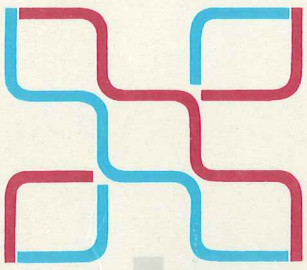
POLYSYS

UTILITIES

MANUAL



POLYSYS

UTILITIES

MANUAL

VERSION 2

MAY 1983

POLYCORP

New Zealand Limited

Progeni House, 14/18 Pretoria Street
Telephone (04) 693-302, P.O. Box 30-243, Lower Hutt
Telex PO LHT NZ3740 Attn Progeni

CONTENTS

1. DISK OPERATING SYSTEM UTILITIES

1.1. INTRODUCTION

The Disk Operating System (DOS) utilities are provided on the programming disks and, where there is room, on the courseware disks. Each utility is a separate file with an extension of .CMD. A utility is executed from DOS by entering the file name (the .CMD extension is not required).

Where the utility is relatively small and requires little work space, it is executed in the POLY utility area so that the BASIC program or TEXT currently loaded is not affected. The utilities that will not run in the utility work space must be run from DOS. These utilities may be run from BASIC and TEXT by putting a + in front of the utility name. The availability of a utility is detailed under the description of particular utility.

For example:

In DOS mode entering

```
CAT 0
```

will provide a catalogue of the files on the disk in drive 0. In BASIC and TEXT modes

```
+CAT 0
```

will do the same.

To enter DOS mode from BASIC or TEXT, enter DOS.
To return to BASIC from DOS or TEXT, enter BASIC.
To enter the TEXT mode from DOS or BASIC, enter TEXT.

The utilities expect either a space or a comma to separate the parameters.

For example:

```
CAT 0 .BAS
```

is exactly the same as

```
CAT,0,.BAS
```

If a semi-colon immediately follows the utility name, then the utility will load and pause, waiting for any key to be pressed before beginning execution. It is then possible to change disks.

For example:

If you have CAT on one disk and you wish to obtain a catalogue of the files on another disk that does not have CAT on it, then enter

```
CAT;
```

When CAT is loaded it will pause so that you may change disks. Press any key to obtain the required catalogue.

Most utilities are loaded from the disk into the POLY for execution and do not interfere with other users. However, for efficiency reasons, some utilities are executed in the disk drive. While these utilities are being executed, other POLYs on the system will not be able to gain access to the disk drive. Such utilities are COPY, FORMAT and FASTCOPY.

In this manual all examples are given as if executed in DOS mode.

1.2. FILE NAMING CONVENTIONS

File names may be up to 8 characters long. The first character must be alphabetic and the remainder must be alphanumeric. File names maybe followed by a "." and a one to three letter extension. Some extensions have been associated with specific types of files.

BASIC source files	.BAS
BASIC compiled files	.BAC
Operating system commands	.CMD
Data files	.DAT
Print files	.PRT
Operating system files	.SYS
Text files	.TXT

If the file is on a particular drive then the drive number may be added to the filename either at the beginning or the end.

For example:

```
1.PROG1.BAS   or   PROG1.BAS.1
```

Both associate the file PROG1.BAS with drive 1.

1.3. COMMAND DESCRIPTIONS

In describing the command syntax, the following conventions are used:

-Words in capital letters must be entered exactly as written.

-Words in small letters must be replaced by the user with a specific filename or other word as required.

-Words enclosed in square brackets ([ ]) are optional and may be omitted.

-Underlined words must be entered if that part of the option is used.

-In examples, the user responses to queries are shown underlined.

For example:

Syntax:- CAT [drive-list] [match-list]

Allows any of the following to be used:

```
CAT
CAT 1
CAT MYFILES
CAT 0 MYFILES
```

1.3.1. BASIC

Availability:- Available in DOS mode only

Syntax:- BASIC [filename]

The BASIC command loads the RAM extensions for POLYBASIC from disk.

If a file name is specified (default extension .BAC), then the file will be loaded and executed as a BASIC program. If no filename is specified, then immediate mode is entered.

For example:

```
BASIC
```

loads the extensions to BASIC and enters immediate mode.

```
BASIC,MENU
```

loads the extensions to BASIC, then loads and executes MENU.BAC

1.3.2. CAT

Syntax:- CAT [drive-list] [match-list]

The CAT utility displays names of the files on a disk. The drive-list may be one or more drive numbers, and the match-list can be a series of filenames, extensions or abbreviations of both to allow 'masked' viewing of the file names in the directory.

For example:

```
CAT 0 PP2.BAS
```

Lists only those filenames on drive 0 beginning with PP2 and having extensions of .BAS.

CAT

Lists all filenames on the current disk drive assigned to that POLY.

CAT,1,P.T,MA

Lists those files on drive 1 which begin with P and have extensions beginning with T, and files beginning with MA.

CAT,0,1,TYP,.SYS

Lists all files on drives 0 and 1 which either begin with TYP or have extensions of .SYS.

CAT also displays other information about the file.

For example:

NAME	TYPE	R	SIZE	DATE	PRT
POLYEX	.BAC		27	26-FEB-82	W

The R if present, indicates the file is a random access type data file. The size is given in 256 byte sectors. The date is the date on which the file was created. The PROTection code is a list of protection attributes assigned to the file (see PROT).

1.3.3. COPY

Syntax:- COPY filename1 filename2
COPY filename drive
COPY source-drive dest-drive [match-list]

The COPY command copies files. If a system has only a single disk drive attached, SDC (Single Disk Copy) must be used to copy files from one disk to another. COPY can only be used in a single drive system to produce a copy of a file on the same disk under a new name.

If the file to be created already exists, a request to delete it is displayed. Pressing Y causes the file to be overwritten.
All copied files retain the date and protection of the original.

For example:

COPY may be used on a single disk in the first form. If this is done then the filenames must be different.

COPY 0.DEMOPR.TXT 0.DEMOUP

This copies O.DEMOPR.TXT to O.DEMOUP.TXT

The extension of the input file must always be specified, but for the output file this is optional as it defaults to the extension of the input file.

COPY may be used to copy between drives. If the drive is not specified, then the current drive for that POLY is used.

For example:

If the current drive is O, then

```
COPY DEMOPR.TXT DEMPOUP
```

copies O.DEMOPR.TXT to O.DEMOUP.TXT

When copying from one drive to another, the file may retain its original name.

For example:

```
COPY O.FILE23.BAC 1
```

This copies FILE23.BAC from drive O to drive 1.

Finally, COPY can be used to copy all files from one drive to another or only those corresponding to a match list.

For example:

```
COPY O 1
```

will COPY all files from drive O to 1

```
COPY 1 O .BAS .TXT
```

will COPY all files from drive 1 to drive O that have extensions of .BAS or .TXT.

```
COPY O 1 DATA PROGR.C
```

will copy all files from drive O TO 1 that have names beginning with DATA as well as those that begin with PROGR and have extensions beginning with C.

The name of each file copied is displayed on the POLY screen.

NOTE: While COPY is running, all other disk activity is suspended.

1.3.4. DATE

Syntax:- DATE

The DATE utility allows the user to input the date and time for the network controller from a POLY, if the network controller has been reset or otherwise initialised. If the network controller already has a valid date, then DATE will not ask for a new date and time, and the POLY date and time will be updated from the network controller.

Only the POLY from which DATE was run, and the network controller date and time are changed. Other POLYs on the network have the previously set date and time until they are reset or logged off.

