected.

6. RESETTING DISKS AND SELECTING DRIVES

During the course of running COMM you may run out of disk space, or you may wish to receive files on a disk different to the one you use to send files. When you change disks it is necessary (under many operating systems) to tell the computer, otherwise it will regard the new disk as read only, and will display an error message.

Resetting the drives enables you to save files on to another disk if your operating system requires it. Even if it doesn't, no harm will ever be done by resetting.

Take a disk, and place it in drive B.

Press R from the off-line menu or <ESC> R from the on-line menu to reset the disks . You will see the message

Reset drives to Read/Write? (Y/N)

Enter either Y or N for Yes or No.

Having reset drive B we can now select it for the next disk operation. From the off-line menu, pressing V (or <ESC> V while on-line) will offer you the choice of two drive, A or B. If, for instance, you want to select drive B, simply enter B at the prompt.

Notice that the filename has remained unchanged, and only the prefix letter has changed from A to B; all further action involving saving and reading etc. will take place on the disk in drive B.

When inputting a file name, a drive can be specified in the usual format by prefixing it with the drive letter and a colon. Thus TEST.TXT on drive A can be selected by typing A:TEST.TXT. If the logged disk drive was drive B, directory requests will still give a directory of drive B, but file-specific disk operations such as saving or appending will use the named file with its named drive.

This can lead to some apparently odd results in that such files can be either read into memory or sent from disk, but not deleted, renamed or appended to : and an attempt to resave them will, though apparently successful, simply create another file of the same name on the current directory or user area.

Uploads of files via the file transfer options differ from text file sending in that the former will only send files from the current directory or user area, but the latter will send files either from anywhere on the APPEND path or from user area 0.

7. INPUTTING FILES TO MEMORY FROM THE OFF LINE MENU

If you want to transmit a disk file by inputting into memory and sending memory instead of sending directly (perhaps you want to use the formatter) you have the option of doing a continuation read, which will read the file from the last point accessed, or inputting from the beginning of the file if not.
The prompt

Continuation read (Y/N) ?

will appear on the screen, and you should enter Y or N. If the file is one you have just named, the file will not have been accessed before, so it will make no difference whether you press Y or N.

However, if you had previously accessed the file and then choose to do a continuation read, the disk file will be input from the last point at which it was accessed. So if you had just sent a file to the screen to look at it, the last point accessed would be the end of the file, and nothing would be input to memory at all if a continuation was selected. You might care to try this out.

This feature means that you can send files larger than the capacity of the memory buffer via memory by inputting them in sections until the file is completely read.

8. THE FORMATTER

The option to format before editing memory works as follows.

All the lines in the text in memory are wrapped around to the chosen width. However, lines which end with a full stop or which are followed by a blank line are kept intact - the next line will not be wrapped to continue on the same line.

Full stops at the beginning of lines are stripped out: however, the line beginning is kept.

This gives three options for keeping a carriage return hard: either end the line with a full stop, begin the next line with a full stop (which will be stripped) or leave a blank line.

Where lines are wrapped, the wrap will occur at the end of the word preceding the maximum line length. If a word is itself longer than the maximum line length chosen it will be chopped at the maximum length and the remainder continued on the next line.

Where a carriage return is taken out, it is replaced with a space. Trailing spaces at the ends of lines are ignored.

Where telex format has been chosen, all illegal and non-international telex characters (including currency symbols) are replaced with spaces. All lower-case characters are turned into upper case ones.

The formatter also replaces all control codes with spaces.

Formatting notes

a) Selecting a particular width affects the pos- t which COMM will dynamically word-wrap whilst typing - see the section of this reference guide on the EDITOR.

b) Please note that the algorithm (the formula) which COMM uses to format text will add extra lines in some cases. Paragraphs separated by a single blank line will not be affected, but ones separated by two blank lines will, after formatting, be separated by three lines rather than two. Repeated formatting will increase this spacing each time - avoid use of multiple blank lines in text you wish to reformat. The formatter is thus an imperfect instrument for some uses, though for a one-off format of a telex it does its job very well with minimal memory overhead.

9. RENAMING FILES

When you press Z from the off-line menu, or <ESC> Z whilst on-line, you will be prompted for a new file name. The file with the name displayed on the menus which actually exists on the disk will be renamed to the name you enter here. However, the name displayed on the menus and used for saving text remains the same.

This enables you to save under your current name without deleting the old file. You can rename it instead.

10. HELP LEVELS

From the off-line menu, if you press H, the current level of help will be displayed, and you are given the opportunity to change it. The help level can also be changed via <ESC> J whilst on-line.

In the on-line modes help is available in three basic levels. It will automatically default to level 2, the most helpful level. In help level 2, if the command key <ESC> is pressed and no other key is pressed within a few seconds, COMM will assume that you have forgotten the commands and will display the on-line menu, followed by the prompt:

***** Command:**

If level 1 is selected, the delay after pressing the <ESC> key only brings the prompt:

***** Command:**

with no menu display. Level 0, however, produces no help whatsoever. Additionally, the requests for confirmation whilst on-line are suppressed. Thus pressing <ESC> E to go off-line, which normally produces the prompt

Exit to Off-line Menu <Y/N> ?

will, on help level 0, exit without asking. The facility for altering the help level can avoid the screen being destroyed whilst on-line by the display of the on-line menu. It also turns the on-line mode of COMM into a command-driven programme rather than a menu-driven programme. Proficient communicators often prefer to eliminate time-consuming menu displays : an additional expert help level enables every COMM message and menu to be eliminated.

11. PROGRAMME SAVE

Pressing P from the off-line menu will save the programme to disk. The current user key definitions, any changes in default handshake protocols or other settings, and the driver module, will all be saved. The original version of COMM will be overwritten and lost, so ensure your backup copy is intact. If you do NOT have a copy of COMM on the current disk when you attempt to save, the message

File not found

will appear, and the programme will not have been saved.

The driver is saved to disk along with the programme, and COMM will be able to run without the driver being present on disk. It will NOT be loaded even if it is there. We assume that the defaults in the driver file are ones you wish to override. (This doesn't apply to GRAPHICS files which must be on disk to be loaded). The message

Found driver .... loading

when COMM is being loaded will not appear when a saved version is used.

Ensure that you have enough space on disk to save COMM - a saved version may sometimes be slightly larger than the original by up to 128 characters.

EXPERT HELP LEVEL

An extra help level is available for use in situations where you want no COMM messages or menus to appear on the screen at any time - this is usually because you are using a terminal emulation and don't want the display disturbed.

This might be the case with any emulation on non-PCDOS systems which don't use pages or status line for menus and messages. On PCDOS systems, this is a way of preserving a 25 line mode in (for instance) an Ampex emulation where issuing a COMM command causes the 25th line to revert to a 24-line display as it needs to use the 25th line for its own messages.

Simply press control-X for the help level when prompted : a message saying that all COMM messages and menus are suppressed will appear. If you did this from the off-line menu, you will be placed in on-line mode also.

Setting expert help level has exactly the same effect as the NOMENU command in a JCF file - this is described fully in the Language Manual. To restore a more normal help level, just set a help level in the normal way, but remember that no COMM commands will be accompanied by the usual messages and prompts. Leaving the expert help level has exactly the same effect as a MENU command in a JCF file : therefore resetting an expert help level via a JCF will have the same effect as the MENU command.

12. TYPEWRITER MODE

When you select T from the off-line menu, everything typed at the keyboard will not only be displayed on the screen, but will be printed too. Carriage returns and line feeds can be typed as usual, but other control characters can only be typed by prefacing them with ^P. exit from the typewriter mode press either <DEL> or control Z and you will return to the off-line menu.

If you have a printer with no keyboard, using COMM in typewriter mode effectively turns your computer keyboard into a typewriter keyboard.

HISTORICAL NOTE ON TYPEWRITER MODE: We've often been asked what on earth this routine is doing in a communications programme. The answer used to be simply that the routine took up very little code, was really quite useful for those who used it, and wasn't at all in the way of those who didn't. But since we put the language in COMM and turned it into COMM+, the typewriter mode, like typing directly into memory (remember option B when in on-line mode ?) has become indispensible for formatting and titling customized printouts from within a JCF.

13. DUPLEX

By pressing X from the off-line menu, or <ESC> X from the on-line menu, you will be able to select the duplex setting required.

There are three duplex options:

Half duplex H — All data transmitted is displayed on the screen as it is sent. COMM will not be listening for an echo. Certain handshakes are impossible in this mode. Half duplex will usually only be used if the computer to which you are transmitting is not echoing back to you. If you are typing, and everything typed is doubled (ie. the word SEND is displayed as SSEENNDD) then you are in half duplex, and should be in full duplex.

Full duplex F — Your screen displays only what is echoed back, and not what you yourself type except as an echo. If you are typing, but only the response from the other end is displayed, then you are in full duplex when you should really be in half duplex. You will be in full duplex if you're connected to a computer that is echoing back.

Host mode O — Host mode is used if the computer to which you are connected requires you to echo back to it.

The usual duplex combinations are as follows:

YOUR COMPUTER	OTHER COMPUTER
half duplex	half duplex
full duplex	host mode
host mode	full duplex

Host mode is sometimes called ECHOPLEX.

Do not connect two computers together with both of them in host mode; the first character sent will be echoed back and forth like a hall of mirrors, and because COMM gives priority to incoming characters, it is difficult to recover from this without rebooting the program.

Do not connect two computers together with both of them in full duplex when sending files with echo checking - after sending the first character, transmission will cease as both machines wait for an echo. Full duplex is a listening mode.

HALF DUPLEX: Please note that our half-duplex setting does NOT mean that modem lines turnaround of RTS and CTS lines will occur - therefore half-duplex modems and couplers cannot be used. All half-duplex means to COMM is whether or not characters are being echoed from your host. Whilst this is strictly speaking an incorrect use of the term, it has become an increasingly common one.

14. HANDSHAKE

This section is very important, for without the correct handshake you may be unable to communicate with another computer. The term handshake is used to define a set of rules that must be obeyed to ensure an orderly exchange of information.

You access the handshaking menu by pressing Y from the off-line menu, or <ESC> Y from the on-line menu. After a prompt the page of handshake options will display.

Most of the options have default values which may be displayed in reverse or dim video. You might have to change them when you get around to transmitting and receiving. The handbooks covering any machine you need to connect to will tell you which characters to use.

There is a printout of the handshake selection menu on the next page, and the details of the options available follow in the order in which they appear on the menu.

Screen Display - Handshake Menu

```
 Handshake Selection: options as follows -
on = ( * )   off = (   ) Code  inoperative =  ^@
 ^ = Control  key  : ^M = Return  ^J = Line Feed

          ( Press K for Kermit setup Menu )

ITEMS 0 TO 4 ARE INOPERATIVE IF HANDSHAKE TOGGLE OFF

 0 Wait and check for errors on echo ....... (    )
 1 Delay after each Character sent ......... ( * )
 2 Length of delay after each Character.....    2
 3 Last Character host uses as Delete echo..    /
 4 Character host gives as a bad echo ......   ^G
 5 Stall transmission Character.............   ^S
 6 Resume transmission Character ...........   ^Q
 7 Request Acknowledgement .................   ^@
 8 Acknowledge Character ...................   ^@
 9 Block length before Acknowledge Request . (    )
 ^ Accept non-printing Characters .......... (    )


SELECTION (CR to return) :
```


0. **Wait and check for errors on echo.**

This option has to be turned on from the on-line mode by pressing <ESC> I (toggle handshake) whilst on-line, as well as being primed here. If switched on here and invoked, COMM will, when sending memory or disk files, wait for the echo of the last character sent, and compare it with that character, before sending the next one. If the character echoed back is not the same as the one sent, COMM will send a backspace (to erase the incorrect character), and then re-transmit the character again.

Please note that this is inevitably a slightly hit-and miss affair. This is because of what is called LINE NOISE. We assume that (with exceptions like line feeds on carriage returns) there will be a one-to-one relationship between characters sent and received. Line noise makes this assumption invalid - spurious characters can appear which nobody has typed. If a character is sent, and the echo appears incorrect, it is posssible that the incorrect character is SPURIOUS - it is a result of line noise. This means that there is no need to correct it. However, there is no way of telling.

Also, many hosts take time to respond to backspaces. COMM may enter a loop whereby it sends backspaces continually. If you hold down the <ESC> key, you can break out of this condition if it occurs.

Note also that this will only work in full duplex. If you attempt to do echo checking in any other duplex mode, COMM will ignore the setting.

1. **Delay after each character sent.**

When interfacing with multi user mainframe systems, or with machines that are slower than yours, you may probably want to set a delay after each character sent, to ensure that no data is lost. This function must also be invoked from the on-line menu, with <ESC> I, but would not usually be used if full echo checking is taking place. Most electronic mail services like a delay after each character sent, especially at 1200 baud and at peak times.

You will also need a delay if you are communicating via an adaptor which is lowering the effective transmission speed to ensure that you do not transmit too fast - for instance, if you are communicating via a 1200 baud line through a convertor to a 75 baud transmit speed (such as the modems

made by DaCom, Tandata or Master Systems which incorporate speed conversion) you need to restrict your transmission speed to 7.5 characters per second even though you could nominally transmit at 120 characters per second.

If your version of COMM includes the option to set a split baud rate of 1200/75 via B from the off-line menu, you will need to set a delay too, to avoid losing incoming characters whilst COMM is changing speeds - a long delay will mean that echos will almost certainly arrive before the delay is completed, and no incoming characters should be lost.

2. **Length of delay after each character.**

Here you set the delay invoked in the previous option by inputting a number from 1 to 9. Please see the section on timings for the exact value that this should be : note that if your operating system is a CP/M 2.2 system with no accessible clock then this value is not absolute, as it depends upon the speed of the clock in your computer and the way your operating system has been implemented. Only trial and error will enable you to find the optimum value for your own installation.

Note that this delay will be applied to all characters sent out, whether they are sent from memory, from disk, or from the keyboard buffer.

3. **Last character host uses as a delete echo.**

When in full error checking mode (0), COMM will send a backspace to delete the last character if it detects a bad echo. Some computers enclose characters to be deleted between symbols, such as /. For example, if the last character in the word SENT, has been received as a D, COMM will send a delete thus:

SEND/D

then the T will be sent:

SEND/D/T

The receiving computer has enclosed the offending D and it will not be saved or printed. However, the / will be echoed back twice, once as an

acknowledgement of the delete command, and then again preceeding the T. Since COMM only sent one delete, it will think that the second / is an incorrect echo of the T, unless we tell COMM that the host will echo back this second delete. By inputting the correct delete echo, COMM will ignore the second slash, and full error checking will proceed unhindered.

4. **Character host gives as a bad echo.**

When you are transmitting you may find that the receiving computer is busy. When this happens, the receiving computer might echo back a busy character. So that this is not interpreted as a bad echo of a character, we have to know the receiving computer's busy signal, so that COMM can suspend transmission briefly. Many computers use a ^G (BELL) as the busy signal - notably, the IPSS intercontinental packet switching system (IPSS).

This character does double duty. When any of the receive buffers in COMM become nearly full, the current stall character is sent to the host to make it stop sending: and any subsequent characters received will also be stalled. If the stall character is itself echoed, a loop will result and the two computers will carry on stalling end echoing the stall indefinitely. Many hosts will beep when any characters are sent after a stall - in effect they will echo any stall after the first as ^G. The echo of the stall character should be put here, so the COMM will know that it is simply an echo of its own stall and not another character to be received.

5. **Stall transmission character.**

This is where you nominate the STALL character mentioned in the file and memory sending section. When installed, either end can send it to temporarily suspend transmission. Note that though automatic file or memory sending is suspended after a stall from either end, until the resume character is received or typed, interactive communication can still take place. COMM will automatically send this character when receiving if on-line with memory on if the memory buffer becomes full. It is also automatically sent when the temporary receive buffer that COMM uses for storing incoming characters fills up because characters are being received faster than they can be processed.

6. **Resume transmission character.**

This is where you nominate the resume character, used to re-start transmission after temporarily suspending it. It resumes transmission when either typed at the keyboard or received over the line. COMM will automatically send this character after a disk save automatically triggered by the memory buffer filling up when receiving data with memory on, and will also send this character when the temporary receive buffer is emptied after a stall character had been sent because it was filling up too fast.

Note: The Stall and Resume characters will be displayed on both screens in reverse video, and will have to be edited out of any text before it is saved to disk. You could, if you wished, have embedded the stall character in the text that you want to send. To do this, in the edit mode, you would have to preface it with ^P (control P). The control-P prefix forces literal acceptance of the character. Note that the usual stall of XOFF is the same as the Wordstar underline code: change the stall if this causes problems.

7. **Request acknowledgement.**

This can be a half of the ETX/ACK protocol used by many machines (sometimes called EOB/ACK). When computer A is sending to computer B and comes across a request acknowledge, it stops, and waits for an acknowledgment from B. Upon receipt of the acknowledgement it will continue to send and so on. Here you input the request acknowledgement code.

For instance, if you put in a carriage return, you can delay transmission after each line until a line feed is received from the host, provided that the line feed character is the acknowledge entry.

8. **Acknowledge character.**

This is the character that will instruct COMM to continue transmitting if it happened to be waiting after sending a request acknowledgement. It is sent by the computer you are connected to as a response to your request. If you are using a carriage return as the request, a line feed here would cause a delay after each carriage return until a line feed was echoed back.

Some systems (Istel COMET, for example, and one of the XMIT options on Telecom Gold) send a ^Q at the beginning of every line for this very purpose. To use it, you would specify a carriage return as the request acknowledgement character, and ^Q as the acknowledgement to be received.

9. **Block length before acknowledge request.**

This entry, when activated with a value, alters the function of the two preceding entries for request acknowledge and acknowledge.

You can define a length for sending files in blocks.

You can input any number from 0 to 2^{16}. Common block lengths are 128 and multiples thereof (256, 512 etc.). If you input a block length of 0, you will disable block sending (ETX/ACK) protocol, though the simple request/acknowledge handshake outlined above will still apply.

If you have entered a block length, then while sending memory or disk files, a request acknowledgement will be automatically inserted after the number of characters you enter here. Your computer will then wait until the acknowledge comes back before resuming its sending. Conversely, if you have a non-zero block installed, a request from the host will automatically generate the acknowledge from you as soon as it is removed from the receive buffer.

Using a block length together with suitable request and acknowledge characters will give you, should you need it, an orthodox ETX/ACK transmission protocol.

^. Accept non-printing characters.

This toggle affects the treatment on all non-printable characters with the exception of carriage return, line feed, and backspace.

When it is off, any control characters (non-printable ones) are sent directly to the screen as they are received, but are not stored in memory (if it is switched on). Only printable text is so stored (with the exception of carriage return, line feed, and backspace).

However, when the toggle is on, control codes are stored in memory. They are displayed on the screen as printable equivalents with the ^ prefix - thus control-E would be displayed as ^E but stored as control-E (ASCII 5).

If you want to function as a terminal, you would normally have this toggle off: and if you simply wanted to receive text to memory it would also be off. However, if you wanted to receive a file mixing control-codes and text (such as a word-processor file) you would switch this toggle on. The control codes would be displayed on the screen with the ^prefix, and stored in memory as they were received. This would, of course, inhibit the use of your computer as a VDU emulator since it would no longer receive control codes such as clear screen and cursor address to its own CRT - they would all be redisplayed as the printable equivalents with the ^ prefix.

Please note that if you are running with an emulation selected, any interceptions of control codes by the emulator will override this selection. For example, if the ^ toggle is on, a control-Z code (26 decimal) would only be displayed as ^Z and stored in memory if no emulations were selected. If ADM emulation were selected, the screen would clear since a ^Z is the ADM clear screen code. The ^Z character would not be placed in memory.

K. Kermit Setup Menu.

This option accesses a secondary menu used solely for configuring the Kermit file transfer protocol : this is dealt with in the section on block transfers.

15. AUTOMATIC SAVING TO DISK

When receiving text with memory on, data will be saved to the disk file currently named (or appended if the file already exists) when the memory buffer becomes full. If no file is named, data is saved or appended to a file called TEMPFILE. COMM will then clear memory and resume reception automatically.

This saving to disk occurs without any user intervention provided the stall/resume characters on the handshake menu are set up (to XON/XOFF for instance) but should the resume character only be blank, the automatic save is disabled. Disabling the resume character only will thus still enable COMM to stall sending from the host and await manual instruction after a buffer full message is displayed notifying the operator that manual intervention is required.

On certain versions of COMM (notably those with the default PCDOS driver), if both the stall and resume characters are blank, then DTR handshaking is enabled instead of XON/XOFF or something similar. This feature, where present, enables COMM to be used more reliably as a terminal emulator in situations where response to XON/XOFF type codes is simply too slow for adequate operation, or where no suitable control codes for flow control exist. If both the stall and resume characters are disabled for these versions of COMM, then automatic saving to disk will still occur : the requirement for enabling manual saving is that a stall character be present without a resume characters - for instance, XOFF without an XON.

16. TIMINGS

With the ever-increasing range of machines based on Intel 16-bit CPUs which offer highly variable performances and speeds (ranging from 4Mhz to 32Mhz), methods of establishing delays via timing loops vary too much with clock speeds, and are therefore unsatisfactory. COMM takes its timing from the operating system clock, and the way that delays and timeouts function in the programme is detailed in this section.

Note that if you are using a CP/M 2.2 operating system, which doesn't have a clock, or if you are using a CP/M-86 system on which the operating system clock is not implemented, the programme uses a delay loop which will not be as accurate as a proper clock, and you may have to tune the delay via the handshake menu for proper operation.

The basic delay is that which occurs between characters sent when:

a) handshake is switched on with ESC I
b) option 1 (delay between characters) is set on via the handshake menu
c) option 2 (delay length) counter on the handshake menu is set to 1.

This is timed via the clock on the system to approximately 100ms (that's one-tenth of a second). The timing is approximate because the clock is updated only 18.2 times each second on PCs, and will therefore sometimes be 10ms late in updating - this is unavoidable.

A delay length of 2 on the handshake menu will last for 200ms, and so on up to the maximum delay of 9, which lasts for 900 ms (that's nine-tenths of a second). The delay counter in a JCF is similarly timed: a delay of 10 will time out after 1 second when the delay counter on the handshake menu is set to 1, and will time out after 9 seconds if the delay counter is set to 9.

These timings are irrespective of the settings of either the on-line handshake toggle or either of the delay flags on the handshake menu - the length of a delay in a JCF is determined by multiplying the delay actually set in the JCF by the delay counter on the handshake menu - the result is the delay in tenths of seconds. This gives a minumum timout of 10 ms (one tenth of a second) in a JCF of 10ms with the JCF delay

and the counter on the handshake menu both set to 1, and a maximum timout of 11430 ms (1 minute 54.3 seconds) with a JCF delay of 127 and the counter on the handshake menu set to 9.

This should be sufficient for most purposes: longer timings can be accurately established with either nested loops, or by means of the time variables HOURS, SECONDS and MINUTES.

Other delays in the programme are similarly exact:

a) the delay in displaying the on-line help menu after pressing ESC when in help level 2 is set to 2 seconds x the counter on the handshake menu and can therefore be from two seconds to eighteen seconds.

b) The timeout during ESC sequences in emulations is set to 1 second x the counter on the handshake menu and can therefore be from 1 second to 9 seconds. PSS users should note that a delay of one second is not uncommon during packets, and that should a packet split in the middle of an ESC sequence, the screen display may be affected should the emulator timeout. Delays should be set to account for this.

c) The same delay is used after each line sent when transferring text files from disk or memory when handshake is switched on : when handshake is switched off, lines are sent consecutively with no delays.

d) The delay for timeouts in block transfers is set to 5 seconds x the delay counter: since transfers are aborted after 9 timeout errors, this allows from 45 second minumum to over six minute maximum delays during such transfers.

All the above variations of the basic 10ms x the delay counter formula are patchable for OEM or specialized use where necessary : registered users should contact us for details.

The delays operate slightly differently under both Digital Research operating systems and also under certain implementations of MSDOS, for which the clock is only updated every second. Thus the minimum delay available is therefore one second.

i. MSDOS versions of COMM which are running on systems which only update the clock every second function in the same way as described above : the difference is simply that the minimum delay will be one second no matter what you are doing and delays greater that one second are timed in multiples of seconds.

ii. CP/M-86 and Concurrent versions work slightly differently : a JCF delay of 1 gives a one second delay, a delay of 10 gives a ten second delay and so on. This is not modified by the delay counter on the handshake menu : however, the counter on that menu does affect timeouts in emulations and file transfers, and the time taken before the menu is displayed in help level 2. Users who need to ensure compatability of JCF timing across all versions of COMM should ensure that they set a delay of 9 on all versions. While this will not affect JCF delays on Concurrent and CP/M versions, it will ensure that the base one second delay is approximately the same for all implementations.

17. BAUD RATE

Baud rate, parity and word length can be changed via the off-line menu, by pressing B, and following a separate menu. This may also include a facility for changing ports on systems with multiple serial outputs.

If changing serial port parameters within COMM is not possible a message saying **'not implemented'** may appear instead - or alternatively, the B option may do nothing at all. The functions changed via option B from the off-line menu are dependent on the hardware you use. We have to write different routines for each computer COMM runs on - sometimes, the information we need is not available, or else speed changing is simply not possible. In such cases you will be able, from your operating system, to use a utility programme called SETUP or CONFIG or PORTSET or SETPORTS (or something similar) to do this instead. A few machines have switches on the back for this.

The implementation of the baud rate setting may also include the facility for outputting a break level. Unlike many terminals, microcomputers do not as a rule have a break key on the keyboard. The break is output for about 250 milliseconds. The break output is also more usually available from the on-line menu by pressing <ESC>/ even when it doesn't appear on the baud rate menu. But if speed changes are not available on your system, the break facility from the on-line menu via <ESC>/ will not work either.

Implementations of COMM offering split speeds of 1200/75 will usually need a delay setting also. Note too that viewdata emulation is selected via <ESC> W whilst on-line, and is not automatically invoked when split speeds are selected.

You should be aware that COMM doesn't restore the serial port to its original state on exiting - you should do this yourself.

18. TOGGLES AND SWITCHES AVAILABLE WHILST ON LINE

The following toggles and switches are starred in the on-line menu if they are activated.

a) Switches

T= On-line Memory off (*)

This is the mode that you would use if you are using COMM to connect up to another computer. When in this on-line mode, the computer will function as a virtual or absolute terminal. This means that all keyboard input (what you type) is output to the port, and all port input is displayed on the screen; just as happens with an ordinary terminal. You can thus run applications software on a distant (host) computer with full terminal performance).

