Psion Developer Manual

Contents

Introduction

Debugging programs, making packs

Used in conjunction with the Comms Link, the Organiser Developer is a powerful tool. You can:

- Backup the entire contents of a datapak or rampak to the PC, and optionally break it into its constituent files.
- Write. edit and debug procedures on your IBM PC or compatible and then send them to the

  Organiser via the Comms Link,
- Build on your PC a `datapak image', made up of the procedures you have worked on, together with other Organiser files you have transferred to the PC. Then link your Organiser and PC via the Comms Link and send this pack to an empty pack inserted into the Organiser.

System requirements

The Developer requires:

- 512k of memory
- DOS version 2.0 or later.

This manual assumes a certain knowledge of and familiarity with the features of the Organiser and IBM PC in general and OPL and the Organiser Comms Link in particular.

The disk

Before using the Developer disk, copy it to another floppy disk or a hard disk. Keep the original disk as a backup and run the Developer from the copy.

The Developer disk contains the following files:

ORG2.EXE

The Organiser emulator.

OPLTRAN.EXE

Translates procedures to run on the Organiser.

BLDPACK.EXE

Builds a pack image' on the PC from specified files.

MAKE.EXE

Sends a `pack image' to a pack inserted in slot C: on the Organiser attached via the Comms Link.

AMAKE.BAT

Appends files to a pack in slot C: on the Organiser attached via the Comms Link.

UNMAKE.EXE

Loads a datapak from the Organiser via the Comms Link and breaks it into its constituent files.

HELP.TXT

Help information for the emulator.

BOOT.BIN

Example Bootable header.

MAKE.BIN, UNMAKE.BIN, AMAKE.BIN

Communications software used during MAKE etc.

You may wish to use the DOS command to set up a path to the directory containing these files. Then you can run the Developer from any directory.

1 Overview

The heart of the Developer is the emulator, in which you can write, edit, translate and run procedures exactly as if they were on a real Organiser, but using increased editing and debugging facilities.

The Developer therefore has two basic modes of operation:

- **Organiser emulation mode**

An area of the screen acts like the Organiser display, showing the PROG menu.

|	|
Select EDIT	Select RUN
- editor	**- debugger**
facilities	facilities
active.	active.

- **Emulation command mode**

The emulator is suspended and you are entering commands at a DOS-style prompt.

The editor and debugger are discussed in Chapter 2.
This Chapter discusses the emulator generally.

1-1 Starting the emulator

> If you haven't set a PATH, make the directory in which you installed the Developer files the current directory.

>Type ORG2 {return}

The emulator is displayed, made up of 3 windows.

LCD window

Prog menu l

l

When the program begins, you enter Organiser emulation mode: the LCD window presents the PROC menu, with a cursor flashing on the EDIT option as on a real Organiser.

The Developer emulates a 4-line Organiser model LZ. You can develop procedures for any model of the Organiser, though, since you can translate procedures for either 2- or,4- line display mode.

You cannot exit from the PROC menu to the top-level menu. This is because the emulator is designed to create and test procedures; other Organiser facilities aren't relevant in the emulator. (The Developer does offer facilities for handling other Organiser file types - see Chapters 3 and 4.)

While you're in Organiser emulation mode,

- **F1** works like the **ON/CLEAR** button - e.g. to clear a file name from a SAVE: prompt.
- **F2** works like the **MODE** button - e.g. to get the TRAN/SAVE/QUIT/XTRAN menu.
- Left, right, up and down {keys} move you around the PROG menu.
- {return} works like EXE on the Organiser - e.g. to select a menu item.

STATUS window

e `Indicators`

e

The indicators in the STATUS window provide information about your `mock' Organiser:

- **RAM SIZE** shows the amount of available RAM (internal memory, device A:) in the emulator Organiser. This figure cannot be changed.
- **SLOT B: and SLOT C:** show the `mock' packs which are fitted to the back of your Organiser (none when you first enter the emulator).
- **DEBUG** indicator and **DISK** indicator should not normally be changed from their defaults. They are explained in Chapters 2 and 3 respectively.

EMULATOR COMMAND window

> >

>Command prompt

To move between the emulator's two basic modes,

- Press **F4** to move from Organiser emulation to emulation command mode.
- Press **F1** to move from emulation command to Organiser emulation mode.

When in emulation command mode, the following groups of commands are available to you:

- Commands handling packs for slots B: and C: in the emulator (see Chapter 3).
- General commands for handling the emulator (see below).

Emulator commands

Square brackets indicate optional parameters; neither square nor angle brackets should be typed in as part of the command.

QUIT {enter}

Quits from the emulator to DOS.

HELP [<command name>] {enter}

Gives information on commands. Only the first 3 letters of the command name need be entered. Enter just HELP{enter} for a list of the keys and commands on which help is available.

DOS [<command line>] {enter}

If no command line is entered, the screen is cleared and you return to the DOS prompt. All your usual DOS operations are open to you. This is different from using quit, since the emulator is still using RAM. To return to the emulator, type EXIT{return}, not ORG2{return} again.

If a command is issued in the same line as you invoke DOS, then this command executes. After it completes, control automatically returns to the emulator after a `Press any key...' message.

For example:

DOS CD\dira\subd {return}

would change the directory which the emulator saves files to or loads files from.

The DOS command will work like this only if COMMAND.COM is in the current path. See your Dos manual for information about how to do this.

1-5 Printing from the emulator

If you have a printer connected to your PC, you can \ print from within the emulator. You can:

- Print out a procedure without having to quit the emulator. Select PRINT from the emulator PROG menu and enter the procedure name in the usual way.

or

- Select RUN from the emulator PROG menu and run a procedure which uses LPRINT statements - e.g. to print out records from a data file.

