TASK SWOPPER

for the Sinclair QL Computer

USER GUIDE

by Mark Hughes

QL Task Swopper Manual: Issue 2 - July 1, 1987

Compware, 57 Repton Drive, Haslington, Crewe CW1 1SA.

TABLE OF CONTENTS

1. Introduction

Have you ever been using Quill, or one of the other Psion packages and wished you could return quickly to Basic and copy some files or run a Basic program? Or perhaps you would like to be able to look up an address in Archive for inclusion in a letter, or would find it useful to access Abacus to do some calculations whilst Quill is printing a long document? If so, *Task Swopper* will be a very useful program indeed.

Whilst there are many programs available which will multitask properly on the QL, there are also several important ones which will not. Properly multitasking programs are invoked by the *exec* command and selected using *CTRL/C*, whilst the others must be invoked with *exec_w* and can only be used one at a time. The Computer One Assembler (available from Compware) is an example of a program which will multitask properly, whereas the four Psion packages are examples of ones which will not.

Task Swopper is designed to let you multitask programs which must normally be run one at a time, and which would normally be invoked by *exec_w*. If you have sufficient RAM on your QL, then *Task Swopper* will enable you to have all the Psion packages, and Basic, available in an instant on the same QL. You will even be able to have more than one copy of any Psion program in the QL at once. Even on a QL with only the standard 128K of RAM, you can have any one of Quill, Archive, Abacus or Easel running, and still be able to swop back to Basic. If you have a RAM expansion then *Task Swopper* will be even more useful.

Task Swopper is fully compatible with a complementary program called *QATS* written by Cope. *QATS* or *QL Applications Traffic Supervisor*, is designed to minimise the number of keystrokes necessary to perform housekeeping tasks. It is menu driven and easily customised to produce menus for running programs. *QATS* is recommended by Compware and includes the following facilities:

- o Loading of programs
- o Job management
- o Formatting
- o File spooling
- o Command files
- o Wild card copying and deletion of files
- o Fast file copying (optionally selective)
- o Listing directories in alphabetical order
- o Label printing
- o Mail merging

Below is a brief summary of the content of each section in this manual:

Section 1 - is this introduction;

Section 2 - lists the components of the *Task Swopper* package;

Section 3 - gets you started and introduces the main programs;

Section 4 - contains a comprehensive description of *Task Swopper's* facilities;

Section 5 - describes how to use *Task Swopper* with floppy discs;

Section 6 - explains how to customise *Task Swopper;*

Section 7 - discusses how to write your own programs making use of *Task Swopper;*

Section 8 - describes *Task Swopper's* true multitasking capability and how it provides an optimum solution;

Section 9 - describes the format of each of the new commands added to QL Basic by *Task Swopper;*

Section 10 - is a help section designed to sort out any problems you may encounter.

2. What You Get

With this manual you should have received a single microdrive cartridge containing the following files:

BOOT - a simple program to load *Task Swopper;*

SMALL_BOOT - a version of BOOT to load SMALL_SWOPPER;

BIG_BOOT - a version of BOOT to load SWOPPER;

SWOPPER - this contains the machine code that does the hard work;

SMALL_SWOPPER - a cut down version for use when memory is limited;

MAIN_MENU - a configurable menu program for automatically loading and setting up swop-tasks when you reset your QL;

DRIVER_MENU - a program to let you select different printer drivers for use with the Psion programs;

CLOCK - a multitasking clock;

INSTALL - a program to automate the process of installing for use from floppy discs;

BACKUP - an easy to use general purpose backup program;

LAST_TIME - a file used by MAIN_MENU to store the last setting of the QL clock;

PREV_DRIVER - a file containing a description of the last installed printer driver;

LETTER_DOC - a quill document (for use in examples);

PRINTER_DAT - the current printer driver (updated each time you use DRIVER_MENU);

ABA_DRVR, ARC_DRVR, EAS_DRVR, QUI_DRVR, QUI_FX80_DRVR, QUICON_FX80_DRVR, RAWPC_DRVR - a selection of ready made printer drivers already set-up for use by DRIVER_MENU.

3. Getting Started

This chapter explains how to get *Task Swopper* up and running quickly, and introduces the BOOT, MAIN_MENU and DRIVER_MENU programs. This will show you the recommended way of using *Task Swopper* and its most important facilities. Once you have grasped the basics, you will be able to read the more detailed explanations given later, enabling you to get the most out of this extremely useful program.

First, we start with the most important step after buying a new program - making a working copy. Please **please** don't just press on and use your master cartridge - follow the instructions below closely.

3.1. Making a Working Copy

Before you can get started, you will need to make a working copy of *Task Swopper* on a new cartridge and can use the BACKUP program provided to do so. First, make sure you have a blank formatted microdrive to hand. Then, insert the *Task Swopper* master cartridge in *mdv1_* and type "lrun mdv1_backup". The backup program will load and ask you for the source device, so type "mdv1_". It will then ask for the name of the destination device, so insert your blank medium in *mdv2_* and type "mdv2_". The backup process will proceed, printing the name of each file as it is copied. When complete, remove the master and store it in a safe place.

3.2. The BOOT Program

Insert your newly created working cartridge in *mdv1_*, reset the QL and press F1 or F2 as normal.

The BOOT program will start automatically and will load *Task Swopper* before running the MAIN_MENU program. This is all the BOOT program does, and is kept separate from the MAIN_MENU program for reasons explained in Chapter 6.

3.3. The MAIN_MENU Program

The MAIN_MENU program will start by asking you to input the correct time. (It actually checks to see if the clock has already been set by comparing the year of the QL *date* function with the value stored in the file called LAST_TIME, held on your *Task Swopper* cartridge.)

If the clock is wrong, you are prompted to input the correct year, month, day etc. To minimise typing, you need only type in new values if they have changed since the last time the clock was set-up. Once the time has been set, the program presents the main menu for selection of swop-tasks.