G= On-line Memory on ( )

In this mode, the memory is turned on, and all port input as well as keyboard output (what you type), is saved. If you are in full duplex, the memory and screen contents will depend upon what is echoed back from the other end (keys sent are reflected back and become port input). Once saved in the memory buffer the data can be edited, saved, appended to a file, reviewed or ignored as you choose.

B= Type Directly to Memory ( )

Although this mode has been discussed before, there are certain conditions that you should know. This 'offline text entry', or buffer mode is somewhat similar to the entry of text in the Edit modes, but the backspace/delete keys are the limit of the editing capabilities. Whilst you are in this mode, any port input is stored in the buffer, and displayed on the screen. The only difference between this, and Terminal Memory on (on-line with memory on mode) is that keyboard input is NOT sent out to the port. Additionally, this mode temporarily sets the duplex to half, and turns the auto linefeed to on. When you select another of the on-line modes, the linefeed/duplex conditions are restored to the settings that you previously selected.

Whilst you are in this mode, if you send a file to the screen or printer, the temporary half duplex/linefeed on settings remain in force, so you could if you wished, exit from one terminal mode into this one, compose a quick memo, save and print it, and return to the previous terminal mode without having to reset handshakes etc.

Once any of the three on-line modes outlined above are activated, they remain in force until explictly changed : thus setting memory on will leave memory on when on-line even if you exit to the off-line menu and then return on-line afterwards. However, when the programme is loaded, the default mode is On-line mode Memory off - in other words, switching memory on isn't something that is preserved via a programme save from the off-line menu, even if you to a programme save while memory is switched on.

b) Toggles

There are three on-line On/Off Toggles, all of which can be either on or off. To turn a function on, simply select it, and a * will appear in the brackets; to turn a function off, reselect it, and it will be turned off.

L= Line feed ( )
When on, a line feed is automatically added to a carriage return.

P= Printer echo ( )
All port input and output will be sent to the printer as well as to the screen.

I= Handshake ( )
Certain of the options on the handshake menu, (wait and echo in particular) will only be operative if the handshake is turned on. See the handshake section for more details.

(The reason why there is a separate toggle for activating handshakes which had already been selected from the handshake menu is because the need for handshaking varies from time to time. The need for echo checking depends partly on the noise on the telephone network, which in turn depends on time of day, weather and so on. The need for waiting after sending characters also varies on timesharing

systems, depending on the number of users and processing load. It is therefore simpler to be able to have a suitable handshake in reserve to be activated when and if needed with a simple key sequence than it would be to re-enter the handshake menu.)

W= Emulation ( )
Please see the EMULATION section for the W option - this is a multiple-way rather than a simple two-way toggle.

19. THE COMMAND KEY

The default value for the command key is ESC. The command key is the key which is used to communicate with COMM itself rather than with the host when on-line. All the commands available whilst on-line are prefixed with the command key.

If your command key is <ESC> and you want to send an <ESC> to the host, you must press <ESC> twice: the first <ESC> will be trapped by COMM, but it will send the second to the host. However, it is sometimes going to be inconvenient to press <ESC> twice, especially when communicating with a host which requires many <ESC> sequences to be sent to it, or when using a software driven modem which is itself programmed using <ESC> codes.

The facility is therefore available, by pressing <ESC> K whilst on-line, to change the command key from <ESC> to any other control code. Possibly the most convenient alternative is Control-J. Though this is the code for a line feed, it is hardly ever typed in normal use. Many keyboards do not even have a key for LF.

Assuming that you have pressed <ESC> K, all you need do is type Control-J on your keyboard when prompted, and the command key will be redefined. Once done, all the on-line commands which were prefixed with <ESC> would be prefixed instead with Control-J, including all user keys and the command key change itself. Note however that <ESC> is still used in the sequences in the editor: only the on-line key has been changed.

Of course, any control key can be used instead - the TAB key is quite a popular alternative as it is present on all keyboards. The designation of the command key is totally flexible. It will be saved on disk with all other default changes if you do a programme save from the off-line menu.

20. GRAPHICS AND COLOUR SUPPORT

COMM includes software hooks for support of viewdata block graphics when in viewdata emulation, and line graphics when in ANSI, Wyse or Televideo emulations. Colour sequences can also be supported for Viewdata and ANSI emulations. Additionally, all emulations are capable of supporting dim/bright, underline, and blink/steady attributes as well as the standard inverse/normal attributes.

The code and tables for the extra features are contained in a graphics driver file called GRAPHICS which is loaded along with the main programme, if it exists on the disk COMM is loaded from. The GRAPHICS file need not be present for COMM to run however : if no graphics file is loaded, then all graphics display as dots, and sequences for colour and other unsupported attributes will be ignored.

Please note that, unlike the DRIVER file and any defaults, the GRAPHICS file is not saved along with the main programme when option P to save the programme is selected from the off-line menu. It must be present on the disk when COMM is loaded for any graphics options to be utilized.

The graphics file is only used for the screen display, and not for any pages saved in memory and thence to disk. This is because most printers cannot cope with graphics or colour, and neither can word-processing and other software - though you will be able to see the graphics on screen, only standard ASCII code is being saved in memory.

Like the code contained in the DRIVER file, the code in the GRAPHICS file is different for each system COMM runs on. Since over 100 machines are supported, graphics drivers for all machines which are able to support them will clearly take some time to become available.

Some machines will never be able to support GRAPHICS enhancements, which generally require a fairly sophisticated video controller. Additionally, some GRAPHICS files do not support all possible extensions available - for instance, a GRAPHICS file might support viewdata mosaics without supporting either colour or line graphics. If there is no graphics driver on your disk, please contact us to find out if and when graphics

will become available on your system . The format of a graphics file is available from us by request if your system is currently unsupported and you want to try and write your own graphics routines. Knowledge of machine-code programming will probably be required for this, but the task is quite straightforward otherwise.

21. TERMINAL EMULATIONS

The purpose of a terminal emulation is to convert the screen codes which the host outputs to screen codes which your own computer understands. Thus, when connecting up to a viewdata system, lack of a Prestel emulation would make screen displays impossible to read.

The emulations in COMM are activated via a menu obtained by pressing <CMD><W> : this will clear the screen and give a menu of options as displayed on the next page.

DISCLAIMER:

The specifications for the COMM emulators should be read carefully by intending users, as no guarantees can be given that any emulation is adequate for all uses. In particular, you should be aware that the emulations operate within the limits of the screen display on your computer : if you have a monochrome screen, you won't get any colours displayed, and if your screen doesn't support blinking characters, then you'll never see characters blink. For instance, IBM PC colour systems cannot support underlining attributes in any emulation.

You must also be aware that apart from the standard ASCII keys on the keyboard, no support for specific features of the keyboards of the terminals that COMM emulates can be assumed unless specifically stated in the description of the emulation : a limited exception to this are arrow keys. Please see the note later in this section for information on cursor key support.

Bear in mind that the adequacy of an emulation depends on the software that the emulation is intended to run. For instance, while no support for 132 column screens is offered, this won't matter unless you are trying to run software written to use columns 81-132 on a screen.

While we offer the emulations in the belief that they are adequate for most uses, it is the responsibility of the user to check that the emulation is sufficient for any specific application. If in doubt, please contact both your dealer and the DP manager for the system you want to connect to, and try to arrange for a test session to be set up.

Emulate:

0 = Viewdata single screen
1 = Viewdata double screen
2 = ADM
3 = WY30/50
4 = TV924
5 = ADDS
6 = Hazeltine
7 = VT-5
8 = ANSI no wrap
9 = ANSI wrap on
A = Blind terminal
B = DG 214 no roll
C = DG 214 roll on
D = Ampex 232 ASCII codes
E = Ampex 232 Scan codes

Press 0 - E: any other key deselects Emulations

The following short summaries of the emulations available should be used to determine what emulation (if any) needs to be used in any particular situation, and what special features and facilities for each emulation need to be noted. It ought to be read in conjunction with the tables which follow the descriptions, which give more detailed specifications of the screen codes supported.

Viewdata: emulates 40-column Viewdata terminals for systems such as Prestel or Topic. This emulation is available in two modes, via either 0 or 1.

When emulation 0 is selected on non-IBM systems, a single 40-column viewdata screen is displayed in the centre of the 80-column terminal screen using columns 20 - 60. On IBM and compatibles with colour cards (either CGA or EGA), the viewdata screen uses the 40-column screen mode, while on monochrome cards the pages display only on the left-hand side of the screen.

When emulation 1 is selected, a dual-screen Viewdata facility is implemented by displaying two 40-column viewdata pages on one 80-column screen on the CRT. The main page is displayed on the right-hand side of the screen on non-IBM systems, and on the left-hand side of the screen on IBM systems. This page can be redisplayed on the other side of the screen by pressing Control-R, where it remains until either the terminal screen is left by displaying another COMM menu, or else until it is replaced with a subsequent screen with another control-R.

The control-R key will redisplay the viewdata page with any hidden data revealed - in other words, it doubles as the reveal key found on dedicated viewdata sets. This is true for either of the viewdata modes.

User key ? is optionally used to hold an ID code that the viewdata system can ask for to enable automated log-on to occur.

While the underline code can be used instead on the hash for keyboard input in accordance with ASCII codes, a hash can also be used instead, and the ASCII decimal 95 code will display on the screen as a hash (or whatever ASCII decimal code 35 signifies on a particular CRT) and not an underline. On IBM systems, the pound sign is supported for both keyboard entry and for display.

Mosaic codes and graphics are replaced with dots if no GRAPHICS file is loaded along with COMM. However, a suitable graphics file will enable either graphics or colours or both to be displayed properly, provided the system you are using is able to support redefinition of text characters.

The major limitations of the emulation even with a suitable GRAPHICS file are that double-height characters will display as single-height and that only contiguous mosaics are supported : separated mosaics display as contiguous.

Note that the Prestel emulation uses its own special memory routines when run with memory on: if a message

Buffer full...

appears on your screen, it is followed immediately by COMM switching memory off. This is because no handshaking is possible with viewdata systems. Normally however, any text will be automatically saved to disk with no problems.

Uploading files or text to viewdata systems is supported via JCF files only : a suitable JCF is the sample MAIL.JCF freely available for this purpose.

Please also refer to section of the manual dealing with file uploading and downloading via block transfer protocols for details on the telesoftware downloader, which is activated by pressing <CMD><.> when in viewdata emulation.

ADM: emulates the Lear-Seigler ADM3A, ADM31, Soroc IQ and similar terminals.

WY30/50: emulates the Wyse 30/50 range. Please note that protect mode on the Wyse terminals is not emulated, and that attribute settings are non-dynamic and take up a space on the screen : some Wyse terminals appear not to do this.

TV925: emulates the Televideo 925 terminal. It also supports most Televideo 910 codes.

ADDS: emulates (partially) ADDS Viewpoint/Regent

Hazeltine: emulates Hazeltine 1500 and Esprit as well as the Hazeltine 1420. Both possible leadin codes and all possible offsets for column and line cursor addressing are supported.

VT52: emulates the DEC VT52 or Heath/Zenith terminals.

ANSI: emulates terminals such as the DEC VT100 either via 8 or 9. The difference is that emulation 8 initially begins without wraparound : that is, characters are not automatically displayed on the next line if they reach column 80 in the screen. By contrast, ANSI emulation 9 works in wraparound mode : characters display on the next line once column 80 is reached on the screen, and characters beyond column 80 on the last line will cause the screen to scroll up. Any software codes for enabling and disabling wraparound which are sent by the host computer will automatically switch between the two ANSI emulations, which differ only in the initial wrap setting.

Blind: This mode suppresses all text that would normally be displayed on the screen when in any of the on-line modes. It has two main uses.

Firstly, it can be used by a JCF to conceal (for either cosmetic or security reasons) any text that comes in the communications port, or any text that would normally appear on screen when in type-to-memory mode. JCF onscreen strings are unaffected by this, as are displays of user key entries and similar command-driven displays : and all text received is still passed to any JCF in the usual manner. Only the display of the text is suppressed.

Secondly, it can be used to speed up receipt of ASCII data. Normally, such data (typically text files) is displayed on screen as it is received, and this takes time. For instance, a standard IBM PC can receive and store text coming in at 9600 baud with no problem, but can only display text at the rate of around 3000 baud via the operating system. Displaying all text thus makes receipt of data take around three times longer than it would if the data were not displayed on the screen : and conversely, suppressing the display via the blind terminal mode enables transfer of ASCII data to be speeded up on any system with a slow screen display.

DG214: emulates the Data General Dasher 214 via either B or C. The difference is that emulation C begins with screen roll enabled, and emulation B begins with screen roll disabled. When

disabled, the screen never scrolls, and the cursor simply wraps around from the end of the screen to beginning (rather like a viewdata page : the screen is treated as one long line). In addition to the codes listed in the table, the DG214 emulator also returns the 6 byte ID 1EH + o#(@Z (sic) in response to a Ctl-^ C interrogatory code, and will automatically switch into ANSI emulation if a Ctl-^ F@ code is received : this will be ANSI with wrap set on if the DG214 emulator had roll set on, or ANSI with wrap off if the DG214 emulator had roll set off.

Ampex: similiar to the Televideo emulation (3). However, unlike both that and the Wyse 30 emulations in which setting an attribute occupies a position on the screen, which maintains an attribute map, the Ampex emulation uses attributes that do not occupy positions on the screen, and which affect characters serially rather than positionally. In other words, switching inverse on in Televideo emulation cannot affect any characters subsequently placed at earlier positions on the screen : whereas in Ampex emulation, any character subsequently placed on the screen has the attributes last defined no matter where this was done. Please see the notes at the end of this section for information on the special details when using the emulation with Ampex Scan codes. The Ampex 232 scan code emulation is compatible with similiar terminals, such as the Wyse 60 in PCTERM mode.

Specifications for the emulations follow.

i) Viewdata emulations

The following codes are supported:

Ctl-E	send ID in userkey ?	Ctl-L	clear screen
Ctl-H	cursor back	Ctl-M	cursor to column 0
Ctl-I	cursor forward	Ctl-X	clear to EOL
Ctl-J	cursor down	Ctl-G	beep
Ctl-K	cursor up	Ctl-^	cursor home

ESC A	alphanumeric red	ESC Q	mosaic red
ESC B	alphanumeric green	ESC R	mosaic green
ESC C	alphanumeric yellow	ESC S	mosaic yellow
ESC D	alphanumeric blue	ESC T	mosaic blue
ESC E	alphanumeric magenta	ESC U	mosaic magenta
ESC F	alphanumeric Cyan	ESC V	mosaic Cyan
ESC G	alphanumeric white	ESC W	mosaic white
ESC H	blink on	ESC I	blink off
ESC \	black background	ESC]	background=foreground
ESC ^	repeat last mosiac	ESC _	release repeated mosaic

ii) Specifications for non-ANSI emulations.

The following codes for the non-ANSI emulations in the table below are accepted : in addition, CR, LF and BEL are accepted as control codes in the usual manner, while TAB will tabulate in columns of 8 on the screen. Variable tabs are no supported. XON/XOFF codes are passed through for COMM to handle in the usual manner for flow control, except in the case of the DG214, which use the XOFF code for disabling screen rolling, and also the Hazeltine and DG214 emulations, which can contain XON/XOFF codes as part of cursor addressing sequences. All other control codes and escape sequences are ignored, as are certain multibyte escape sequences not listed in the table.

The Ampex command to switch to alternate characters is normally used to print out the equivalent code to the one received in the range 128-255 decimal. This command will therefore only have an effect on machines or terminals with a full set of 256 characters instead of just an ASCII set : but the precise effect will depend on both the machine or terminal being used and on the character set currently in operation. The most common use for this code is to enable special graphics, maths, or foreign language symbols to be displayed from a 256 character set, such as the one on a standard IBM PC. This 256-character set is often found as an option on more sophisticated terminals such as Wyse and Ampex units designed specifically for use with microcomputer software, as well as mainframe applications. So while the extended character set command is available on all implementations of COMM, its exact effect will depend on the characteristics of the screen being used.

Support for the 25th line on Ampex terminals for use apart from displaying status messages is only available in PCDOS versions of COMM. Normally, line 25 is normally used for COMM messages. If a 25-line screen has been set during this emulation, you'll see the normal COMM message line disappear from line 25 : however, any COMM commands of any sort that are issued while the 25-line mode is active will cause COMM to reassert its own message line, and the screen will revert to 24-line operation. You can avoid this by using the expert help mode during which no COMM messages will ever appear.

Please also see additional notes on Ampex Scan code emulation later in this section.

	ADM	WY30	TV925	ADDS	Hazeltine	VT-5	DG214	Ampex
Cursor up	Ctl-K	Ctl-K	Ctl-K	Ctl-Z	ESC Ctl-L	ESC A	Ctl-W	Ctl-K
Cursor down	Ctl-J	Ctl-J	Ctl-J	Ctl-J	ESC Ctl-K	ESC B	Ctl-Z	Ctl-J
Cursor left	Ctl-H	Ctl-H	Ctl-H	Ctl-U	Ctl-H	ESC C	Ctl-Y	Ctl-H
Cursor right	Ctl-L	Ctl-L	Ctl-L	Ctl-F	Ctl-P	ESC D	Ctl-X	Ctl-L
Cursor address	ESC =	ESC =	ESC =	ESC Y	ESC Ctl-K	ESC Y	Ctl-P	ESC =
Cursor address		ESC -n	ESC -n		ESC -n			ESC -n
Clear screen	ESC *	ESC *	ESC *	Ctl-L	ESC Ctl-\	ESC E	Ctl-L	ESC *
Clear screen	ESC +	ESC +						
Clear screen	ESC ;	ESC ;						
Clear screen	ESC :	ESC :						
Clear screen	Ctl-Z	Ctl-Z						
Clear screen		ESC y						
Home cursor	Ctl-^	Ctl-	Ctl-^	Ctl-A	ESC Ctl-R	ESC H	Ctl-H	Ctl-^
Home cursor	ESC {	ESC {	ESC {					ESC {
Dim	ESC)	ESC)	ESC)	Ctl-N	ESC Ctl-Y	ESC)	Ctl-\	ESC)
Dim		ESC Gp						ESC Gp
Bright	ESC (	ESC (	ESC (	Ctl-O	ESC Ctl-	ESC (	Ctl-]	ESC (
Clear to E O L	ESC T	ESC T	ESC t	ESC K	ESC Ctl-O	ESC K	Ctl-K	ESC t
Clear to E O S	ESC Y	ESC Y	ESC Y	ESC k	ESC Ctl-X	ESC J		ESC Y
Clear line		ESC t						
Insert Line	ESC E	ESC E	ESC E	ESC M	ESC Ctl-Z	ESC L		ESC E
Delete Line	ESC R	ESC R	ESC R	ESC 1	ESC Ctl-S	ESC M		ESC R
Inverse		ESC G4	ESC G4			ESC p	Ctl-V	ESC G4
Normal		ESC G0	ESC G0			ESC q	Ctl-B	ESC G0
Blink		ESC G2	ESC G2				Ctl-N	ESC G2
Steady		ESC G0	ESC G0				Ctl-O	ESC G0
Underline on		ESC G8	ESC G8				Ctl-T	ESC G8
Underline off		ESC G0	ESC G0				Ctl-U	ESC G0
Status line		ESC f	ESC f		ESC f			ESC f
Status line		ESC F	ESC F					ESC F
Reverse line feed		ESC j	ESC j					ESC j
Line graphics		ESC H						
Return cursor address		ESC ?	ESC ?				Ctl-E	ESC ?
Blink enable							Ctl-C	
Blink disable							Ctl-D	
Inverse							Ctl-^D	
Normal							Ctl-^E	
Clear to E O S							Ctl-^FF	
Roll on							Ctl-R	
Roll off							Ctl-S	
Transparent print on		ESC '	ESC '					ESC '
Transparent print off		ESC a	ESC a					ESC a
Alternate characters								ESC $
Normal character set								ESC %
Protect on			ESC &					ESC &
Protect on			Ctl-O					Ctl-O
Protect off			Ctl-N					Ctl-N
Protect off			ESC '					ESC '
Clear unprotected to E O S			ESC y					ESC y
Clear unprotected to E O S			ESC Y					ESC Y
Clear unprotected to E O L			ESC T					ESC T
Clear unprotected screen			Ctl-Z					Ctl-Z
Clear unprotected screen			ESC +					ESC +
Clear unprotected screen			ESC ;					ESC ;
Clear unprotected screen			ESC :					ESC :
Set 24 line display								ESC g
Set 25 line display								ESC e

Note: E O L = end of line E O S = end of screen
Protected and 25 line modes only implemented in D O S versions

ESC [Ps J - erase screen (leave cursor in place)
- 0 or none = erase from cursor to end
- 1 = erase from beginning to cursor
- 2 = erase entire screen

ESC [Ps K - erase line (leave cursor in place)
- 0 or none = erase from cursor to end
- 1 = erase from beginning to cursor
- 2 = erase entire line

ESC [s - save cursor (same as ESC 7)
ESC [u - restore cursor (same as ESC 8)

ESC [Ps; ... Ps m - set character attributes as follows:

ESC [m	attributes off	ESC [0 m	attributes off
ESC [1 m	bright on	ESC [22 m	dim on
ESC [4 m	underline on	ESC [24 m	underline off
ESC [5 m	blink on	ESC [25 m	blink off
ESC [7 m	reverse video on	ESC [27 m	reverse video off

ESC [3x m - set foreground colour
ESC [4x m - set background colour

where x 0 = black 1 = red 2 = green 3 = yellow
4 = blue 5 = magenta 6 = cyan 7 = white

ESC [n X - delete n characters forward from cursor
ESC [t ; b r - set scrolling window from line t to b

ESC [? 6 h - set origin to window
ESC [? 6 l - reset origin to screen

ESC [? 7 h - sets wrap on
ESC [? 7 l - sets wrap off

ESC (0 - sets line graphics characters as G0
ESC (A - sets standard ASCII set as G0 (default)
ESC) 0 - sets line graphics as G1 (default)
ESC) A - sets standard ASCII set as G1

iii) Ansi Emulator Specifications

The following control codes are accepted: all other are ignored.

BEL - sound bell
TAB - tab in columns of 8 (variable tabs not supported)
BS - non-destructive backspace
LF - move to same position on next line or scroll
VT - same as LF
FF - same as LF
CR - return to first position on line
SI - switch to G0 character set
SO - switch to G1 character set
DC3 - XOFF passed through to COMM's own XON/XOFF processor
ESC - beginning of ESC sequence (see below)

The following escape codes are accepted: all others are ignored.

ESC D - same as LF
ESC E - same as CR plus LF
ESC M - reverse line feed or scroll down if on line 1
ESC 7 - save cursor position (but not attributes - DEC)
ESC 8 - restore cursor position (but not attributes - DEC)
ESC [- beginning of ANSI parameters (see below)

(Note that the scroll down if at top feature of ESC M is not available if line insert is not possible on your CRT.)

The following ANSI codes are accepted. If no parameter are given, the default of 0 is assumed. All other ANSI codes are ignored.

ESC [Pn A - cursor up n lines or till line 1
ESC [Pn B - cursor down n lines or till line 24
ESC [Pn C - cursor right n places or till col 80
ESC [Pn D - cursor left n places or till col 1
ESC [Pl ; Pc H or
ESC [Pl ; Pc f - direct cursor addressing (Pl = line Pc = col)
ESC [6 n - report cursor position

iv) Notes on emulations.

a. Some of the codes in the previous tables require a suitable GRAPHICS file for correct implementation on non-IBM PCDOS systems. These include dim/bright, underline and blink/steady attributes, colour ANSI codes and also all line graphics. If a suitable GRAPHICS file is not available, line graphics display as dots and the colour, underline, dim/bright and blink/steady attributes are ignored. However, inverse/normal attribute codes are contained in the DRIVER file and are thus always implemented.

b. The WY30/50, TV925, Ampex and Hazeltine emulations all treat the ESC - n cursor addressing sequence (where n is a page number) identically to the usual cursor addressing sequence : the page parameter n is discarded. This effectively makes these emulations single-page only. Additionally, the status line display codes for these emulations cause the text that would have been displayed on the status line to be discarded, except on IBM PCDOS systems, when the status line overwrites the COMM prompt on the 25th line of the screen.

c. The clear unprotected data codes for TV925 and Ampex emulations are only implemented on IBM PCDOS versions of COMM. They erase all characters that haven't been protected with the ESC) attribute only if protect mode is on : when protect mode is off, all characters are treated the same. The ESC * clear screen code automatically turns off protect mode.

d. The Hazeltine emulation will accept either the tilde ~ character as a leadin character for certain command as well the ESC code listed in the table

e. The ESC G attribute parameters for the WY30/50, Ampex and TV925 emulations can set multiple combinations of inverse, underline and blink (and dim/bright on the WY30 and Ampex) with a single character : the bits in the character are used as masks for the attributes required. Bit 2 blinks, bit 4 inverses, bit 8 underlines and on the WY30/50 emulation, bit 40 sets dim. In Wyse and Televideo emulations, the ESC G parameter displays a attribute space on the screen, and the attribute selected is active to the end of the screen

in the WY30/50 emulation, or to the end of the line in the TV925 emulation. In Ampex emulation, the ESC G parameter does not display as a space, and the attribute selected is active until cleared.

f. The ESC H graphics parameter for the WY30/50 emulation displays the next character received as a line graphics character if appropriate. However, ESC H Ctl-B sets graphics mode on, when all appropriate characters will be displayed as line graphics: ESC H Ctl-C turns graphics mode off.

g. The only character sets implemented for ANSI emulations are the usual character set for your CRT and the same character set but with line graphics substituted for codes 90 through to 104 decimal. Either set may assign as G0 or G1, and the SI/SO control codes can be used to switch between the G0 or G1 sets : or alternatively, the G0 default set can be redefined. Neither G2 nor G3 character sets are implemented.

h. Keyboard support when in an emulation.

Certain characters on the keyboard will be re-interpreted when an emulation is switched on : these are the characters that are used for cursor control, which are re-interpreted to send the correct cursor control sequence for any particular emulation. Apart from these keys, and any specific keyboard support mentioned in the descriptions of the emulation earlier in this section, no other keyboard support is offered as standard in any emulation. However, you can set up the user keys to generate any specific multiple codes that you might need to send for any application. Remember that if programming the user keys to send an ESC code, you should programme the sequence with a double ESC if you haven't changed the COMM command key from the default.

