Resident Program Manager
Liberation Software

Version 1.0

Introduction

Resident Program Manager is a utility program for creating resident tiles. A resident file contains one or more programs or SuperBASIC procedures and functions designed to be permanently linked to the QL and thus available instantly when required.

RPM produces a resident file by copying one or more source files and some special interface code into a new file. The source file in this context can be any one of the following:

A files of SuperBASIC extensions
An EXECutable program
A Q_Liberator object program, possibly with externals
A SuperBASIC program
Data files (for sequential reading only)

RPM can produce two variants of resident file.

A RAM resident file is designed to be loaded into the resident procedure area at the top of the OL memory map. - Space must be first be allocated in this area using the RESPR function. The RAM resident file can then be loaded using LBYTES from an external medium (eg microdrive) by a BOOT program. When the start address is called once by a program, the routines in the resident file are linked into the QL operating system. They remain present until the system is next reset. This operation will be familiar to moat users as it is the normal way in which SuperBASIC extensions are added.

A ROM resident file is very similar to RAM resident file, but is designed to be copied to an EPROM. Such files have a header at the start which 0005 recognises during initialisation of the QL. The routines which are present are linked automatically and are immediately available.

There are some rules which must be followed for a program to be ROMable. The code must be position independent, should not be self modifying and there should be no data areas contained within the code itself.

Subject to these rules, RPM allows developers to produce prototypes of ROMs in a RAM based form for testing. This saves repeated EPROM blowing and erasing during the testing phase.

The way in which the various source file types are treated by RPM and the manner in which they can be used is described in the following sections.

RESIDENT SUPERBASIC EXTENSIONS

Files containing SuperBASIC procedures or functions are already effectively resident files. RPM makes it easy to put them into ROM and gives you the facility to put all the extensions which are normally loaded individually into one convenient file.

Extensions are used in precisely the same way as before, ie the new procedures and functions are linked into the SuperBASIC name table and can be called by interpreted programs or compiled programs.

Be wary of extensions which use an area within the code as a data area. Such extensions can only be used by one job at a time (ie they are not re—entrant) and can give wrong results when used in compiled programs. They will not work at all if put into ROM. If an extension requires a data area it should either use the buffer in the SuperBASIC area (start address in 0(A6)) or allocate its own space in the common heap.

RESIDENT EXECUTABLE PROGRAMS

Every executable program in a resident file has a user defined procedure name associated with it. This name and some other parameters are specified when the resident file is created. This procedure is linked to the name table in the same way as for extensions. When the procedure is called a new independent job is created to execute the resident program. No parameters can be passed in the procedure call.

Jobs can be started with the equivalent of an EXEC or EXECW call. In the latter case the program which calls the procedure is suspended until the resident program ends.

Not every executable program is immediately capable of running in a resident form. We distinguish 2 categories, pure programs and impure programs. A pure program is designed to be re—entrant so that several jobs can use the program code simultaneously, each with its own associated data area. Q_Liberator object programs from release 3.0 onwards are always pure.

Impure programs are more demanding about the environment they run in and must be treated specially by RPM. They may contain nasties like data areas within the code or address their data area in an invalid way as described below. Code produced by other QL compilers is often impure.

When a program **is** EXECed, QDOS allocates an area of memory for the new job which is created. The program code is loaded at the bottom of this area and a data area follows it of a size determined by the header of the program tile.

When a job is created by an RPM procedure call, the program code within the resident file is used. An area of memory sufficient to hold a short piece of bootstrap code and a complete data area is allocated. Thus the program code and its data are in separate memory areas. Some impure programs cannot handle this, probably because they use addresses calculated relative to the job base. All addressing in the code area will be PC relative in a pure program.

However RPM does provide a mechanism for making impure programs resident, but is it much less memory efficient than that for pure programs. A parameter can be set so that each time an impure program is activated by a procedure call, a complete copy of the code is made in RAM with the required data area immediately following. This mode of operation should work for all EXECutable programs, even those with data areas within their code.