You will notice that if you reset your QL after having set the time, you won't have to set the clock up again unless you switch off in the meantime. Instead you will be presented immediately with the swop-task menu. The swop-task menu provides the following types of option:

to set-up different combinations of swop-tasks (options 1 to 8);
to return to Basic (option 9);
to load and run the floppy disc installation program (option 0).

The options are numbered 0 to 9, with some blank ones for you to fill in later.

The first of the set-up swop-task options (number 1) has been designed for users without a memory expansion whereas the other "swop-task" options require expanded memory in order to work. (Chapter 6 describes how to configure the options for your own system).

If you select a swop-task option (1 to 8) the programs listed will be set-up, a multitasking clock program started and then the DRIVER_MENU program will be run. If instead, you choose to return to Basic (option 9) you will be able to load swop-tasks manually as described in chapter 4, and of course can use Basic as normal.

The installation program (option 0) is used to configure *Task Swopper* for use from floppy discs as explained in chapter 5.

For the purposes of this introductory session, select option 1 by pressing key "1" even if you do have a RAM expansion. You will be prompted to insert Quill in mdv1_, so remove the *Task Swopper* cartridge and insert your Quill cartridge. (Users with floppy discs will learn how to load programs without swopping media later.) When you press <ENTER> to continue, Quill will load and a mock letter head will be read in automatically. After a short delay, you will be asked to re-insert the *Task Swopper* cartridge, so do this and press <ENTER> again. A multitasking clock will appear in the bottom right hand corner, showing hours, minutes and seconds, and finally the DRIVER_MENU program will be run.

3.4. The DRIVER_MENU Program

You will be presented with a menu in a similar format to the start-up menu, but this time for selection of printer drivers. The menu contains several drivers supplied with *Task Swopper*. Chapter 6 explains how to replace these with your own. At the bottom of the menu window, a description of the currently installed driver is given, and in the lower right hand corner the amount of free memory is displayed in Kilobytes.

Try selecting one or two different printer drivers and observe that some activity occurs on drive 1, and that the new selection is displayed in the window. If you select option 0, the screen will clear and a message explaining how to resume the printer driver menu will appear in channel 0. This is done by typing "run", which will cause the familiar menu to reappear.

If you press *SH/ALT/F2* (ie hold the SHIFT and ALT keys down while pressing F2) Quill will be selected and will refresh its screen automatically. The change-over to Quill is slow because *Task Swopper* has used the memory saving options enabling you to use it in an unexpanded QL. (Later, you will learn how to change the MAIN_MENU program to set-up swop-tasks to use *Task Swopper's* many features, including instant screen swopping, to best suit your particular QL configuration and the way you like to work.)

Once in Quill, you should see the mock letter head which you can now modify to contain your own name and address. Having done this, save it and press *SH/ALT/F1* to return to the printer driver menu. Select a suitable driver, return to Quill by pressing *SH/ALT/F2*, and if you wish you can print the letter head out. (If this were a long document, you could return to the menu by pressing *CTRL/SH/ALT/F1*, and select option 0 to return to Basic so that you can do something else while Quill is printing. The CTRL key prevents *Task Swopper* from suspending Quill.) If you had loaded additional swop-tasks, these would be selected using *SH/ALT/F3*, etc. Basic is always selected with *SH/ALT/F1*. Chapter 4 explains task selection in more detail, and in particular how to select from up to ten tasks when there are only five function keys! It also explains how to add and remove swop-tasks without the use of the start-up menu. When you have several swop-tasks in use, the printer menu program is particularly useful for ensuring the correct printer driver is available for the program you are using. Different printer drivers can also be used to select different printer founts and character pitches.

If you have expanded memory, try re-booting your QL and selecting one of the other combinations of swop-tasks from the start-up menu.

4. A Closer Look At Task Swopper

This chapter describes *Task Swopper's* facilities in detail by explaining each of the new commands added to Basic. This will familiarise you with *Task Swopper's* capabilities and enable you to decide how best to use it with your system.

4.1. Loading Task Swopper

In this chapter, manual use of commands is explained, but before you can try any of the examples you need to load *Task Swopper* into the QL. You can do this by inserting your working copy into *mdv1_*, reseting the QL and then pressing F1 or F2 as described already. You will eventually be presented with the start-up menu, and can select option 9 to return you to Basic. (In all the examples, you will need to press <ENTER> at the end of each line, but this has been omitted for clarity.) Instead of loading *Task Swopper* automatically, you could have typed:

```
swopper = respr(7500)
lbytes mdv1_small_swopper, swopper
call swopper
```

Or if you have additional memory:

```
swopper = respr(10000)
lbytes mdv1_swopper, swopper
call swopper
```

(See the note in section 4.7 regarding small_swopper)

Once this has been done, the following commands will have been added to Basic:

exec_s	filename[,size][,flag]
exec_sr	filename,size,flag,delay,command_string
clone_k	key[,size][,flag]
clone_kr	key,size,flag,delay,command_string
exec_b	filename
heap	
respr(bytes)	
nrespr(bytes)	

Throughout this manual, square brackets '[]' are used to denote optional parameters. Thus, the *exec_s* command must always be given the *filename*, but can be given none, either or both of *size* and *flag* parameters. In addition, the commands *es, esr, ck, ckr, ckr,* and *eb* are allowed as abbreviations for *exec_s, exec_sr, clone_k, clone_kr* and *exec_b* respectively. (*es* and *esr* are mnemonics for "exec swop-task" and "exec swop-task and return" respectively. *ck* and *ckr* stand for "clone from key" and "clone from key and return" respectively. *eb* is a mnemonic for "exec background job".)

The *exec_sr* and *clone_kr* commands are provided for automatic loading of swop-tasks and so are covered in a later section. Their effect is essentially the same as the *exec_s* and *clone_k* commands explained shortly, but they have additional features useful for automatic loading.

Chapter 9 describes the full function and syntax of commands and you will find it necessary to refer to it whilst reading this chapter, which explains how to use *Task Swopper* in a step by step fashion using examples.