Thus, if using an ADM type of terminal, the arrow keys generate Ctl-K for up, Ctl-J for down, Ctl-H for left and Ctl-L for right. Pressing the up-arrow key when in ANSI emulation will therefore send not the Ctl-K generated, but will instead send ESC [A in accordance with the ANSI specification for cursor up.

This may occasionally present problems, especially for terminals that use a backspace (decimal 8) as the cursor left code. When in ANSI emulation, for instance, attempting to send a backspace would always result in ESC [D being send. This is not satisfactory if you actually want to send a real backspace - which isn't that unlikely.

In order to do this, you should press the command key before pressing the left arrow in much the same way as you press the command key twice to send it. Assuming that the command key is ESC, in the case outlined above, pressing ESC and the left arrow will send a backspace rather than ESC [D.

i. The Wyse, Televideo and Ampex emulations all support transparent printing, when all incoming characters of any sort, including escape and control codes, are diverted directly to an attached printer until an ESC <a> code is received. This is different to simply setting printer echo on in COMM, as characters aren't either sent to the screen or stored in memory during transparent printing. During transparent printing, the keyboard can only be used to send characters to the host, not to issue commands to COMM. The only exception to this is pressing control-P, which will terminate the print routine in the emulation. This is provided in case the 'ESC a' code to stop transparent printing doesn't arrive due to line noise or some other reason.

j. Ampex Scan codes:

While Ampex 232 emulation D returns ASCII codes for keys pressed in the normal way, emulation E returns IBM scan codes instead. Both emulations behave identically to all data received from the host machine. The only difference is in the way they treat characters sent out.

The IBM scan codes used by this emulation are the standard US keyboard set detailed in the IBM technical manuals. However, on non-IBM systems, scan codes are returned only for equivalent ASCII characters, and also for supported arrow keys on non-IBM keyboard (thus a down arrow pressed would sent the scan code for the down arrow key, which is key 2 on the IBM numeric pad).

On IBM systems, scan codes are additionally returned for those key combinations for which the BIOS is able to provide the scan code : these include all the basic function and numeric pad keys, and combinations of the Alt key and any of the alphanumeric keys.

However, the IBM BIOS provides no support for detecting combinations such as control plus alt, or control plus a numeric key, nor for detecting any of the state-lock keys such as Num Lock or Caps Lock, nor for isolated keypresses of the control or shift keys. The full list of BIOS-supported scan codes is given in the keyboard extended code section of the IBM technical manual. IBM system users should observe that in this emulation, the function keys will not generate user key sequences when on-line : however, the alternative of ESC 0 through ESC 9 can be used instead.

For non-IBM systems, COMM provides additional support for sending scan codes via five extra command sequences available when emulation E is selected. (These are also available on IBM systems, though they wouldn't be needed except for highly unusual and deliberately obscure combinations which are only used by memory resident programmes.) The extra command sequences are as follows.

<CMD> [- sends the scan code for Num Lock
<CMD>] - sends the scan code for Caps Lock
<CMD> \ - sends the next key pressed as an Alt combination
<CMD> _ - sends the next key pressed as a Control combination
<CMD> ^ - sends a scan code based on the next ASCII key pressed.

This last sequence starts from a base of 59 for the scan key (this is F1, which is the first of the non-ASCII keys on an IBM keyboard), and from a base of 0 for the ASCII key pressed - that's 48 decimal. Thus <ESC>^<1> sends an F1 scan code, <ESC>^<2> sends F2 and so on. This enables the commonly-used function keys to be sent from non-IBM systems with the minumum trouble.

The arrow keys are supported, as described above, in the normal course of events : and if needed the entire sequence of the lock keys and those numeric pad can be sent via <ESC>^<;> through to <ESC>^<R>. Commonly used non-ASCII keys can be programmed as

user keys if needed. While the above scheme is certainly complex, there is no simple way of generating scan codes for keys that aren't present on a keyboard, especially as there are no immediate candidates for special treatment (apart from the cursor keys and the function keys, which have been made as straightforward as possible).

Please observe that when using emulation E, all characters sent out are translated into scan codes, not just keys sent after being pressed on the keyboard. Files sent either as text or via the block transfer options are converted to scan keys as they go out. This enables the normal techniques for uploading text to be used : and the conversion of ASCII to scan codes has also been tested successfully with a Kermit server. Though this may seem odd at first, it must be remembered that the keyboard input routines of computers designed to take scan code terminals simple transalate scan codes they receive back into ASCII again. However, people intending to use this emulation for connecting to systems over modems must follow the usual procedure of not enabling the scan code emulation until the modem has connected : autodial modems can neither use scan codes nor convert them into ASCII.

22. USING COMM TO ACCESS PRESTEL

Most of the information on this page is also given elsewhere in the manual, but for Prestel users, a summary is given here.

i) To access Prestel with COMM, a modem or acoustic coupler capable of 1200 baud full duplex operation is needed. If the device is 1200/1200, access will usually have to be via PSS - normal Prestel access numbers require 1200/75 couplers. Though there is limited access both directly and via PSS at 300 baud, the paged nature of the Prestel system makes access at this speed both expensive and inefficient. The Engineering Database on the Prestel System contains information regarding telephone access numbers at 1200/75, 300/300 and 1200/1200 as well as PSS network address codes.

ii) The version of COMM you are using must be capable of 1200/75 split speed operation (though this is true of most versions of COMM, some implementations do not allow this). Alternatively, a modem capable of speed conversion from 75 baud will be needed - DaCom, Tandata and Master Systems all manufacture the necessary equipment. In this case, COMM may be run at 1200 baud (or at any speed the modem can support via its conversion capabilities).

iii) Delays may have to be tuned via the handshaking menu or switched on with <ESC> I from the on-line menu on many systems, especially with buffered modems.

iv) The user ? key should be programmed with your Prestel ID and, if desired, your password. If not used it must be blank.

v) Prestel Emulation should be switched on with <ESC> W 1 or <ESC> W 0 from the on-line menu.

vi) Make sure that your serial port is set to even parity, seven bits with one stop bit for Prestel use.

vii) Prestel, in common with other Viewdata systems, does not use the standard ASCII character set, but uses the closely related ISO7 character set. This makes very little difference except in relation to

the hash # character. The hash # decimal code 35 in ASCII, but is decimal code 95 in ISO7 - this corresponds to the ASCII underline _ character. Consequently, since microcomputers and COMM all use the ASCII character set, you may need to press underline _ to send what Prestel will recognize as a hash. Thus, if this is the case, to access page 1 on Prestel you should use *1_ and **not** *1# as you would on a conventional Viewdata terminal.

viii) We suggest that a version of COMM suitably set up for Prestel be saved (with the Programme Save feature from the off-line menu) in order to avoid having to set up COMM each time you want to use Prestel.

23. BLOCK TRANSFERS

These functions are accessed via the upload-download options, either via L from the off-line menu or via <CMD><.> (that's the command key plus a full stop) when on-line.

They enable error-free transmission or reception of any file, whether text or programme or other binary file, by means of block sending with headers and checksums. Automatic retry for up to ten consecutive errors is allowed for, and manual exit by pressing <ESC> is catered for also.

There are four modes of block transfer, all of which require compatible software at the other end of the link - if there is no compatible software at the other end, you can still achieve error free transfer using the echo checking option in the handshake sub-menu, though this will only work properly for text files.

On most operating systems each file transmitted is accompanied by a display of the file length : and each file received is accompanied by a display of the amount of free space on disk. However, on MSDOS systems earlier than version 2.0, and on CP/M-80 systems, this does not happen.

Whether sending or receiving files, you will always see a display of the number of bytes read from disk (if sending) or the number of bytes saved to disk (if receiving).

i) MODEM4 compatible transfers.

Simply press R to receive a file or T to transmit a file. In both cases, you are prompted for a file name, which must be unambiguous. After the transfer is completed, you are prompted to press any key to return to the off-line menu. This method of transfer will only work if 8 bit word format is selected on both machines.

The machine at the other end of the link can be running either MODEM4 or any other XMODEM compatible software (including COMM).

ii) MODEM7 compatible batch transfers.

Simply press BR to receive files or BT to transmit files. For transferring files, you are prompted for a file name. Wildcard characters such as ? and * are allowable for multiple transfers. The file name is transmitted (with checksums) to the receiving system, which will automatically delete any file existing with that name before proceeding. After all transfers are completed, you are prompted to press any key to return to the off-line menu. This method of transfer will only work if 8 bit word format is selected on both machines.

The machine at the other end of the link can be running either MODEM7 or any other compatible software (including COMM). Please note that many XMODEM compatible programmes are compatible only in single file mode, and do not include the enhancements for mutiple transfers that MODEM7 contains.

MODEM741 users: Users of MODEM741 programmes which incorporate CRC checking as well as the more normal checksums should be aware that there is a bug in the routine which maintains XMODEM compatability across versions. The part which disables the CRC mode in favour of the checksums works first time on multiple receive transfers, but not on subsequent transfers. Wildcard transfers on MODEM741 programmes with this bug are not practical, though the first file of a batch will transfer properly.

iii) COMM batch transfers.

Operation is identical to MODEM7, with the difference that you must press CT to transmit or CR to receive. The other end of the link **must** be running COMM.

This mode will transfer all 8 bits of the words in a file whether 8 bit or 7 bit data format is being used - under most circumstances, the two linked systems can be using two different data formats (one could be using even parity, and the other odd parity, for instance). CRC checks rather than simple checksums are used, which makes for slightly greater reliability as the odds against errors cancelling out when the checksum is computed are considerably less. There is a time overhead of some 15% more per block using this method, since the 8th bit of all bytes is transmitted separately.

iv) KERMIT file transfers.

Kermit file transfer is available by pressing K when prompted after selecting one of the upload-download options. Though the Kermit protocol is copyright material and is not in the public domain, it is made available in COMM at no extra cost, in accordance with the terms of the general Kermit licence.

The Kermit protocol may be configured via an extra menu available by pressing K from the handshake menu - <CMD><Y><K> when on line or <Y><K> from the off-line menu will display the current Kermit options. Since Kermit is capable of dynamically reconfiguring itself depending on what options the system at the other end of the link is using, users will not normally need to change any of these : however, any changes made to the default settings can be saved via the usual programme save option from the off-line menu. Changes to any of these options should be made using decimal codes rather than pressing the actual characters, Entries are verified for range, but not for clashes with other items on the menu.

It is important to note that option 7 on this menu determines whether 8-bit quoting is done, and if parity is set either odd or even, this toggle must be ON. Since the parity settings for COMM are part of the DRIVER code rather than the main programme, it is quite possible, as with XModem, to leave parity enabled when using 8-bit transfers. With XModem this never works but provided 8-bit quoting is ON, Kermit can send 8-bit

files over 7-bit lines. Unlike the COMM protocol this is not parity independent however. The COMM protocol will transfer files between two machines when they are using different parity settings, but Kermit will not ; you must always ensure both ends are using the same word length and parity settings.

As well as normal uploads and downloads, transfers will work with an unattended Kermit server via the usual transmit option for sending, but using the special Fetch option for receiving. A server Logout option has also been made available, giving four sub-options for a Kermit selection from the file transfer menu.

Additionally, if ESC is pressed after the Kermit option is selected from the file file transfer menus, an abort packet is sent down the line.

v) Telesoftware downloader

COMM includes a telesoftware downloader for use with Prestel and similar databases containing software downloadable in strict CET format.

The downloader is activated when <CMD><.> is pressed whilst on-line with Viewdata emulation active. Instead of going into the file transfer menu, when viewdata emulation is on, the programme will immediately begin to initiate reception of telesoftware. Should another file transfer protocol be required while viewdata emulation is switched on (perhaps for downloading via a gateway) the usual block file transfer menu can still be obtained via option L from the off-line menu, Alternatively, viewdata emulation could be switched on before <CMD><.> is pressed to obtain the file transfer menu instead of beginning telesoftware downloading.

You will usually be prompted to commence downloading on frame c of a programme page, where you will see the programme header line at the bottom. After pressing <CMD><.> your screen will clear, a message saying that the downloader is active will appear, and after a few seconds, the name of the programme being downloaded will appear together with a frame count and the amount of free disk space in kilobytes. The frame count is updated each frame, and each frame is converted back from CET format and stored in memory. When memory becomes full, the contents

will be saved to disk, cleared before reception resumes, and subsequent portions of the programme will be appended to what has already been received. When the end of the programme is encountered, the contents of memory will be saved or appended, memory will be cleared, and the "control returned" message will appear. The programme will automatically redisplay the final frame on screen in order that you can see any instructions which might come after the programme itself.

Note from the above that memory will be automatically cleared by invoking the downloader. Also note that no checks are made for disk or directory full errors - especially with chargeable software, you must ensure that your disk has enough room.

If the downloader sticks (because a frame heading or footing is incorrectly received) you can force retransmission of the frame by pressing <RET>. If this doesn't work, or if line noise is so bad that you seem to be downloading the same frame all the time, you may press <ESC> to abort the downloading - you may find due to line noise, you have been misrouted to a completely different part of the database. Because of problems which may be encountered when downloading via a gateway, there are no timeouts in the programme. However, the delay counter in the handshake menu does slightly affect the time taken to load successive frames.

The telesoftware downloader adheres to the strict CET (Council for Educational Technology) format, with additional features to avoid problems which may be caused with software which is uploaded with an extended CET format. Multiple chained downloads are supported also.

The downloader supports two interpretations of the CET |L (end of line) code are supported. Normally, this decodes to a simple 0DH (carriage return). However, if the line feed toggle in COMM is ON (set via CMD L when on-line) then the CET |L code decodes to 0DH,0AH (carriage return + line feed pair). The reason for this is that there is some ambiguity in the CET specification, and software under both competing interpretations is currently being provided on Prestel.

Regrettably, it isn't usually possible to tell which interpretation has been used in advance. Most of the PCDOS software seems to use |L as CR+LF, while most other software uses |L as CR only. Ideally, in view of the

ambiguity, telesoftware oughtn't contain |L at all, since alternative unambiguous ways of specifying the necessary codes are available. As a general rule, PCDOS users should download with the line-feed toggle ON and everyone else should download with the line-feed toggle OFF.

vi) **Notes on block transfers options.**

a. Timeouts in block transfer modes

The timeouts on the block tranfer routines are not be optimized for either other COMM systems or XMODEM type programmes. However, the length of the timeout is tied to the length of the delay as set in the handshake menu, and is thus configurable by a user. This is also used as the basis for setting Kermit timeouts.

b. Aborting transfers

If a transfer in any mode seems to be generating too many errors, you can abort it by pressing ESC. This key is checked for each time the error count is incremented, and if pressed, will terminate the transfer session. On machines which do not detect end-of-transmission characters which come in during disk write operations, this also enables manual ending of transfer sessions.

c. Memory usage

Contents of memory are unaffected by all the block transfer routines with the exception of the telesoftware downloader. Buffering is done via the DMA in 128-byte blocks for both sending and receiving in all the other file transfer modes. Note that any named files will be lost from menu displays, and will have to be renamed.

d. Illegal characters in transmitted filenames

If using COMM to transfer files between CP/M and MSDOS/PCDOS systems, you should note that slash characters in file names are legal for the former operating systems, but are illegal for the latter. Attempts to transfer a filename with either a forward slash or a backslash will actually result in a directory full error. You must rename such files before transmitting the batch.

24. CONFIGURABLE MEMORY BUFFERS

This feature is not available on 8-bit versions of COMM, which use the full transient programme area (TPA) for the programme and use whatever is left over for memory buffers. However, all 16-bit versions of COMM can be configured for a particular size of memory buffer : while this isn't necessarily useful for single-tasking applications, it is available for more efficient memory usage on systems where COMM isn't the only programme that is held in memory. These can be either concurrent operating systems capable of handling multiple programmes (like CDOS or certain variations on MSDOS like Carousel or Desqview) or normal PCDOS or MSDOS versions of COMM using the DOS gateway to access other programmes, or else the optional TSR version of COMM which can multitask on normal PCDOS systems.

In all cases, the size of the memory buffer must be declared before COMM is loaded by means of a parameter typed on the same command line. To configure the size of the memory buffer, COMM should be loaded with a numeric parameter on the command line, which should FOLLOW any file name, and be preceded by a /. This parameter will be used to specify the amount of memory buffer space to be used to store and edit data - the default is 64K.

You can specify any amount between 4K and 64K - e.g. COMM /32 would load up a version of COMM with a 32K memory buffer. The amount of memory available is displayed on the off-line menu, and becomes a default when the programme is saved. If the amount of memory specified is unavailable, COMM will use what it can find left over in its default 64K code/data segment.

Concurrent DOS users should note that the allocation units used by the operating system usually means that the smallest amount of memory that can be allocated is 16K, and thereafter increments of 16K are all that can be used. Thus only 16K, 32K, 48K and 64K memory buffers are available.

MSDOS users should note that this feature is unavailable under versions of DOS earlier than 2.0, which will therefore restrict the amount of the memory buffer to whatever is left over in the initial 64K code/data segment.

25. DOS GATEWAY (DOS versions)

When running on MSDOS or PCDOS systems version 2.0 or later, a DOS gateway is available via the W key from the off-line menu. Any DOS command (including loading a COM, EXE or BAT file) can optionally be entered at this point, or control-C can be typed to cancel the gateway request.

If a command line is typed (up to 45 characters allowed), COMM will resume running once the command has finished. Alternatively, if no DOS command is entered (just press RETURN) the normal EXIT command from DOS will re-enter COMM. In either case, COMM will resume either at the off-line menu in manual mode, or with the next command after DO 27,"EW" and BACK in a JCF.

When using DOS, COMM will remain in memory unaltered - all parameters, memory contents and JCFs remain as they were before exiting. The serial port interrupt routines remain active, so up to 4K bytes could be received and kept in the temporary input buffer while COMM remains in background.

It will be appreciated that the combination of a JCF and the DOS gateway is potentially a very powerful combination, especially if used to run batch files.

CDOS users who are using COMM via the PCMODE DOS emulation facility should note that an unsaved CP/M-86 version of COMM can be run as a DOS programme on Concurrent systems by deleting COMM.CMD (the CP/M loader for COMM.COM) and running COMM.COM directly. This enables directory paths and other MSDOS specific features to be used : however, the DOS gateway cannot normally be used on such a system as there is no capability for running a secondary copy of the DOS command processor.

26. CONCURRENT DOS COMM+ (CDOS Queued version)

This section applies to the version of COMM specifically written for DR concurrent operating systems : as well as CCP/M-86 and CDOS, it also runs under MP/M-86 as well.

The major difference between this version and other versions of COMM that may run under identical operating systems (such as the CP/M-86 - MSDOS version, or the PCDOS version) is that the input/output routines are queued in this version rather than polled. A polled communications programme makes continual use of the CPU on a machine, as it has to monitor both keyboard and communications ports, and this continual use of the CPU slows down other programmes running on the system, even when no communication is actually taking place.

COMM itself creates three queues and three child processes. Additionally, a minimum of two processes need to be created by a DRIVER in order to obtain the proper benefit from queuing. In order to retain the ability to run multiple copies of COMM on different consoles, these queues and processes are tagged with a two digit suffix corresponding to the console which is running them.

The extra queues are COMQOU, which is a queue to which all comms output is written, COMQIN, which is a queue from which all comms input is read, and COMASQ, which is a master queue - when COMM has nothing to do it suspends itself by doing an unconditional read from COMASQ, which causes it relinquish the CPU until COMASQ is written to by one of the three child processes it creates. These processes are COMDEL, which wakes up COMM by writing to COMASQ once every second in order that any timed tasks and delays can proceed properly, COMKEY, which monitors the keyboard and writes to COMASQ when a key is available, and COMINP, which monitors for comms input and writes to COMASQ when a byte arrives at the comms port. This last process, which handles comms input, runs at a priority of 199 - a better one than the other processes, which are set to the default priority of 200.

Though input and output is triggered by queues, the routines for actually doing i/o are still handled by a DRIVER module. A suitable driver will create two more child processes : one process will collect comms

characters from the physical port and write them to COMQIN, which causes the COMINP process to store them in the COMM input buffer, and another process will read COMQOU, and send any characters it receives out of the physical port. Both these processes should be set to the same high 199 priority that COMINP uses.

Thus a concurrent system running COMM will actually be running six processes rather than one, and will be using an extra three queues. It may be necessary on some systems to recreate an operating system via GENCCPM with extra process descriptors and queue descriptors if not enough are available. Three of the processes are running at a higher priority than usual user processes, but this does not degrade performance on other consoles, as they are suspended most of the time, and their high priority when they are needed simply reflects the fact that communications is a real-time activity. Hardware handshaking via DTR/RTS/CTS for input and output buffers should be handled by the driver processes if necessary : should standard AUX drivers be used, the usual auxiliary port handshaking ought to be adequate for this purpose. COMM contains its own XON/XOFF flow control routines : care should be taken that they don't clash with any software flow control used by operating system auxiliary drivers.

When configuring operating system AUX ports, you should try to leave the data throughput as transparent as possible, as many auxiliary port handshaking options can adversely affect performance. In particular, you should note that 8-bit file transfer modes will not work if your auxiliary port is configured for XON/XOFF protocol, as data would not be transferred through the port transparently, and binary files that contain codes 11H or 13H will fail to be transmitted or received correctly.

Though it would in theory be more efficient for one single process to handle input direct to the COMM buffers and for COMM to handle its own output, thus saving the need to create two queues in COMM and two processes via DRIVER, this method has the advantage of being able to be used over DR networks, where queued input and output is essential. The disadvantage of slower speed rating isn't important for usual e-mail and modem applications, which don't usually require speeds greater than 2400 baud. For heavy-duty comms work, the polled version of COMM is probably the best alternative available should the queued version prove too slow, though given the high priority that the I/O routines are set at, this should not generally prove to be a problem.

The normal generic CDOS driver supplied with this version of COMM supports multiple AUX ports and multiple consoles : it requires that the operating system supports both the BDOS and XIOS calls for auxiliary ports. The BDOS call A_CATTACH (0A7H) is used by the driver initialization routine to check that the default auxiliary port for the console is currently unused, and the XIOS calls IO_AUXIN (05H) and IO_AUXOUT (06H) are used for input and output processes respectively.

27. IBM SPECIFIC FEATURES (PCDOS version only)

This section applies only to PCDOS versions of COMM running on IBM-type machines. A number of the added features are accessed by means of command line parameters, which are entered in the same way as the command line parameter for configuring memory buffer size that we discussed earlier. Such parameters are entered on the command line to control both the method of writing characters to the screen, and the colours or screen attributes used in the programme.

Where features are accessed via parameters, they can be combined with each other and with the memory parameter. When multiple parameters are combined on the same line, they should be separated by / characters. Where conflicting attributes are given, the earlier one is discarded. All parameters, like the memory specified, are saved via P from the off-line menu and become defaults and so don't need to be entered more that once.

a. **Support for PC keyboards.**

In normal state, the function keys F1-F10 now act identically to the user keys <ESC>1-0, and are thus fully programmable within COMM. When used on combination with the shift, control, or alt keys, the function keys sequentially generate the mnemonic command codes used in COMM. Thus shifted F1-F10 generate <ESC>A-J, control F1-F10 generate <ESC>K-T, with alt F1-F6 generating <ESC>U-Z.

When using the editor, the INSERT key inserts a space and the DELETE key deletes backwards. The HOME key scrolls the text up one line, and the END key scrolls the text down one line. The PAGE UP key scrolls the text up 22 lines, and the PAGE DOWN key scrolls the text down 22 lines.

The control-break key combinations can be used at any time to send a break signal.

The keyboard support outlined above is in addition to the normal key sequences listed elsewhere in the manual : so control-break doesn't replace <ESC></>, it duplicates it : and the usual control

and escape key combinations in the editor function as well as the special IBM function keys. Similiarly, ESC 1 through 0 can be used for the same sequences as the function keys would normally generate.

b. **Support for multiple paging on IBM screens.**

On colour systems, or monochrome systems using EGA cards or CGA emulations, the programme makes use of four different pages on the video card memory. Being able to switch from one screen page to another means that fewer menu redisplays are needed, and that the screen used for communications can be preserved even in situations where another menu or display needs to be used. The communications screen is however cleared when either a new emulation is selected, or the baud rate is changed.

On monocrome adaptors this paging is not available however : all menus and screens use the same screen page, and therefore will display destructively regarding any information that was on the screen before. This is also the case on any adaptor card when a JCF is in memory : paging is disabled.

c. **Provision of a status/command line on row 25.**

The 25th line of the screen is used for displaying single-line COMM prompts and entry of items like file names and user keys.

d. **Support for direct memory-mapped screen driving.**

The /F, /S or /N command line parameters control the method of writing characters to the screen. The method selected is saved along with other defaults via P (programme save) from the off-line menu, and therefore need not be specified each time COMM is loaded.

If you specify /F (e.g. COMM /F or COMM FILENAME /F) the programme will operate with memory mapped video drivers rather than the default BIOS ones - thus it will be a badly behaved programme. The rate at which characters are displayed on the screen will increase by about 50% however.

The second number determines the foreground and background colours to be used for highlighted menu displays.

The third number determines the colour of the text used for displaying data received (which can be overridden in certain emulations by colour codes received via the communications port).

The format of the numbers used is based on each digit representing a code as follows:

0 = black	1 = blue	2 = green	3 = cyan
4 = red	5 = magenta	6 = brown	7 = white

The units of the number determine the foreground colour, whilst the tens determine the background colour. You can add 100 to this to obtain a light foreground. Thus a 2 (same as 02 or 002) would give green colours on black, 106 would give light brown (yellow) on black, while 41 or 041 would give blue on red. For instance, /A102,107,106 gives green displays, white highlights, and yellow text (all in 'light' mode - hence brown=yellow)

An invalid number (e.g /A340,88,6) causes the whole attribute parameter to be discarded.

It is important to realize that the background colour of the first attribute is used for setting the colour of the whole screen (excluding the border) whenever it is cleared, and that consequently a black background is most practical. In any event, should this background colour be different to the background colour set for data received, you'll see that data coming in to the programme displays with a different background colour to the rest of the screen, which can be rather disconcerting if you aren't expecting this.

If you are using either the Prestel or ANSI emulations, which accept colour escape sequences and assume a default of white on black, the display will look equally peculiar if a non-black background colour has been used to clear the screen first.

The /M parameter restores the defaults monochrome attributes of bright, inverse and dim, whilst the /C parameter restores the colour defaults of green, light magenta and white.

On some CGA screens this may result in "snow" type interference on the display. This can be minimized on most systems by using the /S instead of the /F parameter - the display will be slower, but largely interference-free.

