

1 Scope

This document describes the structure and facilities of the Resource Executive (REX) which runs on Motorola 68k targets which support the Resource. In particular, it describes the interactive facilities of the Supervisor which are present on ^{the target} (the Computrol LP-25 card). The reader should be familiar with the facilities of the 68k XEC [1] and the EI Channels Interface and in particular its the moneic utility [2].

2 Related Documents

- [1] HP024106C308 XEM - 68k Executive
- [2] HP024106C307 EI Channel Interface
- [3] HP024105C??? A Guide to Tuning the Resource
- [4] HP024105C??? A Guide to The Resource Debugger

3 Structure

REX is a set of task handling functions and structures that sits on top of the XEC executive (see [1]).

All references to a word refer to a 16bit (2 byte) word.

3.1 The Background Task

When a target which supports REX is started execution begins in the XEC background task. This task allocates :

- The system memory (§3.4), default 40000 bytes.
- The maximum number of executing tasks, default 10.
- The size of the task graveyard (§3.3), default 10.
- The default task stack size, default 1024 words
- The default task priority, default 50.
- The principal task, default Auto (§3.2).
 - The priority, default 240.
 - The stack size, default 2048 words.

When the background task is started on the Computrol LP-25 it may accept the following switches to modify the above :

- -m <SystemMemorySize>
- -t <MaxNumberOfTasks>
- -g <SizeOfGraveYard>
- -s <DefaultStackSize>
- -p <DefaultPriority>
- -S <PrincipalTaskStackSize>
- -P <PrincipalTaskPriority>

3.2 Principal Task

The principal task is the first task executed (apart from the background task). In principle any task may be used as the principal task but in practice there are only 2 i) Auto (§4) ii) The Supervisor (Computrol LP-25) only(§5).

3.3 Tasks

REX supports a list of pre-defined tasks each of which may be invoked in one of two ways:

run As a sub-procedure of the principal task inheriting its context.

executed As an independent task with its own context.

The context of any task consists of :

Priority The XEC task priority (1-255), default 50.

Stack A stack, default 1024 words. Plus a fixed size system stack of 64 words.

Schedule Interval The interval at which a task (which terminates with a return from main()) is rescheduled for execution, default 0 (do not repeat).

Once a task which has completed execution and terminates with the Exit() or exit() function or a return from main() then the task context (in particular the stack space) is placed in the graveyard. Once a task has an instance of itself placed in the graveyard, then a future instance will use the graveyard context and stack by default (provided a larger stack has not been requested). This helps to conserve the system memory. Tasks that are run are not placed in the graveyard.

In addition tasks may have some of the following attributes :

Unique This indicates that only one instance of the task may be run at any one time.

OnceOnly This indicates that if an instance of the task is found in the graveyard then it may not be re-run.

3.4 System Memory

REX requires an allocation of system memory in which to allocate space for task structures and stack.

4 Auto

The Auto principal task consists of 3 lists of commands to REX :

AutoInitCommands When REX is started for the first time

AutoRestartCommands When REX is re-started

AutoStartCommands Whenever REX is started, after one or other of the above.

Each list is a NULL terminated array of strings. A default set (§7.6) to start a minimal Resource is supplied but a complete set of all 3 may be supplied to the linker to replace these. This allows a programming tool to supply the desired set of auto start commands. This list of commands may consist only of run and execute (§5.2.3 commands).

5 Supervisor

The Supervisor is an interactive principal task which runs on the Computrol LP-25^{and ENM} and accepts task control and Supervisor commands.

The Supervisor communicates interactively using an EIC DMA channel (see [2]). As the Supervisor blocks on this channel, closing the channel effectively suspends the Supervisor until the channel is re-opened. The Supervisor will prompt the user with the prompt `Supervisor>` after the completion of a command and whenever the Supervisor channel is opened. The Supervisor must always be run at priority 240 or lower as otherwise it will lock out the HIN daemon and therefore not be capable of reading or writing to its channel.

The Supervisor accepts the following options on start up :

-a <MaxArguments> The maximum number of arguments to any task or command, default 10.

[-c <DMAChannel> Default 1 (EID1).] *Computrol Only*
 t *eid1*

5.1 Standard IO on Computrol LP-25 →

REX uses 3 EIC DMA or register channels for standard IO, channels `stdin`, `stdout`, `stderr`. By default these[?] are all the same as the Supervisor channel.

Tasks will use these standard IO channels for user interaction and printing to the unix host. Only one task should actually use these channels at a time. The Supervisor expects these channels to remain open and if not will attempt to reopen them. Terminals to the standard IO channels (or indeed any channel) may be set up in one of 2 ways i) By using moneic terminals (see [2]), ii) By using the supplied `etalk` task.