4.2. Starting a Swop-task Using EXEC_S

When invoking a Psion package as a swop-task on an unexpanded QL, the *flag* parameter will always be given as -3. If not, there will not be enough memory for Basic, *Task Swopper*, and the swop-task and an "Out of memory" error will result. Smaller programs (than Quill, Abacus etc) may leave enough memory to take advantage of *Task Swopper's* nicer facilities which are described shortly, and accessed using different values for the *flag*. Examples:

```
exec_s      'mdv1_assembler'
es          'mdv1_editor',82
exec_s      'mdv1_quill',85,-1
es          'mdv1_abacus',80,-2
exec_s      'mdv1_archive',-3
```

Note that the file name should always be enclosed in quotation marks. Any of the above will cause the relevant program to be loaded in apparently identical fashion to if the *exec_w* command had been used thus:

```
exec_w      mdv1_progname
```

When the *size* parameter has been specified (eg as 82, 85 and 80 above), the program will only be allowed to grab the given number of kilobytes of memory. If no size is given, the program can grab as much or as little as it wants. For most programs, this parameter is unnecessary because they only need a fixed amount of memory. The Psion programs however, will grab more memory if it is available, and Quill will grab almost all the memory it can, leaving you no room to load any other programs!

The *flag* parameter is a useful memory saving feature. If unspecified, *Task Swopper* will reserve a whole 32K of RAM for each new swop-task, so that the screen contents can be

saved when another swop-task is active. An additional 32K will also be reserved for Basic's screen when the first swop-task is activated. This is just too expensive in memory terms on a 128K QL and so the *flag* enables you to forfeit the speed of task swopping in favour of using less memory. It will also be useful in any situation where you don't want to be extravagant with memory.

The *flag* can be used to save RAM in three ways, by giving it as -1, -2 or -3:

-1 Will prevent 32K of RAM being allocated for the new swop-task's screen;
-2 Will prevent 32K of RAM being allocated for Basic's screen;
-3 Has the combined effect of -1 and -2.

4.3. Cloning a Swop-Task

Providing you don't run out of memory, it is possible to add up to nine swop-tasks using the *exec_s* command as described above. However, if you want to have more than one copy of the same program running, it is possible to "clone" it using *clone_k*. Cloning has the advantage of not loading in a second copy of the program which means that the second copy only requires additional memory for its data.

Cloning will work with all properly written "re-entrant" jobs, which includes Quill, Archive, Abacus and Easel, and most other programs normally started using *exec* or *exec_w*.

For example, the minimum amount of RAM required to set up your first copy of Quill is 74K, excluding any memory reserved for screen storage. Of this, 52K is program code. By cloning the first copy of Quill using *clone_k*, instead of simply adding a whole new copy with *exec_s*, you will save 52K of memory.

Swopping between tasks is achieved using the SHIFT, ALT and function keys. This is explained in the next section, but the important point in order to understand how to clone a swop-task, is that each swop-task is assigned to a function key when it is started.

clone_k is a mnemonic for "clone key". It is used in the same way as *exec_s* explained already, except that the first parameter is a number from 2 to 10, indicating the key of the task you wish to clone, instead of a file name. Examples:

```
clone_k     2
ck          3,30
clone_k     10,-3
```

Unfortunately you can't clone Basic, which is why you can't clone from key number 1. The meaning of key values 6 to 10 will become clear shortly.

Warning

If you remove a swop-task (by quitting from the program) that has been cloned from, all the cloned tasks will be removed as well. This must happen, because there is only one copy of the program code. You can remove clones without affecting the original copy of course. Hence be careful when quitting from tasks if you have any clones in the system, otherwise you may end up losing your work.

4.4. Swopping Between Tasks

First we will deal with the case where no *flag* has been used, and so all swop-tasks including Basic, have been allocated 32K of RAM for preservation of their screens.

When you start the first swop-task, it will be automatically assigned to function key F2. Basic is always on F1. When the swop-task is first started it will perform exactly as if you used *exec_w*. However, if you if you press *SH/ALT/F1* (ie hold the SHIFT and ALT keys down and press F1), Basic's screen will instantly re-appear and you will be able to load and run Basic programs, copy files et cetera, just as normal. Pressing *SH/ALT/F2* will then return you to the swop-task. Because the *flag* parameter has been left out, swopping is instantaneous, and all screens will be restored exactly as they were when you left them.

As subsequent swop-tasks are started, they will be assigned to the next free function key, ie first F2, then F3 through to F5, and that key, together with the SHIFT and ALT keys will be used to select the corresponding swop-task. Up to nine swop-tasks can be added, so what happens when four swop-tasks have been added and F5 has been used? The fifth swop-task will be selected by pressing SH/ALT/F1 (as you would for Basic) but keeping it held down until the screen changes.

So, the first four swop-tasks are selected by pressing and immediately releasing one of *SH/ALT/F2* to *SH/ALT/F5*, with Basic always on *SH/ALT/F1* The next five swop-tasks are selected by pressing one of *SH/ALT/F1* to *SH/ALT/F5* and holding the keys pressed until the new task is selected. (NB If you keep the keys pressed for too long after the screen changes, you will actually be flipped back to the job selected by just pressing and releasing the key combination.)

When adding several swop-tasks you will find it useful to know how much memory you have spare. This can be found by returning to Basic and typing "PRINT HEAP". *Heap* is a new function, added to Basic by *Task Swopper*, which returns the size of the largest single chunk of free heap in kilobytes. Also note that the DRIVER_MENU screen shows the current amount of free memory in the bottom right hand corner of the menu window.

Normally, you will swop between basic and swop-tasks by pressing the SHIFT, ALT and function keys as described above. When the new job is selected, *Task Swopper* deliberately disables the old job to prevent it from corrupting the new job's screen. However, it may be useful to be able to leave the old job running, to print from one task whilst using another for instance, and this can be done by holding CTRL down as well as the SHIFT and ALT keys. This causes the job which you are leaving to remain active.