The slower BIOS routines rather than direct memory-mapped ones are still used when on-line with emulations off even when direct screen driving is specified, as this invokes the optional ANSI.SYS screen driver included with MSDOS (or any alternative screen driver that may have been loaded). However, the BIOS is not used for any of the emulations or for any other parts of the programmes except on-line displays with emulations off.

The /N parameter can be used to restore the default of BIOS screen driving.

Technical warning : This feature is available only when running on a system with IBM PC style video architecture (i.e. 6845 video controller with memory- mapped video at segment B000H on monochrome and B800H on colour screens) as well as PCDOS style video software interrupts via INT 10H. Use the default BIOS drivers (or load with /N) if using an MSDOS machine which implements INT 10H type PCDOS software interrupts but lacks the appropriate hardware (such as the Sanyo MBC 555 series). Also, if using a multitasking or multiuser operating system such as Desqview or CDOS, direct video memory addressing should probably not be used, and the BIOS default should be used instead.

e. **Support for user-configurable colours.**

The /A, /M, and /C parameters can be used to control the colours or attributes to be used in the programme.

The /A attribute should be followed by three numbers, which can be separated by commas or by spaces.

The first determines the foreground colour which COMM uses for its usual text displays, with the background colour being applied to the rest of the screen; these settings are inverted for the 25th line status display, with any highlighting turned off on the status line background to avoid blinking.

Users of monochrome screens should note that while foreground, intensity and background are selectable, colour selection has no effect except for blue, which will display underlined text in foreground only, while users of LCD screens and other types of psuedo-colour displays will find various combination more than unusually illegible - the /M attributes are recommended in this case.

As well as default colours, /M and /C restore all other attribute defaults including 64K memory buffer and BIOS screen displays.

28. MEMORY RESIDENT COMM+ (PCDOS TSR version)

This section applies only to the special TSR (terminate and stay resident) version of COMM for PCDOS. This is an optional feature within PCDOS versions of COMM, and programmes that can support it have the legend "**PCDOS TSR version**" included on the copyright screen that first displays when COMM is loaded. PCDOS users who don't have the TSR version simply have the legend "**PCDOS version**". All versions of COMM can be upgraded at any time - please contact your distributor or the authors for information on how to do this.

Note that the command line parameters listed here can be combined with any of the other parameters that have been listed earlier, in any order, and can be saved as permanent defaults via a programme save from the off-line menu.

a. Activating or deactivating the TSR option.

The /T or /R parameters determine whether COMM makes use of the memory resident option. The default is T for transient operation, and if this is used then COMM will behave in the same way as the normal PCDOS version of COMM which doesn't have TSR capability.

If you load COMM with the /R parameter, the programme will load into memory and operate in background, and can be called into foreground operation by pressing a hotkey. This defaults to <CONTROL +> for all versions of COMM. A version of COMM saved as a TSR with the R parameter may subsequently be loaded as a transient programme with the /T parameter if needed. Note that the /R parameter is ignored if using a version of DOS earlier than 2.0 and will crash on any machine which is not a component-level IBM clone. Some machines that can run the standard PCDOS version of COMM with either a non-standard or a BIOS-only driver (see Appendix) are of this type.

b. Configuring alternative hotkeys.

Should the default control <+> hotkey be for some reason unacceptable, an alternative hotkey can be specified by use of the /H parameter. This should be followed by a number from 1 to 9 :

the number key specified becomes the hotkey for the version you are running. Thus a /H1 parameter means that you should press control <1> to activate COMM.

A /H parameter with no number will default to the control<+> hotkey. Whether the hotkey is <+> or a number key, you can press the key either on the numeric keypad or on the top row of the keyboard : either will do.

You will find that although two TSR copies of COMM can be loaded at the same time, only the last one loaded actually carries on running concurrently in background while DOS is active. Provided that the programmes have been loaded with different hotkeys, any of them can be activated from DOS (but not from within each other). However, the concurrent running of the last copy loaded will be suspended while any other TSR version of COMM is running.

Consequently, we recommend that should you want to run two copies of COMM at the same time, you do so with one as a foreground process (/T) and one as a background process (/R). The programme has been designed with this in mind, as in such a configuration, the TSR copy is active at all times. Running a COMM/R resident version at the same time as a COMM/T transient programme should present no problems at all, though you'll probably want to set the two up with different display attributes so you don't get them mixed up. Though you can of course run two background copies at the same time, only the second can run concurrently in background, and it will become suspended when the first copy is brought up.

c. **Compatability with other TSRs and time-slicing.**

The time-slicing and context switching that COMM uses when running as a TSR programme may not function properly if another time-slicing TSR programme is running at the same time, or if another TSR loaded before COMM is activated at the same time as COMM is running. Users are advised to check out operation of COMM with any other TSRs they may be using, as no guarantees concerning compatability can be given.

During the period that a TSR version is unable to run concurrently, it does not lose any characters : up to 4K bytes can be received in an interrupt-driven buffer, where they will remain until the programme they relate to is re-activated, when they will be processed. This applies to a non-concurrent TSR that can't run in background because a later TSR COMM has been loaded, and also to the last TSR COMM loaded if it cannnot run in background because either a later time-slicing programme has been loaded, or else because an earlier TSR COMM has been activated.

Certain types of programmes either should not or can not be interrupted by a pop-up - the major categories are are follows:

i. Programmes (like certain compression and backup utilities e.g. PKARC) which bypass DOS to write to disk.

ii. Programmes which don't access DOS (either directly or indirectly) via INT 21H but use FAR calls to the function dispatcher instead.

iii. Programmes which don't use DOS at all - these will cause COMM to suspend till they have finished and returned to DOS. So far, the only programmes found in this category are certain arcade games.

iv. Programmes which neither use the BIOS for keyboard access nor provide an alternative to the INT 16H keyboard vector will allow COMM to pop up and down again, but won't allow any other keyboard input to the programme at all.

d. **Removing COMM from memory.**

Once a resident version of COMM is loaded, it can be ended via the normal quit option and removes itself completely from memory. If other TSRs which use hotkeys, or which use any of same hardware interrupts that COMM relies on, are loaded after COMM, then it is not usually possible to remove COMM as it will detect that other such TSRs have been loaded. Further, if COMM sees that another time-slicing TSR has been loaded, it will immediately stop time-slicing and all context-switching will stop apart from that initiated by a hotkey being pressed.

A second TSR loaded after COMM that doesn't intercept the keyboard, the clock, or DOS will not stop COMM from de-installing itself. However, if COMM is removed from memory, in such an event, a large hole will then be left between where COMM was loaded and where the second TSR lives. Like most programmes of this type, a version of COMM running as a TSR works best as the LAST such programe loaded, and since unlike most TSRs, COMM has the option to remove itself, this should be quite practical. However, other TSRs that may have been loaded prior to COMM should be activated with caution while COMM is resident, as this may result in some system instability - even when in background, COMM is running all the time.

Note that a number of common DOS utilities have TSR elements in them and the TSR portion should be loaded before COMM. These include all keyboard drivers (such as KEYBUK), MODE, PRINT, APPEND and FASTOPEN.

e. **Hotkeying from within a JCF.**

The off-line W DOS gateway, when running a resident version of COMM, has the same effect as pressing a hotkey. The difference is of course that the command can be issued within a JCF, thus enabling COMM to page itself into foreground and back to background again either at specific times or in response to specific occurences. Like other toggles in COMM, the FLAG can be tested from within a JCF after an ESC W - it will be set to 0 if COMM has been placed in background and 1 if COMM has been moved to foreground. The quit command, to remove COMM from memory completely, also works from within a background JCF provided the restrictions above regarding loading other TSRs have been adhered to.

Additionally, the VERSION command in a JCF now returns FLAG=0 if COMM is running in background. If COMM is running in foreground, it returns the version number in the usual way.

f. **Limitations within COMM+ TSR.**

COMM only keeps track of directories on current drives. So though you can be running a programme on the same drive but with a different directory to the one COMM is using, if you switch drives and pop-up COMM, and then go back to the drive COMM is running on after restoring your programme, you'll find that you are using the same directory that COMM is using, as neither COMM nor DOS has kept track of the directory you last used on a different drive.

With no emulations on, COMM normally defaults to using whatever screen driver the operating system has installed (like ANSI.SYS or similar). When running as a resident programme the restrictions on using DOS make this impossible and COMM therefore defaults to a teletype-like emulation, interpreting only carriage returns, line feeds and backspaces as control codes.

COMM always prints out using the default printer. If running as a background programme, this will inevitably interfere with a foreground programme that is also printing. The system will not crash, but the two printouts will appear mixed up.

g. **Screen buffering.**

The amount of memory that COMM needs for buffering screens can be varied by means of the /G parameter, followed by a number. Whilst any number between 4 and 152 is accepted, not all are useful; the only useful numbers are 4, any multiple of 16, any multiple of 28, and any multiple of 38.

i. If a /G4 parameter is used, COMM doesn't use multiple video pages for its menus, and keeps only 8K of memory for screen buffers - one 4K buffer for itself and one 4K buffer for any DOS programme running.

   This means that when running COMM, menus are destructively displayed on the screen (losing any text received) and that when hotkeying between a DOS programme and COMM, only text displayed on the normal text page 0 will be saved. Any graphics screens and multiple pages used by DOS will be lost. COMM

defaults to this setting if it is being used on a monochrome display adaptor, as the MDA itself has only 4K of video RAM, so there is no need to reserve more than 8K of memory to buffer it.

However, the combination of a mono monitor and an EGA card is supported both as regards video paging and extra graphics memory allocation, with the same limitations (as far as attributes are concerned) as colour monitors and EGA or VGA cards. Hercules cards are still unsupported. Hercules users are advised to invest in a cheap EGA clone for their mono monitors.

ii. If a /G16 parameter is used, then COMM uses 4 video pages on the display card for storing its menus, and text received isn't lost from the display unless you switch emulations or baud rates. Up to four multiple video pages and CGA graphics screens are also buffered, enabling COMM to pop-up inside any application running on CGA cards. Note that COMM actually reserves 32K of memory with this setting : 16K is used by COMM screen and 16K is used for storing DOS screens.

If COMM is running on a CGA card, it defaults to this setting if more than 16K of graphics memory is specified, as the CGA only has 16K on board and any extra memory reserved would be wasted. However, if /G4 is specified, then COMM will use that on CGA and EGA or VGA screens as described above, and thus will occupy 24K less.

While graphics screens are saved and restored intact, a number of parameters for all IBM graphics cards are stored in write-only registers and cannot be restored if they were changed from the defaults. These parameters include colour palettes and screen borders, and also the blink disable setting on text screens. There appears to be no way of detecting what these were set to when COMM pops up, and the defaults restored may possibly result in some screen looking a little different from the originals.

iii. Settings above /G16 are only accepted if COMM is being run on an EGA or VGA card, and even where graphics memory above 16K is specified, COMM never reserves more than 16K for itself, as it never used more than this. You should only ever need to

specify more than 16K memory if using programmes that use or run on EGA or VGA graphics screens (such as Lotus or Supercalc graphics, or GEM applications.)

Graphics screens on EGA cards store data in 4 planes which occupy 16K per plane if being used with an CGA monitor (or an EGA monitor in a CGA mode) or else 28K per plane if used in EGA mode. VGA graphics screen use planes that take up 38K each : hence the need to specify graphics memory in multiples of 16K or 28K.

If you reserve memory for one plane only (/G16 for CGA graphics or /G28 for EGA graphics or /G38 for VGA graphics) then you will lose all colours and intensity from graphics screens when popping COMM up in the middle of them, as COMM has to combine the contents of all the planes into one.

iv. If you reserve memory for two planes (/G32 for CGA graphics or /G56 for EGA graphics or /G76 for VGA graphics) then you will lose all colours, but intensity differences will be kept. COMM combines the contents of the red, green and blue planes into one, and saves the intensity plane separately.

v. If you reserve memory for three planes (/G48 for CGA graphics or /G84 for EGA graphics or /G114 for VGA graphics) then COMM will store the intensity plane and the red plane separately and combine the blue and green planes into one. Blue and green both display as cyan, and magenta and yellow both display as white.

vi. If you reserve memory for all four planes (/G64 for CGA graphics or /G112 for EGA graphics or /G152 for VGA graphics) then all screens will redisplay intact. Colour palettes however are returned to the default settings as described above. EGA cards can have 16 colours from a palette of 64 selected, while VGA cards support up to 256 colours. The default palette is effective after COMM has popped-up. You should try to avoid popping up COMM whilst graphics screens are in the middle of being updated, as this has for reason proved to be unreliable, presumably for the same reason.

Note that while /G64 enables all four planes to be saved when using a 640 x 200 EGA graphics screen, it is sufficient for only two planes when using 640 x 350 EGA graphics or one plane in 640 x 480 VGA graphics. In the latter case, either /G56 or /G38 would have performed in exactly the same way. Also note that EGA cards with more than 128K of memory can support multiple video graphics pages and more than four text pages. COMM can only handle the first graphics screen and the first four text pages.

h. **Storing DOS screens in Expanded Memory (EMS).**

If your PC has any LIM-EMS memory installed, then it is possible to use that instead of conventional memory to store graphics screens. This is done by specifying a /E parameter along with the /G graphics memory one. Thus, /G152/E will reserve 152K of EMS memory for storing graphics screens instead of taking up 152K of conventional memory, with a consequent ability to run programmes that themselves use up a lot of conventional memory but cannot utilize any EMS memory installed.

If no EMS memory is installed, or insufficient is available for use, then the /E parameter is ignored. Like other parameters, /E is saved via option P from the off-line menu, and need not be specified each time COMM is run. The parameters for restoring defaults (/M or /C) will reset COMM to use conventional memory if required.

The amount of RAM COMM takes up can be subtantially varied by using the data buffer memory parameter and the /G parameter for graphics and screen memory.

A minimum RAM version of COMM is one loaded without a GRAPHICS driver on disk and with /R/G4/4 parameter. This uses up just under 61K of RAM. The maximum RAM version of COMM is loaded with the /R/G112/64 parameters and occupies just under 241K of RAM. In fact, it occupies exactly 180 Kbytes more than a minimum version. The extra 180K of course comes from COMM using 64K for its data buffer, 112K for DOS screens and 16K for COMM screens. That's 192K of extra memory in total, while the minimum version keeps only 4K for each purpose.

Note that GRAPHICS files take up memory too, for maintaining Viewdata character matrices. On a CGA screen, 1K extra is used and on an EGA screen 3.5K extra is used. If GRAPHICS is removed, this space is freed, and any Viewdata emulations default to phoney graphics. The type of video card being used can be detected without having a GRAPHICS file; it is only used for loading and displaying the Viewdata character set.

If you know the precise memory requirements of any software you want to run concurrently with COMM, then it should be fairly simple to work out how large a version you can run. Assuming a 640K system in which DOS takes up 40K, running a maximum memory version of COMM leaves only about 350K for another programme. This is probably insufficient for most DTP packages, which normally require at least 450K. However, a version with 28K of screen memory and a 4K data buffer takes up a little under 97K, and should run happily in a 640K system with a DTP package, and quite possibly a 125K version (which would enable screen intensities to be preserved) would run also. Of course, if you have EMS memory installed and use the /E parameter, then no matter how much graphics memory you declare, conventional memory is unaffected.

Oddly enough, a version of COMM loaded with /R/4/G4 leaves more space for DOS programmes than one loaded with /T/4/G4, even though the latter doesn't take up 8K in video storage. This is because running a DOS programme via the gateway from a transient programme also loads up another copy of COMMAND.COM, which takes up more space than the COMM minimum video buffer.

COMM+ Language Manual

(this is a blank page)

INTRODUCTION

This part of the COMM manual documents the inbuilt language for generating and executing job control files (JCFs). These enable all features of the programme to be intelligently controlled to a very high degree, and customized screens and menus to be designed to replace the default ones (in foreign languages if necessary).

Unlike many script file processors, the COMM language enables monitoring and intelligent decision-making at all stages of all communications operations, from dialling on intelligent autodialers, logging on to services, to monitoring progress of sessions, saving messages, sending messages and logging off. Multiple JCF files can be nested and run within each other, offering an exceptional degree of flexibility hitherto unknown in communications software.

The language manual contains descriptions of the commands and associated techniques which are available to you when writing programmes, together with many example programmes. The examples are, however, partial. They illustrate the commands and the various programming devices, and none of them should be used as if they were complete programmes in themselves. Though this will usually be obvious from the context, it is worth stressing the point here.

To use any programming language ideally needs an understanding of some programming method (such as flowcharting) and some knowledge of your hardware and operating system. An additional requirement for writing communications software is a reasonable degree of familiarity with the computer you will be communicating with. If you lack any of these background requirements, you should not expect to be able to write good programmes immediately. Learning new skills will take time, so please be patient with yourself and with our software.

TABLE OF CONTENTS

1. WHAT JCFs ARE, AND HOW TO CONSTRUCT AND EXECUTE THEM

JCF stands for Job Control File. A JCF consists of series of instructions - a programme - which supervise the ordinary operation of COMM, and can be used to automate frequently used manual procedures.

JCFs are written in the first place using simple commands (using any text editor such as the one in COMM) and saved to disk with the file extension .SRC.

When a .SRC file has been named, pressing G from the off-line menu in COMM will condense the .SRC file (removing all comments, checking for syntax errors and condensing the command words to single bytes). The resulting JCF file is written to disk.

To execute the JCF file, all you need to do is to have a file with the .JCF extension named in COMM, and press G from the off-line menu. You will go directly to on-line mode, and the JCF will begin executing.

2. SIMPLE JCFs.

Assume that you frequently log-on to various electronic mail systems. The logging-on procedure is the same in each case, and can easily be automated with a JCF.

All you have to do is type GET followed by the characters that your service prompts you with, then DO followed by your response. In both cases, the characters should be inside quotes.

If you need to put a carriage return in your DO sequence, stop the quotes, type 13 (after a comma) and, if necessary, type another comma with more letters inside quotes.

Either command may include sequences up to 80 characters long.

That's all there is to writing a simple JCF. We'll now explain how the GET and DO commands work, and what the number 13 in the above description does. Then we'll give a few simple examples before going on to slightly more complex JCFs.

3. GET

The command GET "aaaaaa" waits until the characters between the quotation marks come in through your communications device, and then carries on with the next line in the JCF.

4. DECIMAL CODES

There are some characters you may want to send which cannot be properly embedded between quotation marks. One of these is the carriage return you send after each part of the process of logging on. To send these, you use the decimal code equivalents of the character - these ASCII codes (as they are known) mean the same thing as pressing the key itself.

The code for the carriage return (or ENTER) key is 13. So, in the examples starting on the next page, we use a 13 without any quotes to indicate that we want to send a carriage return code. A comprehensive table of the ASCII codes with their equivalent character, binary and hexadecimal equivalents is at the end of this guide.

If we had put the 13 in quotes, the actual characters 1 and 3 would have been sent ; but because we didn't use quotes, COMM assumes that we mean ASCII code 13.

5. DO

The DO command works exactly the same way as the GET command, with the simple difference that the letters between the quotes are assumed to be keys you would have typed at the keyboard. You can enter ASCII codes in the same way as you can with GET commands (though the simple examples don't do this).

DO strings have a maximum length of 80 characters and are placed in the COMM keyboard buffer, which has a capacity of 128 characters. This will be emptied when a GET instruction is encountered - if you want to place more that 128 characters in the keyboard buffer before a GET, you should read the section on the BACK instruction.

6. SIMPLE EXAMPLES

a) Logging on to LINK 7500 mailbox ABC000001 with the username of NAME and password of PASS.

```
GET "ID?"
DO "01 ABC000001 NAME.PASS",13
```

b) Logging on to One-to-One mailbox ABC with the password PASS

```
GET "PLEASE ENTER YOUR ACCOUNT/USER NUMBER A SPACE
AND YOUR PASSWORD"
DO "ABC PASS",13
```

c) Logging on to ISTEL Comet mailbox ABC with the password PASS

```
GET "SELECT SERVICE OR PRESS RETURN"
DO "C2",13
GET "PLEASE TYPE YOUR NAME"
DO "ABC",13
GET "PASSWORD:"
DO "PASS",13
```

(Note that in the above examples, the ID and passwords would need to be replaced with real ones if the programmes were expected to work.)

7. DELAY, SET, PART, AND IF.

All of the previous examples are simple because there is no decision making involved in the programmes we have listed. However, your programmes will often have to be a little more intelligent than the ones above. Like most programming languages, the one that COMM uses can do things CONDITIONALLY - that is, only if a certain condition is met. To introduce us to this concept, let's look again at the GET instruction as we used it above.

The examples up until now have all used a simple version of the GET command which will wait forever until the characters you want to get have actually been got. This is all very well for simple applications, but we sometimes need to do things if there is no response from the other end of the communications link instead of just waiting.

To allow this, we can SET a delay counter before the GET command. The delay will count down to itself, and the GET will finish either

i) when the delay has counted down to zero, or
ii) when the characters you want have actually arrived.

In the first case, DELAY will be zero (because it counted down) and in the second case it will be greater than zero.

This type of operation is called setting a timeout - if the GET ends because delay is 0, the GET is said to have timed out.

We set a delay by putting the line

```
SET DELAY= (any number from 0 to 127)
```

on a line by itself in the JCF. If the delay is set to 0, then the timeout feature on the GET is disabled - the GET will wait forever.

After a GET with a non-zero delay, we can then use a simple IF test to see if DELAY is zero, and take action accordingly.

The action we take is to go to another part of the JCF (or not, as the case may be). You can divide a JCF into parts simply by placing the

command PART followed by a part label consisting of up to eight alphanumeric characters on a line by itself in your programme. You can use fewer than eight characters, but if you use more, your label will be truncated at the eighth character. Duplicate part labels or going to undefined parts are illegal operations, and will generate errors. However, you can go to a part anywhere in a programme - you don't have to define parts before you reference them.

For example,

```
PART ONE
```

will define a part as label ONE. The part label 0 is reserved for the beginning of the JCF, and cannot be defined, though you can reference PART 0 whenever you want. The following labels are not allowed:

PART GOTO GO TO THEN

Defining labels with these names will cause errors: however, if these words occur between a label reference and the label itself they will be ignored.

So a reference such as PART 0, TO 0 or TO PART 0 is the same as simply using 0 by itself.

Putting

```
IF DELAY=0 PART 0
```

on a line by itself will go back to the beginning of the JCF if the GET which preceded it timed out.

The examples which follow all use this type of statement - the conditional tests are simple "begin again" instructions referencing the reserved 0 label at the beginning of a JCF.

Notice that we have to reset DELAY after every time we use it (unless we have set it to zero). If we don't, subsequent GET commands will carry on with the residual delay we had from the previous GET.

8. MORE EXAMPLES

a) Logging on to Dialcom (Telecom Gold) mailbox 81:ABC123 with the password PASS. We have to send returns to Dialcom to activate the PAD. So we set a short delay, and carry on sending returns at intervals until the PAD responds, at which point we carry on. All we do is loop back to the beginning each time we get no answer, but set delay of 0 once we are in the system. (This is the procedure used with direct dial to Dialcom - we deal with a technique for PSS access later).

```
SET DELAY=5
DO 13
GET "PAD>"
IF DELAY=0 PART 0
SET DELAY=0
DO "CALL 81",13
GET "PLEASE SIGN ON"
DO "ID ABC123 PASS",13
```

b) Autodialling telephone number 1112222 with a Dacom 2123 modem.

```
SET DELAY=5
DO 13
GET "++"
IF DELAY=0 PART 0
SET DELAY=0
DO "1112222",13
```

c) Autodialling telephone number 1112222 with a Racal CP2123 modem.

```
SET DELAY=5
DO 13
GET "*"
IF DELAY=0 PART 0
SET DELAY=0
DO "DN1112222",13
```

d) Autodialling telephone number 1112222 with a Master Systems 2123 modem.

```
SET DELAY=5
DO 13
GET ":"
IF DELAY=0 PART 0
SET DELAY=0
DO "D1112222",13
```

e) Autodialling telephone number 1112222 with a Hayes Smartmodem.

```
SET DELAY=5
DO "ATZ",13
GET "OK"
IF DELAY=0 PART 0
SET DELAY=0
DO "ATDP1112222",13
```

9. GET CHAR

We've been using the GET command with sequences (strings) of characters. An alternative form of the GET command is

```
GET CHAR
```

which will get just one character from your communications device (or until a timeout, if you have set one).

You can, if you wish, test this character - for instance

```
IF CHAR="A"  PART (n)
```

will go to part (n) of your programme if the character received was an A.

CHAR can also be tested using decimal codes - for example, IF CHAR=65 is the same as IF CHAR="A". CHAR is unique in that it can be referred to either as a string (in quotes) or as a decimal code. This makes it easy to test for control characters - thus IF CHAR=13 would test for carriage returns.

10. UNLESS

A similar command to the IF we've been using is the UNLESS command - the only difference is that IF goes to the part you designate if the condition specified is met, while UNLESS goes to the part you designate if the condition specified is not met. So UNLESS is a negative version of IF.

Here is an example, from a PSS log-on procedure in which we use DELAY, GET CHAR and UNLESS to introduce a short wait in the programme for a PSS PAD we dial up to get its act together.

(The log-on procedure for PSS involves sending two returns, D2, another return, our network user identification <NUI>, and, after the prompt ADD?, the network address we want to connect up to. If we began transmitting to the PAD immediately, it probably wouldn't be ready for us as PSS can't begin taking input immediately. Some PSS pads may need a delay after every character in the sequence - so this may need tweaking if you want to use it directly)

This programme will log on via PSS using NUI ABCDEFG1234567 to address X12345678900.

```
SET DELAY=10
GET CHAR
UNLESS DELAY=0 PART 0
DO 13,13,"D2",13
DO "ABCDEFG1234567",13
GET "ADD?"
DO "X12345678900",13
```

11. JUMP

We've seen that control can be passed to a different part of a JCF with IF or UNLESS conditional tests. Control can also be passed unconditionally with a simple JUMP instruction such as

```
JUMP LABEL2
```

This was implied, of course, in our original discussion of part labels.

```
PART MAIN                          ; main loop
SET DELAY=0                        ; no timeouts
GET WORD                           ; gets words
IF WORD="MORE" PART MORE           ; we filter out more
IF WORD="ACTION" PART ACT          ; and we filter out action
JUMP MAIN                          ; loop for all other words
;
;
;
PART MORE                          ; if we received "more"
SET DELAY=5                        ; set a timeout delay
GET WORD                           ; get another word
UNLESS DELAY=0 PART MAIN
;
; if Dialcom isn't waiting for input, loop
;
DO 13                              ; else do a return
JUMP MAIN                          ; and then loop
;
;
;
PART ACT                           ; if we received "action"
GET WORD                           ; get next word
;
UNLESS WORD="REQUIRED" PART MAIN
;
; if not "action required" loop
;
DO 13                              ; else do a return
JUMP MAIN                          ; and then loop
```


12. GET WORD

As we've seen, you can GET either strings of characters or single characters. A third form of the GET instruction which is extremely useful is GET WORD. Like the other types of GET, the timeouts imposed by setting a delay can be used.