For example:

After the network controller has been reset, enter

```
DATE

 Enter date DD,MM,YY 20 1 83

 Enter time HH,MM 9 12
```

This will set the network controller and POLY dates to 20 January 83 and the times to 9:12:00.

1.3.5. DRIVE

Availability:- Available in DOS mode only

Syntax:- DRIVE drive-number

The utility duplicates the DRIVE command in TEXT and BASIC modes. That is, DRIVE reassigns the current drive for a particular POLY.

For example:

```
DRIVE 1
```

This causes the POLY to use drive 1 as its current drive.

1.3.6. FASTCOPY

Syntax:- FASTCOPY [source-drive dest-drive]

FASTCOPY duplicates all of the information stored on the disk in source-drive onto the disk in dest-drive. Two drives are required. The destination disk cannot have any bad sectors (see FORMAT).

FASTCOPY copies sector by sector rather than file by file and so a corrupt disk, or one in which files are scattered all over it, will be reproduced exactly as the original. FASTCOPY will overwrite anything already on the disk being copied onto.

For example:

FASTCOPY 0 1

will duplicate the information on the disk in drive 0 onto the disk in drive 1.

NOTE: While FASTCOPY is running, all other disk activity is suspended.

1.3.7. FORMAT

Syntax:- FORMAT [drive]

FORMAT is used to format a new diskette or reformat an old one. FORMAT must be used on all new disks before they can be used. If drive is not entered with the command, then FORMAT will ask for the drive.

For example:

FORMAT 1

ARE YOU SURE? Y

indicates that formatting is desired (N is pressed to abort format)

DOUBLE SIDED DISK? Y

(or N)

VOLUME NAME? POLY

(up to 7 characters with the same rules as file names)

VOLUME NUMBER? 1

(up to a 4 digit number)

IS THE DISK TO BE FORMATTED IN DRIVE 0? Y

(N to abort formatting)

NOTE: Inputs to the queries do not require a terminating <ENTER> except in the case of volume name and number.

When the process is complete, a message stating the total number of sectors formatted is displayed. For a single-sided disk this should be 1140 sectors, 2280 for a double-sided disk.

FORMAT checks for surface defects. If a bad sector is found on a part of the disk required by the POLY operating system, then the FORMAT is aborted. If this occurs remove the disk from the drive, reinsert it and try again. If this proves unsuccessful after another try then assume the disk is unable to be used.

If bad sectors occur on other parts of the disk, then these are reported and formatting continues. These disks may be used but FASTCOPY cannot be used to copy onto the disk.

Do not turn single-sided disks over to use the other side, as the protective sleeves have dust collecting surfaces on the inside which are not intended for use in a reversed direction.

NOTE: While FORMAT is running, all other disk activity is suspended.

Using FORMAT with a Single Disk Drive

FORMAT may be used in a single drive, but the disk to be formatted must be placed in the drive before the final question. To prevent formatting the disk containing FORMAT.CMD accidently, it is a good idea to Write Protect by removing the tab from the disk protection notch.

1.3.8. GPRINT

Availability:- Available in DOS mode only

Syntax:- GPRINT

GPRINT is used to print the POLY graphics screens. The program asks for various parameters for printing the screen.

For example:

Start row (0-203) 0

row 0 is the first row to be printed

Start column (0-239) 0

column 0 is the first column to be printed

End row (0-203) 200

row 200 is the last row to be printed

End column (0-239) 239

column 239 is the last column to be printed

Size (1 or 2) 1

the size of the printed picture is normal. (Entering 2 would select double si ze.)

Screens (24) 2

screen 2 is to be printed (Screens 2, 4 or 2 and 4 can be printed).

Output filename ? SCRNDMP

the output filename for the picture will be SCRNDMP. (Any valid filename can be used. The extension defaults to .LST, .PRT extensions should not be specified.)

The result of the above example will be the file SCRNDMP.LST which will contain a complete copy of screen 2, and can be printed with the command

PRINT SCRNDMP.LST NG

NOTE: This utility will only work with EPSON MX-series printers with the bit-image graphics option.

1.3.9. KILL

Syntax:- KILL [drive-list] [match-list]

KILL is used to delete files from disk.

Before deleting a file a check is made.

Delete "FILENAME" ?

Any reply other than Y will leave the file intact and proceed to the next file , but if Y is pressed then the file is deleted. <ENTER> is not required after the reply, and <ENTER> as the reply will cause KILL to terminate.

For example:

KILL MYFILE.BIN

will delete MYFILE.BIN from the disk on the current drive for that POLY.

KILL 1 FILE.CMD 0 DATES.TXT

will delete FILE.CMD from drive 1, and DATES.TXT from drive 0.

Files may be protected against any attempts to KILL them by Delete or Write protecting them (see PROT for the control of such protection).

A file on the PRINTQUEUE may not be deleted using KILL but the print may be stopped using PRINT -filename.

If no filenames are given on a KILL command then the files on the disk are presented one by one for deletion. If Y is pressed, then the file is deleted. Pressing any other key leaves the file intact. Make sure that all write and delete protect flags are

removed from files to be deleted prior to running KILL in this manner.

For example:

```
KILL 1
```

will give the option to delete all files from the disk on drive 1.

```
KILL 0 PP
```

will give the option to delete all files starting with PP on drive 0.

1.3.10. LINK

Syntax:- LINK filename

Where filename is the name of the POLY Disk Operating System file. LINK sets up a pointer on the disk to filename, causing it to be loaded into the network controller memory whenever the network controller is turned off or reset. LINKing must be done before a disk can be used for automatic loading.

For example:

```
LINK 1.POLYNET.SYS
```

will LINK POLYNET.SYS on drive 1.

A disk created using FASTCOPY does not need to be LINKed if the original disk has been LINKed.

1.3.11. LIST

Syntax:- LIST filename [startline - endline] [+N]

LIST lists the contents of TEXT or BASIC files on the screen. Entire files or only selected lines may be listed.

The drive number may be included in the filename. If the file extension is not specified, the extension defaults to .TXT. The numbers of the first and last lines to be displayed may be specified - otherwise the whole file is LISTED.

For example:

```
LIST 1.TESTPR.BAS
```

will produce

```
10 REM TESTPROGRAM
20 REM TESTPROGRAM
50 REM TESTPROGRAM
70 REM TESTPROGRAM
```

```
100 END
```

on the screen.

```
LIST 1.TESTPR.BAS,1-4
```

will produce

```
10 REM TESTPROGRAM
20 REM TESTPROGRAM
50 REM TESTPROGRAM
70 REM TESTPROGRAM
```

on the screen.

```
LIST 1.TESTPR.BAS 3-
```

will produce

```
50 REM TESTPROGRAM
70 REM TESTPROGRAM
100 END
```

on the screen.

Note that the range indicates the actual line numbers of the lines and not BASIC line numbers. +N causes actual line numbers to be printed.

1.3.12. OVERRIDE

Availability:- Available in DOS mode only

Syntax:- OVERRIDE

The OVERRIDE utility may be used to over-ride the protection on a file. Log on to a POLY with initials and password, enter DOS mode and run OVERRIDE. OVERRIDE sends the current initials and password to the disk drive as a master password, and this overrides normal passwords. (PROT can then be run so protection can be removed from files on which the passwords have been forgotten.) Access to this utility must be restricted to supervisors.

