APPENDIX A EXAMPLES

1. Programs Using the RS232 Port.

```
5REM*********************************
10REM This is an example of using the
15REM serial port of the Poly. This
20REM program reads 960 characters
25REM from the screen and outputs them
30REM to the serial port.
35REM
40REM The status register address
45REM  is E004 HEX.
50REM The control register address
55REM  is E004 HEX.
60REM The data register address is
65REM  E005 HEX.
70REM The programmable baud rate
75REM  address is E006 HEX.
80REM
85REM For port details see Motorola
90REM  6850 ACIA description.
95REM*********************************
100REM Master reset of serial port
110scratch%=SWI(47,3,HEX("E004"),0)
120REM Set to divide ratio of 16
130scratch%=SWI(47,1,HEX("E004"),0)
140REM Set baud rate to 9600 bit/sec
150scratch%=SWI(47,0,HEX("E006"),0)
160REM Get 960 characters from the screen
170screen$=TEXT$(0,0,960)
180REM Output each character of the string
190FOR loop%=1 TO 960
200char%=ASC(MID$(screen$,loop%,1))
210REM Lines 150-160 check for the
220REM  port Clear-To-Send signal.
230busy%=SWI(46,0,HEX("E004"),0) AND 2
240IF busy% <> 2 THEN 230
250REM Write the character to the
260REM  data register.
270scratch%=SWI(47,char%,HEX("E005"),0)
280NEXT
10REM*********************************
20REM This program uses software
30REM interrupt 24 to output a
40REM complete file, line by line.
50REM*********************************
60REM Master reset of I/O port
70Z%=SWI(47,3,HEX("E004"),0)
80REM Set to divide ratio of 16
90Z%=SWI(47,1,HEX("E004"),0)
100REM Set baud rate to 9600 bit/sec
110Z%=SWI(47,0,HEX("E006"),0)
120OPEN OLD "PRIMES.TXT" AS 1
130ON END #1 GOTO 230
140Z%=SWI(47,12,HEX("E005"),0)
150INPUT LINE #1,Z$
160REM Output the line
170Z%=SWI(24,-LEN(Z$),DPEEK(PTR(Z$)))
180REM Output a carriage return
190Z%=SWI(47,13,HEX("E005"),0)
200REM  and a line feed
210Z%=SWI(47,10,HEX("E005"),0)
220GOTO 150
230CLOSE 1
```

2. Sharing a File Open for Write.

The following programs provide a simple mailbox system for a network of 10 POLYs.

This program must be run to set up the mailboxes (files MAIL1, MAIL2, ... MAIL10).

```
10REM Set Mailboxes for 10 Polys
20FOR poly%=1 TO 10
30 num$=STR$(poly%)
40 num$=RIGHT$(num$,LEN(num$)-1)
50 OPEN NEW RANDOM "MAIL"+num$ AS 1
60 FIELD #1,252 AS buf$
70 LSET buf$="#"
80 FOR rec%=1 TO 10
90   PUT#1,RECORD rec%
100 NEXT rec%
110 CLOSE #1
120NEXT poly%
130END
```