Unless it times out, GET WORD will end when a complete word comes in via the communications device. A word begins with an alphabetic character (A-Z) and ends with any non-alphabetic character (digits, punctuation characters, carriage returns and so on).

The word received is stored in a variable called WORD. This can be tested for in the same way as CHAR can be tested for. For example,

```
GET WORD
IF WORD="BEGIN" PART N
```

will jump to part N if "BEGIN" is received through the communications device.

An example of the use of JUMP and GET WORD with Dialcom follows. When Dialcom is sending mail to you, it often prompts you with things such as " more " or "Action required". The purpose of this example is to filter out such prompts and send returns back, which enable Dialcom to carry on sending. (We use Dialcom here as it is the most complex electronic mail system in public use, so offers the best examples).

You will notice that we have used semicolons to divide up the parts of the programme. This makes it easier to read. Lines beginning with semicolons are ignored by COMM when creating JCFs, as are any characters on a line after a semicolon (except one inside quotes). It's about time we introduced this feature, since our examples are getting rather complicated.

Semicolons can be used to annotate programmes as well as dividing then up. So here's an annotated version of the programme.

13. ACCESSING COMMANDS IN COMM

Up until now, we've used the DO command only to send characters out of the communications device. However, the DO command is considerably more versatile than that.

The reason why is because the characters following a DO command appear to the main part of COMM as if they are typed in at the keyboard - rather like user keys, in fact. So since we use the keyboard to access the commands inherent in COMM, we can use the DO command in the same way.

We assume that the ESC key is your command key in these examples - the decimal ASCII code for ESC is 27.

So the command

```
DO 27,"N"
```

will ask you for a file name. And the command

```
DO 27,"NTEST.TXT",13
```

will name a file TEST.TXT.

If, after any of the sample logging on programmes we listed earlier, we ended with the command

```
DO 27,"G"
```

we would have switched on memory after logging on.

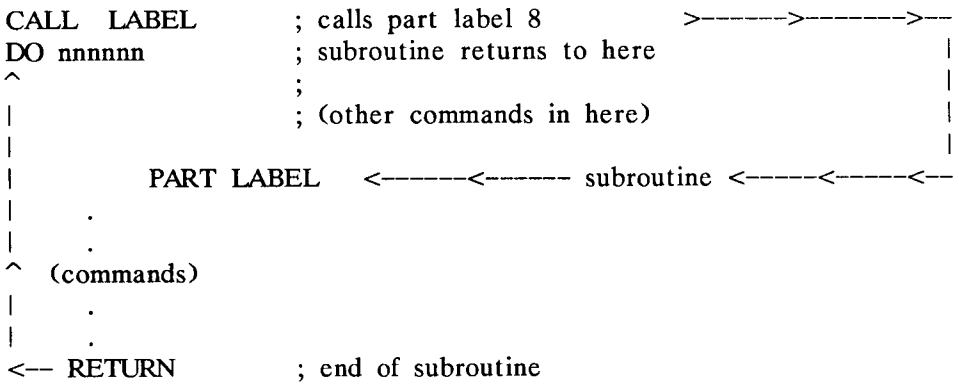
So the DO command makes the full power of COMM available from within a JCF.

14. SUBROUTINES IN A JCF - CALL and RETURN

As well as being able to JUMP to a part of a JCF, you can set up subroutines within a JCF and CALL them. The subroutines must end with a RETURN command, and can be nested (up to 32 levels of nesting are possible, but you are advised not to use that many as it makes your programmes impossible to maintain or debug).

(Those of you unfamiliar with the concept of subroutines are advised at this point to read any elementary programming book - you will not really be able to use many features of JCFs if you have no idea of what subroutines are and how to use them.)

The subroutine returns to the main portion of your JCF at the instruction following the CALL. The format is as follows (symbolically):



ii. The FLAG variable also becomes set when you change any of the toggles in COMM : if a toggle is switched on, FLAG is set to 1, and if a toggle is swiched off, FLAG is set to 0. This applies to toggles such as the handshake, line feed and printer toggles, and also to the on/off toggles on the handshake menu.

iii. The JCF commands QKEY, HOLDPORT, WHATSIGN and VERSION also set the FLAG variable : this is detailed later on.

By using commands such as

IF FLAG=0 PART LABEL

or

UNLESS BUFFUL=1 PART LABEL

you can check whether your disk filing and memory saving needs attention. Note that NAMING files sets FLAG - while a JCF is running, the naming files operation includes a directory search for a file you have named.

A simple use of both the above variables follows, which duplicates the autosave-to disk function which COMM usually implements when its buffers become full - it isn't a programme that you'd need to use unless you had disabled the autosave for some reason (see the Handshake section in the Supplementary Reference Guide for information on this).

You should note that in this example we assume that COMM is running in help level 0. If you want to run under level 2, you'll need extra 'Y' characters in your DO sequences for when COMM asks you to confirm something like a clear memory. If you remember, the confirmatory Y/N prompts for commands issued when on-line are not used in help level 0.

15. SYSTEM VARIABLES

In order to enable you to automate the programmes you write even further, there are two system variables you can test for in your programmes.

Variables are things which are liable to change - these are system variables because they are changed by the system.

The two system variables are BUFFUL and FLAG.

BUFFUL has two possible values - it is normally set at 0, and becomes set at 1 when the memory in COMM becomes full, either through reading files from disk or through receiving characters via the communications ports. Clearing memory is the only way to set it to 0 again.

FLAG usually reflects the status of any disk operations. It has four possible values when used with disk operations -

It is set to 0 by saving, appending, deleting, sending, naming, renaming and getting directories if the named file or files exist.

It is set to 1 by the operations above if the named file or files do not exist.

It is set to 2 by save and append operations if your disk becomes full.

It is set to 3 by save operations if your disk directory becomes full.

NOTE: Non-Disk uses of FLAG.

i. After defining a user key, FLAG will contain the number of characters entered into that key, or 127 if either the 20 character limit was exceeded or the operation was aborted with control-C. This enables you to force entry of a specific number of characters into user-defined string space if you so wish. An example of this is forced password entry.

```
DO 27,"G"                          ; switch on memory
;
PART MAIN                          ; main loop
GET CHAR                           ; gets characters
UNLESS BUFFUL=1 PART MAIN          ; until buffer fills up
;
DO 27,"NMAIL.TXT",13               ; when buffer is full name MAIL.TXT
BACK
;
; *** BACK is explained in next section ***
;
IF FLAG=0 PART APPEND              ; if it exists append
DO 27,"S",27,"C",17                ; else simply save it
                                   ; then clear memory
;
; we end with 17 sent. This is CONTROL-Q (XON) since
; buffer full automatically sent CONTROL-S (XOFF)
;
JUMP MAIN                          ; then loop for more
;
; we jump here if MAIL.TXT exists
;
PART APPEND
DO 27,"A",27,"C",17                ; same as above but append
JUMP MAIN                          ; and then carry on
```

Note that the above programme sample doesn't check for disk full or directory full errors. You might want to work out how to do this yourself, as an exercise.

MSDOS/PCDOS AND CERTAIN OTHER SYSTEMS ONLY:

There are three extra system variables under some systems which enable you to get and set the operating system clock. These three variables are HOURS, MINUTES and SECONDS.

HOURS should be SET to a value from 0 to 23.
MINUTES and SECONDS should be SET to a value from 0 to 59.

Any of these three can by tested for by using IF or UNLESS statements. Thus the following code would execute an automatic reading of mail (in an AUTOREAD routine) every two hours.

```
SET HOURS=0          ; set hours at zero
PART WAIT            ; if we want to be able to break this we
                     ; put a back here enabling us to abort.
BACK
UNLESS HOURS=2 WAIT
                     ; loop if not two hours gone
CALL AUTOREAD        ; do the automatic reading
JUMP 0               ; and then begin again
```

HOURS, MINUTES and SECONDS can be set or tested under non-MSDOS/PCDOS systems, but the values they return might not change if certain standard clock calls are not implemented. (These clock calls are function 104 and 105 under CP/M PLUS, MP/M-80, MP/M-86 and CCP/M-86, or the data area given by the function 49 SYSDAT address offset by 32 under CP/M-86 1.1. These calls are not always fully implemented by system designers, but making them should do no harm. Consult your technical support people if in doubt. CP/M 2.2 systems have no standard clock calls at all so they cannot support clock variables.)

16. BACK

The BACK command (which we used in the examples we just listed) is slightly complex in its use, and needs some understanding of how a JCF routine interacts with the rest of COMM.

Like most communications programmes, the kernel of COMM are routines for getting characters from the comms device and putting them on the screen, and routines for getting characters from the keyboard and sending then out through the comms device.

There is only one point at which the JCF supervisory routines are called, and that is directly after the comms device input routine. The JCF retains control of the programme until it encounters a GET instruction. Note that the GET instructions are the primary purpose of the JCF routines.

DO commands, which place strings in the keyboard buffer, are thus not acted upon until the JCF comes to a GET command. The capacity of the keyboard buffer is 128 characters. This should not present any difficulties in normal use, but sometimes one needs to be certain the keyboard buffer is empty before continuing a JCF. For instance, the two commands

```
DO 27,"S"          ; save file on disk
IF FLAG=0 .....    ; see if save is successful
```

would not normally work because the DO command simply places the save command in the keyboard buffer - it doesn't actually save memory. So the status of FLAG after the DO would not reflect whether the save were successful or not. Use of a BACK statement between the DO and the IF ensures that the buffer has been emptied (and therefore the instruction to save acted upon) before the flag is tested.

However, any characters coming into the system during the BACK command are not available for GET commands to check - normally, any character coming into the system is buffered (held) until a GET command occurs, and the JCF doesn't return control between GETS, as explained above. So the BACK command should only be used when it is essential to ensure that a DO has been done.

17. USER VARIABLES

As well as the two system variables, there are 120 possible user variables a JCF can access. You refer to them as VAR0 to VAR119, and unlike the system variables FLAG and BUFFUL, you can SET them as well as test them. The command for doing this is the same SET command we used to set delays for the GET functions. Thus

```
SET VAR0=0
```

sets user variable 0 to 0. The command

```
SET VAR1=VAR0
```

would set variable 1 to zero as well. The command

```
SET VAR1=VAR0+1
```

would, when variable 0 is 0, set variable 1 to 1.

The maximum value for a variable, like the delay, is 127.

Variable can be subtracted too.

```
SET VAR0=VAR1-10
```

would set variable 0 to ten less than variable 1.

There are, as you can see, a number of ways you can use the SET command with user variables.

One obvious use of user variables is in loops. The concept of a loop is familiar to most programmers - all it means is doing things a certain number of times. For instance, the example on the next page will transmit 10 lines of "HELLO".

```
SET VAR1=49                    ; var1 to "0"
;
PART EXT2                      ; 2nd char extension
SET VAR2=49                    ; var2 to "0"
;
PART EXT3                      ; third extension
SET VAR3=49                    ; and var 3 to "0"
;
;
PART NAME                      ; for file naming
DO 27,"NMAIL.",VAR1,VAR2,VAR3,13
;
; first time names "mail.000"
;
BACK                           ; wait till done
IF FLAG=1 PART DONE            ; not found=save
;
SET VAR3=VAR3+1                ; last number+1
UNLESS VAR3=58 PART NAME       ; loop unless past 9
;
SET VAR2=VAR2+1                ; if past 9 middle+1
UNLESS VAR2=58 PART EXT3       ; loop to 0 for last
;
SET VAR1=VAR1+1                ; unless middle is 9
UNLESS VAR1=58 PART EXT2       ; when we increment first
                               ; and zero other two
PART DONE
DO 27,"S"                      ; save file now
```

You can modify this routine and put it in any of the examples we listed earlier, for detecting buffer full conditions. And as a further exercise, see if you can work out a way to automatically save incoming data to disk every 128 characters.

You may want to know that if you add 1 to a variable with an existing value of 127, it becomes 0. Observe also that a more elegant routine is possible - the above routine would take 998 directory searches to locate MAIL.999. However, it is possible to do it in a maximum of 30. This brainteaser is thrown in for nothing.


```
SET VAR1=0                     ; initialize variable
;
PART DO                        ; do loop
DO "HELLO",13                  ; does hello and a return
SET VAR1=VAR1+1                ; increments variable
UNLESS VAR1=10 PART DO; and loops till we've done 10
```

However, the most useful thing about the user variables in COMM is that they can be embedded in character sequences for use in the DO command. We've already seen that characters between quotes and ASCII decimal codes can be embedded in DO strings. You can put user variables in DO strings as well. They are interpreted as the decimal equivalent of the ASCII value they happen to have at the time of the DO.

In the next example, we do the same as we did in the last one, with the difference that instead of sending just "HELLO" we send "HELLO 0", "HELLO 1" and so on. We use the ASCII decimal code for 0, which is 49, as the starting point for our user variable instead of 0.

```
SET VAR1=49                    ; initialize variable to "0"
;
PART DO                        ; do loop
DO "HELLO ",VAR1,13            ; does hello, number, return
SET VAR1=VAR1+1                ; increments variable
UNLESS VAR1=58 PART DO         ; loops till we've reached "9"
```

The principle should be quite clear by now. The next example is a more useful one which automatically numbers mail messages you may want to save.

18. ABORT and FINISH

These commands are really quite simple to use.

The ABORT command will stop execution of your JCF. You may want to do this if you come across something that needs manual attention, such as changing disks. For instance, your disk may be full. Here is an example.

```
DO 27,"S"                    ; save a file
UNLESS FLAG=2 PART nnn ; carry on if OK
ABORT                        ; but abort if disk is full
PART nnn                     ; carry on here
```

You can resume a JCF manually after an ABORT command by pressing ESC @ from the on-line menu. This is in fact a toggle. If a JCF is executing, pressing ESC @ will abort it manually - if it is suspended either because of a manual abort or by use of the ABORT command within a programe, pressing ESC @ will resume the JCF with the next command.

The FINISH command is just what it says. It finishes the JCF. You can if you wish put a FINISH as the last line of any JCF, but you need not. The JCF will stop after the last command in any case. The main use of the FINISH command is so that you can place subroutines after the end of the main part of the JCF instead of having to jump around then to the real end. Obviously, you need to keep subroutines separate from the main part of your programmes, and FINISH lets you do this.

19. VERSION, WHATSIGN, QKEY and HOLDPORT

The following miscellaneous JCF commands all reset the FLAG variable.

VERSION returns the COMM version number in FLAG - for instance, 40 for 01.4-0 - this has been introduced to facilitate version independent programming. When a TSR version of COMM is running, the VERSION command can also be used to detect whether or not COMM is in foreground or background - VERSION returns a value of 0 if COMM is in background, or else returns the real version number if COMM is in foreground.

WHATSIGN can be used after conditional arithmetic jumps and returns with FLAG set 0 for negative and 1 for positive results - this allows greater than and less than operations. For instance, to see if a number key has been pressed on the keyboard :

```
PART INKEY                   ; label for loop
KEY VAR0                     ; get a key
IF VAR0=48 PROCESS           ; 0 is OK
WHATSIGN                     ; set FLAG
IF FLAG=0 INKEY              ; loop if below zero
IF VAR0=57 PROCESS           ; 9 is OK
WHATSIGN                     ; set flag
IF FLAG=1 INKEY              ; loop if above 9
PART PROCESS                 ; carry on here if 0-9 pressed
```

QKEY sets FLAG to the number of keys in the COMM keyboard buffer - this allows detection of key presses.

HOLDPORT toggles the communications port on and off, leaving all characters when off in a 4K buffer until the port is toggled back on. Like the other toggles in COMM, the system variable FLAG is set to 1 when the toggle is switched on, and set to 0 when the toggle is switched off. The main use of HOLDPORT is to avoid incoming data coming in at inappropriate times such as during a menu display on the screen. The recommended method of using HOLDPORT - or indeed, of setting any toggle - is illustrated by a couple of subroutines as follows:

```
PART PORTON             ; label for loop
HOLDPORT                ; switch port
IF FLAG=0 PORTON        ; woops - repeat if switched off
RETURN                  ; back with port on
;
PART PORTOFF            ; label for loop
HOLDPORT                ; switch port
IF FLAG=1 PORTOFF       ; woops - repeat if switched on
RETURN                  ; back with port off
```

20. CHAINING JCFs

You are not restricted to using one JCF at a time. A JCF that is executing can itself load and execute another JCF. There are two commands which will achieve this. (In both, the .JCF extension is assumed.)

The command

    LOAD "JOB2"

in a JCF will replace the currently executing file with JOB2.JCF (assuming that there is such a file on disk).

The command

    EXECUTE "JOB2"

in a JCF will nest JOB2.JCF within the currently executing programme. Then, when JOB2 has finished, control will return to the next instruction after the EXECUTE command in the original file.

It may help to regard LOAD as analagous to JUMP, and EXECUTE as analagous to CALL.

Up to five JCFs can be nested within each other: when a JCF ends at any level, it returns to the JCF at the previous level. There is, however one command which will stop all JCFs at all levels. This is the command

    TERMINATE

on a line by itself.

Unlike FINISH, which stops the current JCF and returns you to the previous level JCF (if there is one), the TERMINATE command at any level is equivalent to a FINISH at level zero - that is to say it stops all JCF execution. This can be achieved manually as well by pressing G from the off-line menu.

So if you are in the middle of a series of JCFs which have gone wrong somewhere, press ESC @ whilst on line (this will suspend the JCF currently running), press ESC E to go to the off-line menu, then press G to terminate all JCF operations at all levels.

21. DESIGNING YOUR OWN MENUS AND PROMPTS

It is possible, using JCFs within COMM, to completely or partially customize not merely the operation of the programme, but also the screen displays and prompts.

The command for displaying your own menu on the screen is, appropriately enough, ONSCREEN. You follow this with exactly the same format for the text you want to display as you would if you were placing the text in the keyboard buffer with a DO command. Like the DO string, an ONSCREEN string has a maximum size of 80 characters (though you can easily put longer text on the screen by separating it into 80 character segments). So

    ONSCREEN "THIS IS A MENU"

will display the text THIS IS A MENU on the screen. (All ONSCREEN text is displayed in UPPER-CASE)

    ONSCREEN 10,13,"THIS IS A MENU"

An example of a custom-designed menu and prompt sequence follows. It simply loads the right JCF for one of a number of possible electronic mail services you might want to connect to.

```
ONSCREEN 0,2,10,0          ; clear,line 10
;
; display this menu
;
ONSCREEN "SELECT THE SERVICE REQUIRED"
ONSCREEN 2,12,0,4," 1 ",3," LINK 7500 (Mercury)"
ONSCREEN 2,13,0,4," 2 ",3," TELECOM GOLD (Dialcom)"
ONSCREEN 2,14,0,4," 3 ",3," ONE-TO-ONE"
ONSCREEN 2,15,0,4," 4 ",3," COMET"
ONSCREEN 2,20,0,4," PRESS A NUMBER 1 - 4 : "
;
PART LOOP                  ; loop if no good
KEY VAR0                   ; key to var0
IF VAR0=49 PART K1         ; if "1"
IF VAR0=50 PART K2         ; if "2"
IF VAR0=51 PART K3         ; if "3"
IF VAR0=52 PART K4         ; if "4"
ONSCREEN 8," ",8,7         ; none - erase entry and loop
JUMP LOOP                  ; (sound bell) to try again
;
PART K1                    ; these are the files to load
LOAD "LINK7500"            ; only you have to write them
PART K2                    ; yourself.
LOAD "DIALCOM"
PART K3
LOAD "121"
PART K4
LOAD "COMET"
;
; by the way, we don't need a FINISH here.
```


will do the same thing but on a new line.

The codes 0-4 have a special meaning in an ONSCREEN sequence.

0 will clear the screen and home the cursor.
1 will just home the cursor.
2 signals a cursor addressing sequence - the numbers following indicating a line number and a column number respectively. So 2,12,25 will position the cursor at line 12 column 25. (The position 0,0 is the top left hand corner of the screen and on a 24 x 80 screen, position 23,79 is the bottom right hand corner.
3 displays succeeding text in normal video display.
4 displays succeeding text in dim video or reverse display.

So, with the ONSCREEN command, you have control over cursor positioning and text attributes.

22. GETTING RESPONSES TO MENUS AND PROMPTS

The other side of designing your own menus is getting keys in from the operator and seeing what they are, and taking action accordingly. This can be done with the KEY VAR command. Entering

```
KEY VAR0
```

will wait for a key to be typed at the keyboard and place the key (turned into upper-case if a lower-case key was pressed) in VAR0 (if you wanted to use VAR1 or VAR119, simply enter KEY VAR1 or KEY VAR119). The key is not echoed to the screen (though you could easily do this yourself with ONSCREEN VAR0).

Once the key is in the variable, you can simply use IF and UNLESS commands to decide what you want to do next, testing the variable you KEYed into. Remember to use the ASCII decimal code - thus pressing A (or a) in response to a KEY VAR1 will put decimal code 65 in VAR1.

23. SUPPRESSING COMM'S OWN MENUS

If you have written a comprehensive set of menus and prompts yourself, you will probably want to suppress all the things that COMM usually puts on the screen. You can do this with the command

NOMENU

on a line by itself. The command

MENU

on a line by itself switches the COMM menus and prompts back on.

When NOMENU is in force, the following changes to the programme occur:

i. The PRESS ANY KEY delays are all CANCELLED and do NOT wait for a key.
ii. All (Y/N) requests in MENU mode are assumed to have a Y response.
iii. If the programme is at the off-line menu and the keyboard buffer is empty, you will automatically be put into on-line mode.
iv. Entry into the editor switches NOMENU off - exit from the editor switches NOMENU back on.

The above changes mean that DO strings which work in MENU mode (especially under help level 2) may not work in NOMENU mode. It is the users responsibility to carefully check and debug all parts of any programmes written to run in NOMENU mode. Remember that ALL screen output coming from COMM rather than from the communications device is suppressed when NOMENU is in force - this includes screen echo of input of file names, user keys and so on. As a general rule, all screen ouput which would not be echoed to the printer were the printer echo toggle on will be suppressed completely.

The only exceptions to this are:

i. the 'Buffer full' message which appears when memory is switched on and becomes full.
ii. the 'Exit to System' quit message.
iii. the display of keys pressed when entering a user-defined key sequence via ESC U n.
iv. Directory displays from the off-line menu. Thus

DO 27,"DTEST.*",13

will simply set FLAG to signify whether any TEST.* files exist on disk. However,

DO 27,"EDTEST.*",13

will display a directory of any TEST.* files on disk.

24. ADVANCED DATA RETRIEVAL - XTRACT and ZERO

The final two commands are especially useful in advanced applications requiring some degree of control over data within memory.

The XTRACT command is identical in syntax to the KEY command. However, whilst KEY VARn gets the character from the keyboard to a variable, XTRACT VARn gets a character from memory to a variable, using the internal reference which COMM maintains pointing to the beginning of sendable memory. This pointer is of course incremented by one each time the XTRACT command is used. When end of memory is reached, the XTRACT command returns nul (0) to a variable, and carries on returning nul each time XTRACT is used thereafter.

The ZERO command is simply a one-word command (like BACK or RETURN). It will reset the pointer used by XTRACT to the beginning of memory, which is the point it is at when the programme is first loaded.

The pointer is also reset each time memory is cleared using the C command in COMM itself: however, ZERO resets the pointer without clearing memory. The pointer used is the same one that COMM keeps to reference the beginning of memory when memory is being sent to screen,

```
DO 27,"CY"              ; clear (yes)
DO 27,"EY"              ; off-line menu (yes) and then
DO "IN",13,13           ; input file (no continuation)
BACK                    ; wait
PART LOOP
     XTRACT VAR1        ; extract char from memory
     IF VAR1=10 LOOP    ; ignore line feeds
     IF VAR1=0 ENDLOOP  ; out when VAR1 returns nul
     DO VAR1            ; if none of above conditions
                        ; are true DO char
                        ; (place in keyboard buffer)
     BACK               ; send and gets any echos
     JUMP LOOP          ; carry on until gone
PART ENDLOOP
     TERMINATE          ; end JCFs
```

This sends the same text as the commands

```
DO 27,"FYT"             ; send file (yes) transmit
BACK                    ; wait till done
```

However, control over the sending process using the latter sequence is lost. The XTRACT command maintains complete control over sending (which is why the line feeds have to be stripped from CR-LF pairs in the example above). Users ought also, if necessary, make sure that their own XON/XOFF processing and/or echo checking is included. This way of sending files will be slower than the normal DO 27,"FYT" or DO 27,"MYT" commands - the overhead is about 10% at 300 baud, but is slightly greater at higher speeds.

The reason why you would want to maintain control is if, for example, you were sending a file to a DIALCOM computer which contain the command/address line

MAIL SEND ABC001

printer or keyboard. So sending memory after using XTRACT will send not from the beginning of memory but from the last character XTRACTed from memory. To send from the beginning rather than this point, preface your memory send command with a ZERO instruction.

There are three main uses of XTRACT.

i) First, it enables you to maintain control of sending operations in cases where COMM's own file transfer procedures are inadequate.

ii) Second, it provides the missing link in enabling transfer of data between variables, user key strings, memory and disk files.

iii) Third, it enables you to develop simple database applications. Examples of these three uses follow.