But **beware;** leaving jobs running in the background can lead to confusing situations, and not only because the screen can become corrupted. Further confusion can arise when a job finishes some background activity and decides to look for keyboard input, it will very often snatch the cursor away from the job you are currently using which is disconcerting. To get the cursor back where you want it, you need to press the relevant *SH/ALT/F* combination. If you are using a Psion package whose screen has become corrupted, you can easily refresh it by pressing *SH/F5*. If you are trying to retrieve Basic's cursor, and *SH/ALT/F1* seems to have no effect, pressing *CTRL/C* or *CTRL/SPACE* should bring it back.

4.5. The Effect of the Flag Parameter on Task Swopping

If the *flag* is left out when a swop-task is started, swopping between tasks is instantaneous, with each task's screen being preserved when a different task is active. As described in section 4.2, the *flag* can be used to save memory, but at the expense of slower task swopping. If the *flag* has been used to prevent Basic from having memory allocated for storage of its screen, when you swop back to Basic *Task Swopper* will clear away the swop-task's screen by simulating typing of "cls #0:cls #1:cls #2" to Basic.

(NB. This will not clear all of the screen, but is a compromise because *Task Swopper* does not know how Basic's windows are organised, and so cannot fully refresh them using *border* etc. The *mode* command won't do any better because it will start by refreshing Basic's windows, and then will refresh the swop-task's windows, ruining Basic's windows in the process.)

If you have used the *flag* to prevent allocation of memory for saving of the swop-task's screen, *Task Swopper* will simulate the typing of *SH/F5* when the swop-task is restarted. If the swop-task is one of the Psion packages, this will have the effect of telling the program to refresh its screen.

4.6. Task Swopper and Ordinary Jobs

Task Swopper will not prevent you from running ordinary jobs which multitask properly on the QL, ie those that are run with the *exec* command. Such jobs can be "exec'ed" from Basic in the normal way, and will be available as part of Basic's screen. When Basic is selected, you will be able to use *CTRL/C* to select which of these ordinary jobs you want, just as you would normally. If you wished, there would be nothing to stop you running an ordinary job as a swop-task, enabling you to take advantage of *Task Swopper's* screen preservation facilities.

In the same way that Basic will be suspended when you leave it by selecting a swop-task, most ordinary jobs will also be suspended until you return to Basic unless you used the CTRL key when you left Basic, to keep them running. However, *Task Swopper* is clever enough not to suspend jobs which have not opened any screen windows of their own, because they should not overwrite a swop-task's screen. This is useful when you are using something like a spooling program, which you want to carry on running in the background whichever swop-task is selected. See also *exec_b* described next.

The *exec_b* command has exactly the same effect as *exec* except that any jobs created with it become background jobs. Jobs created with *exec* are suspended by *Task Swopper* when you leave Basic unless you use the CTRL key as explained above, but background jobs are never suspended by Task Swopper. So *exec_b* is useful for setting up jobs such as a multitasking clock which you want to remain running no matter which swop-task you are using. A multitasking clock program is included as part of *Task Swopper* and can be run as a background job using the command "exec_b 'mdv1_clock'".

4.7. A Smaller Version of Task Swopper

In addition to the normal version of *Task Swopper*, a smaller version has been provided for use when memory is particularly tight. The smaller version is identical in its capabilities to the normal version, except that it can support fewer swop-tasks and background jobs. It is loaded using the following commands, or the program SMALL_BOOT provided:

```
swopper = respr( 7500 )
lbytes mdv1_small_swopper, swopper
call swopper
```

4.8. Task Swopper and RAM

In an earlier chapter, you have already seen how the *flag* can reduce the amount of memory used by *Task Swopper*. This section gives further details on what RAM *Task Swopper* will use, how much RAM the Psion packages use and the effects of different values of the *flag*.

If when you use the *exec_s/clone_k* commands to add a swop-task, no *flag* is given, then in addition to letting the swop-task have access to the amount of RAM you have decided to allocate, *Task Swopper* will use 32K of RAM for Basic's screen and 32K for each swop-task's screen. This is the most convenient method of using *Task Swopper* if you have enough RAM, because swopping is very fast indeed.

The *flag* can be used to reduce the RAM used by *Task Swopper* in three ways, by giving it as -1, -2 or -3:

-1 Will prevent allocation of 32K for this swop-task's screen;
-2 Will prevent allocation of 32K for Basic's screen;
-3 Will have the combined effect of -1 and -2.

When the *flag* parameter has been specified, *Task Swopper* compensates for the loss of screen contents as follows. When returning to Basic, it simulates the typing of "cls#0:cls#1:cls#2" to clear the three main windows of Basic. When returning to a swop-task whose screen has not been saved, it simulates pressing SH/F5 which causes a Psion package to refresh its own screen fully.

When trying to decide the best usage of RAM for your own machine the following information may be useful (the figures shown are based on using small_swopper on a 128K QL):

Psion package	Version	Code size		Data size		Total job size	Minimum RAM required	Free heap on 128K QL	Package workspace
Quill	2.00	51752	+	1280	=	53032	74k	9k	-
Abacus	2.00	51406	+	1280	=	52686	66k	16k	3k
Archive	2.00	51474	+	1280	=	52754	72k	14k	5330
Easel	2.00	62210	+	1280	=	63490	76k	6k	2k

Thus the smallest value of the *size* parameter you should specify when invoking Quill is 74. If you used this value for *size* and specify the value of *flag* as -3 on a 128K QL, 9k of heap will be left over for use by Basic. The rightmost column of the table shows the amount of RAM that each package will have for storing its own data if the given minimum values of *size* are used. Note that if you do give these minima, you won't leave the respective Psion package with much RAM for itself.