If your printer is not connected to the parallel port, use the following command:

DOS MODE LPT1:=COM1 {return}

or

DOS MODE LPT1:=COM2 {return}

This runs the MODE command to redirect output from the parallel port to serial port 1 or 2 respectively.

2 Procedures in the emulator

Developing procedures in the emulator involves the same three stages as on the Organiser:

1 - Editing.

2 - Translating and saving.

3 - Running - debugging as errors occur.

This Chapter describes these emulator facilities.

In a typical Developer session, you will probably: edit a procedure - run it - edit it again to correct errors - run it again - repeatedly, until the procedure works. This is done by simply selecting EDIT and RUN in turn from the PROC menu.

2-1 Restrictions on OPL

Because of the difference between Organiser and PC hardware, some features of OPL cannot be emulated.

The following **have no effect:**

- UDG, POKEB and POKEW commands.
- ADDR PEEKB. PEEKW, USR and USR$ functions.
- Top-slot access - e.g. XTSEND, XTRECV.

- so do not include them in any of your procedures.

The clock (produced with the CLOCK command on model LZ) shows only as ##:##

2-2 Editing

When you first enter the emulator, your Organiser is `empty' . If there are any procedures in your current directory, however, the Developer recognises these by their tile extension - .OPL.

- To edit a procedure already existing as a .OPL file in your current directory, select EDIT from the PROC menu.
- To create a new procedure in the emulator, select NEW from the PROC menu.

When you enter the procedure name. the emulator behaves differently from a real Organiser:

- The EDITOR STATUS window shows just the file name and the current line number of the procedure being edited.

> \>

- The OPTIONS window provides a summary of the editor keys.

>

- The procedure itself is listed in the SCREEN EDITOR window.

2-3 Editor keys

The Developer editor provides all the basic screen-editor facilities. However, if you prefer, you can edit procedure files with your usual ASCII text editor - or word processor, provided you save the file as ASCII (text) only.

Exit - F2

Exits to the usual TRAN/SAVE/QUIT/XTRAN menu. This is equivalent to pressing the MODE key during editing on the real Organiser. At this point you may press F4 to return to emulator command mode.

Movement

Move cursor one word to the left	**F9**
Move cursor one word to the right	**F10**
Move cursor to left end of line	**HOME**
Move cursor to right end of line	**END**
Scroll up one screen	**PG UP**
Scroll down one screen	**PG DN**
Top/bottom of	**F3**

If the cursor is at the top of the buffer, moves to the bottom; otherwise moves to the top.

Cutting and inserting text

Delete current line - F1

Restore line - SHIFT-F1

Restores the line most recently deleted with F1. Always inserts before the current line.

Mark - F1

Marks the first line of a block of text to be cut.

Cut - F6

Cuts to the paste buffer all lines from the mark down/up to and including the current line. Only whole lines may be cut.

Paste - F4

Inserts the contents of the paste buffer before the start of the current line.

Search and replace

The OPTIONS window prompts for the search (and replace) string. The search is case independent.

Search - F7

Searches forwards from the current cursor position for a specified text string. If the string Is found, you are prompted to:

- Continue searching - press C

  or

- Stop searching - press X.

Replace - F8

Searches forwards from the current cursor position for occurrences of a given text string, and optionally replaces that string with another. If the string is found, you are offered three options:

- Continue searching without replacing the string - press C
- Replace the search string with the new string and continue searching - press R

  or

- Stop searching - press X.

Buffers

The screen editor has two buffers, each of which may be used to edit text. Initially the second buffer is empty. This buffer may be used, for example, to write or edit .TRN and .BLD files as described M Chapter 3, File names are prompted for in the OPTIONS window. They must include file extensions, and may include a Dos directory path if required.

Swap buffers - SHIFT-F2

Swaps from one buffer to the other.

Merge - SHIFT - F3

Merges (pastes) a file into the current buffer before the line where the cursor is currently positioned.

Save current buffer - SHIFT - F4

Saves the contents of the current buffer as an ASCII file.

2 Translating and saving

As on the Organiser, you have to translate procedures in the emulator before you can run them. When you have a complete procedure in the editor.

- Press F2.
- Select TRAN or XTRAN, then SAVE the procedure under a file name.

2- and 4-line mode

Procedures translated in 4-line mode will run only on model LZ: they will immediately terminate if you try to run them on model XP or CM.

2-line mode procedures can be run on any model of the Organiser, as well as in the emulator.

- TRAN translates into 4-line mode.
- XTRAN into 2-line mode.

Translating while not editing

Instead of using TRAN or XTRAN from the PROG menu in Organiser emulation mode, you can use the following commands while in emulation command mode:

TRAN <filename>{return} **(4-line mode)** or

XTRAN <filename>{return} **(2-line mode)**

To translate files not in your current directory, specify a path name - e.g.

TRAN \dira\subd\file{return}

.LNO files

TRAN and XTRAN, from the PROG menu or as commands, create .LNO Files from the .OPL file <filename> in your current directory.

The .LNO file runs when you select RUN from the PROG menu in Organiser emulation mode. However, it wouldn't run on the Organiser itself, since code on the PC is different to code on the Organiser. If you want to translate a .OPL file into a .OB3 file which the Organiser can run, use the OPLTRAN command (see Chapter 4).

Translating more than one procedure

If you want to translate more than one procedure at a time, use the TRAN or XTRAN command like this:

TRAN @<filename>{return}

or

XTRAN @<filename>{return}

The command uses the command file <filename>.TRN Write this in your current directory. This file lists the names of the files to be translated - e.g.

```
proc1 calls proc2
proc2 input procedure
proc3 calculates result
```

In this file, the first word on each line is taken as the name of a .OPL file in your current directory. The rest of the line is ignored, so you can include comments.

Progress and errors are reported while the TRAN or XTRAN command proceeds.