`etalk` accepts the following options

<Channel> EID or EIR channel, default `eid1`.

-b Use blocking mode, default non-blocking.

-f <Seconds> Delay to flush output before terminating, default 20 seconds.

-x Exit if the channel is closed.

It is recommended that `etalk` is used to talk to the Supervisor, but otherwise either should be adequate.

5.2 Commands

The following Supervisor commands may be issued either interactively or in an Auto start up command list (§4).

5.2.1 EIC Channels on Computer LP-25

The following commands map directly onto the EIC DMA or register channels.

A channel number may be referred to as one of

- EID<n>, where n = 1 .. max
- EIR<n>, where n = 1 .. max
- stdin
- stdout
- stderr

open

open <Channel> Open the channel. The options -block, -nonblock may be supplied. The default is non-blocking.

close

close <Channel> Close the channel.

ioctl

ioctl <Channel> <Command> ioctl on the channel

Where the command is one of

QNDELAY Query if blocking or non-blocking

SNDELAY Set non-blocking

SDELAY Set blocking

QCONNECT Query if connected

QNR Query number of unread bytes

QNW Query number of unwritten bytes

stdio

stdio [stdin|stdout|stderr|stdio|stdall] <Device>

This command may be used to list out the current allocation of the standard IO channels and to change this allocation.

stdio List out the allocations

stdio <StandardChannel> <EICChannel> Change standard channel

Example

Supervisor>stdio
 stdin = eid1
 stdout = eid1
 stderr = eid1
 Supervisor>open eid2
 Supervisor>iocctl eid2 QCONNECT
 1
 Supervisor>stdio stdout eid2
 Supervisor>stdio
 stdin = eid1
 stdout = eid2
 stderr = eid1
 Supervisor>

5.2.2 XEC

The following commands provide a limited interface to XEC.

✓ **priority**

priority Report the priority of the Supervisor

priority <Priority> Change the priority of the Supervisor.

✓ **gettime**

Reports the time since the Supervisor was started. The value is converted to XEC ticks if the **-ticks** option is supplied.

Example

Supervisor>gettime
 2 Days 02:32:55.336 → Elapsed Time Since StartUp = 0 days, 00 hrs, 01 min, 59 sec
 Supervisor>gettime -ticks
 90990664 → System Time: 15-Feb-2007 16:46:21 GMT

setevent

Number of Ticks Since StartUp = 134209
 System Time: 15-Feb-2007 16:46:37 GMT

setevent <TaskName> <Event> Set an XEC event for a task.

setevent <Event> [-all-instances] [-quiet] <TaskName>

wakeup

wakeup <TaskName> Issue an XEC wakeup to a suspended task.

wakeup [-all-instances] [-quiet] <TaskName>

5.2.3 Tasks

Some of the following commands use a task name specification. This specification matches all tasks whose names match the characters of the specification up the first occurrence of *. eg * matches all tasks, r* matches all tasks whose names begin with r.

✓ **directory**

directory List all tasks

directory <TaskNameSpec> List tasks whose name match the task name spec

directory [Disk:] [<FilePattern>] [-c] [-l] [-s]
 If the -l option is supplied then the task attributes (§3.3) are listed. ?

(no attributes are listed with -l option ?)

U Unique

O Once Only

Example

Supervisor>dir -l l*
 listRes --
 loader UD
 Supervisor>

execute

*dir -l **

RESOURCE.RLW	63	2007:02:15:16:44:30	rw
RESOURCE.VRF	1682	2007:02:15:16:44:32	rw
RESOURCE.AIF	29798	2007:02:15:16:44:35	rw
RESOURCE.RIF	1109	2007:02:15:16:44:36	rw
FSIR-DIR.DAT	75	2007:02:15:16:44:40	rw

execute <ExecuteOptions> <TaskName> <TaskOptions> Execute a task.

execute [-free] [-i<Interval>] [-p<Priority>] [-s<StackSize>] <TaskName>
 Where the execute defaults (§3.3) may be overridden with the options

<CommandLine>

-p <Priority> XEC priority to run at.

-s <StackSize> Stack size in words.

-i <ScheduleInterval> Task repeat interval in seconds.

When a schedule interval is supplied the task begin execution again at main(). All the originally supplied arguments (argcand argv) are still present and may be read again.

✗ **run**

<TaskName> <TaskOptions> Run a task.

This is not a command as such, but runs the task as a procedure of the Supervisor. Such a task must not call Exit() or exit() to terminate otherwise the Supervisor will itself terminate.

✓ **graveyard**

List out the contents of the graveyard.

✗ **status**

status <TaskNameSpec> List tasks whose name match the task name spec.