1.3.13. PCOPY

Availability:- Available in DOS mode only

Syntax:- PCOPY filename1 filename2
PCOPY filename drive
PCOPY source-drive dest-drive [match-list]

The PCOPY command is the same as the copy utility, except it allows the user to selectively copy files. PCOPY can only be used on a single drive system to produce a copy of a file on the same disk under a new name.

For example:

PCOPY 0 1 GEOG

will display all files whose names begin with GEOG, giving the option of having each one copied (Y) or not (N). <ENTER> is not required after the Y or N. Pressing any other key will cause PCOPY to abort.

COPY COMPLETE

is displayed when PCOPY is complete.

NOTE: While PCOPY is running, other disk activity continues.

1.3.14. PRINT

Syntax:- PRINT [filename] [options]

The PRINT command is used to print a file on the printer. The default extension for the filename is .TXT. There are six options available:

D - Double spacing: causes double vertical spacing on the printout.

N - No headings: where this is not specified headings take the form of:

```
**************************
  LISTING OF FILENAME.EXT
**************************
```

C - Compressed print: this will cause compressed characters to be printed.

P - Paging: prints 60 lines per page then skips to the top of the next page.

E - Elongated print: this will cause elongated characters to be printed. In the case of the Epson MX80 printer, there are normally 80 characters per line. In compressed mode, there are 132 characters per line; in elongated mode, 40 characters per line.

G - Graphics print: this disables automatic line feeds and other special translations by the print spooler, allowing the user to control the spacing and to print control characters. This is especially intended for printing graphics.

To delete a file from the print queue (even if it is printing) put a "-" sign in front of the filename.

PRINT without any parameters will display the contents of the print queue.

For example:

```
PRINT POLYPR DNC
```

will print the file POLYPR.TXT using double spacing with 132 characters per line and with no heading.

```
PRINT -POLYPR
```

will abort the printing of POLYPR.TXT.

1.3.15. PROT

Syntax:- PROT filename [option-list]

This command is used to change the PROTection attributes of a file. Protection against deleting and writing to a file is set by PROT. Until PROT has been run, a file may be rewritten, renamed or deleted.

The filename is the name of the file to be protected. The option-list may contain any number of the following protection codes.

D - Delete: to protect a file so that it cannot be deleted by KILL or from within a program. The file may still be changed.

W - Write: to write protect a file so that it may not be deleted or renamed or have anything written to it. It is automatically delete protected.

C - Catalogue: to catalogue protect a file so that it will not be displayed when CAT is executed. To display these files, KILL with no file names may be used.

P - Password: to protect a file with a password generated from the user's initials and password entered when the user logged on. Access to this file will only be allowed by using the same initials and password. (It is remotely possible for another set of initials and password to generate the same password.)

X - Remove: will remove all protection from the specified file.

For example:

```
PROT FASTCOPY.CMD WP
```

will write protect and password protect the file FASTCOPY.CMD.

```
PROT SECRET.TXT XDC
```

will remove all previous protection from the file SECRET.TXT, delete protect it and prevent it being displayed by CAT.

1.3.16. RECOVER

Availability:- Available in DOS mode only

Syntax:- RECOVER [drive]

The RECOVER utility tries to recover missing sectors from a disk. If drive is not entered, then the current disk drive is assumed.

For example:

```
RECOVER 1

Are you sure ? Y
```

indicates that recovering is desired (N is pressed to abort recover)

```
Disk to be recovered in drive 1? Y
```

(N to abort recover)

1.3.17. SDC

Availability:- Available in DOS mode only

Syntax:- SDC filename1 filename2
SDC filename drive
SDC source-drive dest-drive [match-list]

The SDC command (Single Disk Copy) is used to copy files from one disk to another using a single drive. This utility is similar in operation to COPY. The source and destination drive specified must be the same.

Before each file is read, the message:

```
Insert source disk & press a key
```

is displayed. Insert the disk that the file is to be copied from. When a key has been pressed, the file being copied is read into the memory of the Poly.

When complete,

```
Insert destination disk & press a key
```

is displayed. Insert the disk that the file is to be copied to then hit a key. When the file has been copied a message is returned. When copying multiple files, the above process repeats until all the specified files have been copied.

For example:

```
SDC TESTPR.BAS
```

copies the file TESTPR.BAS from one disk to another using the current drive.

If a multiple drive system is available use COPY or PCOPY, they are much faster.

If the file being copied is larger than the amount of memory available, then the source and destination disks will have to be interchanged as many times as necessary until all the file is copied. If an error occurs it will be necessary to insert the destination disk so that SDC may delete temporary files.

1.3.18. TEXT

Availability:- Available in DOS mode only.

Syntax:- TEXT [filename]

The TEXT command loads the POLY text editor (see section 4 of this manual). If a file name is specified (default extension .TXT), then the file will be loaded ready to be edited.

For example:

```
TEXT
```

loads the text editor

```
TEXT MYPROG
```

loads the file MYPROG.TXT ready for editing.

1.3.19. WTD

Availibility:- Available in DOS mode only

Syntax:- WTD filename utility-command
WTD + utility-command

If WTD (Write To Disk) is placed in front of a POLY utility command, then the output from the utility normally displayed on the screen will be directed to the specified file or to the printer (the + option).

The default extension for filename is .PRT i.e. a printfile.

For example:

```
WTD PRFILE LIST TESTPR.BAS 1-3 +N
```

will list lines 1 to 3 of the file TESTPR.BAS into the file PRFILE.PRT which will then be automatically printed and deleted.

For example:

```
WTD + CAT 0
```

will print the catalog for drive 0, using a spooler filename for the printfile.

2. SETTING UP A NEW DISK

Users may purchase and set up disks themselves. The disks may be either single or double sided and be either single or double density. Currently, all disks are recorded as single density, either as single or double sided.

2.1. FORMATTING THE DISK

Each disk must be formatted before it can be used for storage on the POLY system. The format used is identical to that used by the FLEX (copyright trade name) operating system. The DOS utility FORMAT is used to format the disk. This may be run either from BASIC as +FORMAT, or from DOS as FORMAT.

2.2. LOADING THE OPERATING SYSTEM ONTO THE DISK

All courseware module disks and programming disks contain the Disk Operating System files. These must be copied from one of these disks, using either COPY, SDC or PCOPY.

The operating files that are required for a disk with automatic loading are:

POLYNET.SYS	- operating system (essential)
POLYSYS.SYS	- operating system (essential)
BASIC.CMD	- BASIC language extensions (essential)
ERRORS.SYS	- text file containing error messages
MENU.BAC	- menu program (essential for auto startup)
LINK.CMD	- link program (essential if only single disk drive is available - FASTCOPY automatically copies the link information.)
TEXT.CMD	- TEXT command (essential for the POLY Text Editor)

When a disk unit is switched on or reset, an automatic startup and load occurs of both BASIC and MENU.BAC. The MENU program can be either a standard supplied MENU or a user compiled MENU written in BASIC.

For example:

If on startup the BASIC programming mode is required then a simple program such as

```
10 CLS:NEW
```

can be created and compiled as MENU.BAC.

If DOS mode is required a simple program such as

can be created and compiled as MENU.BAC. If another program is required then that program can be compiled as MENU.BAC.