The following program may be run in order to send and receive messages.

```
10REM Mailbox program
20INPUT "I am POLY number"; poly%:REM To identify the POLY
30IF poly%<0 OR poly%>10 THEN 20
40INPUT"(S)end, (R)eceive or (Q)uit";a$
50IF a$="S" OR a$="s" THEN GOSUB 200
60IF a$="R" OR a$="r" THEN GOSUB 400
70IF a$="Q" OR a$="q" THEN END
80GOTO 40
200REM Send routine
210INPUT"To Poly";op$
220IF VAL(op$)<0 OR VAL(op$)>10 THEN 210
230IF VAL(op$)=0 THEN RETURN
240ON ERROR GOTO 1000
250OPEN OLD RANDOM "MAIL"+op$ AS #1
260FIELD #1,252 AS buf$
270LOCK #1
280GET #1,RECORD poly%
290IF LEFT$(buf$,1)="#" THEN 320
300PRINT"Poly #";op$;" hasn't read your last message."
310GOTO 370
320INPUT LINE(252)inp$
330LSET buf$=inp$
340PUT #1,RECORD poly%
350UNLOCK #1
360ON ERROR GOTO 0
370CLOSE 1
380RETURN
400REM Receive routine
410us$=STR$(poly%)
420us$=RIGHT$(us$,LEN(us$)-1)
430ON ERROR GOTO 1000
440OPEN OLD RANDOM "MAIL"+us$ AS #1
450FIELD #1,252 AS buf$
460LOCK #1
470flag%=-1
480FOR snder%=1 TO 10
490  GET #1,RECORD snder%
500  IF LEFT$(buf$,1)="#" THEN 560
510  PRINT"From ";snder%;":";buf$
520  LSET buf$="#"
530  PUT #1,RECORD snder%
540  flag%=0
550  WAIT 100
560NEXT snder%
570IF flag% THEN PRINT"No messages!"
580ON ERROR GOTO 0
590CLOSE 1
600RETURN
1000IF ERR=34 THEN RESUME
1010STOP
```

APPENDIX B VERSION 2 ENHANCEMENTS

(This appendix also includes Version 1.8 enhancements not previously documented.)

Operating System

1. Multi-user file control system.
2. Extensive error-checking and upgraded error messages.
3. RS232C serial port software including a terminal option.
4. Password protection of files.
5. Easy modification of the print spooler to use other printers.
6. EXIT key works immediately.

Utilities

The following utilities have been added or upgraded. Refer to the Version 2 POLYSYS Utilities Manual for full details. (Note also that lower case file extensions are now converted to upper case.)

1. TEXT - allows a file to be loaded.
2. PROT - sets P(assword) protection.
3. WTD - generates print spooler files.
4. PRINT - provides more options.
5. KILL - prompts for deletion.
6. GPRINT - prints graphics screens.
7. SDC - handles files larger than POLY memory.
8. DRIVE - sets default drive from DOS.
9. RECOVER - recovers lost sectors.
10. DATE - sets date in network controller (after reset).
11. COPY, PCOPY - more efficient.
12. CAT - prints P(assword) protection.
13. OVERRIDE - to over-ride P(assword) protection (supervisor only.)

Pascal

The order and format of error messages has been changed, providing more clarity. As well, modifications were made to allow Pascal compilation from drives other than 0.

Basic

General

1. Short error messages replace the error numbers. The <HELP> key provides a more detailed description of the error.

2. String allocation is more efficient.

3. Arithmetic operations are up to three times more efficient.

4. "Ready" and error messages clear the rest of the line and next line.

5. Certain commands (DEL, RENUM, AUTO) are not usable from program mode (for obvious reasons).

6. Random files now utilise the multi-user file control systems.

7. Prompts to save a file being edited are made before the file is destroyed.

8. All checks for the disk being booked for write have been removed (no longer necessary with the multi-user file control system).

Commands and Functions

The following commands and functions have been added or upgraded. Refer to the Version 2 POLYBASIC Manual for full details.

AUTO	CLOSE	DATE$
DEL	DRAW	DRIVE
FIELD	FILE$	FILL
GET	GOSUB	GOTO
INPUT LINE	LOAD	LOCK
LPRINT	MERGE	MID$
NEW	ON END	ON KEY
OPEN	RENUM	RESOFF
RESON	SAVE	SOUND
STORE	TEXT	TIME$
UNLOCK	VAL	

APPENDIX C ADVICE

1. Corrupted Disks - Prevention and Recovery

Corrupted disks can result from files being opened for write and not subsequently closed. If this happens the structure of the free space on the disk is corrupted and subsequent use of the disk may cause problems.

To prevent the corruption of disks, remember the following rules.