You can see why you need to specify the *flag* as -3 on a 128K QL. Before you start, there will only be about 90K of free memory available (immediately after reset), because 32k is used for the QL's screen, and a few "K" for system variables and Basic. Loading "swopper" takes 10k, (or 7.5k using "small_swopper") and the smallest Psion package needs 66k. This leaves only 90k-(66k+10k)=14k which is simply not enough to store a copy of either Basic's or the swop-task's screen.

An additional 128K memory expansion would leave 14k+128K free after loading Abacus and letting it have 66k. So you could afford to use 32k+32k of this for the Basic and Abacus screens. This would then leave 14k+128k-64k=78k free. With this you might simply decide to let Abacus have some more RAM for its own use, because you can't do much with the minimum 3k (see table), or you might decide to load in another Psion package. The second Psion package would have to be loaded with a *flag* value of -1, because there will not be enough RAM in the spare 78k for the package and another 32k of RAM for its screen. Hence swopping to Abacus would be very vast, but swopping to the second Psion package would take longer as it will take time to refresh its screen.

The above may need reading through a few times, but I hope it will enable you to see how to juggle the use of available memory to suit your needs.

4.9. Removing Swop-tasks and Releasing Memory

When you quit a swop-task (by abandoning Quill for instance), RAM is automatically released by the QL, but may be invisible to the *heap* function described in section 4.4. (In practice, most rather than all of the task's RAM will be released. This is a fundamental problem caused by the fact that the Psion programs do not properly release the RAM that they use, and so you will not get all of it back until you next reset the QL.) The RAM released may be invisible to the *heap* function if you still have a swop-task (or ordinary job) loaded that was

set-up after the one you have removed. This happens because the remaining swop-task sits between the area of memory released by the abandoned task and the main body of free memory. The memory released by the swop-task is in fact free but does not increase the value returned by the *heap* function because it is not contiguous with the unused free memory. If the newly liberated block of memory is bigger than the previously largest block of free memory then the size of the new block will be returned by *heap* (because *heap* returns the size of the largest block of free memory).

When a swop-task has been abandoned and you have moved on to another swop-task using a *SH/ALT/F* press, trying to return to the abandoned swop-task will have no effect. Other swop-tasks will be unaffected and when a new swop-task is started, it will be assigned to the first available *SH/ALT/F* key slot, so filling in any gaps left by abandoned swop-tasks.

4.10. Automatic Loading of Swop-tasks

The *exec_sr* and *clone_kr* commands have been specifically provided to enable you to construct a simple Basic program to automatically load and initialise swop-tasks each time you reset your QL. You can either write your own programs to do this, or simply modify the start-up program, MAIN_MENU, allowing you to select different combinations of swop-tasks.

Two extra parameters have been added to the new commands, and unlike the *exec_s* and *clone_k* commands, all parameters are compulsory. This means that the *flag* parameter must always be given, and so in addition to the values of -1, -2 and -3, a value of 0 is also acceptable which does not prevent the allocation of RAM for screens.

The two new parameters are *delay* and *command_string*. *Delay* is a number which defines how long the swop-task will be left running after it has been loaded, before it is suspended and control is returned to Basic. By returning to Basic, further swop-tasks can be loaded automatically by the program. The purpose of the delay is not only to let the swop-task set up its screen, but also to give it enough time to obey the instructions given to it by the second new parameter, the *command_string*. The *command_string* is a string of characters which will be sent to the swop-task immediately after loading, just as if they were typed at the keyboard. This can be used to load your default document into Quill which can have your address in the top right hand corner and all the margins set up as you require. Alternatively, it can be used to run an Archive program which opens up a particular database for you, or load a spreadsheet into Abacus and so on and so forth.

You will need to find the optimum value of *delay* by trial and error, because it will depend on which program is being started, how long the *command_string* takes to be obeyed, and whether you are using microdrives or discs.

Examples of using the *exec_sr* and *clone_kr* commands are given below, but assume that the values of some Basic variables (ENT$, F3$ etc) have been defined as in the MAIN_MENU program explained in chapter 6. For example:

```
exec_sr   'flp1_quill',100,0,900,F3$&'lflp1_letter'&ENT$
```

would load Quill from floppy disc drive 1, allowing it 100k of RAM plus 32k for its screen plus 32k for Basic's screen, and would instruct it to load a default document called "letter" from *flp1_*. The *delay* of 900 should be just sufficient to allow Quill to do this before it is disabled. Subsequently:

```
esr         'flp1_archive',120,0,900,'run "flp1_arcinit"'&ENT$
```

would load Archive, allowing it up to 120K of RAM plus 32K for its screen, and instruct it to run an Archive program called "arcinit". Then:

```
clone_kr  2,30,0,0,''
```

Would create a second copy of Quill (which is already on function key F2), using the same program code as the first. This would also be allowed 32K for screen storage, but would only take 30K of data space.

5. Installation For Use From Floppy Discs

Task Swopper is easily set up for use with floppy discs, and an installation program has been provided to automate the process. Ensure that you have a blank formatted disc ready, and insert it in disc drive 1. Insert your *Task Swopper* cartridge in *mdv1_* and type "lrun mdv1_install".

You will be asked for the name of the floppy (or hard) disc on which to install *Task Swopper*, so type *flp1_*, or *fdk1_* as appropriate. (Hard disc users should include any directory prefix, for example "hdk1_swopper_", but note that all programs to be run by *Task Swopper* will need to be held in the same directory.)

The files will be copied from the *Task Swopper* cartridge and listed on the screen. The BOOT, MAIN_MENU and DRIVER_MENU programs will be modified during the process, so that they will access the floppy disc when looking for programs to load or printer drivers to install. MAIN_MENU will also be changed to prevent it from asking you to swop media when loading swop-tasks, because you should be able to fit all the programs onto your *Task Swopper* disc.