2.3. LINKING TO THE INITIAL FILE

The POLY system needs to know the file to initially load when the disk unit is switched on. The DOS utility LINK must be run to set this up. The command is

```
LINK 0.POLYNET.SYS
```

2.4. SETTING UP OF A NEW DISK WITH THE OPERATING SYSTEM FOR A SINGLE DRIVE POLY SYSTEM

1. Start the system up in the normal manner using the Programming disk and enter

   ```
   DOS
   ```

2. Format the new disk. Enter

   ```
   FORMAT 0
   ```

   When the question ARE YOU SURE? is asked, change the programming disk for the new disk before proceeding.

   Answer the questions. On conclusion of format, the number of sectors formatted on the disk is displayed. This should be 2280 sectors for double-sided disks and 1140 for single-sided disks.

3. Copy the operating system onto the new disk using SDC. Put the Programming disk in the drive. It is wise to write protect the Programming disk by removing the tab from the disk protection notch. Use SDC to copy each of the operating system files from the Programming disk onto the new disk. (See SDC for instructions).

   The files required are:

   ```
   POLYSYS.SYS -(essential)
   POLYNET.SYS -(essential)
   BASIC.CMD   -(essential)
   ERRORS.SYS  -(essential)
   MENU.BAC    -(essential)
   LINK.CMD    -(essential)
   TEXT.CMD
   ```

4. Link to the initial file by putting in the new disk and entering

2.5. MULTIDRIVE SYSTEMS

On multi-drive systems, if the programming disk is placed in drive 0 and the new disk in drive 1, there is no need to change disks during the copying of the operating system files. It is wise to write protect the Programming disk by removing the Write Protect tab from the disk protection notch.
The sequence of commands from DOS is:

```
FORMAT 1
COPY 0 1 POLYNET POLYSYS BASIC BASERR ERRORS MENU TEXT
LINK 1.POLYNET
```

2.6. USE OF FASTCOPY IN MULTIDRIVE SYSTEMS

It is good practice to keep a master programming disk. To create a new programming disk the sequence of commands is:

```
FORMAT 1
FASTCOPY 0 1
```

with the new disk in drive 1. This copies everything across including the file linkage.

3. USING THE MENU PROGRAM

Each disk containing educational modules supplied by POLYCORP provides a MENU program to give easy selection and loading of the individual modules.

3.1. SELECTION

Selection is made by using the up and down arrow keys to move the flashing magenta square into the required box. The ENTER key is pressed to make the selection.

Menu also contains a number of other facilities.

3.2. BASIC PROGRAMMING

To enter the POLY BASIC programming mode, type PB.

3.3. CHANGING DISKS

Each disk has a unique menu which contains only those programs on that disk. If a disk is changed during a module, the correct menu for the new disk loads automatically. However, if a disk is changed while the MENU is being displayed, then CD must be typed in order for the new menu to be loaded.

3.4. CHANGING DRIVES

To change from the menu on one disk drive to another, type Dn, where n is the number from 1 to 4 which selects the new drive.

3.5. TEXT EDITING

To enter the POLY Text Editor, type TX.

3.6. DISK OPERATING SYSTEM

The DOS mode may be entered by typing DS.

3.7. LOGOFF

Pressing <EXIT> will log the student off and return the POLY to the magenta log on screen.

3.8. BROADCAST MODE

Broadcast mode is used to log off all POLY units on the network and load a specified program to all of them. To enter Broadcast mode, type in BC on the unit attached directly to the disk drive. The word BROADCAST appears in the bottom right hand corner of the screen. A program may then be selected from the menu in the normal manner. As soon as the selection has been made, all POLY units are logged off and the magenta log on screen appears with the words BROADCAST PROGRAM LOADING at the top. When the program has loaded, the words BROADCAST PROGRAM LOADED appears. The user then logs on in the usual manner (see the POLY System Operating Manual) but gets the loaded program, rather than the MENU.

To escape from the Broadcast mode in the Menu, BC may be typed in again. The word BROADCAST then disappears.

If BC is typed on any POLY other than the first on the network, that unit is logged off but none of the others are affected.

3.9. HELP

The <HELP> key may be pressed at any time to get a descripion of the module. Pressing the <HELP> key again, gives a list of instructions for using the MENU.

3.10. NEXT PAGE

Pressing the <NEXT> key displays the next page of modules for selection.

3.11. PREVIOUS PAGE

Pressing the <BACK> key displays the previous page of modules for selection.

3.12. CREATING YOUR OWN MENU PROGRAM

A utility is available from POLYCORP which creates MENU programs. This allows users to set up MENUs to their own specifications.

4. THE POLY EDITOR

The POLY system provides a full screen editor which can be used to edit either BASIC or TEXT files.

BASIC mode is available from TEXT and DOS via the BASIC. The prompt Ready will appear printed in yellow. The default extension for files is .BAS. In BASIC mode line numbers are part of the file. In BASIC mode, whenever a program is edited, all variable values are reinitialised and any files left open are closed.

TEXT mode is entered from BASIC or DOS via the TEXT command. The prompt Ready is always printed in cyan. The default extension for files is .TXT. In TEXT mode line numbers are not part of the file, they are added to the lines when loading (starting at 10, with intervals of 10) and deleted when saving. In TEXT mode line numbers are used to reference lines for listing, deleting and inserting.

4.1. ENTERING NEW LINES

All new lines are entered with a line number at the start which indicates the position in the file into which the line is to be inserted. If the line number is omitted the line is treated as an immediate command. Entering a line is the act of typing the line and pressing the <ENTER> key.

The cursor may be moved back to an incorrect line and the line corrected. The line is re-inserted into the file on pressing the <ENTER> key. If <ENTER> is not pressed, the line is only stored on the screen and is not updated in memory.

4.2. USING THE AUTO COMMAND

The AUTO command is used to save time when entering new lines, it automatically sets up the line numbers.

Syntax:- AUTO [start-line] [,increment]

The start-line is the first line number at which the automatic numbering will start. If not specified, 10 is used.

The increment is the amount added to each line number to get the next number. If not specified, 10 is assumed.

For example:

```
AUTO
```

starts automatic line numbering at 10 with an increment of 10, i.e. 10 20 30 40 ...

```
AUTO 100, 200
```

starts automatic line numbering at 100 with increments of 200, i.e. 100 300 500 ...

In BASIC mode, the next line number is displayed, as soon as <ENTER> has been pressed for the previous line.

In TEXT mode, the line numbers are not displayed on the screen but are incremented in memory each time <ENTER> is pressed.

To exit from AUTO mode either enter a null line (i.e. just press <ENTER> at the start of a new line) or press <EXIT>.

AUTO will not allow the entering of lines with line numbers the same as those already entered.

4.3. LOOKING AT LINES ALREADY ENTERED

The LIST command displays text already entered, on the screen.

Syntax:- LIST [startline] [-] [endline]

Startline and endline refer to the line numbers as entered. If startline is not specified, then the listing starts at the beginning of the file. If endline is not specified the listing will stop at the end of the file. The <PAUSE> is used to halt the listing at any time. To restart the listings, press any key. If the <SPACEBAR> is pressed following <PAUSE>, then the lines are listed one at a time. If the <EXIT> key is pressed, then the listing is terminated.

If only the startline number is specified then only that line is displayed.

For example:

```
LIST
```

displays the whole file.

```
LIST 100
```

displays only line 100.

```
LIST 100-
```

displays all lines from 100 to the end.

```
LIST -100
```

displays all lines up to 100.

LIST 100-200

displays lines 100 to 200 inclusive.

4.4. ALTERING LINES

To alter a line, list it on the screen using LIST, move the cursor up to the line using the arrow keys, make the alterations necessary, and press <ENTER>.