Saving to disk or to a pack

Different files are produced if you translate with DISK OFF from when you translate with DISK ON:

- DISK ON - procedures are saved as .OPL (ASCII) and .LNO (PC code) files in your current directory.
- DISK OFF - procedures are held in RAM; you can RUN the procedure during your editing session, but the procedure is lost when you QUIT from the emulator.

DISK is ON by default. When DISK is ON, you cannot copy procedures to packs in B: or C: in the emulator. Set DISK OFF **only** when you wish to do this (see Chapter 3).

3 Debugging

When running OPL procedures on the emulator, you may have the debug facility either on or off.

- DEBUG OFF - the LCD window shows procedures running as they would on the real Organiser.
- DEBUG ON - the debugger offers all the usual language debugging tools.

Extra code is added when a procedure is translated with DEBUG ON, which makes the code larger than it would be on an Organiser. Procedures translated with DEBUG OFF are the same size as Organiser procedures.

To turn DEBUG ON/OFF:

- Press F4 to enter emulation command mode and type DEBUG OFF {enter}

- To turn it back on, type DEBUG ON {enter}

Starting the debugger

Procedures can be debugged only if they were translated whilst the DEBUG indicator was set to ON. DEBUG is set ON by default.

To start the debugger:

- Press F1 to enter Organiser emulation mode.
- Translate the procedure with TRAN or XTRAN.
- Select RUN from the PROG menu and enter the procedure name.

When you select RUN, the procedure does not run immediately - provided DEBUG is ON - but the screen divides into four areas:

- LCD window operates in the same way as the LCD on the real Organiser.
  |

> >

- **DEBUG COMMAND** window shows a ? command prompt.

>

- **SOURCE** window shows the source code, with a * marking the line currently being executed and a > marking the cursor position.

>

- **DEBUG** window shows the output of some of the commands - e.g. HELP information is shown here.

When the procedure completes, you are returned to the PROG menu.

Using the debugger

If you haven't used a debugger before, try:

- Using the GO command to run a procedure for the first time, to see whether it works.
- If the procedure doesn't run adequately, set TRACE ON and select GO again. This lets you watch the progress of the procedure as it runs.
- Use the WATCH command to see how the values of variables change.
- For more control over the execution of the lines of the procedure, use one of the STEP commands - to execute line by line - or one of the BREAK commands - e.g. to run a loop and then pause.

2-11 Debugger keys and commands

While the debugger is running, the commands listed below are available to you. To make changes to the source code, QUIT the debugger and select EDIT .

To back out of a command, press **F1** or **Esc** to return to the ? prompt.

Square brackets indicate optional parameters.

<Procedure name> does not include the colon.

General

Quit - Q {enter} ~ Ends execution of the OPL program.

Help - F3 ~ Lists all the commands available.

Catalogue C {enter} ~ Shows a list of all .OPL files on the disk.

Movement (in source window)

- **Move cursor up a line - Up**
- **Move cursor down a line - Down**
- **Scroll code one screen up - PG UP**
- **Scroll code one screen down - PG DN**

Running

Go - F7 or G {enter}

Continues program execution from the current position.

Trace - T {enter}

Toggles TRACE ON/OFF. OFF by default.

If TRACE is ON, the source code is displayed with a * marking the line currently being executed. The procedure pauses and returns control to the debugger command line if:

- a break point is reached
- a watched variable changes
- an untrapped error occurs

or

- **F1** is pressed (other than in a GET, KEY, MENU or INPUT statement).

Step to next instruction, or - F5 or into a sub-procedure - S/I {enter}

Executes a single statement, stepping into a sub-procedure if one occurs, and then stops.

There may be more than one statement in a line- e.g. the line CLS :PRINT "x" :RETURN contains three statements.

To execute a certain number of statements, enter S/I<number>{return} at the command line - e.g. S/I5{enter} would step 5 statements.

Step to next instruction in current procedure - F6

Steps to the next instruction in the current procedure, executing any intervening sub-procedures without stepping through them.

If you type S <number>{enter} that number of statements is executed.

If you type S /R{enter} the program runs to its end (unless the procedure breaks for any reason).

Break into a running program - F1

Provided the procedure is not in the middle of executing a GET, KEY, MENU or INPUT instruction.

Breaks

To stop a running procedure at the beginning of a certain Line, set a break point. No checking is done that a specified procedure or Line number exists. Up to 8 break points may be set at any one time, including temporary break points.

Break points

B[<procedure name>[:<number>]] {enter}

If no procedure name is specified, the break point is set at the current line in the current procedure.

If no number is specified, a break point is set at the beginning of the first line of the procedure (i.e. after the procedure name).

If a number is specified, a break point is set at the beginning of that line - e.g. Bproc:5 sets a break point at the beginning of the fifth line of the procedure proc:.

Typing B on its own gives a list of the current break points.

To delete a break point re-enter it exactly as it was first entered.

Set a temporary break point - F10

Sets a break point at the beginning of the current line (marked with >). The break point will stop the running program only once - it is deleted when reached.

Variables

Deposit

D [<procedure name>:]<variable name>{enter}

<new value>{enter}

Changes the current value of a variable.

At the Deposit at: prompt, enter the variable name. At the with: prompt enter the new value.

Note that:

- Deposit will not replace field variables.
- An error is reported if the new value is not the right type of data - e.g. a string for an integer variable. In this case the variable's value remains unchanged.

Examine

E [<procedure name>:]<variable name>{enter}

Returns the current value of any variable in any procedure.

E.g. to And the current value of the variable C% in the procedure TEST: type ETEST:C% {enter}

If you do not specify a procedure then the command looks for the first instance of the variable.

Variables

V[<variable name] {enter}

Lists the variables currently available. Use the * wildcard to specify a subset of variables.

E.g. V ABC*{enter} lists all variables starting with the letters ABC. V{enter}lists all current variables. v *.* {enter} lists all field variables (since only field variables have the `.' character in their names.)