RESIDENT SUPERBASIC PROGRAMS

SuperBASIC programs can also be made resident but in this case they are not activated as procedures. Instead a new serial device driver with a name chosen by the user is created. The familiar procedures to LOAD, LRUN or MERGE programs can be used to access this pseudo device and bring the SuperBASIC program into memory. The device is treated as read only and only supports sequential access.

This feature is primarily intended for loading short bootstrap routines because the inherent slow loading of the interpreter means that resident SuperBASIC programs will not load appreciably faster than from a real physical device such as a microdrive. The current version of RPM does not support the creation of resident QLOAD files which would remove this restriction, but of course the best way to make SuperBASIC programs resident is to compile them with Q_Liberator and load them as either extensions or EXECutable programs.

RESIDENT DATA FILES

Data files can be made resident in the same manner as SuperBASIC programs, by associating them with a device driver. They can be accessed sequentially by OPENing the device driver and the use of INPUT or INKEYS in the same manner as accessing a serial port. Such files are of course read only.

USING RPM

RPM is supplied as an EXECutable program produced using Q_Liberator. Before running the program please copy it from the supplied master to a working disk using the COPY command.

To run RPM, simply type

```
EXEC mdv1_rpm_obj
```

or if you have the Q_Liberator extensions loaded, QX rpm.

RPM opens a window then prompts for the name of a control file.

THE RPM CONTROL FILE

RPM receives all its instructions from a control file. This must be first be created by a text editor or by a short SuperBASIC program. Examples of control files are given later.

The control file contains a: number of statements. Each statement comprises a 4 character directive followed by 0 or more parameters. The directives define which source files are to be linked into the resident file and how they are to be called up when later used by programs. Comments can be put into the control file by starting a line with a *.

As a convention we suggest appending _rpm to the control file name.

RPM processes the control file in a manner similar to a compiler. The statements are read in turn, checked for errors and all being well, the various source files are copied and linked into the resident file. Each statement is printed on the screen as it is processed. When complete, the size of a ram resident file is displayed. RPM can be terminated by entering a blank control file name. The resident file can then be loaded by an appropriate BOOT program.

For example if RPM has produced a resident file called RESFIL with a size of 3276 bytes then the SuperBASIC line

```
abase=RESPR(3276): LBYTES MDV1_RESFIL,abase: call abase
```

would load and initialise the resident routines.

THE RPM DIRECTIVES

The first statement in the control file defines whether a ROM resident file or a RAM resident file is to be created.

ROMH CREATE A ROM HEADER

Syntax:	ROMH target_file, ROM_size [, ROM_name]
target_file	is the name of the resident file to be produced.
ROM_size	is the size of the target ROM specified in Kb. This figure is used at the end of compilation to check that the ROM size has not been exceeded.
ROM_name	is the message which is printed on the initial screen. It can be of any length but should be limited to 36 characters if the message is to fit on one line. If ROM_name is not specified, no message is printed.

The ROM header which is created has the standard format of a QL ROM driver:

```
00 $4AFB0001    flag to indicate ROM is present
04 $0000        unused
06 pointer to initialisation routine
08 $0000        unused
```

Example: ROMH mdv1_myrom,16,SUPER ROM installed

Create a 16k ROM file with the name mdv1_myrom. On initialisation print “SUPER ROM installed “

RAMH **CREATE A RAM READER**

Syntax: RAMH target_file [, message]

target_file	is the name of the resident file to be produced.
message	is an optional string which is displayed on channel 0 when the RAM file is loaded.

Example: RAMH mdv1_tools_res,Tools version 1.0

EXEC **Link an EXECutable file**

Syntax: EXEC source_file,proc_name, [no_wait [,jmpure]]

source_file	must be an executable file, with a type = 1 in the file header.
proc name	is the procedure name which will invoke this program.
no_wait	is either Y or N. If Yes is selected then when the job is activated by a program, the activating program continues to run. This corresponds to the action of EXEC. If No is selected then the activating job is suspended while the resident job is running, corresponding to EXEC_W. The default if the parameter is not specified is Y.
copy	is either Y or N. Yes specifies that the code of the program is to be copied from the resident file into RAM prior to execution. Copy should only be specified when a program is impure as it is less efficient In memory Usage. The default if the parameter is not specified is N.