While changing a line, the <CHAR INS> and the <CHAR DEL> keys may be used for insertion and deletion of characters on that line.

<ENTER> may be pressed when cursor is anywhere on the line, it does not necessarily need to be at the end of the line.

The <LINE INS> and <LINE DEL> keys enable lines to be inserted and deleted on the screen but do not cause changes to the file.

4.5. DELETION OF LINES

A line may be deleted by either:

(i) entering the line number with no data following it, or

(ii) by use of the DEL command.

The DEL command may be used to either delete individual lines or a group of lines from memory.

Syntax:- DEL startline [-endline]

The startline must be given. If the -endline is missing, only the startline is deleted.

For example:

DEL 280

deletes line 280.

DEL 280 - 1000

deletes lines 280 to 1000 inclusive.

NOTE that the following forms are NOT allowed:

DEL 280-

or

4.6. RENUMBERING OF LINES

At times, all available line numbers in a particular sequence may have been used. Alternatively, due to a large number of insertions and deletions the line numbers may be badly distributed. In both these cases, it is advisable to use the RENUM command to renumber the file.

Syntax:- RENUM [startline] [,increment]

Renumbering a BASIC file not only changes the line numbers but also changes all references to them in GOTO, GOSUB and other statements. RENUM may also be used to renumber part of a file (see the description of the RENUM command in the POLYBASIC manual).

Renumbering a TEXT file only changes the line numbers.

The startline is the first line number allocated. If not given, 10 is used.

The increment is the amount added to each succeeding line number. If not given, 10 is used.

For example:

```
RENUM
```

renumbers the file from line 10, in increments of 10, i.e. the new line numbers are 10, 20, 30, 40 ...

```
RENUM 100
```

renumbers the file from 100 in increments of 10, i.e. the new line numbers are 100, 110, 120, 130 ...

```
RENUM ,100
```

renumbers the file from 10 in increments of 100, i.e. the new line numbers are 10, 110, 210, 310 ...

```
RENUM 1000,100
```

renumbers the file from 1000 in increments of 100, i.e. the new line numbers are 1000, 1100, 1200 ...

4.7. SAVING THE EDITED FILE ON DISK

At any stage during editing, the file may be saved using the SAVE command. A BASIC file is saved with line numbers, a text file is saved without line numbers.

Syntax:- SAVE "filename"
SAVE

The filename may specify the extension and the drive number.

For example:

```
SAVE "0.MYFILE.TXT"
```

If the drive number is not given then the file is written to the current drive for that POLY.

If the extension is not given then a BASIC file is given the extension .BAS and a TEXT file the extension .TXT.

Following a SAVE, the file is still in the POLY memory and further editing may be performed.

For example:

```
SAVE "MYFILE"
```

If the POLY is in TEXT mode and the current drive is 0, then the file will be saved on drive 0 as MYFILE.TXT.

SAVE may be used without a file name if the file has been previously LOADed from disk. In this case the user will be prompted with

```
Save filename (Y/N) ?
```

where filename is the name of the file that was LOADed.

4.8. LOADING FILES FROM DISK

A file stored on disk is loaded into POLY memory using the LOAD command. This clears any program or file currently in POLY memory, and loads the file from disk.

Syntax:- LOAD "filename"

The filename may specify the drive number and the extension.

For example:

```
LOAD "1.MYFILE.BAS"
```

will load MYFILE.BAS from the disk in drive 1.

If the drive number is not given, then the file is loaded from the current drive for that POLY.

If the extension is not specified then .BAS is used in BASIC mode and .TXT in TEXT mode.

When a TEXT file is loaded, line numbers are added, starting at 10 and incrementing in steps of 10.

For example:

```
LOAD "MYFILE"
```

If entered on a POLY with the current drive as 1 and in TEXT mode, then the file 1.MYFILE.TXT will be loaded into the POLY memory, starting at line 10 and incrementing in steps of 10.

4.9. MERGING FILES ON DISK WITH THE FILE BEING EDITED

The MERGE command merges a file from disk into the file currently being edited. BASIC files are merged on line number such that where the same line exists in both files, the new line replaces the old line.

In TEXT mode, the disk file is appended onto the end of the file being edited and line numbers above those currently in use are allocated.

Syntax:- MERGE "filename"

The filename may specify the drive number and the extension.

If the drive number is not given, then the file is loaded from the disk on the current drive for that POLY.

If the extension is not specified then, for BASIC .BAS is assumed, and for TEXT, .TXT is assumed.

For example:

If a POLY (in BASIC mode) contains the following file:

```
10 CLS
20 FOR row = 0 TO 10
30 PRINT @(row,0) "11Q"
40 NEXT row
```

and the file MYFILE.BAS on disk contains:

```
30 PRINT @(row,0) "  R";
50 REM DRAW A CAR
60 REM etc...
```

then when the command:

```
MERGE "MYFILE"
```

is entered, the resulting file in the POLY will be:

```
10 CLS
20 FOR row = 0 TO 10
30 PRINT @(row,0) "  R";
40 NEXT row
50 REM DRAW A CAR
60 REM etc...
```

If a POLY (in TEXT mode) contains the following file:

```
100 THIS IS A TEXT FILE
200 CONTAINING ONLY
300 3 LINES
```

and the file MYTEXT.TXT contains:

```
THIS IS MYTEXT
FILE WHICH HAS
ONLY 3 LINES
```

then following the command:

```
MERGE "MYTEXT"
```

the POLY file becomes:

```
100 THIS IS A TEXT FILE
200 CONTAINING ONLY
300 3 LINES
310 THIS IS MYTEXT
320 FILE WHICH HAS
330 ONLY 3 LINES
```

4.10. DELETING THE FILE BEING EDITED

The NEW command deletes the file currently being edited from memory.

For example:

```
NEW
```

If the file being edited has not been changed the Ready prompt will appear on the screen. If the file has been changed since the last SAVE the user will be prompted with

```
Save (Y/N) ?
```

or

Save filename (Y/N) ?

The filename will only appear if the file was LOADed. In the first case if Y is typed, the NEW is aborted; if N is typed, the NEW is executed. In the second case, if Y is typed the file will be SAVEd and NEW executed; if N is typed, NEW will be executed. Only Y,y,N or n will be accepted.

5. USING THE PRINTER

The printer in a POLY system is shared by all the POLYs. All print requests are queued onto disk before printing. There are several ways of having information printed.

5.1. PRINTING EXISTING FILES FROM DISK

These are printed using the DOS PRINT command in the form

```
PRINT filename
```

They may be removed from the queue, even after printing has started by entering

```
PRINT -filename
```

For further details, see PRINT in this manual.

5.2. PRINTING DOS UTILITY OUTPUT

DOS commands normally print messages on the POLY screen. These may be printed instead by using the WTD (Write To Disk) command in conjunction with the DOS utility command.

For example:

```
WTD + CAT 0
```

will print the catalog on the printer using the file SPnnnnnn.PRT as the intermediate file, where nnnnnnn is a unique 6 digit number

or

```
WTD filename.PRT CAT 0
```

where a specific filename is given as the print queue file name. Any file that is created with .PRT extension is automatically printed and then deleted.
Further details of WTD are given under the description of WTD in this manual.