Next, copy any programs you want to use with *Task Swopper* onto the disc, including Quill, Archive, Abacus and Easel. You can use the backup program described in section 3.1 to copy all the files on a microdrive cartridge to the disc, or you might prefer to copy the files manually, enabling you to save space by only copying those files that are absolutely necessary. For instance, unless you use the on-line help feature of the Psion packages (accessed by pressing F1), you only need the main program files (Quill, Archive, Abacus and Easel) to be copied to the *Task Swopper* disc. If you do use the on-line help, then also copy the file ending with "_HOB" from each cartridge. The Psion INSTALL_BAS and CONFIG_BAS programs need not be on the *Task Swopper* disc, but won't do any harm - they just take up space.

If you have not already configured the Psion programs for use from discs, you should do this now, by running the program CONFIG_BAS found on your Abacus cartridge.

6. Customisation of Task Swopper

This chapter describes how to customise the programs described in chapter 3 to provide alternative combinations of swop-tasks and printer drivers.

Before attempting to customise your system in this way, you should make sure that you understand the function of these programs and *Task Swopper's* commands by reading chapters 3 and 4. Whilst making changes you will also find it necessary to refer to chapter 9 which describes the syntax of commands.

6.1. The BOOT Program

The BOOT program simply loads *Task Swopper* and runs MAIN_MENU. On the cartridge supplied, BOOT is a copy of SMALL_BOOT which actually loads small_swopper saving 2.5K of RAM. A similar program called BIG_BOOT is provided and can be used to replace BOOT if you have a memory expansion. BIG_BOOT loads the full sized swopper. The small version of *Task Swopper* is identical to the normal version, except that it only supports five swop-tasks (including Basic) and fewer background jobs. To make sure you are using the full version of *Task Swopper*, type the following (users without a RAM expansion should not do this):

```
delete    mdv1_boot
copy      mdv1_big_boot to mdv1_boot
```

6.2. Modifying the MAIN_MENU Program

This program is run by the BOOT program after *Task Swopper* has been loaded as described above. The version supplied includes the following options:

1 - Quill only
2 - Quill + Archive
3 - Quill + Quill
4 - Quill + Archive + Abacus + Easel

Options 5 to 8 are blank and can be added to. Options 9 and 0 should be left unchanged.

Option 1 is configured to load Quill into a standard 128K QL, whereas 2, 3 and 4 will require a memory expansion in order to work.

List the MAIN_MENU program and identify the lines which print out the menu options listed above. You can modify the menu printed to the screen by altering the print statements. If this is to make sense, ie to make the program do what your modified menu offers, you will also need to to modify the subsequent lines which contain the *exec_sr* commands to reflect the changes made. For example, to change option 1 to set-up Quill and Archive, and to use *Task Swopper's* fast screen update facility, do the following:

First modify the menu string from "1 - Quill only" to "1 - Quill and Archive". Next, modify the group of statements:

```
=49: REMark 1
swop_cart 'Quill'
EXEC_SR dev$&'quill',75,-3,3500,F3$&'lletter'&ENT$
swop_cart ts$
mult_clock
LRUN dev$&'driver_menu'
```

(Line numbers have been omitted as these may change from version to version.) Change the above to read:

```
=49: REMark 1
swop_cart 'Quill'
EXEC_SR dev$&'quill',100,0,3500,F3$&'lletter'&ENT$
swop_cart 'Archive'
EXEC_SR dev$&'archive',100,0,3500,''
swop_cart ts$
mult_clock
LRUN dev$&'driver_menu'
```

Two new lines have been added, and the existing *exec_sr* statement has been modified. The modification has allowed Quill 100k instead of 75K of memory, and changed the *flag* parameter from -3 to 0. We can give Quill more RAM, as you must have a RAM expansion in order to run Quill and Archive simultaneously, and changing the *flag* enables fast screen updating. The two lines following the existing *exec_sr* statement are new. The first causes the user to be asked to insert the Archive cartridge before any attempt is made to load (installing for floppy discs disables this). The next *(exec_sr)* statement then loads Archive in a similar way to Quill, but has a null (ie '') *command_string*. A null *command_string* prevents any automatic set-up of Archive. Alternatively, the string could have been 'run'&dev$&'arcinit'&ENT$ to load and run an Archive program called "arcinit_prg". (Refer to the QL User Guide for details of how to write Archive programs - but note that they can be used to open up your database etc).

Referring to chapters 4 and 9 should enable you to modify the *exec_sr* statements in MAIN_MENU to suit your needs.

6.3. Modifying the DRIVER_MENU Program

When you print files from Quill, Abacus or Archive, a file called PRINTER_DAT is used to determine the printer control codes to be used for selecting different print modes and so on. In the past, you will probably have used the program called INSTALL_BAS supplied on the Quill cartridge to install different printer drivers. This was fine when you were using only one program at a time, but *Task Swopper* enables you to swop between these programs and you may well want to use a different printer driver with each program. The DRIVER_MENU program enables you to do this. Changing printer drivers will also allow you to select different printer founts or character sizes on your printer. Details of how to select printer drivers was given in section 3.4 earlier. Here, we describe how to modify the DRIVER_MENU program to use printer drivers you have created yourself.

6.4. Creating a Printer Driver

Insert your Quill working microdrive cartridge in *mdv1_* and type "lrun mdv1_install_bas" and create and install the printer driver you require as you would normally, and as explained in the QL User Guide. The installation process will create a new version of the file PRINTER_DAT for the printer you have described.

Next, copy the newly created printer driver to your *Task Swopper* working cartridge, but at the same time, changing its name to something reflecting its function. For example, you might type "copy mdv1_printer_dat to mdv2_fx80condensed_drvr". This might be a driver for the Epson FX80 printer, which puts the printer into condensed print mode. You could also create a driver to return the printer to normal mode in which case you could call it "fx80normal_drvr".

Repeat the above process until you have created all the drivers you require. With *Task Swopper* you will find some ready made drivers, and the DRIVER_MENU program has been configured to use these, with some spare places in its menu.