Watch

W<variable name>{enter}

Sets a watch on a variable, checking it after each statement. The program pauses and returns control to the debugger if any of them have changed. The change in value is reported.

Watches can be set on up to 4 variables, excluding field variables. wet on its own lists all the variables currently being watched.

To stop watching a variable, enter W<variable name>{enter} exactly as it was first entered.

The program may run slightly slower during use of this command.

Display

Display another .OPL file. - F2

Prompts for a procedure name.

While the procedure is displayed, you may set temporary break points in it. To return to the current procedure press **F2** again.

Procedures - P{enter}

Shows the procedures above the current procedure in the calling structure with their current line number.

3 Packs and data files in the emulator

The real Organiser has two slots, B: and C:. in which datapaks and rampaks can be fitted. In the emulator, too, you can fit packs and copy data and procedure files to them.

For example, you might want to:

- Test a procedure which accesses data files on a different device.
- Create a data file with a procedure and then save the data file to disk.

You can now do this with the LOAD and SAVE commands, new in this version of the Developer.

3-1 Creating packs

The STATUS window shows what packs the emulator contains, in slots B: and C:. To start with, these are empty. Press **F4** to enter emulator command mode and then use the following commands to put in or take out packs:

Create <packname> <size> [R}{enter}

Creates and sizes an 8, 16, 32, 64 or 128k datapak. If the command is terminated with R, <size> must be 32, and a 32k rampak is made. These packs are held on the PC as files with the extension .IPK. The packname must be different to any existing .IPK filename.

DELETE<packname>{enter}

Deletes a .IPK file. The pack to be deleted must not currently be fitted to either slot.

ERASE <packname>{enter}

Erases the contents of a .IPK file. This is equivalent to formatting a pack in a formatter and then sizing it. The pack to be erased must not be fitted to either of the slots.

FIT <packname> <slot> {enter}

Fits an existing pack into either B: or C:.

REMOVE <slot>{enter}

Removes the pack currently in the specified slot.

In these commands, <packname> is up to 8 alphanumeric characters starting with a letter, and <slot> is either B: or C:, including the colon.

3-2 Loading data files into the emulator: LOAD

Once you have created and fitted packs, you can fill them with data and procedure files. Two new commands have been added to the Developer:

- LOAD - loads a .ODB file held on disk into a device (A:, B: or C:) in the emulator.

- SAVE - saves a .ODB file created in the emulator (by OPL) to disk.

These are emulator commands; they aren't called from DOS.

If you have a .ODB file on your disk - either created in your usual text editor, or transferred from your Organiser via Comms Link - LOAD the file onto one of the devices in the emulator in the following way:

- Press F4 to enter emulation command mode.
- Type LOAD <filename> [.<extension>]<device>:[newname]{enter}

E.g. LOAD Data b: {enter} loads the datafile Data.ODB into the pack currently in slot B:.

If no extension is specified, .ODB is assumed. [newname] is the optional new name for the file on the pack.

<filename> can include a pathname - e.g. LOAD \dir\ subd\ file b:{enter}. If no path is specified, the file is looked for in the current directory.

The LOAD command loads the ASCII information into the .IPK `pack image' file, or into RAM if <device> is A:.

3-3 Creating and saving data files through OPL: SAVE

To create a data file on a pack through OPL, write a procedure like the following. RUN the procedure from the emulator PROC menu.

```
file:
CREATE "A:names",A,field1S,field2S,field3$
A.field1$="name"
A.field2$="address"
A.field3$"phone"
```

If you had a pack fitted to B: or C:, you could create this File on that device.

To save this data File to disk:

- Press **F4** to enter emulation command mode.
- Type SAVE <device>:<filename> [<newname>[.<extension>]]{enter}

This saves <filename> from device A: B: or C: to disk. It will be given the name <filename>.ODB by default.

You may specify a <newname>, and if you do you may also specify an extension. <newname> may include a pathname- e.g. SAVE b:file \dir\file.odb{enter} . If no path is specified, the file is created in the current directory.

3-4 Procedures on packs

By default, DISK is set ON. This means that the emulator looks for and saves .OPL Files on the disk. If you wish to erase or copy files on a pack fitted In B: or C: in the emulator, DISK must be set OFF.

Saving on a pack

Follow these steps to save a procedure on a device:

- Press **F4** to enter emulation command mode. If desired, FIT a pack into slot B: or C:.
- Press **F1** to return to Organiser emulation mode. Select EDIT and load the procedure.
- Press **F2** to get the TRAN/SAVE/QUIT/XTRAN menu in the LCD window.

- Before translating go back into emulation command mode with F4 and set DISK to OFF.
- Press **F1** to return to the TRAN/SAVE/QUIT/XTRAN menu. Select TRAN or XTRAN, then SAVE. mess **F2** to toggle the device from A: to B: or C:.

The procedure can now be copied between packs, or erased.

When a procedure is saved or copied to a pack in B: or C:, the code is inserted into the .IPK file of the pack fitted. The .IPK file is saved on your disk, whereas A: is RAM that is cleared when you QUIT the emulator.

Retrieving from a pack

A procedure held in a pack image can be retrieved as a .OPL file: select EDIT and load the procedure from the pack, press **F2** then set DISK ON before you translate and save the procedure.

Renaming procedures

In the emulator, procedure names are protected just as on the real Organiser. To change the name of a procedure, either:

- Use the RENAME command in DOS

or

- Use the COPY option in the PROG menu in the emulator to make a copy under a new name. Then use ERASE from the PROG menu to delete the original.

If DISK is OFF, you can't copy from A:prog to A:newprog.

4-1 Making packs for the Organiser

The process of making a pack on the Organiser involves the following stages:

1. Translate procedures to Organiser code with the OPLTRAN command.
2. Write a .BLD file, naming those files to be built into the pack. Build the pack image with the BLDPACK command.
3. Connect the PC and the Organiser with the Comms Link and send the pack image to the Organiser with the MAKE or AMAKE command.