5.3. PRINTING FROM A PROGRAM

To print from a BASIC program use the LPRINT command (see the POLYBASIC manual). Output from LPRINT will be printed when the program stops executing. It is also possible to print from a program by creating a disk file with a .PRT extension. When the CLOSE of that file is issued, that file is automatically printed and deleted. This method may be used from both BASIC and PASCAL programs. If it is necessary to retain the file, it may be created with any other extension, and the DOS PRINT command used to print it out.

6. SOFTWARE INTERRUPT FUNCTIONS.

The number on the left-hand-side of each description below is the "Software Interrupt Function number". The BASIC function "SWI(number,parameter)" calls the relevant software interrupt function. This function is performed before the program continues. The PASCAL manual contains details of calling software interrupts for PASCAL programs.

For example:

To check the status of the keyboard, insert the code

```
X%=SWI(0)
```

at the appropriate position in your program.

Some software interrupt functions require one or more parameters to be specified. Values returned by software interrupts may be used as normal function values.

For example:

```
X%=SWI(3,1000,100)
Y%=SWI(1)
```

The first call does not return a value. The second returns a value which is assigned to the integer variable Y%.

0. Check Status of Keyboard.

This function checks to see if a key has been pressed on the keyboard.

Input Parameters: None
Value Returned:
If SWI(0) AND 255 = 1 a character is waiting for the first time.
If SWI(0) AND 255 = 128 the key is still depressed.
If SWI(0) AND 255 = 129 a character is waiting for the first time and the key is still depressed.

1. Input Single Character.

This function turns on the cursor and waits until a character is typed on the keyboard.

Input Parameters: None
Value Returned:
The ASCII value of the key pressed (this is in the 2nd byte, the first byte remains unchanged).
In BASIC, INCH$(0) accomplishes the same task.

2. Line Edit.

The line editor must first be initialised, otherwise it will not function properly. To do this first put the current teletext screen into ASCII mode by: ' PRINT" N"; '. Then call SWI 2 with the following parameters:

Input Parameters

Parameter 1:

Byte 1 - If zero then when the <ENTER> key is pressed (indicating End-Of-Line) the buffer is returned beginning from the start of the buffer to the cursor position. If byte 1 is non-zero, then the buffer returned is from the start of the input buffer to the end of input buffer, no matter where the cursor is positioned.

Byte 2 - must contain 31 (hex $1F) to initialise the editor.

Parameter 2: The start address of user's input buffer.

Parameter 3: The maximum number of characters allowed (or the maximum buffer length).

Value Returned: None

The line editor is now ready to accept input characters. The input character can be program generated or input from the keyboard using SWI 1 (i.e. INCH$(0)).
When a character is input the line editor will do the appropriate function e.g. insert character, delete character, cursor left or cursor right.
When the <ENTER> key (13 or $0D) is input to the editor, the edit (or input)is complete, and the input buffer is copied into the user's buffer, whose address was specified in parameter 2 of the initialisation SWI call.
The length of the buffer will be returned via parameter 3.

An example in POLYBASIC is:

```
 10 CLS
 20 REM Ensure teletext screen is in ASCII mode
 30 PRINT@(10,5)" N";
 40 REM Buffer$ is the user's buffer
 50 Buffer$=STRING$(20)
 60 REM Ba% is the address of the user's buffer
 70 Ba%=DPEEK(PTR(Buffer$))
 80 REM Initialise editor to return whole
 90 REM buffer to address Ba%, and maximum
100 REM number of characters to input is 20.
110 Z%=SWI(2,256+31,Ba%,20)
120 REM Get a character from the keyboard
130 A%=ASC(INCH$(0))
140 REM Input the character to the line editor
150 Z%=SWI(2,A%)
160 REM If the character wasn't ENTER then go back for another input
170 IF A%<>13 THEN 130
180 REM The key was ENTER, so now the input buffer has been
185 REM copied into the user's buffer.
190 CLS
200 REM Print the user's buffer which now contains the
205 REM edited line.
210 PRINT@(5,0) Buffer$
220 END
```

3. Sound Generator.

This function generates sound from the speaker. the pitch is calculated as (502400 / frequency) -1. The length is specified in 10 millisecond lengths.

Input Parameters:
Parameter 1: The pitch of the sound.
Parameter 2: The length in 10 millisecond intervals.
Value Returned
None.

4. Reserved For System Use.

5. Put Character

This function prints a character to the current teletext screen.

Input Parameters:
Parameter 1: The second byte contains the character to be printed on the screen.
Value returned: None

6. Write Character to Specified Position

This function prints a character to a specified position of the current teletext screen. It does not affect cursor position.

Input Parameters:
Parameter 1: The second byte contains the byte to be printed.
Parameter 2: Not used.
Parameter 3: Contains the print position in teletext row, column format. e.g. To specify row 2, column 3 use 2*256+3
Value Returned: None

7. Read the Keyboard

This function reads the keyboard. If a key has been pressed, it will return the ASCII character, otherwise a null (CHR$(0)) will be returned. This is essentially the same as INCH$ in BASIC.

Input Parameters: None
Value Returned: The second byte contains the ASCII value of the key pressed, otherwise null. The first byte is unchanged.

8. Copy From Screen to a String

This function copies a string of characters from the current teletext screen to a memory area specified by the user. Nulls in the string will be converted to spaces (hex 20).

Input Parameters
Parameter 1: Start address of string (not above hex E000)
Parameter 2: Length of string to be copied.
Parameter 3: User specified position in row, column format
Value returned: None
If the user tries to copy beyond the end of the teletext screen (i.e. row 23, column 39), the carry bit of the condition code will be set. In this case the string is copied up to the end of the current teletext screen.

9. Set Cursor Position

This function sets the cursor to a specified position on the current teletext screen. If the split screen is active then this function allows the user to move the cursor to the other portion of the split screen. If the input cursor is invalid (i.e. beyond row 23, column 39) then the cursor will not be moved.

Input Parameters:
Parameter 1: Contains cursor position in row, column format. The first byte is the row, and the second byte the column (e.g. 2*256 + 5 sets the cursor to row 2 column 5).
Value Returned: None
If an invalid cursor format is specified then the carry bit of condition code is set on return.

10. Set Relative Cursor Position on Current Teletext Screen.

This function moves the cursor by the specified number of positions, relative to its current location.

Input Parameters:
Parameter 1: A number in the range -23 to +23
which is the number of rows that the cursor is to be moved.
Positive is down, negative is up.
Parameter 2: A number in the range -39 to +39
which is the number of columns that the cursor is moved.
Positive is to the right, negative is to the left.
Value Returned:
None.

11. Read Cursor Position

This function reads the cursor position of the current teletext screen. It does not affect the cursor position.

Input Parameters: None
Value Returned:
The first byte contains the row and the second byte contains the column number of the cursor position.

12. Read Cursor Character

This function reads the character on the current teletext screen at the current cursor position.

Input Parameter: None
Value Returned:
The first byte is always null. The second byte contains the ASCII value of the character read.

13. Split Screen Into Two Portions

This function splits the current teletext screen into two portions, one above the other. The two portions are to all intents and purposes independent of each other, in particular, scrolling on one portion is independent of scrolling on the other, and the cursor remains in one portion until specifically moved to the other using PRINT@ or SWI(9) or SWI(10).

Input Parameters:
The first byte contains the start row of the second half of the split screen. (If =0 or >23 then the split screen is turned off §i.e. reset †)
Value Returned: None

14. Clear Teletext Screen

This function is used to clear one of the two teletext screens. Clearing the current teletext screen can be accomplished by printing a character 12 (Hex $0C, Home) to the screen using SWI(5).