It will be seen from the examples that, though primitive, XTRACT is an extremely powerful addition to the available commands.

i) **MAINTAINING CONTROL OF SEND OPERATIONS:** Sending files or memory using commands such as DO 27,"FYT" involves losing control of the sending process whilst COMM's internal file or memory sending routines take over - you can press any key to suspend, and any key or ESC to resume, but the JCF is temporarily suspended until the transmission is ended or aborted. With XTRACT, you have an alternative way of sending text which enables you to maintain complete control. An example of sending a complete file via memory using XTRACT follows.

```
DO 27,"CY"          ; clear memory also reset pointer
DO 27,"NVAR0"       ; names file with VAR0 value
DO 27,"EY"          ; off-line (yes)
DO "IN"             ; inputs file (no continuation)
BACK                ; wait for all that
XTRACT VAR0         ; and restore variable 0
```

This technique may be useful for complicated routines which need to store passwords, file names etc. on disk and be able to recall them. User keys can similiarly be saved to disk by transferring them to memory (via a DO 27,"B"). A simple adaptation of the above routine can therefore be used to restore saved strings from disk files to user keys, or transfer data between user keys, memory and variables. This facility is developed further below, where we show how to develop a simple data retrieval system with a JCF which transfers data stored on disk to a user key.

iii) **DATA RETRIEVAL:** Simple database applications are possible using the XTRACT and ZERO commands. The obvious example is a communications context is telephone directories. Assume that you have a file on disk called NUMBERS with the following four-letter names and phone numbers in the following standard data format:

```
GOLD,5803000
LINK,9283600
PRES,618
```

The following programme will enable you to display the directory on screen and recall numbers.

```
NOMENU
DO 27,"C"                ; clear memory
DO 27,"NNUMBERS",13      ; name file
BACK                     ; wait till do done
DO 27,"EI"               ; input to memory
BACK                     ; wait for that too
PART ONERR               ; in case of error
ZERO                     ; point at beginning
```


at the beginning. You would want to stop sending until the prompt

Text:

appeared. Using the XTRACT command, this can be done quite simply.

However, we recommend that unless there is a specific need to bypass the standard text file sending procedures, the existing COMM routines be used wherever possible as they are more efficient and quicker than character-by-character JCF programming. This is why the XTRACT command uses the same pointer as the memory send command. It enables you, in the case outlined earlier, to use character-by-character JCF routines for the first line or two of the text to be sent, and then, once the difficult bits in the address lines have been sent, transfer to the standard DO 27,"MYT" command. This will carry on sending using the normal memory sending routines from the last byte you XTRACTed to the end of memory, giving you the best of both worlds.

ii) **SAVING SETS OF VARIABLES OR USER KEYS:** Though variables and user keys are saved along with the programme using the P command from the off-line menu, the XTRACT command enables variables to be saved in files on disk. An example of saving VAR0 to disk follows.

```
DO 27,"CY"              ; clear memory
DO 27,"B"               ; type into memory mode
BACK                    ;
DO VAR0                 ; enter variable 0 to memory
DO 27,"NVAR0",13        ; names a file
DO 27,"SY"              ; saves the file
BACK                    ; all done
```

Here is an example of a restoring VAR0 if saved as above.

```
ONSCREEN 0                          ; clear screen
ONSCREEN "Select service from this list: ",13,10
                                    ; message
SET VAR0=49                         ; start with one
;
PART DISPNUM
ONSCREEN 13,10,VAR0,". "            ; number of service
     PART THISNUM                   ; first field is ..
          XTRACT VAR1               ; .. service name
          IF VAR1=13 THISNUM        ; ignore linefeeds
          IF VAR1=10 THISNUM        ; and returns
          IF VAR1=0  ENDNUM         ; out if end of file
          IF VAR1=44 NEXTNUM        ; comma=end of name
          ONSCREEN VAR1             ; print char in name
          JUMP THISNUM              ; loop till comma
     PART NEXTNUM
          XTRACT VAR1               ; leave chars
          IF VAR1=0 ENDNUM          ; till end
          UNLESS VAR1=10 NEXTNUM
                                    ; of file or line
          SET VAR0=VAR0+1           ; next number
          IF VAR0=58 ENDNUM         ; this only does 9
          JUMP DISPNUM              ; next number
PART ENDNUM
ONSCREEN 13,"       "               ; erase number dot
ONSCREEN 10,10,13,"Select number to dial up: "
KEY VAR0                            ; get entry
ONSCREEN VAR0,10,10,13              ; echo it
SET VAR0=VAR0-48                    ; decimal from ASCII
ZERO                                ; point to start
PART FINDNUM                        ; to find line
     SET VAR0=VAR0-1                ; one off entry
     IF VAR0=0 FOUND                ; until reached zero
     PART NEXTLINE                  ; next entry
          XTRACT VAR1               ; increment pointer
          IF VAR1=0 ONERR           ; if at end, retry
          UNLESS VAR1=10 NEXTLINE
                                    ; new line found
          JUMP FINDNUM              ; so loop again
```



```
PART FOUND
     XTRACT VAR1                    ; next char is name
     IF VAR1=0 ONERR                ; error if at end
     UNLESS VAR1=44 FOUND           ; loop until comma
ONSCREEN 10,13,"Your number is "    ; message
DO 27,"U="                          ; set up userkey =
     PART NOS                       ; do not yet done !
     XTRACT VAR1                    ; get next digit
     DO VAR1                        ; char in buffer
     UNLESS VAR1=13 NOS             ; until cr
BACK                                ; set up user key
FINISH                              ; no. in user key =
```

The programme just listed will display the selected number on screen and place it in user key = as well. So with an intelligent autodialler simply issue the dial commands and then DO 27,"=" to send the phone number to the modem.

NOTE ON PROGRAMME LAYOUT:

You can see that in the last programme (which is the most complex we've listed up till now) we have use indentations to help structure the code and make it simpler to read. This is a common programming trick, and tabs or spaces can be used to indent lines in COMM programmes, as we have just done.

All characters before the first printable one on a line are ignored when a .SRC file is condensed to a .JCF file. So spaces or tabs will simply aid you in a visual representation of your programme structure.

25. JCF–COMM INTERFACE

We've now covered the operation of all the JCF commands available in COMM, with examples and discussions of how to use them. This section will tell you how your JCF files are incorporated in the main body of COMM itself.

a. GETTING JCFs

A JCF should initially be written as a .SRC file which is then condensed into a JCF file. The condenser takes the first printable character (ignoring tabs and spaces and indentations) as the command. (It will be noticed that all 18 commands begin with different letters). All text on a line after a semicolon is assumed to be remarks. Thus, lines beginning with semicolons are ignored. This enables layout of source code in an easily readable form.

When G is pressed from the off–line menu, and the file named has the .SRC extension, the .SRC file is condensed and checked for syntax (but not logic). The condenser listing is sent to the screen (and to a printer if printer echo is on). The condensed file is written back to disk as a .JCF file if no syntactical errors are encountered. Users should compare the condenser listing with their .SRC source to ensure that what COMM will do is what they really want. Note that only that first character of commands is meaningful to the condenser (even though the whole command is printed). Thus a non–existent TRANSMIT command would actually TERMINATE execution.

Characters after the first one in a line are actually ignored until the first non–alpha character. So DIMWIT "FRED" is the same as DO "FRED" but DØ "FRED" – mistyping a zero for an O – won't do FRED but would do a nul instead. COMM is very forgiving of typing errors, but only on its own terms.

Errors in syntax cause the condenser to stop at the line containing the first error, and the message "**Error in above line**" is displayed.

Duplicated part labels are flagged as error when the duplication is discovered. A message "**Label error**" appears. The same message will appear for an undefined part reference, but at the end of the condensation process.

When G is pressed from the off–line menu and a .JCF file is named, the .JCF file is loaded, and the programme moves into on–line mode, at which point the JCF file will supervise proceedings.

If G is pressed from the off–line menu and neither a .SRC nor a .JCF file is named, no action is taken unless a JCF file (either active or aborted) is in memory. If a JCF is in memory, the whole JCF process is terminated.

If a .JCF file is included on the command line before COMM is loaded, it will be automatically executed.

A file called COMM.JCF on the disk will be automatically executed if no other .JCF file is included on the command line.

b. ABORTING JCF FILES (Manually or under programme control).

Under programme control, this is done with the ABORT command. It can be done manually with ESC @ from on–line mode. This acts as a toggle – thus ESC @ will resume an aborted file. In both cases, a message is printed on the screen saying whether the JCF has been aborted or resumed.

JCFs may be completely cancelled with either a TERMINATE command or G from the off–line menu while a file is in memory. This means that condensing .SRC files while a JCF is running is NOT possible, and that any attempt to do so (or to run a JCF) will terminate JCF operations.

26. JOB CONTROL FILES - technical details.

1. COMMANDS AVAILABLE

1. Programme structure within a JCF

```
PART $          ; label for a part of the programme
JUMP $          ; jumps to part labelled $
CALL $          ; calls a subroutine at part $
RETURN          ; returns from CALL
```

2. Programme structure between JCFs
   (variables and filenames preserved, memory cleared)

```
EXECUTE $       ; nest the $.JCF file within current one.
LOAD $          ; replaces the $.JCF file at current level
```

3. Programme structure between JCF and COMM

```
ABORT           ; suspends execution of a file
FINISH          ; ends JCF at current level
TERMINATE       ; ends all JCFs at all levels
```

4. Decision making and logical commands
   (see also rules for expressions below)

```
IF x=(y) PART $         ; when true, jump $
UNLESS x=(y) PART $     ; when not true, jump to $
SET VAR n = (y)         ; set var n to value x
SET DELAY = i           ; set timeouts to i (disable=0)
WHATSIGN                ; returns sign in FLAG
VERSION                 ; returns COMM version number in FLAG
```


c. MEMORY USAGE

All loading of .JCF files clears memory. Restoring old memory will usually enable only part of this to be recovered. This applies particularly to EXECUTE and LOAD commands, which will clear out any memory. Saves should be done by any programme which runs with memory prior to another JCF being used. File names, user variable contents and user defineable string space are all retained between JCFs LOADS and EXECUTES.

Source files for condensing must fit into memory. Additionally, they should leave ten bytes free for each part definition and each part reference.

The JCF files must leave at least 5K of free memory after loading.

A message saying that the file is too big will display if either of the above two limits are exceeded at any time. Should a JCF be too big, it must be split into at least two sections, which may be linked via EXECUTE or LOAD.

2. RULES FOR EXPRESSIONS

Expressions for IF and UNLESS commands must be of the form

x=(y)

and for SET commands must only be of the form

(y)

where x is either

```
VAR n       ; any variable 0-119
DELAY       ; timeout value or residue after GET
FLAG        ; system flag                          (not settable)
BUFFUL      ; memory full flag                     (not settable)
HOURS       ; hours on clock                       (not all systems)
MINUTES     ; minutes on clock                     (not all systems)
SECONDS     ; seconds on clock                     (not all systems)

or for string expressions                          (not settable)

CHAR        ; character received by GET CHAR
WORD        ; word received by GET WORD      (80 chars maximum)
```

and y is either

```
VAR n               ; any variable 0-119
VAR n + (i)         ; variable n plus integer i          (127+1=0)
VAR n - (i)         ; variable n plus integer i          (0-1=127)

or for string expressions

"aaaa..."           ; a string of characters in quotes
```


5. Commands for menus and prompts

```
MENU            ; enable COMM prompts and menus
NOMENU          ; suppresses COMM prompts and menus
ONSCREEN $      ; displays string $ on screen in upper-case
```

6. Output commands (to comms device or programme)

```
DO $            ; places string $ in the keyboard buffer
BACK            ; wait until keyboard buffer is empty
HOLDPORT        ; toggle comms port on/off
```

7. Input commands (from comms device or keyboard or memory)

```
GET WORD        ; get word from comms device (or timeout)
GET CHAR        ; get char from comms device (or timeout)
GET $           ; get string from comms device (or timeout)
KEY VAR n       ; keyboard (u/c) input to VAR n
QKEY            ; set FLAG = no. chars in keyboard buffer
XTRACT VAR n    ; memory input to VAR n, increments pointer
ZERO            ; points XTRACT at beginning of memory
```

Note that characters returned via KEY n from the keyboard to a variable are all translated to upper-case and can only be referred to by their ASCII decimal codes (since all user variables are decimal). However, characters returned via GET CHAR to the CHAR variable can be referred to by either a "a" type string expression, or as ASCII decimal values despite the fact that CHAR is a string variable.

3. PART LABELS

JCF files are divided into PARTS rather than having numbered lines. PART labels can be up to eight alphanumeric characters in length: PART, GOTO, GO, TO, and THEN are illegal part labels.

The words PART, GOTO, GO, TO, and THEN before the part label in IF and UNLESS statements are optionally included by users, and are ignored by COMM. PART 0 is always reserved for the beginning of the JCF. Parts need not be defined in any order but must not be duplicated or undefined (except for PART 0).

4. RULES FOR COMMAND STRINGS AND WORDS

A string $ is up to 80 characters long, and is of the form

```
"aaaa" , i , VAR n , .....
```

where

"aa.."	chars between quotes taken literally
i	integer decimal codes 0-127
VAR n	takes ASCII value of variable

The three types can be intermixed in any order. All "aaaa" type input is converted to upper-case in all DO and ONSCREEN strings.

ONSCREEN STRINGS ONLY: Strings in the ONSCREEN command can use the following decimal codes for screen formatting:

0 clear screen and home cursor
1 home cursor
2 cursor position row (x 0-23) column (y 0-79)
3 normal video
4 reverse or dim video (if available)

The $ character is inadmissable in ONSCREEN strings. Note that $ is decimal code 24, and positioning the cursor on column 24 via a command such as:

```
ONSCREEN 2,1,24
```

would put a $ in the string. You can get round this problem by putting the decimal code 24 in a variable. So instead of the above sequence, use this one instead:

```
SET VAR24=24
ONSCREEN 2,1,VAR24
```

Cursor positioning values are NOT checked for valid lines and columns. Users must do this for themselves.

WORDS: When a GET WORD command is executed, the first alphabetic character GOT is taken as the first letter of the word, and the first non-alphabetic character (including punctuation marks or digits) is taken as being the end of the word. The alphabetic word is returned in WORD, and the character which ended the word is returned in CHAR. All GETS are done via the CHAR variable, which is not preserved between GETS. If 80 characters are received with no non-alphabetic character, the word will terminate at the 80th character.

You should be aware that CHAR is affected slightly differently by a GET WORD as compared to a a GET $ command. For instance, if the string

More-?

comes in the comms device, a GET WORD command will end with "more" as the WORD and "-" as the CHAR, and the next GET will have "?" returned to it. However, GET "MORE" will end with "E" as the CHAR, and the "-" is not returned until the next GET command begins.

Note that ALL word, character or string input is irrespective of case. GET "MORE" will get either "MORE" or "more".

5. USER VARIABLES

There are 120 variables, from VAR0 to VAR119. They are set to zero on first entry into the programme, and their value is saved with a programme save from the off-line menu. Thus, if you are automatically numbering saved messages, saving the variables used to store the numbers enables you to locate the next number without needing to do sequential directory searches.

6. SYSTEM VARIABLES

a. DELAY

This is set by users within programmes. A GET command counts down the value set in DELAY until it is zero, at which point the GET terminates. However, if DELAY was zero to begin with, then the GET continues indefinitely. A delay of one lasts approximately one second with a system having a 4MHz CPU - however, this may vary considerably with the speed of the operating system being used and its implementation, since keyboard input is buffered during GET operations, and this is accessed via the operating system. After a GET with delay >0, use

IF DELAY=0

or

UNLESS DELAY=0

to check if the GET timed out or was successful. Delay values from 0 to 127 can be set directly: longer delays can be achieved by nesting with a variable counter.

b. BUFFUL

This is set to 1 if memory is full.
It is reset to 0 on clearing memory.

c. FLAG

This is set after each disk operation or file naming.
It takes any of the following values:

0 file found on disk
1 file not found on disk
2 disk is full
3 disk directory is full

It is up to the user to monitor the status of FLAG after each disk operation, and take action accordingly. Note that when JCFs are executing, the name file operation will do a directory search for the named file and set FLAG accordingly - under manual operation, this does not occur. Successful SAVE, APPEND, RENAME and KILL file operations will leave FLAG set at zero, as will SEND file operations. DIRECTORY displays will set FLAG to 1 since all directory displays continue till no more files are found on disk.

The following are the five non-disk uses of FLAG :

FLAG is set after a is defined to the number of characters entered into the key (or 127 if the process was aborted with either control-C or more than 20 character input).

FLAG is set by WHATSIGN to indicate the sign of the last IF/UNLESS comparison : positive sets 1, negative sets 0.

FLAG is set by QKEY to indicate the number of characters in the keyboard buffer.

FLAG is set by VERSION to indicate the version number of the copy of COMM a JCF is running on : or if a TSR COMM+ is running, VERSION returns 0 if in background.

BINARY	HEX	DECIMAL	ASCII	CONTROL KEY
0000 0000	00	00	NUL	^@
0000 0001	01	01	SOH	^A
0000 0010	02	02	STX	^B
0000 0011	03	03	ETX	^C
0000 0100	04	04	EOT	^D
0000 0101	05	05	ENQ	^E who ?
0000 0110	06	06	ACK	^F
0000 0111	07	07	BEL	^G sound bell
0000 1000	08	08	BS	^H backspace
0000 1001	09	09	HT	^I tab
0000 1010	0A	10	LF	^J line feed
0000 1011	0B	11	VT	^K
0000 1100	0C	12	FF	^L form feed
0000 1101	0D	13	CR	^M return
0000 1110	0E	14	SO	^N
0000 1111	0F	15	SI	^O
0001 0000	10	16	DLE	^P
0001 0001	11	17	DC1	^Q xon
0001 0010	12	18	DC2	^R
0001 0011	13	19	DC3	^S xoff
0001 0100	14	20	DC4	^T
0001 0101	15	21	NAK	^U
0001 0110	16	22	SYN	^V
0001 0111	17	23	ETB	^W
0001 1000	18	24	CAN	^X
0001 1001	19	25	EM	^Y
0001 1010	1A	26	SUB	^Z end file
0001 1011	1B	27	ESC	^[escape
0001 1100	1C	28	FS	^\
0001 1101	1D	29	GS	^]
0001 1110	1E	30	RS	^^ ^caret
0001 1111	1F	31	US	^_ ^underline

FLAG is set to 0 when any toggle in COMM is switched off : and conversely, it is set to 1 when any toggle in COMM is switched on.

d. HOURS, MINUTES and SECONDS.

These variables should be used under non-MSDOS/PCDOS systems with caution. When tested with IF or UNLESS they return the time taken from the system clock.

HOURS can be SET to a value from 0 to 23.
MINUTES and SECONDS can be SET to a time from 0 to 59.

If a non-valid number is SET for any of these variables, the SET command will NOT reset the system clock, but no error will result.

On CP/M 2.2 systems these three variables can be set and tested, but the values they return will not change (so seeing if SECONDS change during a delay loop could provide a simple test for operating system type). In other words, they will behave just like user variables, and DELAY loops will have to be used for timings instead.

7. TIMINGS

The JCF supervisory programme buffers the ports whilst running. Character loss should not occur at normal speeds on most systems. This does NOT apply during disk accesses - any DO commands which may access the disk, or LOAD and EXECUTE commands, may result in character loss. This can be prevented by sending (for instance) an XOFF to a host computer followed by a GET CHAR until a timeout occurs. After the disk command is executed, an XON issued either by a JCF or manually will resume host communications.

BINARY	HEX	DECIMAL	ASCII
0010 0000	20	32	SPACE CHAR
0010 0001	21	33	!
0010 0010	22	34	"
0010 0011	23	35	# UK keyboard pound
0010 0100	24	36	$
0010 0101	25	37	%
0010 0110	26	38	&
0010 0111	27	39	' apostrophe
0010 1000	28	40	(
0010 1001	29	41	)
0010 1010	2A	42	*
0010 1011	2B	43	+
0010 1100	2C	44	, comma
0010 1101	2D	45	- hyphen
0010 1110	2E	46	.
0010 1111	2F	47	/
0011 0000	30	48	0
0011 0001	31	49	1
0011 0010	32	50	2
0011 0011	33	51	3
0011 0100	34	52	4
0011 0101	35	53	5
0011 0110	36	54	6
0011 0111	37	55	7
0011 1000	38	56	8
0011 1001	39	57	9
0011 1010	3A	58	:
0011 1011	3B	59	;
0011 1100	3C	60	<
0011 1101	3D	61	=
0011 1110	3E	62	>
0011 1111	3F	63	?

BINARY	HEX	DECIMAL	ASCII
0100 0000	40	64	@
0100 0001	41	65	A
0100 0010	42	66	B
0100 0011	43	67	C
0100 0100	44	68	D
0100 0101	45	69	E
0100 0110	46	70	F
0100 0111	47	71	G
0100 1000	48	72	H
0100 1001	49	73	I
0100 1010	4A	74	J
0100 1011	4B	75	K
0100 1100	4C	76	L
0100 1101	4D	77	M
0100 1110	4E	78	N
0100 1111	4F	79	O
0101 0000	50	80	P
0101 0001	51	81	Q
0101 0010	52	82	R
0101 0011	53	83	S
0101 0100	54	84	T
0101 0101	55	85	U
0101 0110	56	86	V
0101 0111	57	87	W
0101 1000	58	88	X
0101 1001	59	89	Y
0101 1010	5A	90	Z
0101 1011	5B	91	[
0101 1100	5C	92	\
0101 1101	5D	93	]
0101 1110	5E	94	^
0101 1111	5F	95	_ viewdata #

COMM+ Manual: Appendix

BINARY	HEX	DECIMAL	ASCII
0110 0000	60	96	‘ open single quotes
0110 0001	61	97	a
0110 0010	62	98	b
0110 0011	63	99	c
0110 0100	64	100	d
0110 0101	65	101	e
0110 0110	66	102	f
0110 0111	67	103	g
0110 1000	68	104	h
0110 1001	69	105	i
0110 1010	6A	106	j
0110 1011	6B	107	k
0110 1100	6C	108	l
0110 1101	6D	109	m
0110 1110	6E	110	n
0110 1111	6F	111	o
0111 0000	70	112	p
0111 0001	71	113	q
0111 0010	72	114	r
0111 0011	73	115	s
0111 0100	74	116	t
0111 0101	75	117	u
0111 0110	76	118	v
0111 0111	77	119	w
0111 1000	78	120	x
0111 1001	79	121	y
0111 1010	7A	122	z
0111 1011	7B	123	{
0111 1100	7C	124	|
0111 1101	7D	125	}
0111 1110	7E	126	~ tilde
0111 1111	7F	127	DEL delete

1. INTRODUCTION.

This appendix is essential rather than optional reading. It contains information on things which you will need to get right in order to communicate properly. Items in the appendix are here not because they are unimportant, but for two possible reasons.

a) They are not part of COMM itelf. They are concerned with your other hardware or operating system software. Putting such information in the other parts of the manual would confuse users by mixing up different operating systems and hardware configurations.

b) They apply only to highly specific versions of COMM. In such cases, putting such information in the other parts of the manual would be irrelevant to users of other systems.

It is actually possible, by combining the number of different machines and operating systems COMM runs on with the number of possible modems and the different data network services users may want to connect up to write a few thousand different manuals. Clearly this is not feasible, so users of COMM simply have to accept that information they need is found in lots of different places. This appendix and the rest of the manual are only two of those - your modem manual, electronic mail or host computer manual, your telex directory and your other software manuals are also sources of information you may need at some time. Communications by its nature is unlike spreadsheets or word processors in that a stand-alone communications system is a contradiction in terms.

Read the parts of the appendix that concern your system and understand them fully since if you have any problems, the answers may well be here.

TABLE OF CONTENTS

2. OPERATING SYSTEM LOADING

We are not allowed to sell operating systems with copies of COMM. Consequently, your master disk as supplied will NOT load up if you simply switch your machine on and put the disk in the drive. You have to boot up the machine with a systems disk, and then run COMM.

We recommend that your COMM master disk is backed up as soon as you receive it, and that you don't attempt to place an operating system on the master - some computers (Apricots, for instance) destroy the directory when you do this, and you wouldn't then be able to use the programme. However, if you really must place an operating system on a disk you can do so as follows -but **at your own risk**:

i) CP/M-80 Systems: Place the master disk in drive B, a systems disk in drive A, and enter the **SYSGEN** command. Follow instructions.

ii) MSDOS/PCDOS Systems: Place the master disk in drive B, a systems disk in drive B and enter the **SYS B:** command. If SYS cannot be used on your system, then you had better forget about this.

iii) CP/M-86 Systems: Place the master disk in drive B, a systems disk in drive A, and enter the **PIP B:=CPM.SYS[OVR]** command.

We emphasize again that you should try NOT to write anything on to your master disk. You are entitled to copy your disk for legitimate backup purposes, and we urge that you this first before doing anything else.

The recommended procedure for backing up your disk is to make a bootable empty disk up using your system's **FORMAT** command. This might be called something else or even involve you loading up a separate disk utility programme. Make sure that if this is the case, you invoke any options to transfer the operating system too.

IBM users should note that running FORMAT/S on their hard disks (the /S transfers the operating system) might, without prompting, reformat the hard disk. Use FORMAT A:/S instead, with the blank floppy disk in drive A.

CP/M-80 users would normally use **SYSGEN** on the blank disk after formatting.

CP/M-86 users would copy **CPM.SYS** (or **CCPM.SYS** for Concurrent) on to the floppy disk. All CP/M users should also place **PIP** on their new disk.

After generating a blank disk, you should then use either the CP/M command **PIP** or the MSDOS command **COPY** to copy all the files from the master disk to the new one. The put away the master for update and emergency use only. Note that should you at some time in the future need an upgrade, the master disk will have to be returned.

3. CABLES FOR COMMUNICATING ON ALL SYSTEMS

It is essential that your modem is connected to your computer with the appropriate cable. These notes are for advice only - if you are unsure about what you need to do, seek professional help from a reputable dealer.

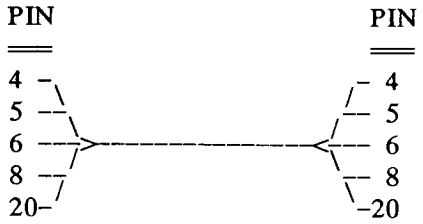
The serial communications link between your modem or coupler and your computer needs only three wires to work properly. These are the receive data and transmit data wires, and the signal ground wire.

On the standard D-shaped 25-way connector used for RS232 serial communications, the receieve data is on pin 3, the transmit data is on pin 2, and the signal ground is on pin 7. This is known as data terminal wiring (DTE).

All modems are constructed with receive on pin 2 and transmit on pin 3. This is known as data communications wiring (DCE).

When the two are linked, the transmit pin on one goes into the receive pin on the other, and communication can take place.

However, on some computers and using some equipment, modifications may be needed to the cable used to take account of possible crossovers, and possible handshaking lines.