6.5. Using Your Own Printer Drivers with DRIVER_MENU

To use the printer drivers created by the above procedure, you will need to modify the DRIVER_MENU program. Examine the program to see how the menu options correspond to the action taken by the program when a key is pressed to select a particular driver. The menu gives a brief description of the available drivers, and when one is selected, the existing PRINTER_DAT file is deleted and replaced with a copy of the new driver. The variable *last$* is also updated with a description of the new driver, and this will be displayed on the menu screen.

To add one of your own drivers to the DRIVER_MENU program, you need to modify the *data* statements at the beginning of the program. You should not change the number of *data* statements, by adding or deleting lines, nor should you change the number of parameters in a *data* statement. Each *data* statement contains two parameters, first a string which describes the printer driver, and then the name of the corresponding driver file. You can either modify the existing strings and driver file names, to use your own drivers, or change the "empty" values in the last two *data* statements which do not have proper driver descriptions or filenames.

7. Writing Programs to Use Task Swopper

This chapter offers some guidance for those wishing to write their own programs using *Task Swopper's* facilities.

You may be wondering why a separate BOOT program is used to load *Task Swopper*, rather than including this at the start of MAIN_MENU. This is necessary because of a QL bug. Versions of the QL earlier than JS, including AH and JM, will sometimes fail to initialise new Basic procedures that are referenced in a program that contains references to those procedures. For example your boot program will probably be similar to:

```
100  swopper = respr(10000)
110  lbytes mdv1_swopper, swopper
120  call swopper
130  lrun mdv1_load_stasks
```

The file "load_stasks" is intended to be a Basic program which contains *exec_sr* and *clone_kr* statements to load the swop-tasks.

In addition to loading swop-tasks, your program can be used to set up background jobs such as the multitasking clock program included with *Task Swopper*. When run, this program provides an hours minutes and seconds display in the bottom right hand corner of the screen. To activate it, you should use the following (channel 0's window is adjusted to stop it scrolling the clock display upwards):

```
exec_b 'mdv1_clock'
window #0,458,50,0,206
```

When writing your own programs, you will probably find it useful to study the programs provided for automatic set-up of *Task Swopper* described in chapters 3 and 6.

8. True Multitasking With Task Swopper

The benefits of multitasking are pretty obvious - essentially allowing you to leave a program to get on with one job, whilst you carry on using another to do something else. More importantly, you can swop quickly from one program to another and back again without losing your work. The most common examples include using one copy of Quill to print a document, which may take several minutes, and using a second copy of Quill to write another. Background printing (from Quill, Abacus, Easel, Archive or any other program come to that) is a very common example of an activity that would tie up a non-multitasking computer, preventing you from doing anything useful until it has finished. There are many other examples of course, but could there possibly be any disadvantages to multitasking?

8.1. Disadvantages of Multitasking

Most of the time, you won't actually want your programs to multitask, because when two programs are really running concurrently, both will be slowed down dramatically. The more programs you have running, the slower they will be. 90% of the convenience of multitasking on the QL is probably realised through the ability to swop between programs instantly, rather than by enabling them to run concurrently.

Another problem with true multitasking is that if several jobs are sharing the same screen, the display can become confused unless the programs have been written with multitasking in mind. Unfortunately, few QL programs have been written to multitask properly. Providing a mechanism for restoring the display of one job that has been corrupted by another concurrent job, can be expensive in terms of memory and further slow the QL.

Task Swopper has a unique approach which gets round the disadvantages of multitasking without losing any of the benefits. *Task Swopper* suspends all jobs (with certain special exceptions) except the current swop-task, preventing inadvertent corruption of job screens. This means that all tasks run at full speed, no matter how many you have in memory, but enables you to switch instantly between them. The exceptions to this feature have been carefully designed to preserve the full benefits of true multitasking.

The first exception involves use of the CTRL key to exit from a swop-task without suspending it. (See also the end of section 4.4.)

8.2. Background printing

Thus, to have Quill printing a letter in the background while you continue to use another program (or return to Basic) the procedure would be: Select Quill with the relevant *SH/ALT/F* combination. Load (or write) the relevant document and set it printing. Then, leave Quill holding CTRL down in addition to the relevant *SH/ALT/F* combination for the program you want to use. When Quill finishes, it will overwrite part of the screen and probably snatch the input channel from the program you are using. To retrieve the input channel from Quill, simply press the *SH/ALT/F* combination for the current swop-task. If you are using one of the Psion packages, you can also refresh the corrupted screen by pressing *SH/F5*. In addition to being able to prevent suspension of swop-tasks by using the CTRL key, there are two more ways in which jobs can be left running.

The first is through an intelligent feature of *Task Swopper:* Section 4.6 explains how jobs that have not opened any screen windows of their own are assumed not to write to the screen and hence are not suspended. This means that any multitasking document spoolers or similar programs will continue to run which is obviously desirable.

The second mechanism is through use of the *exec_b* command. This is equivalent to *exec*, except that any jobs set up by *exec_b* will never be suspended by *Task Swopper*. This is useful for starting a multitasking clock or similar program that you want to continue running regardless of which swop-task you are using.

Thus *Task Swopper* allows access to all the major benefits of multitasking without gobbling up massive amounts of memory, or needing an over complex user interface. Task swopping could not be much simpler or quicker, and has been deliberately implemented without the use of menus which were deemed inappropriate in this application. Menus would have taken up significant extra memory and increased both the time taken and number of key presses required for task swopping.

9. Details Of Additional Commands

Task Swopper adds the following commands to QL Basic:

EXEC_S, ES
EXEC_SR, ESR
CLONE_K, CK
CLONE_KR, CKR
EXEC_B, EB
HEAP
RESPR, NRESPR

9.1. EXEC_S, ES, CLONE_K, CK

Syntax:

exec_s	filename[,size][,flag]
es	filename[,size][,flag]
clone_k	key[,size][,flag]
ck	key[,size][,flag]

Parameters:

filename - (*exec_s* and *es* only) name of an execable file enclosed in quotation marks, including device name, as would be expected by the Basic command *exec_w*.

key - (*clone_k* and *ck* only) the number (2 to 10) of the function key assigned to the swop-task you wish to clone.

size - the number of kilobytes you wish to be available to this swop-task, excluding any RAM used for saving screen images.

flag - A flag to indicate the use of a memory saving feature:

-1 prevents allocation of 32K RAM for this swop-task's screen;
-2 prevents allocation of 32K RAM for Basic's screen;
-3 has the combined effect of -1 and -2.