In the descriptions of the OPLTRAN, BLDPACK MAKE and

AMAKE commands,

- <packname>, <filename> and <newname> are up to 8 alphanumeric characters starting with a letter.
- <size> is 8, 16, 32, 64 or 128.
- <slot> Is either B: or C:, including the colon.
- Square brackets [ ] indicate optional command parameters, a vertical bar indicates either/or parameters. Do not type in the brackets and bars.

HELP

Help on these commands is available in DOS by typing <command name> ?{enter} In the emulator, call this with the Dos command in emulation command mode.

4-2 Translating procedures to run on the Organiser

OPLTRAN

When you select TRAN or XTRAN from the PROG menu in the emulator, code is produced to run the procedure only in the emulator - not on the Organiser. The translated PC file has the extension .LNO; procedures translated on the Organiser have the extension .0B3.

To produce .OB3 files on the PC, use the OPLTRAN command.

4- and 2-line mode

- To produce 4-line mode .0B3 files for Organiser model LZ, use OPLTRAN.
- To produce 2-line mode .0B3 files for any model of the Organiser, use OPLTRAN with the -x flag.

OPLTRAN is a DOS command. When you are in DOS, call it like this:

OPLTRAN <filename>{enter}

or

OPLTRAN <filename>{enter}

When you are in the emulator. call it like this from emulation command mode:

DOS OPLTRAN <filename>{enter}

or

DOS OPLTRAN <filename> -x{enter}

- where <Filename> is the name of the .OPL file to be translated. You don't need to include the extension.

When the OPLTRAN command has completed, the File <filename>.oB3 will have been added to your current directory and you will be returned to the emulator. During execution, progress and errors are reported to the screen as the files are translated.

Translating files in different directories

For OPLTRAN to access files in directories other than its own, specify the pathname - e.g.

Der

However, OPLTRAN cannot access files on another drive. if you don't have a hard disk, this means that all the files will have to be held on the same floppy for the duration of the translation.

Source or object only

The command can be entered like this:

OPLTRAN <filename> [flag]{enter}

where the flag can be one (and only one) of the following:

-s	source only
-o	object only (the default)
-t	source and object

The -x flag can be used in addition to any of these flags.

Translating more than one procedure at a time

- Write a command file, in your usual text editor or the second editor buffer, with the extension .TRN. In this file, list the names of the .OPL files which are to be translated. List them on separate lines **without** the .OPL extension, as this is assumed.
- Call this .TRN file with the OPLTRAN command like this:

OPLTRAN @<filename> [-x] [flag]{enter}

For example, If Filelist.TRN looked like this:

```
proc1
proc2
proc3
```

the command **DOS OPLTRAN @filelist -x -o{enter}** would produce the Files proc1.0B3, proc2.OB3 and proc3.OB3 to run in 2-line mode. These would be object only; you could not edit them, only run them on the Organiser after transferring them there.

4-5 Building a pack image:

BLDPACK

BLDPACK builds a 'pack image' to be made into a pack on the Organiser. The pack can contain any combination of the following:

- .OB3 files (untranslated .OPL Files aren't accepted)
- .OBD database files
- Image Files from the Organiser - e.g. Pocket Spreadsheet files
- Bootable code (see Appendix C).

The `pack image' is held on the PC as a .OPK File.

The BLDPACK command calls a command File with extension .BLD. This must be an ASCII File, written on your usual text editor or on the second editor buffer, containing the list of Files to be built into the pack.

Example .BLD file

```
EXAMPLE 16 NOCOPY
TABLE ODB            !Data file
PROC1 OB3            !First procedure
PROC2 OB3            !Second procedure
```

(The exclamation marks let you add comments.)

This .BLD file will produce a 16k non-copyable pack called EXAMPLE, containing one data File called TABLE and two procedures PROC1 and PROC2. On the PC the pack has the filename EXAMPLE.OPK.

Pack specification

The first line of the .BLD file has the following format:

<packname> [<size>] [NOCOPY] [NOWRITE]

where <packname> is the name of the pack produced.

Size

If <size> is specified it is the size of the pack generated. Otherwise the pack size is the smallest pack onto which the data will fit. If the data will not fit in 128K or the size specified, further packs will be generated: the second pack will have `1' appended, the second `2' and so on. These packs will all have the same protection.

Protection

By default the pack will be unprotected. You can instead make the pack copy-protected (NOCOPY), write-protected (NOWRITE) or copy- and write-protected (NOCOPY NOWRITE).

Files included

After the first line, subsequent lines of the .BLD file give the references of the files to be included on the pack, as shown below.

In these references, there is no dot, just a space between the filename and its extension. The file created on the pack has the same name as the <filename> specified unless a <newname> is given.

Data files

<filename> odb [<newname>] [,<separator>] [!<comment>]

The data file must have extension .ODB in your current directory and be in ASCII format. It must have one line per record, and fields delimited by <TAB>s (the default) or by the given <separator> (any single character).

Image files

<filename> <obx> [<newname>] [!<comment>]

<obx> is a file extension in the range .OB2 to .OBF, which covers all Organiser image files. See Appendix A for a list of extensions. BLDPACK won't accept .OPL files.

Calling files from a different directory

BLDPACK must be able to access all the files listed in the .BLD File. To include files in other directories, give the pathname - e.g.

\dira\subd\PROC2 OB3

However, BLDPACK cannot access files on a different drive, if you do not have a hard disk, this means that all the Files must be on the same floppy for the duration of the build.

Calling the .BLD file

Call the .BLD file by typing at the DOS prompt:

BLDPACK @<filename{enter}

<filename> does not include the file extension. You can only build one pack at a time.