Whilst it is unlikely that all these will be needed, it should do no harm at all if you connect them all together. Some computers may, however, hang if a line is held high when the computer expects it to be low – ideally, you should either read your manuals carefully, or consult your hardware supplier, or specify the equipment you have when you order a modem or coupler and insist the supplier provides the correct lead. If your supplier refuses to do this, find one who is more co-operative. Communications is difficult enough for a first-time user without having to contend with the ignorance of some dealers.

4. MODEM NOTES

COMM will work with virtually any modem currently available : however, the following notes are provided in case of difficulty.

i) Some modems (such as the Racal CP2123 modem, the Prism 2000, or the Pace Nightingale or Linnet) are only suitable for use on computers which hold lines on the serial port high. A special crossover cable might be needed for this.

ii) There are a number of various modem standards. Most European counties use the CCITT standard, but the USA uses Bell standards. Though these are near-compatible at 1200 baud, at 300 baud they are NOT. Modems manufactured outside the UK are not necessarily suitable for use in the UK. UK.

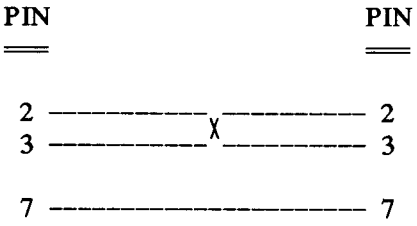
iii) BABT approval is a legal requirement for using modems in the UK on public telephone systems. It is an offence to use equipment without the such approval in the UK. For other countries there are similiar local regulations – consult your telephone company for details.

iv) COMM cannot be used with synchronous modems without special arrangements being made : if you need to connect your microcomputer to a host that only permits synchronous communications, we suggest you contact your local dealer for help.

a) Crossover cables

If two computers with identical wiring have their receive pins connected and their transmit pins connected, they will not be able to communicate. The recieve pin on one has to be connected to the transmit pin on the other.

Some computers have their serial output wired for data communications (DCE), usually so that they can be linked up to printers, which are wired as data terminals (DTE). If this is the case on your computer, you will need a crossover cable, which should be wired as follows



If your computer can drive a serial printer, you will usually need to use a crossover cable.

b) Handshaking lines

Pin 4 (RTS or request to send), pin 5 (CTS or clear to send), pin 6 (DSR or data set ready), pin 8 (DCD or data carrier detect) and pin 20 (DTR or data terminal ready) are known as the handshaking lines and can output a high signal. On some computers and some modems, one or more of these need to be held high for communication to take place. If your computer is one of these, you may need to link pins 4,5,6,8 and 20 together at each end of the cable and connect them all up together as follows –

v) Half-duplex modems : Many modems are capable of 1200 baud half-duplex operation. COMM does not (in its standard form) support this protocol - but then neither do any public access computers or databases. In our modem table, all speeds indicated refer to FULL duplex operation. For the technically-minded, use of a half-duplex mode requires receive and transmit modes to be turned around with hardware. A special driver file is needed for this - contact us should you have a need for this type of communication.

5. MACHINE-SPECIFIC NOTES

The information that follows is provided for guidance only and since hardware manufacturers reserve the right to change specifications without notice, we cannot guarantee it to be correct at all times.

The notes that follow are also liable to change without notice: however, if you find a file called READ.ME on your disk, this may contain supplementary information regarding a new implementation on a particular machine. Changes to this published information will usually be for the better - for instance, a machine which we are currently unable to provide certain features on due to lack of technical information may, at some time in the future, have those features added. We believe that it is better to provide them at the risk of inaccuracies in our manual than to hold back on improvements.

i) DTE and DCE

We use DTE to refer to terminal wired computers, which need no crossover, and DCE to refer to communication wired computers, which do need crossovers.

ii) PATCH

We use PATCH to indicate the need to patch a cable for handhaking lines in those instances where COMM cannot avoid it. As described above, we aim to need only pins 2,3 and 7. Of course, your modem itself may need a patched cable. We cannot cover such all eventualities here - though the modems listed in the previous section indicate what requirements they have.

There are two major reasons why a computer may need a cable with the handshaking lines patched.

The first is that certain serial port chips (notably the 8250 UART and derivatives such as the Signetics 2561) have transmit lines that are CTS conditioned - that is, they will never transmit while the CTS line is low. Machines with such serial port chips (such as Sanyo 555 and Intertec Superbrain) always need cables which feed a high signal into the CTS pin.

The second reason is that where a COMM driver works through the operating system, it is often the case that the software that the operating system uses to control the serial port implements similiar features. For instance, the BIOS on the IBM PC implements both CTS-conditioned transmit pins and DSR-conditioned receive pins, and won't transmit unless CTS is high or receive unless DSR is high. This is not unusual, and could in fact be considered to be a kind of standard - the ICL DRS range works in the same way. Regrettably, there is no guarantee that any particular modem will hold the necessary lines high, or that any particular cable will even have the necessary pins wired. We can only repeat our earlier advice to insist that your dealer supplies you with a suitable cable for your system.

iii) INTERRUPTS

This is noted for those 16-bit versions of COMM which are interrupt-driven (i.e we do not poll for characters coming in, but wait to be interrupted by the serial port whenever a character arrives). For these machines, COMM should be exited only via the normal routes (ESC Q from on-line menu or Q from the off-line menu), or via a system reset. Any other forced exit (such as opting to Abort from a DOS error message if trying to use a non-existent disk or printer) should be followed by a system reset. This is because COMM contains its own interrupt drivers which replace the normal operating systems ones, and only resets the system interrupt vectors on normal exit, but not on abnormal ones.

iv) AUTO FLOW CONTROL

Some machines which are not interrupt-driven may lose characters when run at higher speeds into some systems - upgrades for the DRIVERS used may be made available at some time in the future. Many of the versions which are not interrupt-driven do not need to be - COMM will work in polled mode on most microcomputers.

However, some CP/M-80 implementations have special arrangements for software flow control. Please note that all these systems requires proper XON/XOFF handshaking at the host end to avoid losing characters at speeds above 300 baud. This is automatically invoked with ESC I whilst on line, when an XOFF is sent to the host before each new line is displayed on the screen or screen clear character is received, and an XOFF is sent afterwards. Use the handshake menu to disable the stall/resume characters if you do not require this feature, or else run with handshaking off - most intelligent modems will need to be set up with handshaking off as they do not like XON or XOFF characters.

This feature is available on all CP/M-80 versions of COMM by patching the byte at 09A3H. When this byte is set to 00H, characters are sent to the screen as received by the communications port. However, when it is set to 0FFH, each line feed or clear screen sequence received from the communications port triggers an XOFF (or stall) to the host, followed by a short delay (configurable via the DELAY length in the handshaking menu). After this delay, the line feed or clear screen character is sent to the screen, and then an XOFF (or resume) is sent to the host to resume transmission.

On the machine notes, an AUTO-FLOW CONTROL indicates that the driver has this patch already installed. The systems on which this has been done are those which cannot communicate at 1200 baud without character loss on a screen scroll. As far as we are aware, no CP/M-80 systems are unable to work at 300 baud without flow control: but a number cannot work at 1200, and many more need flow control at speeds above 2400. Since most communications take place at 1200 and below, it was not though worth patching the drivers concerned for implementations where 1200 baud worked well without.

Prestel users should be aware that character loss after a clear screen in somewhat more common than after a screen scroll, and since most Prestel frames begins with a clear screen, character loss after a clear screen results in the top line - possibly the top two lines - becoming misplaced. Unfortunately, Prestel doesn't support flow control, so nothing can be done about this problem. Luckily, the top line of each frame contains page numbers and billing information, and the second line is often either blank or used with an IP message, so usually the actual page content will remain intact.

v) FULL SPEED CONTROL, NO SPEED CONTROL OR DUAL SPEED ONLY

Where COMM allows for full control of serial port speeds within the programme from the baud rate menu, this is noted. Generally, speeds begin at 300 baud and double up to 9600 baud, giving six possible settings. You should be aware that 9600 baud is often inaccurate, and two systems may be unable to communicate reliably at 9600 baud with each other, though both may work happily with a third system at 9600. Slight inaccuracies in calculations of clock speed are responsible for this - one system may be 3% below 9600, one may be 3% above, and the third may be a true 9600. The first two have an unacceptable 6% difference in speed, and so cannot communicate, whilst the last works with any system.

On some systems no speed setting at all is possible, and where relevant this is noted too.

On certain systems, either access to the clock controlling the speed on the serial port is not available due to lack of technical data, or else there isn't a separate clock apart from the one for the CPU. One such systems,

vii) **BREAK SUPPORT**

A break is a continuous signal transmitted over a dataline, usually for at least a quarter of a second. It is often used to terminate a currently executing command. Where we note that break is supported, ESC / whilst on-line will generate a break signal - the option has also been included on the baud rate menu on some systems. On some systems no break is possible - this too is noted. The reason may be either lack of technical data, or systems software prohibiting the direct port access needed.

viii) **ANSI.SYS**

On many COMM systems, one of the emulations provided is redundant since the screen will emulate one of these terminal types in native mode. Where this is the case (it will be apparent by comparing screen codes in the systems guide for your machine with the table in the supplementary reference guide) faster displays can be obtained for the native mode by leaving emulations OFF rather than invoking the one corresponding to the native mode. Users of PCDOS or MSDOS version 2 or later should be aware that ANSI emulation is available as native mode only if the file ANSI.SYS is included in the CONFIG.SYS file as described in your systems guides - ANSI is a subset of the DEC VT100 codes. Consult your systems manuals or dealer for advice on this if unsure about what to do.

ix) **FUNCTION KEYS** and **ARROW KEYS**

Unless specifically stated in the notes below, any function keys on your machine are not supported. If they are configurable keys, you may be able to configure them yourself. For PCDOS systems, see the Supplementary Reference Manual for details on special PC features supported (such as keyboard, screen, colours etc).

Arrow keys are also not supported unless specifically stated - however, the editor in COMM supports most usual assignments for arrow keys, so many users will find that their function keys behave perfectly well.

x) **PULSE DIALLING**

A number of systems are listed as having standard and pulse-dialling drivers available. Most modems would use the standard driver : the pulse

it is often possible to obtain some degree of speed switching from within COMM by altering the divisor which the serial port controller uses to determine the way it should count down the speed. Usually this divisor is 16. Since many chips have the option to use a divisor of 64, we can run at one quarter of the default speed as well.

Where a DUAL SPEED ONLY note occurs, the programme is capable of altering the speed used in this way. If you have the facility to set the default speed from your operating system, we recommend that you set the serial port to run at 1200 baud. COMM, on first loading, will also run at 1200 baud. However, the option to use a different divisor means that we can also run at a quarter of 1200 baud - this, of course, is 300 baud. This can be saved within COMM. We recommend setting a system default of 1200 since this gives a useful 300 baud option as well. But if you use a 300 baud default, your secondary divisor will give you a 75 baud option, which is of little use.

vi) **SPLIT SPEEDS**

Some systems have separate clocks for receive and transmit data and thus give an option for true split speeds of 1200/75 for V23 access as used by Viewdata systems such as Prestel. This is noted as hardware split speed.

Other systems have only one clock for both receive and transmit data, and thus cannot offer a true split speed of 1200/75. Where such systems can run at 75 baud, it is possible to give a software split speed option by altering the speed to 75 baud whenever a character is sent out. This effectively runs V23 in a sort of half duplex, since characters coming in at 1200 baud whilst the serial port is set to 75 baud for sending will be lost. For Viewdata systems, which use single-key responses and have relatively slow response times, this makes little difference. However, such systems are difficult to use for teletype applications at 1200/75 split speed - this is therefore not recommended.

Clearly, if a note is given for either no speed control or dual speed control, split speed of any type won't be possible. We recommend a speed conversion modem be used for V23 access on such systems, to be run at 1200 baud. The Supplementary Reference guide mentions this.

dialling driver is a special version that autodials on otherwise dumb modems (such as Voyager, WS2000, or Demon modems) by pulsing the DTR line in line with the required telephone number. Such drivers need special cables to work properly, as a number of modem control lines are used as well as the normal data lines, and are often unsatisfactory for use with other types of modem.

For most modems, including all intelligent and Hayes-compatible modems, you need only the standard driver.

The actual machine notes themselves now follow. For convenience, we have grouped the CP/M-80 systems first, the MSDOS/PCDOS systems second, and the CP/M-86 systems last. Each group lists the systems alphabetically. Compatible systems are not entered separately - each entry corresponds to one driver and operating system. Where compatabilities are known and tested, we have listed them. All machine types appear in the index.

CP/M-80 SYSTEMS

Generic CP/M+ version

As well as the machine specific drivers listed below, a generic CP/M+ version is also available : it relies on correct implementation of the operating system auxiliary port calls for data and status. It is available for machines which run CP/M+ but for which no hardware specific driver has been written : examples include the Cifer and Spectrum. However, a hardware driver will usually give much better performamce than one which has to make operating system calls all the time, so where a suitable hardware driver is available, a generic driver will not be provided unless specifically ordered.

Alphatronic P2/P3	Dual speed only, no break support, auto-flow control. Very slow screen clear affects Viewdata access. Not the same as Alphatronic PC. DTE.
Alphatronic PC	Dual speed only with break support, auto-flow control. Very slow screen clear affects Viewdata access. Not the same as Alphatronic P2/P3. DTE.

Amstrad 6128	Full speed control include hardware split speeds with independently selectable receive and transmit rates. No break. Only works with CP/M+ (not 2.2). Apparently satisfactory screen scrolling, but slow at clearing screen, which affects Viewdata access. DTE.
Amstrad 8256	Full speed control include hardware split speeds with independently selectable receive and transmit rates. Note that the ESC key on the PCW 8256 keyboard is labelled EXIT and placed at the bottom right of the alpha keys rather the usual top left. The key at the top left where an ESC key is usually placed is labelled STOP and generates ^C. The Amstrad function keys are not supported in COMM. DTE. A number of different drivers are available, including pulse diallers, BIOS-only, and interrupt-driven ones. GRAPHICS file available for Viewdata access. Same version for Amstrad 8512 and Amstrad 9512, but 6128 models are different.
Apple II	COMM works on all Apple II machines except the IIc (basic II, II+, Europlus and IIE). However, you MUST have a standard Z-80 card and a CP/M disk installed in your Apple, an 80-column card, and a serial communications card (NOT the Apple high-speed card) installed in slot 2. If you don't have the above you cannot use COMM on the Apple II. The Apple III and Macintosh will not run COMM due to CPU incompatabilities. The Microsoft Gold CP/M card is not suitable for running COMM, though the standard Microsoft Softcard, and various other Z-80 card are fine. On standard Microsoft CP/M use the **COPY/S B:** command to copy operating systems rather than SYSGEN. As explained above, you MUST have a communications card installed in slot TWO of your Apple computer. There are three types of serial card which COMM supports, and driver routines for these are provided on your disk. If you have a Computech Diplomat Communications card, rename the file 8250 to DRIVER by typing REN

DRIVER=8250 (press return). Make sure that the red connector is set up for modem use as described in the manual for your card.

If you have an Apple Super-serial card, rename the file 6551 to DRIVER by typing REN DRIVER=6551 (press return). Make sure that the jumpers on the card are set up for modem use as described in the manual for your card.

If you have an Apple Communications card, a CCS, Simon, U-Micro or Digitek serial card, rename the file 6850 to DRIVER by typing REN DRIVER=6850 (press return). Apart from the Apple communications card, all cards with a 6850 chip are set up for printer use. You will need a special cable which switches over pins two and three in the connectors, and which also straps together the handshaking lines (usually pins 4 and 5, and pins 6, 8 and 20). If you are not sure which type of card you have, carefully remove it from your machine and look at the large chip on the card itself. It will have the chip type written on it.

Note that Apple can be used to emulate a variety of terminals by running the BASIC CONFIGUR program. However, COMM expect the Soroq IQ default to be used. Reverse video is not implemented as most 80-column boards do not properly support it.

PRESTEL USERS NOTE: User key : (unshifted star) is set up to produce an ASCII underline, which should be used for paging instead of the hash Prestel Terminals use. Apple II keyboards lack an underline.

Apple 6551	Full speed control, break support, software split speeds, auto-flow control.
Apple 6850	Dual speed only. Auto-flow control. Note for Apple Communications Card: The 110 baud rate on this card is in fact an alternate clock which is selectable by setting

RTS high on the 6850 (which we can do in software with this special dual speed driver). This clock runs at 440 baud with a divisor of 16: hence you can run at 110 by selecting the 64 divisor. Users may consider asking their dealer to modify the Apple card as described in the Communication Card manual so that it defaults to 1200 baud - this will allow 300 access via the 64 divisor.

Apple 8250	Full speed control, break support, software split speeds, auto-flow control.
BBC B	This version runs with the Acorn Z-80 tube add-on. It will NOT work with Torch CP/N diskpacks. A special cable for the 5-pin RS422 socket is needed - specify it for a modem. The editor and the emulations are designed to run on the 80x25 line screen selected via MODE 3. To select this, type the following - **BBCBASIC** (to load Basic) **MODE 3** (Screen clear and mode 3 selected - must use capitals) ***CPM** (return to CP/M) **COMM** (load COMM) Full speed control. Hardware split speeds - transmit and receieve speeds are independently selectable. 1 stop bit is used. Break is NOT implemented.
Caltex	No speed control, no break. Same as Ceedata. DCE.
Eagle	Break support, no speed control. Uses 7 bit word. DCE.
Epic	Speed control to 2400 only, no split speeds, break support. DTE.
Epson QX-10	Full speed control, software split speeds, no break support. DTE. Standard or pulse dialling drivers available.
Epson PX-8	Full speed control, software split speeds, no break support. DTE. Use control with arrow keys to simulate 80x24 screen on PX8 8x8 screen. Adviseable to upgrade

Mimi 802	Full speed control, auto-flow control, break support with software split speeds. Auto-flow control. Not the same as Mimi 803. DTE.
Mimi 803	Full speed control, auto-flow control, break support with software split speeds. Auto-flow control. Not the same as Mimi 802. DTE.
Morrow Microdecision	Dual speed control. Software may be supplied on Osborne disk format (can be read). DTE.
Multicomputer M2000/Trio	Full speed control, support with software split speeds. Specify disk format and VDU type when ordering. DTE.
NCR DM5	Full speed control, break support with software split speeds. Auto-flow control. The version of the program as provided asks whether a printer or communications board is being used when it is loaded. This feature is saved via P on the off-line menu. Keep a copy of the original version if you think you may wish to change from printer board (211) to comms board (212). DTE on Comms board, DCE on Printer board. A serial board which is link-selectable as a 211 or 212 is also available. Configure as appropriate. Standard and pulse dialling drivers available.
NEC 8001	Dual speed, break support. Initialize serial port for non-interrupt driven polled access via operating system utilities before using COMM. Not the same as the NEC 8801. Auto-flow control.
NEC 8801	Dual speed, break support. Initialize serial port for non-interrupt driven polled access via operating system utilities before using COMM. Not the same as the NEC 8001. Auto-flow control.
Olympia ETX II	Full speed control, software split speeds and break support. Peculiar character set displays fractions instead of ^ sign. Warning: check disk format before ordering, as both single and quad density disks are in use. DTE.

	available RAM due to very slow microcassete load and lack of RAMDISK space when running COMM.
Equinox	Turbodos system (CP/M-80 compatible). Full speed control, no software split speeds, break support. DTE. Graphics file for viewdata available : assumes Ampex VDU in use.
Gemini Galaxy	Same as Quantum and Kenilworth. Full speed control, auto-flow control, break support with software split speeds. Auto-flow control needed with GM812 IVC, probably not with GM832 SVC. Uses on-board serial port on GM813 card. DTE usually - depends on implementation. Single sided 96 tpi disk supplied as standard. Graphics file for Viewdata access available.
Gnat	Same as Millbank 10. Full speed control, no split speeds, break supported. DTE.
Hytec 4500	Dual speed only, break support. DTE.
ICL PC1	No speed control, no break. DCE. Supplied as standard with ICL VDU driver for CP/M 2.3. or CP/M 2.2 - specify which operating system and disk format when ordering. Small charge for other VDU drivers. Also works under MP/M II. Under CP/M 2.3 and MP/M II, the speed must be must be configured using TTYSET from the operating system. The TTY port to be used must be assigned to NUL: using ASSIGN, and must be set within the programme via B from the off-line menu. Under CP/M 2.2, set the speed for the correct port with SET, and make sure that the correct port is being used also via B on the off-line menu.
Kaypro 2/4/10	Full speed control, software split speeds, break support. On first running COMM, you are asked whether your Kaypro supports reverse video. This is needed for menu display and emulations. Once programme is saved, question will no longer be asked. DTE. Standard and pulse dialling drivers available.

Olytext	Full speed control, software split speeds and break support. Supplied on 3.5" disk. Also known as ETX III.
Osborne 1	Dual speed. Break support only via TTL modem port, not via RS232. Sendata and K&N make modem port compatible couplers. DCE.
Osborne Executive	Full speed control, software split speeds and break support. Uses modem port. DTE.
Osborne Express	Full speed control, software split speeds and break support. BIOS dependent interrupt-driven driver. DTE.
Philips P2000C	Full speed control, software split speeds and break support. Needs cable patched to connect pins 15, 17 and 24. DTE. Warning: check disk format before ordering, as both single and quad density formats are available.
Pied Piper	Full speed control, software split speeds, break support, auto-flow control. Uses port B. DTE. The program itself needs the HAZELTINE VDU mode to run properly, in which case Hazeltine is native.
Portico Miracle	No speed control, but break support. Uses 7 bits only out of modem port. DTE.
Rair Black Box	No speed control, no break. DCE. Supplied as standard with ADDS VDU driver for CP/M 2.3. or CP/M 2.2 - specify which operating system and disk format when ordering. Small charge for other VDU drivers. Also works under MP/M II. Under CP/M 2.3 and MP/M II, the speed must be must be configured using TTYSET from the operating system. The TTY port to be used must be assigned to NUL: using ASSIGN, and must be set within the programme via B from the off-line menu. Under CP/M 2.2, set the speed for the correct port with SET, and make sure that the correct port is being used via B on the off-line menu.

Sanyo 1150	No speed control, but break support. Uses 7 bits only.
Sharp PC3201	Dual speed, break support. Uses CE-340R card configured as channels 0 and 1, inserted in slot 1 or slot 3. Programme asked once only which channel you are using (programme save to make permanent). DCE.
Superbrain	Dual speed, break support. Specify disk format if standard single-sided double-density Superbrain format not readable. DCE. On first running COMM, you are asked to select either the main port or the auxiliary port - you press either M or A. After a programme save, you are not asked this again. Both standard and pulse dialling drivers available.
Tatung Einstein	Needs 80-column card and MOS/DOS 1.13 or later. Needs special cable with 5-pin DIN plug : must hold CTS high to transmit. Full speed control, true 1200/75, break support. Auto-flow control.
Televideo 802	Full speed control, no split speeds, break support. Works with Televideo 800A workstation (see MMMost below) but not with 803 or 800. Uses modem port. DTE.
Televideo 803	No speed control, no break. Works with TPC1 also. Cannot be used with MMMost. Uses modem port. DTE.
Televideo MMMost	COMM runs in the workstation, not in the 806 or 816 controller. This means that modems must be attached to the workstation and not to the controller. An 803 or 800 workstation cannot be used with MMMost systems since it uses the modem port as its networking port. The 800A workstation and 802 can be used: both use the same 802 driver.
Transtec	Speed control to 2400 only, Break support, no split speeds. DTE.

Xerox 820	Full speed control, software split speeds, break support. DTE.
Zorba	Dual speed, break support. Uses modem port: beware of high voltages on this port. Use special cable if possible. DTE.

MSDOS/PCDOS SYSTEMS

Generic PCDOS version

As well as the machine specific drivers listed below, a generic PCDOS version is also available : it relies on correct implementation of the INT 14H operating system auxiliary port functions calls for setup, data and status. It is available for machines which either run PCDOS compatible operating systems or implement the INT 14H comms port interface correctly, but for which no hardware specific driver has been written : examples include the Tandem and the Hyperion. However, a hardware driver will usually give much better performamce than one which has to make operating system calls all the time, so where a suitable hardware driver is available, a generic driver will not be provided unless specifically ordered.

Advance 86B	This is a modified version of the IBM PCDOS version of the programme offering the same screen and function key facilities. Polled rather than interrupt driven : loses characters at speeds above 600 baud on scrolling, and also after a screen clear. Standard and pulse dialling drivers available.
Apricot F series	Full speed control, software split speeds, break support, interrupt driven. DTE. Note that all Apricot disks are supplied single sided as standard, but can be read by any Apricot system. Standard and pulse dialling drivers available.