Function:

exec_s will set-up a swop-task on the next available function key. Up to nine swop-tasks (plus Basic) may be in the QL at once, providing there is sufficient RAM available. *es* is an abbreviated form of *exec_s*.

clone_k is equivalent to *exec_s* except that the new swop-task uses the same program code as the one already loaded. This means that the second and suentuent copies of any swop-task need not take up as much memory as the first. For cloning to work, the first swop-task must be written to be properly re-entrant, which includes the four Psion packages supplied with the QL. *ck* is an abbreviated form of *clone_k*.

The first four swop-tasks are selected by pressing one of *SH/ALT/F2* to *SH/ALT/F5*. Basic is always selected with *SH/ALT/F1*. The next five swop-tasks are selected by pressing one of *SH/ALT/F1* to *SH/ALT/F5* and holding the keys pressed until the screen changes. If the keys are held for too long after the screen change, the effect will be as if you had just pressed and immediately released the relevant *SH/ALT/F* key combination.

9.2. EXEC_SR, ESR, CLONE_KR, CKR

Syntax:

exec_sr	filename,size,flag,delay,command_string
esr	filename,size,flag,delay,command_string
clone_kr	key,size,flag,delay,command_string
ckr	key,size,flag,delay,command_string

Parameters:

filename, key and size - see *exec_s* above.

flag - see *exec_s* above, and note that an additional value of 0 is acceptable which has the effect of ensuring that *Task Swopper* will try to allocate 32K of RAM for the swop-task's screen and for Basic's screen.

delay - specifies the amount of time that the swop-task is to be left running before it will be suspended and control returned to Basic.

command_string - A string of characters that will be sent to the swop-task as if they had been typed on the keyboard.

Function:

These commands are very similar to *exec_s* and *clone_k* above, but have been provided so that swop-tasks can be loaded and initialised from a Basic program. Once the delay specified has

expired, the swop-task will be suspended, and the next command in the Basic program will be executed. The delay used is entirely up to you, but is intended to allow sufficient time for the swop-task to set-up its screen and open any files it requires, in addition to dealing with the command string (see below). If too short a delay is given, the swop-task may not finish any file operations it performs (eg Quill opens a def_tmp file), and may not be properly suspended. If this happens, some corruption of Basic's screen is likely to occur, so increase the value of the *delay*.

The *command_string* specified will appear to the swop-task to have been typed on the keyboard and is intended to allow default files to be loaded. If you do not wish to use this facility, a null string must be provided by including a pair of empty quotes thus: "".

9.3. EXEC_B, EB

Syntax:

exec_b filename
eb filename

Parameters:

filename - name of a file enclosed within quotation marks, that would normally be run using the *exec* command.

Function:

Creates a background job. The effect is identical to using *exec*, except that the job will not be suspended when a swop-task other than Basic is selected.

9.4. HEAP

Syntax:

variable = HEAP
PRINT HEAP

No parameters.

Function:

heap is a new function which returns the size of the largest free area in the common heap in kilobytes. It is useful for finding out roughly how much spare memory you have. To do this, type "PRINT HEAP".

9.5. RESPR, NRESPR

Syntax:

address = RESPR(bytes)
address = NRESPR(bytes)

Parameters:

bytes - the number of bytes of RAM you wish to reserve.

address - will be assigned a floating point value which is the address of an area of RAM of the size requested.

Function:

This version of *respr* replaces the existing *respr* command explained in the QL User Guide. It functions in apparently the same way, except that instead of allocating RAM from the resident procedure area, it allocates RAM from the common heap. This means that you will be able to use *respr* even after creating a job or swop-task. The existing *respr* command will not work once there is a job running, because jobs sit in the transient program area immediately below the resident procedure area, preventing the latter from being expanded.

nrespr has been provided as a way of getting round a QL bug. Some versions of the QL ROM fail to overwrite the existing version of the *respr* command resulting in a "Not complete" message when *respr* is used after a job has been started. The solution is to alter references to *respr* which suffer from this to *nrespr*.

10. What To Do If Things Go Wrong

This is a chapter to help sort out problems.

BAD OR CHANGED MEDIUM

If your *Task Swopper* cartridge has become corrupted and you do not have a backup cop (shame on you!) all is not lost. If you return the original cartridge to Compware within 30 da of purchase, quoting your invoice number, and enclosing a cheque for £1 handling, we w replace the cartridge. If the 30 day period has expired the charge will be the same as a ne version upgrade, which includes the latest version of the software and documentation. Conta us for pricing.

SCREEN CORRUPTION

You may find that at some time during your use of *Task Swopper*, one swop-task manages corrupt the screen of Basic or another swop-task. This can happen if a swop-task is not pro erly suspended when a new swop-task is selected. It is possible to do this deliberately, usi the CTRL key (see the end of section 4.3), or inadvertently by leaving a swop-task whilst it performing some operation which cannot be disabled - opening a file for instance.

Screen corruption of this type can be minimised by sensible use of *Task Swopper*, but in a case is only a minor inconvenience. The screen of a Psion package can be refreshed at a time, by pressing *SHIFT/F5*.

NOT COMPLETE - when using RESPR()

This may occur when *respr* is used after a job has been started. It happens with some versio of QL because the new version of the *respr* command fails to overwrite the existing one. T solution is to alter references to *respr* which show this problem to *nrespr*.

IF ALL ELSE FAILS

If you find that *Task Swopper* does not appear to be operating in the way described in t manual, please try and help us before writing or telephoning by:

Thoroughly reading this manual to ensure that the problem in not a simple misunde standing;