(i) Before switching off or resetting a POLY, either LOGOFF or enter CLEAR from BASIC.
(ii) If your BASIC program ends with an error or stops for any reason with files open, execute CLEAR to close all files (if you need to examine variables for debugging purposes do so first).
(iii) Do not reset or switch off the disk unit until all POLYs have closed their files as above.
(iv) Do not remove a disk from the disk unit, or even open the door unless you are sure that all users have closed all the files they have open on that disk.

N.B. A file is open for write if is is opened NEW or RANDOM. A file is also open for write if LPRINT is used (though this is automatically closed if an error occurs) and when SAVE is being executed. Many utilities have files open for write as well so execution of these should not be interrupted.

The RECOVER utility should be used to clean up a disk under any of the following conditions.

(i) If any of the above rules are broken.
(ii) If a disk door opened error is given (the disk which was in the drive when the door was opened should be RECOVERed).
(iii) If, after executing the CAT command, the "SECTORS=" number (the number of sectors used by the files) plus the "FREE=" number (the number of sectors not used by the files) do not add up to 2280 for a double-sided disk or 1140 for a single-sided disk. Note that even if these numbers do add up to 2280 or 1140 as appropriate, the disk is not necessarily uncorrupted. Also note that sometimes when a disk is FORMATted, the "TOTAL SECTORS=" number may be less that 2280 or 1140 as appropriate due to physically crashed sectors (see below). We suggest that you throw away such disks.

2. Reset Button

Caution must be exercised when turning on or resetting (either normal or warm - see the POLYBASIC Manual, page 26) a POLY that is part of an active network. While the reset button is being depressed and for one second after it is released all communications through a POLY are suspended. They are also suspended for one second after a POLY is turned on. This is to allow stabilisation of the circuitry. During this time, POLYs on the daisy chain that are further away from the disk unit than the POLY being reset or turned on, will be invisible to the disk unit. If such POLYs are performing disk accesses (either read or write) they may occasionally hang-up (because they do not receive correct acknowledgement to their requests). If such a hang-up occurs during a disk write, then the disk being written to should be RECOVERed. To prevent this happening

(i) ensure that all POLYs on a network are switched on at the outset,

(ii) use LOGOFF rather than reset, and

(iii) if it is necessary to reset, check what others on the network are doing and don't depress the reset button for longer than necessary.

3. Crashed Disks

Disks may crash for a variety of reasons - physical damage, wear, dirt, power surges, etc. Normally such disks will contain one or more unreadable sectors and access will result in disk read or disk write errors (errors 9 and 10). Cleaning disk heads every so often may help (cleaning kits are available from POLYCORP) but remember the following rules.

(i) Back up your disks regularly.

(ii) Once you have noticed a disk crash error, do not continue to use the disk.

If you do have a crashed disk then the following procedure will recover as much as possible.

(i) Get a catalog of the crashed disk (if this isn't possible the disk is irretrievable; a partial catalog will allow some files to be recovered).

(ii) Decide which files you want to recover.

(iii) Copy as many of these files as possible, one at a time to another system disk using PCOPY, COPY, or SDC.

(iv) Copying a file containing an unreadable sector will present problems. For example, the copy may stop in the middle, in which case only part of the file will be recovered. Some times the file may have unwanted lines in it and these will have to be edited out. Sometimes, the file may appear

infinitely long and an attempt to copy it will continue for ever (this is a bad one, the network controller will have to be reset to turn off the disk drive so the disk can be extracted and the file deleted).

(v) After copying the wanted files, reformat the crashed disk, mirror the system onto it and if you're sensible, back up your recovered disk.

4. Deadly Embrace

Now that it is possible to share files several precautions must be taken. If a file is opened for write by more than one program, and one of the programs is to get a record, alter it, and then put it back, then the other programs must be denied access to that record while it is being changed. To accomplish this, the updating program LOCKs the whole file (i.e. all records) before it gets the required record and then UNLOCKs the file after it has put the record.

In the simple cases this is all very well, but consider a more complex example.