Input parameters:
Only the second byte is used.
If the number >0 and <127 Clear screen 3
=0 Clear current screen
>128 and <=255 Clear screen 1
Value Returned: None

15. Set Screen and Display Characteristics.

This function is used to set teletext and graphics characteristics as follows:

Input Parameters:
Parameter 1: A 16 bit integer where each bit is set or reset according to the functions required above.
Parameters Returned:
None.
bit 15 - not used
bit 14 - mix/priority bit: 1=mix, 0=priority
bit 13 - 1=Display 2 (Graphics 1 screen)
bit 12 - 1=Select 5, 0=Select 2
See Bit 13 for displaying these.
bit 11 - 1=Display 1 (Teletext 1)
bits 10 & 9 - Select screen 2 mix colour
00=blue, 01=green, 10=red,11=none
bits 7 & 8 - not used

bits 6 5 & 4 - Select Background colours
i.e.
bit 6 - 1= BLUE
bit 5 - 1= GREEN
bit 4 - 1= RED
bits 3 & 2 - Select Screen 4 mix colour as for Screen 2 (See bits 10 and 9)
bit 1 - 1=Display 4
bit 0 - 1=Display 3

16. Read Display Mode.

This function reads the current screen display mode.

Input Parameters:
None.
Value Returned:
A 16-bit integer defined as follows:
bit 15 - not used
bit 14 - mix/priority bit: 1=mix, 0=priority
bit 13 - 1=Display 2 (Graphics 1 screen)
bit 12 - 1=Select 5, 0=Select 2
See Bit 13 for displaying these.
bit 11 - 1=Display 1 (Teletext 1)
bits 10 & 9 - Select screen 2 mix colour
00=blue, 01=green, 10=red,11=none
bits 7 & 8 - not used
bits 6 5 & 4 - Select Background colours
i.e.
bit 6 - 1= BLUE
bit 5 - 1= GREEN
bit 4 - 1= RED
bits 3 & 2 - Select Screen 4 mix colour as for Screen 2 (See bits 10 and 9)
bit 1 - 1=Display 4
bit 0 - 1=Display 3

17. Set The Clock

This function allows the user to change the current value of the clock. The clock is initially programmed to interrupt every one second. The user can use this function to stop the clock or restart it. (Warning-- Sound generation is dependent on the clock, if the clock is turned off the sound will not work).

Input Parameters:
Parameter 1: Only the second byte is used. If zero then stop the clock otherwise run the clock as specified by the user.
Parameter 2: The first byte contains the most significant byte of a 3 byte value of time (in 10 millisecond units) after midnight.
Parameter 3: Contains the least significant two bytes of a 3 byte value of time (in 10 millisecond units) after midnight.
Value Returned: None

18. Return The Time

This function returns the current time in 10 millisecond units. If the clock is not running the carry bit of condition code is set and nothing is returned.

Input Parameters: None
Value Returned:
Parameter 1: The least significant 2 bytes of a 3 byte value of time (in 10 millisecond units) after midnight.
Parameter 2: The second byte contains the most significant byte of a 3 byte value of time (in 10 millisecond units) after midnight.
(In BASIC only one parameter is returned.)

19. WAIT Routine

This function waits for a period of time specified by the user. It is done by sampling the clock until it reaches the specified time. Therefore the clock must be running otherwise the carry bit of condition code is set.

Input Parameters:
Parameter 1: A 16 bit value of time (in 10 millisecond units) to wait.
Value Returned: None

20 to 22 Reserved For System Use.

23. Read Through The Serial Port

This function allows the user to read a string of characters through the RS232 port. Either all characters up to and including a specified character are read, or a specified number of characters are read and stored in a string in the user area. A time out will occur if no character is received for approximately one second.

Input Parameters:
Parameter 1: If negative, the absolute value is the count of characters to be read. Otherwise, this is the ASCII value of the delimiter character denoting the end of the input string.
Parameter 2: The address of the memory location into which the characters are to be stored.
Value Returned: The number of characters actually input. If negative or zero a timeout has occurred (the absolute value represents the number of characters input before the timeout).

24. Write Through The Serial Port

This function allows the user to output a string of characters through the RS232 port. Either all characters up to and including a specified character are output or a specified number of characters are output from a string in

the user area. A timeout will occur if the serial port is not ready for transmission for approximately one second.

Input Parameters:
Parameter 1: If negative, the absolute value is the count of characters to be output. Otherwise, this is the ASCII value of the delimiter character denoting the end of the output string.
Parameter 2: The address of the memory location from which characters are to be output.
Value Returned: The number of characters actually output. If negative or zero, a timeout has occurred (the absolute value represents the number of characters output before the timeout).

Note: When using the RS232 port it is necessary to
(i) initialise it (i.e. write 3 to memory location $E004),
(ii) set the baud rate (i.e. write an appropriate number to memory location $E006 - see Software Interrupt 25), and
(iii) set the word structure (i.e. write an appropriate number to memory location $E004 - see Software Interrupt 25).

For example:

```
 10 scratch% = SWI(47,3,HEX("E004"),0)
 20 scratch% = SWI(47,0,HEX("E006"),0)
 30 scratch% = SWI(47,17,HEX("E004"),0)
 40 buffer$ = STRING$(255)
 50 bufpnt% = DPEEK(PTR(buffer$))
 60 ret% = SWI(23,13,bufpnt%)
 70 IF ret%<=0 THEN 110
 80 ret% = SWI(24,-ret%,bufpnt%)
 90 IF ret%<=0 THEN 110
100 GOTO 60
110 PRINT"TIMED OUT"
120 STOP
```

This program will read characters through the RS232 port until a carriage return is encountered. The string is then echoed out the RS232 port (using the character count). The initialisation in lines 10,20,30 sets the port up for 9600 baud, with a word consisting of 8 bits and 2 stop bits.

25. Select Terminal Mode

This function allows the POLY to act as a dumb terminal. Characters input via the keyboard are transmitted through the serial port. Characters received at the serial port are displayed on the screen.

Input Parameters:
Parameter 1: A number corresponding to the baud rate as follows:

0 = 9600
2 = 4800
4 = 2400
6 = 1200
8 = 600
10 = 300

Parameter 2: The word structure (i.e. word size, parity on/off). The bits specifying this are 000xxx01 so this will be a number between 1 and 29 as follows:

1 = 7 bits + even parity + 2 stop bits
5 = 7 bits + odd parity + 2 stop bits
9 = 7 bits + even parity + 1 stop bit
13 = 7 bits + odd parity + 1 stop bit
17 = 8 bits + no parity + 2 stop bits
21 = 8 bits + no parity + 1 stop bit
25 = 8 bits + even parity + 1 stop bit
29 = 8 bits + odd parity + 1 stop bit

Value Returned: None.

26. Select Standard Memory Map 2

This function selects the predefined memory map 2. This map contains the 64k RAM addresses from $0000 - $FFFF. The two graphics screens are also included in this memory map. The address of the first graphics screen occupies $E000 - $FFFF, the second graphics screen occupies $8000 - $9FFF. The operating system occupies $C000 - $DFFF. To display the graphics screen see SWI(15) and SWI(16).

Input Parameters: None.
Value Returned: None.

27. Switch To Memory Map 1

This function allows the user to switch from memory map 2 to memory map 1. If the user is already in map 1 nothing will change. The only difference between memory map 1 and standard memory map 2 is that in map 1 addresses from $A000 - $BFFF and $E000 - $FFFF are the 16k BASIC ROMs while in map 2 these addresses are 16k of RAM.