Saving information to a file

As the `build' proceeds, the status and any errors are reported. This information can be written to a File during the `build' by using the -map flag:

BLDPACK @<filename> -map {enter}

The map File produced will have the extension .MAP and the same filename as that of the pack which is being built.

Building a pack for the emulator

Use the command like this:

BLDPACK @<filename>{enter}

to create a .IPK, not a .OPK, File. This is equivalent to using the CREATE command, then copying procedures to the created pack and loading data files to it with the LOAD command.

If you use the -i flag, .LNO files are the only image files which may be included.

4-9 MAKE and AMAKE

MAKE and AMAKE download .OPK Files created on the PC with BLDPACK directly to a Datapak or Rampak in slot C: on the Organiser, via the Comms Link cable.

The difference between the commands is that:

- MAKE requires a blank pack - i.e. formatted and unsized.
- AMAKE appends Datapaks - adding files to the pack without affecting any files already there. It can either append, or format and then fill, Rampaks.

Restrictions on AMAKE

AMAKE does not check the consistency of the pack it is making. This means that you could end up with more than one file with the same name on the pack. This would not corrupt your files, but make handling them unpredictable. To avoid this, check the names of the files on the destination pack and in the .BLD file for duplicates.

AMAKE does not append records within individual data files. It adds files to a pack, not records to a file.

You cannot append a pack with bootable code. E.g. you would get a "ERROR - write error" if you included BOOT.BIN in the .BLD file, and the AMAKE would end.

Comms Link setup

AMAKE.BAT actually just calls MAKE.EXE with a *-a* flag. Therefore the setup for these two commands is largely the same. But the messages and prompts during the execution of the commands are slightly different.

It is worth using an Organiser power supply when using MAKE or AMAKE, since the power drain while making the pack (unless it is a Rampak) is substantial.

When making a pack, don't run the `Comms Link' program as you usually do when communicating with the Organiser. The MAKE and AMAKE commands use their own communications software.

Issuing the MAKE or AMAKE command

The commands have the syntax:

MAKE <packname> [<baud> [<port>]] [-a]{enter}

AMAKE <packname> [<baud> [<port>]]{enter}

<packname> is the name of the .OPK file on the PC.

These commands must be able to access the .OPK file they are calling. Either have the MAKE.EXE/AMAKE.BAT and .OPK files in the same directory, set up a path or specify the pathname - e.g.

MAKE \dira\subd\pack{enter}

<baud> may be 9600, 4800, 2400, 1200, 600, 300, 150, 110, 75 or 50. If you specify a baud rate you may also specify a port - 1 or 2. 9600 baud, port 1 is the default. Make sure that BAUD in the Organiser SETUP list is set to the same rate.

- Link the PC and the Organiser with the Comma Link cable, as described in the Comms Link manual.
- Enter the MAKE or AMAKE command.
- As asked, select the BOOT option from the COMMS menu on the Organiser, then at the `Name:'prompt simply press **EXE**.

After this point, all significant prompts appear on the IBM PC screen.

MAKE

For a few seconds the message *Loading code...* is displayed. Then you get the prompt: *Place unsized pack in C:, then press a key on the PC.*

- Insert a newly formatted and unsized pack in slot C:. Don't do this before the message prompts you.
- Press any key on the PC.

The PC shows a block count of data sent as the specified .OPK file is downloaded directly to the pack on the Organiser. If the pack is a Datapak, not a Rampak, this may take some time. The Organiser shows the message:

> *Sizing pack C: please wait*

and then:

> *Making a pack in C:*

The PC then displays the message:

> *Pack successfully made. Do you want to make another pack (Y/N)?*

If you press Y you are prompted to insert another pack in slot C:. Alternatively. press N to return to DOS. On the Organiser you are returned to the COMMS menu.

AMAKE

For a few seconds the message Loading code... is displayed. Then you get the prompt: Place pack in C:, then press a key on the PC.

- Insert the desired pack in slot C:.

- Press any key on the PC.

The .OPK file is downloaded to the pack on the Organiser.

If the pack is a Rampak. you are first asked:

Press A to append, press F to format, ESCAPE for neither.

If the pack is a Datapak, you are first asked:

Pack already formatted. Do you want to append (Y/N)?

During the transfer. the Organiser shows the message:

Adding to pack C:

and the PC:

Sending pack data block nn

-where nn is a block count. The PC then displays the message:

Pack made successfully. Do you want to make another pack (Y/N)?

If you press Y you are prompted to insert another pack in slot C:. Alternatively, press N to return to DOS. On the Organiser you are returned to the COMMS menu.

Errors

The error messages which may be presented during the MAKE or AMAKE process are listed in Appendix B.

If the pack is different in size to that specified in the .BLD file but there is enough room for the data, then the message:

WARNING: PACK IS DIFFERENT SIZE

is given. Sometimes this is fatal, since the way that the Organiser reads a pack is different according to what size it thinks the pack is. Some different sizes are read in the same way, some not.

Abandoning the MAKE or AMAKE

If you have to abandon the transfer, don't pull the Comms Link cable out of the Organiser during the transfer. This could cause electrical damage.

Break into the transfer on the PC instead. Wait for the Organiser and the PC to timeout. Remove and reformat the pack you were trying to MAKE or AMAKE.

5-1 Loading a pack to the PC: UNMAKE

The UNMAKE utility

- Loads an entire Datapak or Rampak from an Organiser into the IBM PC as a .OPK file.
- Optionally breaks this .OPK file into its component files - procedures, data Files, etc.
- Optionally displays data file records which were deleted on the Organiser (deleting records on a Datapak does not actually erase them, only make them unreadable on the Organiser).

Command syntax

```
Unmake <packname> [<baud> [<port>]]{enter}

(unmake name 9600 2)
```

<packname> is the name of the .OPK file to be generated on the PC. This file will be created in the directory that is current when you issue the UNMAKE command.