Example: EXEC flp1_game_obj,GAME

When resident the program game_obj can be activated by typing GAME. It would multitask alongside the interpreter.

```
EXEC mdv1_demo_task,Demo,n,y
```

The program demo_task can be activated by the procedure Demo. A copy is made in RAM and execution of the interpreter (or other activating job) is suspended until demo_task ends.

ASME **Link a file of assembler extensions.**

Syntax:	ASME source_file
Source_file	is the name of the file containing the extensions This must be a file which itself can be loaded into the RESPR area. it can also be a Q_Liberator object file containing external procedures and functions.

Example: ASME mdv2_extensions

Include all the procedures or functions from Mdv2_extensions in the resident file.

SBAS **Link a SuperBASIC program**

Syntax:	SBAS source_file,driver_name
Source file	is the name of a SuperBASIC source program.
Driver_name	is the name of the new device driver which will be created to access the resident SuperBASIC program. If this is called BOOT then it is treated specially when used in a ROM resident file as explained below.

AUTO STARTING PROGRAMS

When the QL is initialised, after any ROMs have been linked in an attempt is made to load a SuperBASIC program from a device called BOOT. If this fails (as it will if no additional ROMs are fitted) then the file MDV1_BOOT is sought (FLP1_BOOT on floppy based systems). By creating a ROM resident SuperBASIC program with the associated device driver named BOOT, control of the system can be gained by a program as soon as F1 or F2 is pressed. Such a BOOT program could if required startup one or more resident EXECutable programs by using the activating procedure name. Remember however that if a resident program is not free running the BOOT program will be suspended until the resident program ends.

Completely dedicated systems can be created in this way so that naive users can enter an application without ever worrying about BOOTing from the correct disk.

ERRORS REPORTED BY RPM

If RPM finds that a statement contains an error then it stops, prints the offending statement and beneath it an explanatory message from the list below is displayed. No resident file is created.

```
Directive not recognised.
File must start with ROMH or RAMH.
Bad parameter.
Too few parameters.
File greater than ROM size.
```

In addition the familiar QDOS errors such as 'bad name' and 'drive full' can occur during I/O operations.

When RPM has successfully produced a resident file it prints some additional information. For RAM resident files, the size to be used in a RESPR call is displayed.

For ROM files, the total size and the length of the unused part of the ROM is shown, unless the file is greater than the ROM size in which case the size of the overflow is shown.

EXAMPLES

To put Quill in a ROM such that it can be invoked by typing QUILL.

```
ROMH mdv1_quill rom,64,QUILL v 2.3
EXEC mdv1_quill,QUILL
```

To put all the Q_Liberator release 3.2 extensions in one file:

```
RAMH flp1_qlib_sys
ASME flp1_qlib_run
ASME flp1_glib_ovl
ASME flp1_qlib_ext
ASME flp1_qlib_bin
```

To create a ROM containing a toolkit and an auto starting SuperBASIC program. For those without access to a text editor to create the control file, this example shows how it can be done with a simple SuperBASIC program.

```
10 OPEN_NEW #3,"mdv1_tools_rpm"
20 REPeat loop
30   IF EOF THEN EXIT loop
40   READ a$ : PRINT #3,a$
50 END REPeat loop
60 CLOSE #3
70 :
80 DATA "ROMH mdv2_tools,16,Tools installed"
90 DATA "ASME mdv2_tools_bin"
95 DATA "SBAS mdv2_tools_boot,BOOT"
```

PROBLEM PROGRAMS

If you find that a program which normally gives no problems crashes when used in a resident file, it is likely that the code is in some way impure. Use of the copy option in the EXEC directive should solve the problem. SuperBASIC extensions which refuse to work when resident (particularly when in ROM) will require amendment and should be avoided. Liberation Software is unable to undertake changes to 'purify' code.

EPROM PROGRAMMERS

We can unreservedly recommend the QEP III EPROM programmer from QJUMP to those who wish to create their own EPROMS. The file produced by RPM can be directly loaded by the software which accompanies this product.