Program A opens two files, say X and Y, locks X and tries to read Y. In the meantime, program B opens X and Y, locks Y and tries to read X. Using the error trap as in

```
      ON ERROR GOTO 1000

1000  IF ERR=34 THEN RESUME
```

the two programs will wait on each other forever. This is know as "Deadly Embrace" and must be carefully avoided.

5. Security

If you have files you want to protect from accidential or purposeful access by other users, log onto the POLY with your initials and a password that you will remember. Then run the PROT utility specifying the P option and the password will be associated with the file. Future access to the file will only be permitted to users logging on with the same initials and password.

If you forget your password, an OVERRIDE program (with limited distribution) exists so that the password may be removed.

6. Noteworthy

(i) GRAPH is a new reserved word so if you wrote programs with GRAPH as a variable name or GRAPH as part of a variable name, they will have to be altered.

(ii) Previously it was possible to store the contents of the graphics screen(s) into a string, e.g.

```
STORE (0,0),(200,200) A$
```

This string could then be written to a file opened as

```
OPEN NEW PRINT "PICTURE" AS #1
```

by

```
PRINT #1, A$
```

Then, after closing the file and re-opening it as

```
OPEN OLD "PICTURE" AS #1
```

the picture could be redrawn using

```
DRAW #1
```

In actual fact PRINT should not be used to save graphics strings to be later drawn directly on the screen. While possible in version 1.8, it is not possible in version 2.

The file picture should be opened as follows

```
OPEN NEW GRAPH "PICTURE" AS #1
```

The screen may then be saved either by

```
STORE (0,0),(200,200) A$
SAVE #1, A$
```

or directly by

```
STORE (0,0),(200,200) #1
```

STORE and DRAW are opposites with respect to input and output of graphics strings.

To convert graphics strings stored using PRINT to a format suitable for use with DRAW the following program may be used.

```
10 OPEN OLD "Filename" AS 1
20 FIELD #1, 252 AS Buf$
30 ON END #1 GO TO 80
40 GS$=""
50 GET #1
60 GS$ = GS$ + Buf$
70 GO TO 50
80 CLOSE 1
90 IF RIGHT$(GS$,1)=CHR$(0) THEN GS$ =
LEFT$(GS$,LEN(GS$)-1): GOTO 90
100 GS$ = GS$ + CHR$(0)
110 SAVE #"Filename", GS$
```

(iii) Sometimes users wish to load a program from disk into the POLY memory, disconnect the POLY and then run the program. (Clearly, the program cannot use disk files while it is executing.) If this is to be carried out, users are advised to issue a CLEAR command prior to disconnecting the POLY. The reason for this is that all file handling is now carried out in the disk drive/network controller - essential for the multi-user environment. Issuing the CLEAR ensures that the disk drive and the POLY are co-ordinated.

CLEAR may also be used when the error message "Illegal file control block specified" is displayed. This message normally follows a user error when files are accidentally left open. CLEAR ensures that these are closed. It is not done automatically because when debugging, users often wish to interrogate variables, etc.

(iv) With the new random file mechanism, the size of a random file may be calculated as follows.

If the record length is r%, then the number of records per disk sector is the integer value of 252/r% i.e. INT(252/r%).

If the maximum record number you PUT to the file is max%, then the number of sectors allocated to the file is

2 (for the index) + max%/INT(252/r%)

If this value is not an integer, it must be rounded up.

For example, if r%=252 and you PUT record 92, the file will be 94 sectors long. If r%=20 and you PUT record 200, the file will be 19 sectors long.

This is different from Version 1.8 when random files could be longer then the maximum record written. Version 2 random files will thus often be smaller than Version 1.8 random files. However, Version 1.8 random files may still be used with Version 2.

When using random files, it is good practice to allocate a random file large enough to handle your application before writing to the file. Do this by PUTting to the maximum record number.

APPENDIX D VERSION 2 RELEASE

1. Programs

The following is a catalogue of all files on a Version 2 Programming (with Pascal) disk.