<baud> may be 9600, 4800, 2400, 1200, 600, 300, 150, 110, 75 or 50. if you specify a baud rate, you may also specify a port - 1 or 2. 9600 baud, port 1 is the default.

Comms Link setup

When unmaking a pack, don't run the `Comms Link' program as you usually do when communicating with the Organiser. UNMAKE uses its own communications software.

Make sure that BAUD in the SETUP list on the Organiser is set to the same rate as used in the command line on the PC.

It is worth using an Organiser power supply when using UNMAKE, since the power requirement can be quite substantial.

To issue the UNMAKE command

- Link the PC and the Organiser with the Comms Link cable, as described in the Comms Link manual.
- Enter the UNMAKE command line.
- As prompted, select the Boot option from the Comms menu on the Organiser, then at the `Name:' prompt simply press EXE.

After this point, all significant prompts appear on the IBM PC screen. For a few seconds the message *Loading code...* is displayed. Then you get the prompt: *Place pack to be loaded in C:, then press a key on the PC.*

- Insert the pack in device C: and press a key on the PC.

The transfer completes. On the Organiser you are returned, after a short wait, to the Comms menu. On the PC, you are offered choices as to what to do with the received pack file.

Breaking into constituent flies

Once the transfer has completed, the PC asks:

Do you want to break the pack into parts (Y/N)?

If you answer N to this you are returned to the DOS prompt. The single .OPK file created will not be intelligible in your text editor, and cannot be used by the emulator. Use this option if you want to backup a pack and have maximum security on the files.

If you answer Y a separate file is made in your current directory on the IBM PC for each file on the pack.

- If there are any files in the destination directory with the same name and extension as the files received from the pack, the Files are overwritten.
- Empty files may be created.

If you are in doubt as to the contents of a pack, it is worth creating a temporary sub-directory on your IBM PC and loading the files into there.

Source code, deleted records

If you choose to break the pack File into constituent files, you are offered 2 further options:

> *Do you want to produce .OPL source code files (Y/N)?*

If you answer N .OB3 Files only, which can't be edited or run in the Developer, are created.

> *Do you want to show deleted records (Y/N)?*

Deleted records are those which, like work crossed out in a book, are still present on a Datapak but can't be seen by you.

If you answer Y you are told `Deleted records will be indicated by (*)'. The asterisk appears when you view the records in one of your data files in a text editor. If you answer N, when you look at your records, only those which hadn't been deleted will be shown.

As the pack is loaded to the PC, the Files created on the PC are listed to the screen - e.g.

CREATING DATAFILE.ODB

CREATING PROC1.OPL

The File type extensions are listed in Appendix A.

Help on UNMAKE

To get help about the UNMAKE command, type UNMAKE ?{enter} from DOS, or DOS UNMAKE ?{enter} from emulation command mode.

Appendix A-1

File types

File type	default extension on PC	extension on Org. model LZ
OPL source code (ASCII)	.OPL	.OPL
OPL source/object code	.OB3	.OPL
OPL object code for IBM PC	.LNO	
Database or LZ Diary (ASCII)	.ODB	.ODB
Diary (model XP/CM)	.OB2	.DIA
Comms Link setup	.OB4	.COM
Pocket Spreadsheet	.OB5	.PLN
Pager setup	.OB6	.PAG
LZ Notepad	.OB7	.NTS
Unspecified	.OB8-F	
LZ Notepad(ASCII)	.NTS	.NTS
Object code (binary)	.BIN	
Translate command file (ASCII)	.TRN	
Build command file (ASCII)	.BLD	
Organiser Datapak image	.OPK	
IBM PC Datapak image	.IPK	

Appendix B-1

Error Messages

MAKE, AMAKE and UNMAKE errors

No Link

The link between the Organiser and the PC has not been correctly established.

No Pack In Slot

No pack has been inserted in slot C: on the Organiser.

Pack Not Blank

The pack contains old data and should be reformatted.

Pack Too Small

The pack in device C: is too small to accept the data in the specified .OPK File.

Write Error

The data cannot be written to the pack; ensure the pack is correctly fitted.

All these errors will end the attempt to make that pack. In the case of a `No Link' error you will have to start again; in the other cases you will be given the opportunity to blow another pack.

LOAD errors

Bad Destination File Name
The name for the file on the pack is invalid.

Destination File Error
The emulator couldn't create the File on the pack.

Error Opening Source File
The emulator couldn't open the source File.

Load Error
The emulator couldn't load the file to the pack.

Invalid Source File name
The name of the source file in the PC directory is invalid.

SAVE errors

Bad Source File Device
The source device name in the emulator is invalid.

Bad Source File Name
The source file name is invalid.

Error Opening Destination File
The emulator can't create the destination file.

Invalid Destination File Name
The destination file name in the PC directory is invalid.

Save Error
The emulator couldn't save the file.

Source File Does Not Exist
The specified source file does not exist.

Source File Error
The emulator couldn't open the source file.

Appendix C-1

Bootable packs

When you use Comms Link, the menu item COMMS is added to the Organiser's top-level menu when you press ON/CLEAR (or the second time you press it if you had to press it once to turn the Organiser on). It is possible for a pack, made using the BLDPACK and MAKE commands, to insert a menu item automatically into the top-level menu in the same way - when you first turn the Organiser on with the pack in one of the slots. In other words, you can make your pack bootable.

To do this you need a .BIN File, which we have provided with the BOOT.BIN file on the accompanying disk. With a pack including this file, when you press ON/CLEAR the Organiser immediately executes BOOT.BIN, which tells it to run an OPL procedure called BOOT:. You can write this procedure yourself and then translate it as usual into a .OB3 file. On completion of the OPL procedure you are returned to the top-level menu.