Input Parameters: None
Value Returned: None

28. Switch To Memory Map 2

This function allows the user to switch from memory map 1 to memory map 2 (non standard). It is usually called after SWI(29) which changes the configuration of map 2 but does not switch to it.

Input Parameters: None
Value Returned: None

29. Change Configuration of Memory Map 2

This function uses the user specified address translation table to configure memory map 2. The address translation table is written into the system dynamic address translator (DAT). This configuration does not change until the next call to this function or until the system is reset. In the latter case the configuration is the same as standard memory map 2. (Note: the BASIC interpreter uses a different configuration from the standard map 2, therefore one cannot assume that memory map 2 is always the same as the predefined memory map 2.) This function does not switch the current memory map to memory map 2, to do this use SWI(28).

Input Parameters:
address of the 8 byte translation table.
Value Returned: None

30. Select Current Teletext Screen

This function selects one of the two teletext screens. Once it is selected, all references to a teletext screen (e.g. print a character, read a character etc.) will refer to this screen. One of the two screens will be current at any one time. This function only selects the current teletext screen, it does not display it. (See SWI(15) and SWI(16) to display teletext and graphics screens.)

Input Parameter:
If the second byte contains a 2, the second teletext screen (Screen 3) is selected, otherwise the first teletext screen (Screen 1) is selected.
Value Returned: None.

31. Select 24-Line Display for Text Screens.

Calling this function sets both text screens to display 24 lines.

Input Parameters:
None.
Value Returned:
None.

32. Select 12-Line Display for Text Screens.

Calling this function sets both text screens to display 12 lines.
Functions 33 & 34 determine whether the top 12, or bottom 12 lines are to be displayed.

Input Parameters:
None.
Value Returned:
None.

33. Display First 12 Lines.

When used in conjunction with 32 above, this function displays the top 12 lines (lines 0-11) on each text screen.

Input Parameters:
None.
Value Returned:
None.

34. Display Last 12 Lines.

When used in conjunction with 32 above, this function displays the bottom 12 lines (lines 12-23) on each teletext screen.

Input Parameters:
None.
Value Returned:
None.

35. Set Scroll Mode On Teletext Screen

The teletext screen can operate under scroll mode or wrap mode. In scroll mode, when the cursor reaches the end of the screen, the whole screen is scrolled up one line and the cursor is positioned at the start of the bottom line. In wrap mode, the cursor wraps around to the top of the screen, and there is no scrolling. This function allows the user to set both teletext screens to scroll or wrap mode.

Input Parameters:
If the second byte is zero, both teletext screens are set to scroll mode. Otherwise both teletext screens are set to wrap mode.
Value Returned: None

36. Reserved For System Use

37. Test EXIT Flag

This function returns the <EXIT> key flag. If the <EXIT> key has been pressed the value returned is not zero. On return the <EXIT> key flag is always cleared to null.

Input Parameters: None.
Value Returned:
The first byte is unchanged. The second byte contains the <EXIT> key flag. If non zero, the <EXIT> key has been pressed.

38 to 42 Reserved For System Use

43. Send Message to Master

This function is used to send a message to the network controller. There should be no need for a user to call this function, as all message handling is done by the operating system.

Input Parameters:
Parameter 1: The second byte contains the message type.
Parameter 2: Start address of message.

Parameter 3: Length of message.
Value Returned: None.

The carry bit of the condition code is set if an error occurs in the communication system.

44. Receive A Message From Master

This function is used to receive a message from the network controller. Normally the master does not send messages to the POLY unit unless the POLY has requested some information from the master.(e.g. read a sector from disk). As in the case of SWI(43) the user should not need to call this function.

Input Parameters:
Parameter 1: Not used.
Parameter 2: Start address of message to be received.
Parameter 3: Maximum length of message allowed.
Values Returned:
Parameter 1: The second byte contains the message type.
Parameter 2: Not changed.
Parameter 3: Length of message actually received.

If there is a communication system error or no message is forthcoming, then an error has occured and the carry bit of the condition code is set.

45. Log Off

This function returns the user to the log on screen.

Input Parameters: None.
Value Returned: None.

46. Read System Input/Output

This function allows the user to read the status and data registers of various peripherals attached to the POLY.

Because all input/output ports appear only in the system memory map mode (which is protected), it is necessary to use this function to read these ports e.g. PIA, TIMER.

Input Parameters:
Parameter 1: If the second byte is zero then one byte is read, otherwise 2 bytes are read.
Parameter 2: Address to read from (accessible ranges are $E000-$E02F, $E800-$EFFF).
Value Returned:
The value read. If only one byte is read, then it will be in the low order byte and the high order byte will be unchanged.

47. Write System Input/Output

This function allows the user to write to the control and data registers of various peripherals attached to the POLY.

Input Parameters:

Parameter 1: The 1 or 2 bytes of data to be written to the specified address.

Parameter 2: Address to write to (accessible range is $E000-$E02F)

Parameter 3: Zero indicates one byte is to be writen, otherwise 2 bytes will be written.

Value Returned: None.

6.1. MEMORY MAPPING

The 6809 micro-processor is capable of addressing 64K bytes of memory. However a feature called Dynamic Address Translation has been incorporated into the POLY which allows more than 64K to be used.

This feature allows the user to switch into addressable memory any of 8 blocks of 8K memory, from a maximum of 16 blocks. These may be placed in any order in any of the 8 positions in addressable memory. Two memory maps are available, and the user indicates which is currently in use.

Memory map 1 is fixed at initialisation of the system, but the user may alter memory map 2 to suit an application by using software interrupt 29. The selection of a memory map is made using software interrupts 27 and 28.

The physical memory in the POLY is allocated as follows

Blocks 10 to 15	Not present
Blocks 8 and 9	16K RAM bank 4
Blocks 6 and 7	16K BASIC ROM
Blocks 4 and 5	16K RAM bank 3 Block 4 contains screen 4
Blocks 2 and 3	16K RAM bank 2 Block 2 contains screen 2
Blocks 0 and 1	16K RAM bank 1

The System ROM and Teletext screens are within a protected area which is not accessable directly by the user. These may only be accessed by software interrupts.

Memory map 1 is configured as follows

Starting address	Physical block	Contents
$E000	7	BASIC ROM
$C000	5	DOS RAM
$A000	6	BASIC ROM
$8000	4	Screen 4 or user RAM
$6000	8	User RAM
$4000	3	User RAM
$2000	1	User RAM

$0000	0	BASIC disk extensions up to approximately $1B00

Memory map 2 may vary depending on the last use of it.

To setup and call a special memory map 2, carry out the following steps.

1. Set up a string of 8 bytes, each byte containing the physical block number to be put into the map, starting from address 0.

2. Call software interrupt 29 giving the address of the 8 bytes as parameter 1.

3. Call software interrupt 28 to select memory map 2.

For example:

In BASIC

```
100 A$=CHR$(0)+CHR$(1)+CHR$(3)+CHR$(8)+CHR$(2)+CHR$(6)+
CHR$(5)+CHR$(7)
110 A%=SWI(29,DPEEK(PTR(A$)))
120 A%=SWI(28)
```

This puts screen 2 in place of screen 4 in addressable memory.

BASIC graphics commands should not be used while using memory map 2, as these use a special memory map 2 and will overwrite the memory map defined by the user.