The dates are important in identifying the most up-to-date versions of programs.

```
DIRECTORY OF DRIVE NUMBER 1
DISK: POLY #2007 CREATED: 13-JUN-83

   NAME    TYPE  R  SIZE     DATE    PRT

 POLYNET  .SYS        40  13-JUN-83 WD
 POLYSYS  .SYS        16   9-JUN-83 WD
 ERRORS   .SYS  R     43  17-MAY-83 WD
 BASIC    .CMD        44  10-JUN-83 WD
 TEXT     .CMD         1   5-MAY-83 WD
 FORMAT   .CMD         7  25-MAY-83 WD
 DATE     .CMD         2  27-JUL-82 WD
 FASTCOPY.CMD          7   9-JUN-83 WD
 COPY     .CMD         4   9-JUN-83 WD
 SDC      .CMD         7  13-JUN-83 WD
 LINK     .CMD         1  29-JAN-80 WD
 KILL     .CMD         3  16-MAR-83 WD
 PCOPY    .CMD         4   9-JUN-83 WD
 WTD      .CMD         2   9-MAY-83 WD
 PRINT    .CMD         4  23-MAY-83 WD
 CAT      .CMD         5  16-DEC-82 WD
 LIST     .CMD         3  13-JUN-83 WD
 PROT     .CMD         1  12-MAY-83 WD
 RENAME   .CMD         1  29-JAN-80 WD
 DRIVE    .CMD         1  31-JAN-83 WD
 RECOVER  .CMD         7   3-MAY-83 WD
 GPRINT   .CMD        30  10-JUN-83 WD
 MEMTST3  .CMD         3  29-JAN-82 WD
 POLYTEST.BAC         26   5-MAY-83 WD
 VDEOTEST.BAC          2   2-MAR-82 WD
 MENU     .BAC         1   1-MAR-82 WD
 PASCAL   .CMD         4   6-DEC-82 WD
 PC       .CMD       111  15-DEC-82 WD
 RE       .CMD         1  26-MAR-82 WD
 RA       .CMD        88  26-MAR-82 WD
 LL       .CMD        53   8-APR-82 WD
 PASLNK   .RO          1  15-DEC-82 WD
 RL       .RO         59  15-DEC-82 WD
 PRIMES   .TXT         3  30-MAR-82 WD

FILES=34, SECTORS=585
LARGEST=111, FREE=1695
```

The last eight files are necessary to compile and run Pascal programs.

2. Upgraded Courseware

New MENU's have been provided on all courseware disks. The new MENU program is described in the Version 2 POLYSYS Utilities Manual.

Note that CATalog protect is not used on the Version 2 courseware disks.

With the release of Version 2, the following courseware files have been upgraded.

Physics
- PP24PY (Free Fall Simulation)
- PP25PY (Hit the Tank)
- PP26PY (Frames of Reference)

Farming
- PP2NCA.DAT (Data for Red Mite Control)
- PP21FM (Managing a Dairy Farm)

Games
- PP23GM (Shoot)
- PP24GM (Battleships)
- OCTOPUS
- DOGFIGHT
- OOOXXX

Computer Awareness
- PP2ICA (Bank Teller)
- PP23CA (Computer as a Filing Cabinet)
- PP2LCA (Teacher Security Program)
- PP2ECA (Loan Repayment)
- PP22CA (Data in a Computer)
- PP29CA (Personal Information Service)
- PP21CA (Crossroads)
- PP2KCA (Personal Information Database)
- DESIGNS1
- DESIGNS2
- POETRY

Maths
- MATHS1
- GRAPH

Geography
- PP2HGG (Chained from Developing Graph Skills)

English
- SPELLING

Other
- POLYMENU
- PROFORMA

3 Replacement ROMS

4 BASIC (dated 10/6/83)
1 POLY System (dated 19/5/83)
1 Disk drive/network controller (dated 3/9/82)

4. Replacement Manuals (All May 1983)

POLYBASIC Manual
POLYSYS Utilities Manual
POLY System Operating Manual
Pascal on POLY