The .BLD file

In the .BLD file you must include the names of the BOOT.BIN and BOOT.OPL file which you want on the pack. The name of the BOOT.BIN file must be the second line in the File, like this:

```
EXAMPLE 16, NOCOPY
BOOT BIN          [!Boot file]
TABLE ODB         [!Data file]
PROC1 OB3 RUN     [!First procedure]
PROC2 OB3 RUN2    [!Second procedure]
BOOT OPL          [!Boot file]
```

After BOOT.BIN, you can list the other files in whichever order you want.

If no .BIN File is included, a standard header is put down at the start of the pack. If you want to write your own .BIN file, see the end of this Appendix. For most purposes the BOOT.BIN file provided will be sufficient.

Given below is an example BOOT.OPL file which you could use in conjunction with BOOT.BIN. When you turn the Organiser on with this pack inserted, a new menu item, is inserted into the top-level menu, to which you are returned.

WARNING: This program uses the POKEB and POKEW commands, which if incorrectly used can cause the entire contents of the Organiser's internal memory to be deleted, Make sure you type it in exactly as it appears here.

As a precaution, back up the contents of A: to a Datapak.

BOOT:

```
ADDTOP: ("INVEST",0)
ADDTOP: (item$, pos%)
LOCAL I%, A%(2)
IF LEN(item$) >8
RAISE 202 :REM Menu too big
END IF
POKEB $2187,LEN(item$)
I%=l
WHILE I%<=LEN(item$)
POKEB $2187+I%,ASC(MIDS(item$,I%,1))
I%=I%+1
ENDWH
POKEW $2188+LEN(item$),0
A%(1)=$3F65 :REM Careful here
A%(2)=$3900 :REM Careful here
USR(ADDR(A%()), pos%)
```

BOOT: calls ADDTOP: with two parameters:

- "INVEST" is supplied as the value for parameter item$. Replace INVEST with whatever name you want inserted in the menu. Make sure it is a valid name, up to 8 characters long.
- The position at which you want the item to appear in the menu is supplied to the parameter pos%. 0 stands for the first position in the menu (where FIND usually is) and -1 stands for the penultimate position (just before OFF). Don't supply an invalid position to pos%.

The menu item inserted with this procedure will not be automatically deleted when you press **ON/CLEAR** after taking the pack out. You will have to delete it by hand.

Advanced use

If you want to write your own .BIN files you will need to contact either PSION Technical Support or your local country distributor.

You will need to purchase a copy of the Organiser Technical Manual (to see what the significance of the header bytes are on the Organiser) and the disks containing the Organiser Assembler (to translate from source to object code).

Given overleaf are those details of the header created by BOOT.BIN which you might wish to change.

BYTE	**NAME**	**DESCRIPTION**
0	Control	
BIT		
(0)		
1	EPROM or RAM	Set by MAKE, which finds out via Comms Link whether the attached pack is a Rampak or a Datapak and page counted or not.
2	Page counted or not	
3	WRITE protection	Set by you in .BLD file
4	NOBOOT	Whether pack is bootable; set to bootable by default.
5	COPY protection	Set by you in .BLD file
(6)		
1	**Size**	Gives the size of the pack (8, 16, 32, 64 or 128k); set by you in .BLD file or set automatically by .BLDPACK. Warning message if wrong.
(2)		
(3)		
(4)		
5	**Priority**	Determines whether this pack will be booted before another pack, depending on which has the higher priority.
(6)		

Appendix D-1

Function keys and commands

Screen editor

Cutting and inserting text	
Delete Current Line	F1
Restore Line	SHIFT-F1
Mark	F5
Cut	F6
Paste	F4
Search and replace	
Search	F7
Replace	F8
Buffers	
Swap Buffers	SHIFT-F2
Merge	SHIFT-F3
Save Current Buffer	SHIFT-F4
Exit	F2
Movement	
Move to left end of current line	HOME
Move to right end of current line	END
Scroll up one screen	PG UP

Scroll down one screen	PG DN
Top/Bottom of Buffer	F3
Move one word to the left	F9
Move one word to the right	F10

Debugger

General

Quit	Q{enter}
Help	F3
List of .OPL files (CATALOGUE)	C{enter}
Movement	
Move cursor up one line	Up
Move cursor down one line	Down
Scroll source code 1 screen up	PG UP
Scroll source code 1 screen down	PG DN
Displaying	
Display another .OPL file.	F2
List of procedures (PROCEDURES)	P{enter}
Running	
Go	F7
Trace on/off	T{enter}
Step to next instruction/into procedure	F5
Step to next instruction (same procedure)	F6
Break into running program	F1
Breaks	
Set temporary breakpoints	F10
Set BREAK points	B[<procname>]:<number>{enter}
Variables	
Change value of variable	
(DEPOSIT)	D[<procname>:]<variable name>{enter} <new value>{enter}
EXAMINE value of variable	E[<procname>:]<variable name>{enter}
List variables (VARIABLES)	V[<variable name>]{enter}
Set WATCH points	W[<variable name>]{enter}

Packs for the emulator

CREATE	<packname> <size> [R]{enter}
DELETE	<packname>{enter}
ERASE	<packname>{enter}
FIT	<packname> <slot>{enter}
REMOVE	<slot>{enter}

Packs for the Organiser

BLDPACK	@<filename> [-i] [-map]{enter}
MAKE	<packname> [<baud> [<port>]] [-a]{enter}
AMAKE	<packname> [<baud> [<port>]]{enter}
UNMAKE	<packname> [<baud> [<port>]]{enter}

Translating

.OPL > .LNO

```
TRAN[@]<filename>{enter} (4-line mode)
XTRAN [@]<filename>{enter} (2-line mode)
```

.OPL > .OB3

```
OPLTRAN [@]<filename> [-s | -o | -t]{enter} (4-line mode)
OPLTRAN [@]<filename> -x [-s | -o | -t]{enter} (2-line mode)
```