Apricot Modem	Internal modem driver MODEMAPR.SYS needed in CONFIG.SYS file. Works on all Apricots at 300/300 and 1200/75 only. Break not supported. Separate instruction sheet comes with disk. DTE. Note that all Apricot disks are supplied single sided as standard, but can be read by any Apricot system.
Apricot PC/Xi	Full speed control, hardware split speeds, break support, interrupt driven. DTE. Note that all Apricot disks are supplied single sided as standard, but can be read by any Apricot system. Standard and pulse dialling drivers available.
Apricot Portable	Full speed control, hardware split speeds, break support, interrupt driven. DTE. Note that all Apricot disks are supplied single sided as standard, but can be read by any Apricot system. Standard and pulse dialling drivers available.
Canon AS100	Usually supplied on CP/M-86 disk, to be transferred to MSDOS if needed. See CP/M-86 systems. Needs ANSISEQ to run properly.
Canon TX50	Runs modified PCDOS version with special drivers, either for standard serial port or optional dual SCC 4-port card. All speeds, break supported, interrupt driven.
DEC Rainbow	Usually supplied on CP/M-86 disk, to be transferred to MSDOS if needed. See CP/M-86 systems. Standard and pulse dialling drivers available.
Epson QX-16	Full speed control, software split speeds, break support, interrupt driven. DTE. ANSI.SYS. Implements same function and arrow keys, and same status line, as IBM PCDOS version.
Fujitsu 16S	Full speed control but no split speeds. Break support. DTE. The function keys are predefined, and are shown on the 25th line of the display. COMM uses the main serial

NEC APC I	Usually supplied on CP/M-86 disk under single-sided single-density IBM3740 format, to be transferred to MSDOS if needed. See CP/M-86 systems.
Olympia People	Speed control, software split speeds, break support. Runs under CP/M-86 also. Graphics file available. DTE.
Panasonic JB3000	Needs optional serial board - please wire as DTE. Full speed control, software split speeds, break support, interrupt driven.
Sanyo MBC 555	Needs optional serial board. Full speed control, software split speeds, break support, interrupt driven. DTE. Needs patched cable as any disk access drops DTR, which disconnects many modems. Supports IBM-type function key display and editing keys, with appropriate Sanyo modifications for five function keys. See IBM details and your Sanyo manual for details. Supplied on single-sided 8 sectored disk format as standard. ANSI.SYS only with DOS 2.0 or later. Supplied with batch file for patching DOS to avoid DTR problems if necessary.
Sirius/Victor	Full speed control, software split speeds, break support. DTE. Single-sided disk is supplied.
Sharp 5000	Full speed control, software split speeds, break support, auto-flow control. Supplied on IBM compatible disk and NOT on bubble memory. DTE.
Tandy 2000	Full speed control, software split speeds, break support. Supports IBM-style status line, function keys and editing keys - see section on IBM COMM for details. DTE. ANSI.SYS
Wang PC	Full speed control, software split speeds, break support, interrupt driven. DTE. COMM supports a limited number of the WANG function keys. The arrow keys are supported in the editor, as are the INSERT and DELETE keys. Shift INSERT will insert a new line, and shift

	port on the machine rather than any extra serial ports which may be installed. Terminal emulations at speeds of 9600 are not reliable.
Future FX20/30	Full speed control, software split speeds, break support, interrupt driven. DTE. Doesn't work with Future network. Standard and pulse dialling drivers available. Can run a modified PCDOS or standard MSDOS version of COMM.
IBM PC	Full speed control, split speeds (software, but very reliable), break support, interrupt driven. DTE port. ANSI.SYS available. GRAPHICS file available supporting MDA, CGA, EGA, VGA screens (but not Hercules). See main manual Reference guide for PCDOS specific features. The IBM PCDOS version works with most clones and can work with any version of DOS from 1.0 upwards. Tested on many PCs and clones under DOS including all IBM systems (PC, XT, AT, PS/2), Amstrad (1512, 1640, PPC), Olivetti, Compaq, Tandon, Toshiba, Zenith, Ericsson, Tandy, Epson etc etc. Very wide range of drivers available : standard version is interrupt-driven for either COM1 or COM2, supports DTR handshaking. Optional alternative drivers include following features: Pulse dialling, CTS handshaking on output, V23 half-duplex (simulated 1200/1200), 9600 baud EGA Viewdata driver, BIOS only polled driver (this is the generic driver, suitable for non-standard serial ports, networked ports, INT 14H interceptors etc.), Polled direct port driver, SCC card driver, and many others.
Missing Link Modem	Works at 300/300 or 1200/75. Break supported. ANSI.SYS. Comes with instruction sheet accompanying disk. Renamed Breakout Modem, same thing.

Canon AS100	Full speed control, software split speeds, break support, interrupt driven. DTE usually but can use any port - on first use defaults to lowest available but this can be changed. Note that neither RSHND nor RSINIT are required as COMM contains its own interrupt handlers and port initializers. The system supplied ANSISEQ file should be run under MSDOS both for ANSI capability and to ensure correct screen displays in COMM itself. Please load up COMM before rather than after connecting your modem - this has been found to cause less problems when initializing the ports. Note that the CP/M-86 version can be run under MSDOS (if still unsaved) by copying all the files on disk to an MSDOS diskette using the supplied READCPM utility. Consequently, disks usually supplied on CP/M-86 format unless specifically requested.
Comart CP2000	Also for Comart CP3000, but different from Quad. No speed control, no split speeds, no break support. Runs under Concurrent. XIOS driven, DTE.
Comart Quad	No speed control, no split speeds, no break support. Runs under Concurrent. BIOS driven, DTE.
DEC Rainbow	Full speed control, software split speeds, break support. DTE. Switch off smooth scrolling to avoid character loss. COMM supports a limited number of the DEC function keys. The arrow keys are supported in the editor, as are the FIND and INSERT keys. The PREV SCREEN key will scroll up one line, while NEXT SCREEN will scroll down a line. The REMOVE key will delete forwards. The HELP key will produce help screens in both on-line mode and within the editor, and will enable the help level to set from the off-line menu. The MAIN SCREEN key will return you to the off-line menu from either the editor or on-line mode, and will put you in the editor from the off-line menu. The EXIT key will quit back to system level from either on or off-line modes. Note that the F11 key is the ESC key on the DEC. Note that the CP/M-86

	DELETE will delete to end of line. The PREV key will scroll up one line, shift PREV will scroll up 22 lines, while NEXT will scroll down a line and shift NEXT will scroll down 22 lines. The SRCH key will do a find, and the ERASE key will delete forwards. The WANG PC has no ESC key. The combination shift+control+] will generate an ESC code, but for ease of use, the HELP, EXEC and CANCEL keys may be used instead. Any of these will produce an ESC code. Disk supplied on the IBM compatible 160K single-sided format as standard.
Zenith Z100	Full speed control and break support. Disk supplied on the IBM compatible 160K single-sided format as standard.

CP/M-86 AND RELATED SYSTEMS

Generic CP/M-86 version

As well as the machine specific drivers listed below, a generic CP/M-86 version is also available : it relies on correct implementation of the operating system auxiliary port functions calls for setup, data and status. It is available for machines which either run CP/M-86 or compatible operating systems such as DOS+, MP/M-86 or CCP/M-86 and implement the comms calls correctly, but for which no hardware specific driver has been written : examples include the BBC Master 512. However, a hardware driver will usually give much better performamce than one which has to make operating system calls all the time, so where a suitable hardware driver is available, a generic driver will not be provided unless specifically ordered.

NOTE: The driver supplied with the queued CDOS version of COMM documented in the Supplementary Reference Manual is a generic version which supports either ADM, ADDS, VT52 or ANSI type screen addressing and works on most Concurrent DOS systems later than 3.1 : the generic driver depends on proper implementation of the AUX XIOS calls, which on some systems were not implemented on 3.1 - for instance, Comart CP2000 needs 5.0, Rair needs 5.2. The versions mentioned above refer NOT to the generic XIOS CDOS version but to the generic CP/M-86 BIOS version of COMM+.

Jarogate Sprite	BIOS version only. No speed control, split speeds or break support. DCE.
Logica Kennet	No speed control, no split speeds, no break support. Runs under Concurrent. Interrupt driven. DTE. Also sold as a BT Merlin machines (but since BT have badged both IBM clones and systems from Rair and ICL as well as Logica, people with BT Merlin systems really must find out what incarnation they have).
Missing Link Modem	Works at 300/300 or 1200/75. Break supported. Comes with instruction sheet accompanying disk. Renamed Breakout Modem.
NCR DM5	Runs a 16-bit driver with same features as the 8-bit NCR DM5 version.
NEC APC I	Full speed control, software split speeds, break support. DTE. Note that the CP/M-86 version can be run under MSDOS (if still unsaved) by copying all the files on disk to an MSDOS diskette using the supplied READCPM utility which works with IBM3740 single-sided single density disks. Consequently, disks usually supplied on IBM3740 format for CP/M-86 (which reads them happily) unless specifically requested. Note that certain screen functions such as insert and delete line are not available in the emulators. The NEC serial board must have its jumpers set for ASYNCHRONOUS operation - as supplied by the manufacturer, the synchronous mode is the default. Consult either your dealer or your manuals.
NEC APC III	Full speed control, software split speeds, break support. DTE. Not the same as APC I. Note that the NEC APC IV Powermate range are IBM clones.
Octopus	No speed control. Break supported. Interrupt driven. Two versions available, for either port 2 (standard) or port 6 (optional). Runs under CP/M-86 or Concurrent.

	version can be run under MSDOS (if still unsaved) by copying all the files on disk to an MSDOS diskette using the supplied READCPM utility. Consequently, disks usually supplied on CP/M-86 format unless specifically requested.
Equinox	Turbodos system (CP/M-86 compatible). Full speed control, no software split speeds, break support. DTE. Graphics file for viewdata available : assumes Ampex VDU in use.
FTS	No speed control, no split speeds, no break support. Runs under Concurrent. Suitable for Email and Viewdata only. Polled.
Fujitsu 16S	Speed control, no split speeds, break support. Runs under Concurrent also. DTE. The function keys are predefined, and are shown on the 25th line of the display. COMM uses the main serial port on the machine rather than any extra serial ports which may be installed. Terminal emulations at speeds of 9600 are not reliable.
Future FX20/30	Full speed control, software split speeds, break support, interrupt driven. DTE. Doesn't work with Future network. Runs under Concurrent. Standard and pulse dialling drivers available.
IBM PC	Speed control, software split speeds, interrupt driven, break support. DTE. Standard and pulse dialling drivers available. For all IBM clones.
ICL DRS	No speed control or break support. Interrupt routines handled by BIOS - make sure that data is coming through the port transparently, as the DRS can double all 10H codes up for some reason, as well as having the usual XON/XOFF problems.
ICL PC2	No speed control or break support. Supplied with ICL VDU as standard. Interrupt routines handled by XIOS. DCE. Same as ICL Quattro.

COMM+ Glossary

Olympia People	Speed control, software split speeds, break support. Runs under MSDOS also. Graphics file available. DTE.
Orb	Full speed control but no break support. Supplied with Tatung VDU as standard. Interrupt routines handled by XIOS. DCE. Make sure that the port being used is currently unassigned. Defaults to port 8.
Rair Supermicro	No speed control or break support. DCE. Supplied with ADDS VDU driver as standard. Interrupt routines handled by XIOS. Use TTYSET from the operating system to set the speed. Whilst any port can be set up (with B from the off-line menu) you must ensure that the port you are using is NOT set up as a console or list device. ASSIGN will tell you if a port is free. Also works with updated BIOS on the Rair 360/365.
Sirius/Victor	Speed control, software split speeds, break support. We have to exit by doing a complete reset - our apologies for this. Reason for normal exit not working is unknown.

GLOSSARY OF TERMS

(Cross-reference may be needed)

ACOUSTIC COUPLER – A type of modem which can be used anywhere, in which a telephone handset is physically inserted into two (usually rubber) cups. It converts outgoing digital computer data into acoustic tones of differing frequencies, and converts them back again when they come in – effectively two computers whistle at each other.

ANSWER MODE – When two modems, acoustic or direct connect, are linked up to each other over telephone lines, one of them must be in originate mode and the other must be in answer mode. The frequencies used to modulate the signals (the tone of the whistle) is different in each case. The answer modem listens for originate tones, but outputs answer tones, whilst the originate modem listens for answer tones but outputs originate ones. Two modems or couplers in the same mode will NOT work together.

ASCII – Stands for American Standard Code for Information Interchange. An industry-standard method of encoding letters into binary form.

ASYNCHRONOUS – A type of serial protocol which does not depend on accurate timing, in which the beginning and end of a word are marked by start/stop bits.

BACKSPACE – A key on a computer which moves the cursor back one space.

BACKUP – A copy of computer information, either program or data, kept securely in case the original is damaged. It is most important to make and keep backups!

BAUD RATE – A way of referring to the speed of transmission of serial computer data. Equivalent to bits per second. 300 baud thus equals 300 bits per second. If the word length (qv) is 10, 300 baud=30 characters per second.

BINARY – The way most computers store all data. Combinations of 0 and 1.

BIT – A bit is one digit of binary information, either 0 or 1.

BREAK – A continuous high signal on a data link. In asynchronous communications, a low (marking) signal indicates no activity on a link, and a pattern of high and low signals indicates data.

BUFFER – A store.

BYTE – A computer word. Usually consists of 8 bits. Confusingly, one character takes up one byte.

CARRIER – The signal on a modem or acoustic coupler which conveys the fact that something is connected at the other end. No carrier = no connection.

CDOS – Digitial Research Concurrent DOS operating system.

CGA – Colour Graphics Adaptor. One of the IBM PC display standards.

COMMUNICATIONS – Relating to exchange of information. On a computer, a communications port must be able to both send and receive.

CONCURRENT – A feature of programmes and/or operating systems whereby they appear to be doing more than one thing at once.

CONTROL – A key on a computer keyboard. Like a SHIFT key, when held down at the same time as another key it alters that character.

COPY – A utility program used to make a backup.

CP/M – Control Programme for Microprocessors. (Actually nobody seems to know for certain what it stands for.) A standard operating system, marketed by Digital Research (U.S).

CRT – Cathode ray tube. Commonly used to refer to a screen (qv).

CTS – Clear To Send. One of the modem control pins used in serial communications.

CURSOR – The indication of where you are on the screen. Could be a solid block or a line, flashing or steady.

DATA – Computer word for information. Could be a program, or could be text, values etc.

DCD – Data Carrier Detect. One of the modem control pins used in serial communications.

DELETE – A key on a computer keyboard. Has the effect of a destructive backspace.

DIALCOM – An Electronic Mail system marketed in the UK as Telecom Gold.

DIRECT-WIRED – Connected permanently and physically via wires.

DIRECTORY – A catalogue of files and programs on a disk.

DISK FILE – A block of logically contiguous data stored on a disk, with its own name and size. Just like a paper file.

DISK(ETTE) – Magnetic storage media for computers – often 5.25" in diameter, but can also be 8", 3.5" or 3". The two latter types have a rigid enclosed case, whilst the former are really floppy and thus more fragile. All type are susceptible to damage from heat, dust and magnetic fields in particular.

DRIVE – Device in which a disk/ette is inserted. It reads and writes information to the disk magnetically.

DSR – Data Set Ready. One of the modem control pins used in serial communications.

DTR – Data Terminal Ready. One of the modem control pins used in serial communications.

DUPLEX – A type of communication, either half or full (qv).

ECHO – What you send comes back to you. If a computer is echoing to you, any character you send is sent back. Useful to check whether your connection is good.

EDIT – Change content of text.

EEPROM – Electronically Erasable Programmable Read Only Memory.

EGA – Enhanced Graphics Adaptor. One of the IBM PC display standards.

ELECTRONIC MAIL – A computerized system of information transfer over what can best be described as electronic pigeonholes in a large computer. Each user can read only from their own pigeonhole, but can send, answer, forward messages to any other pigeonhole on the system.

EMS – Expanded Memory Specification. A method of accessing paged memory above the 640K limit imposed by MSDOS and PCDOS (see LIM-EMS).

EPROM – Eraseable Programmable Read Only Memory.

ESC – A key on a computer keyboard. It has no printable equivalent, but is a key in its own right, like alphanumeric keys.

FILE – See disk file

FLOPPY DISKETTE – See diskette. The term floppy is relative. Do not mistreat them

FORMAT – A utility program which prepares disks for use by a computer. Imagine it as like ruling lines and columns of a blank sheet of paper before writing on it. Format generally means prepare.

FULL-DUPLEX – When a computer operates in full-duplex it can send and receive at the same time. The most noticeable effect is that it makes echo possible – what you see on the screen is what is echoed back, not what you type directly.

GENDER CHANGER – A connector with which turns a plug (qv) into a socket (qv) or vice-versa.

HALF-DUPLEX – When a computer operates in half-duplex it can only send or receive at any one time, not do both. Echo isn't usual in half-duplex, so what you see is either what you are sending out or what the host has sent to you – the disadvantage of no echo is that you don't know whether what you you sent was the host received.

HANDSET – Another description for the telephone receiver that you talk into and listen to.

HANDSHAKE – The way that computers and computer devices regulate their flow of information is called a handshake.

HEXADECIMAL – A way of representing binary data using numerics from 0 to 9 with alpha characters A to F to extend the decimal set. Thus 15 decimal = F hexadecimal

HOST – A computer which a terminal connects up to. You can't have a terminal unless it is a terminal to something, and that something is called a host.

HOTKEY – Jargon word denoting a key combination to be pressed to activate a TSR (qv).

INTERFACE – The part(s) on a computer which connect to something else – thus disk interface, keyboard interface etc.

IPSS – International packet switching system. Used to communicate internationally with computers or services in foreign countries. See PSS.

JACKET – The paper envelope a disk is stored in and protected by. Also used to refer the the black teflon–like material the magnetic part is wrapped in.

KBYTE – 1024 bytes.

KEYBOARD – The part of a computer which looks almost like a typewriter without any paper handling.

LIM–EMS – The Lotus–Intel–Microsoft Expanded Memory Specification. An EMS (qv) specification agreed on by those three companies, which has become a de facto standard for expanding the memory on DOS systems.

MDA – Monochrome Display Adaptor. One of the IBM PC display standards.

MEMORY – Where data storage occurs.

MEMORY BUFFER – A part of memory separated off as a discrete store for program use.

MNP – Microcom Networking Protocol. A series of error–checking protocols pioneered by Microcom, which have become widely adopted by other modem manufacturers.

MODEM – Stands for modulator/demodulator. See acoustic couplers. These are also called acoustic modems. Modem by itself is usually used to refer to the direct–wired type. Most modems are controlled by manipulation of voltage levels on their connectors. However, some are software driven, enabling control by issuing simple commands to them.

MSDOS – An operating system similiar to CP/M, but marketed by a company called Microsoft rather then Digital Research. Only runs on 16–bit computers. (CP/M is available for both 8 and 16 bit machines).

NULL MODEM – A length of wire with connections which effectively runs like two modems back–to–back.

OPERATING SYSTEM – The part of the software which integrates all the bits of the computer system together –files, peripherals, memory, input, output, etc. CP/M and MSDOS are operating systems.

ORIGINATE MODE – See answer mode.

PACKET – A bundle of data collected and sent as one unit for addressing and error–checking.

PAD – Packet Assembler–Disassembler. A device for generating packets (qv) of data from a byte stream, which are enclosed with headers and trailers for error–checking and addressing purposes. Normally use a protocol such as X.25 to talk to each other, but when a terminal is connected to a host via a pair of PADs, they can normally be treated as transparent error–checking devices.

PAGE – Can be used to denote a discrete portion of computer memory, as in EMS page or graphics page.

PARALLEL – A method of transferring many bits of information at the same time down a number separate wires, one for each bit. Usually contrasted with serial.

PARITY – Part of a serial communications protocol. Either odd, even or none. Since ASCII (qv) is a seven bit code, and most computers operate wih 8 bit words, the 8th bit is unused. It is called the parity bit, and under even parity is set to make the number of binary 1s in a word even, while under odd parity it is set to make the number of 1s odd. So if you are under even parity and you receive a binary word comprising 00111000, the odd number of 1s indicates a failure to communicate properly. No Parity means the bit is sent transparently, space means it is always 0, mark means it is always 1.

PCDOS – A version of MSDOS which runs on the IBM personal computer.

PIP	- A CP/M utility program for copying files (amongst other uses). Stands for peripheral interchange program.
PLUG	- A connector with pins on it - mates with a socket. Often referred to as a male connector.
PRESTEL	- A large public viewdata network part-owned by British Telecom - see Viewdata.
PRINTER	- A machine that produces printed copy on paper when linked to the computer.
PROGRAM	- A series of computer instructions. Correct English is actually programme, but the American variant of the word is now more common in connection with computers.
PROM	- Programmable Read Only Memory.
PROMPT	- A request for operator input
PROTOCOL	- The things that two computers must agree on to be able to communicate.
PSS	- Packet switching system. A service run by British Telecom enabling access between computers simply by dialling a local PSS node and entering the PSS network address of the computer you wish to contact. Access is possible over the PSTN at speeds up to 1200 baud (full duplex only), but higher speed access is available by leasing dedicated lines from BT.
PSTN	- Public switched telephone network
RAM	- Random access memory.
READ ONLY	- Only for reading. Cannot be written to or on.
READ/WRITE	- Can be both read from and written to.
RETURN	- A key on a keyboard. Simulates both the carriage return on a typewriter and the enter key on a calculator pad.
ROM	- Read-only memory.
RS232	- A computer standard serial data transfer protocol.
RI	- Ring Indicator. One of the modem control pins used in serial communications.

RTS	- Ready To Send. One of the modem control pins used in serial communications.
SAVE	- Write a file to disk.
SCREEN	- The cathode ray tube on a computer on which information is displayed.
SCROLL	- Move a screenful of information vertically.
SERIAL	- A way of transferring binary data. Each bit is sent sequentially, one after the other.
SOCKET	- A connector with holes in it - mates with a plug. Often referred to as a female connector.
SPLIT SPEED	- Typically used by Viewdata systems. Transmission speed and reception speed differ.
STOP BITS	- The bits which end a word, in asynchronous (qv) communications protocols. Could be one, two (or three sometimes). Part of a protocol (qv).
STRING	- Any sequence of characters.
SYNCHRONOUS	- Serial communications protocol where data transfer is regulated by clocks, and is time-critical. Used by (amongst others) IBM.
TERMINAL	- Normally a keyboard and VDU, connected to a host.
TEXT	- Letters, numbers and words in readable form or a variation thereof, encrypted or otherwise.
TSR	- Terminate and Stay Resident. A type of MSDOS programme that stays in the memory of the computer and is capable of working whenever needed without being reloaded. Effectively it has become part of the operating system.
UPPER CASE	- Capital letters.
USER AREA	- A logically (not physically) distinct part of a disk. Under CP/M, there are up to 16 possible distinct user areas on a disk. Files written in one user area cannot be accessed directly from another area. Under MSDOS or PCDOS user areas as such do not exist, but release 2.0 and later allows mutually exclusive tree-structured directories to be created, which serve a similiar purpose.

COMM+ Index

UTILITY PROGRAM	– A program which performs essential functions for the operating system.
V21	– 300/300 baud full duplex communications.
V22	– 1200/1200 baud asynchronous full duplex communications.
V22 bis	– 2400/2400 baud asynchronous full duplex communications.
V23	– Either 75/1200 baud or 1200/75 baud full duplex, or 1200/1200 baud half duplex asynchronous communications.
VDU	– Visual Display Unit. Commonly used to refer to a screen (qv).
VIEWDATA	– A communications standard based on 40x24 colour graphics pages, using a character set similiar but not identical to ASCII (qv). Communication is typically at 1200 baud to the user, and 75 baud back to the system – the concept is designed around page access with single-character responses from users. Prestel is the leading viewdata system.
VOLATILE	– Liable to vanish without power.
WORD	– Confusingly, a word is a byte. Therefore a letter is a computer word long. Eight-bit computers deal with eight-bit words, 16-bit computers deal with 16-bit words. Since the ASCII code (qv) is actually 7 bits, 8 bits are quite satisfactory for most purposes. The COMM language uses WORD more conventionally.
WORD PROCESSOR	– A program or computer which enables editing, manipulating, saving, retrieving and printing of text.
X.25	– A packet-switching protocol used by PADs (qv).

THE END

Screen Display - On-Line Menu

```
O N - L I N E   M E N U      File =  A:TEST      .TXT
-------------------------------------------------------
  ENTRY COMMANDS                        |      TO EXIT
T= On-line  -  Memory off ( * ) |  E= Off-line Menu
G= On-line  -  Memory on  (   ) |  Q= to System
B= Type direct to Memory  (   ) |-----------------------
-------------------------------  ON/OFF TOGGLES
  HELP                          |  L= Line feed     (   )
J= Help level     ( 2 )         |  P= Printer echo(     )
H= This On-line Menu            |  I= Handshake     ( * )
--------------------            |  W= Emulation     (   )
  PROTOCOL                      |-----------------------
Y= Set up Handshake             |  SPECIAL KEYS
X= Duplex   ( F )               |  K= Change Command key
--------------------            |  U= Userkey Changes
  DISK                          |-----------------------
N= Name File  Z= Rename File    |      MEMORY
F= Send File  V= Select Drive   |M= Send Memory
S= Save File  A= Append to File |C= Clear Memory
D= Directory  R= Reset Disk Drives|O=  Old Memory
-------------------------------------------------------
   ( * )= on (   )= off: Command key is now <ESC>
```


User-friendly message

You have opened the manual upside-down.

Inside the "start point" box is the reminder "ensure all connections are in order". This is important: one thing COMM does from the time it is loaded on many systems is to look all the time to see if anyone is trying to communicate with you. (Technically, this procedure is called 'polling'.) On some machines, if nothing at all is connected to the computer, it will look as if there is something sending a continual stream of characters, which have no value, but still appear to be sent - this can cause problems. If you have made sure that your connections are in order, even if there is nothing at the other end (you do NOT have to dial up, only plug in) this problem will never arise even if your hardware is susceptible to it.

It is also true that some machines don't like connections being made whilst they are on - this is especially true of equipment which involves input voltages, such as printers and modems.

The first instructions proper appear as an option:

<RETURN> to go on-line or
<ESC> for off-line operation

(the words and letters enclosed in <> angle brackets denote keys to press throughout the manual - you don't need to type the brackets themselves).

Press the <ESC> key, and the screen will clear again, and display the OFF-LINE MENU. A printout of this screen is on the next page.

(If you have pressed the <RETURN> key by mistake, and ended up in on-line mode, type <ESC> again followed by <E> followed by <Y>.)

The OFF-LINE MENU is divided into three sections. We suggest that you read through the next section of the manual carefully to understand all the functions on this menu. (You will find that it is not nearly as complicated as it may appear at first.)

3. FUNCTIONS IN THE ON-LINE MENU

The on-line menu contains another set of options, this time divided into eight sections. To call each function you have to preface the single letter on the menu with the command key (as displayed at the bottom of the menu) - this is initially <ESC>. The Supplementary Reference Guide contains details on changing the command key should you need to do so. Most users will find <ESC> to be quite satisfactory.

1) ENTRY COMMANDS -

T= On-line - memory off (*)
G= On-line - memory on ( )
B= Type directly to memory ( )

The above allow you to select which on-line mode you enter. The brackets and star signify which option is currently active.

(*) is ON
( ) is OFF

The above convention will be used relatively frequently in the on-line part of the program.

2) TO EXIT -

E= to off-line menu (takes you back to the off-line menu)
Q= to system (releases you from COMM and returns you to the operating system)

Screen display - Off-Line Menu

```
O F F - L I N E  M E N U      File =  A:TEST      .TXT

-------------------------------------------------
                  DISK COMMANDS
 S= Save File       |N= Name File   | D= Directory
 F= Send File       |Z= Rename File | V= Select Drive
 A= Append to File  |K= Delete File | R= Reset Disks
        J= Change area being used on Disk
-------------------------------------------------
                  MEMORY
 M= Send Memory     |E= Edit and/or | C= Clear Memory
 O= Get old Memory  | format Memory | I= Input File
-------------------------------------------------
                  UTILITIES
 U= Define Userkeys |X= Duplex      | H= Help level
 L= Block Transfer  |Y= Handshake   |
 P= Programme Save  |B= Baud + word |----------------
 T= Typewriter Mode |G= Get JCF     | Q= EXIT
-------------------------------------------------

      ** Press RETURN for operation On-line **

                      Select:
```