Example

```

Supervisor>status *
Supervisor      2   766/2048   46/64
rouRecover      4   214/1024   42/64   00:00:10.000   0
router          3   222/1024   46/64
task            8   229/4096   46/64
task            7   229/4096   46/64
task            6   229/4096   46/64
task            5   229/4096   46/64
Supervisor>

```

The listing contains in order :

- The task name
- The XEC task id
- The user stack. The number of bytes "used" / total allocated
- The system stack. The number of bytes "used" / total allocated
- The schedule interval.
- The schedule overruns.

The number of bytes "used" in a stack is determined by walking back from the end of the stack and detecting the first non-zero byte encountered. The stack is cleared on allocation and so this method will result in a figure which may be slightly less than the number of bytes actually used by a task.

unschedule

unschedule <TaskName> Unschedule the task, do not repeat at end of current interval.

unschedule [-all_instances] [-quiet] <TaskName>

5.2.4 Help

help

help Give a list of help subjects.

help <Subject> List help on the subject

help [<Command>]

5.2.5 Exit

exit

→ should be #exit ?

Will cause the Supervisor to terminate. All other tasks will continue to execute. The Supervisor channel will be closed. The stdio channels will only be closed if they are the Supervisor channel.

6 Built-in Tasks

X 6.1 eilin

The EILIN task (eilin) may be used to execute the separate parts of the EILIN. The task may be used either to set up EILIN with the following options :

- node <NodeNumber> Initialise EILIN with this node number.
- aSize <Size> Sizeof buffers
- arxBD <Number> Allocation of receive buffer descriptors
- atxBD <Number> Allocation of transmit buffer descriptors
- arxFD <Number> Allocation of receive frame descriptors
- atxFD <Number> Allocation of transmit frame descriptors
- rRWNR <Retries> Set the number of retries for RWNR requests.
- rRWR <Retries> Set the number of retries for RWR request.
- t<Type> <Timeout> Set the timeout in SSM ticks of <Type>

Request

Confirm

Abort

Indication

Response

Connection

Base Connection timeout base count.

wks Implement the Well Known SAP task.

and/or to periodically run the individual EILIN sub-tasks on a fixed interval with the following options :

- cycle <XEC Ticks> Set the tick interval.
- ssm Run the SSM task every tick interval.
- tick Run the SSM tick every tick interval.

X 6.2 timex

The timex task provides a clock which is updated on every timex cycle. The task is provided as XEC only supplies a 32 bit tick counter which is insufficient to hold a full IEC/Unix time and date. The task should first be used to set the time and date and then scheduled (say, once an hour) to check against any overflows. Timex also provides the unix compatible time() entry point.

-quiet Do not report the time and date, default is to report. This should be supplied when scheduled.

Year:Month:Day:Hour:Minute:Seconds The current time and date. All fields are integers.

6.3 rallTasks

This task is used to load all Resource TASKs with the same set of options. It is principally for use as a default way of starting a Resource.

`rallTasks <ExecuteOptions> * <TaskOptions>` This executes each TASK as "exe <ExecuteOptions> * <TaskOptions>" where * is the TASK name.

7 The Resource

Running the Resource under REX has certain differences from Unix. In general the `-q` option must be supplied if Auto is being used. The `-y <Key>` is accepted but is ignored.

7.1 Shared Memory

The amount of shared memory available is the total spare RAM - the system memory. Therefore tuning the system memory can increase available shared memory.

The `md:tools` accepts two additional options (`-p` for physical memory and `-s` for system memory).

7.2 The Loader

The loader will not fail if all the shared memory requested is not available, it will just limit itself at the maximum available.

The loader task has the `OnceOnly` attribute but is normally run and not executed and therefore it will not be possible to detect if it has already been run. The loader should only appear in the `AutoInitCommands` list of Auto (§4) If the loader is run again then the system will crash. This is because the loader alters the original C Resource structure. If the loader is executed then this problem will not occur but the stack will not be recoverable.

7.3 Tasks

All Resource TASKs are unloaded by setting event 1 which is read at the start of each task cycle.

7.4 The Router

The router is run as 2 separate tasks under REX. The media recovery part of the router is executed as a separate task (`rouRecover`) which must be scheduled with a suitable repeat interval and at a higher priority than the router.

If the router is to support EILIN then it must be at a lower priority than the EILIN task (§6.1).

7.5 The Unloader

The unloader may be used to unload TASKs but the -l option must be used if that TASK is no longer executing as otherwise another task which is using the same task id as the RESOURCE task may be unloaded. Also if no task is running with that task id then the system will crash as XEC does not check task ids. In particular when the unloader is used in AutoRestartCommands (§4) the -l option must be supplied.

7.6 Example Resource Auto Start

This very simple example of the 3 lists of Auto commands (§4) runs up all tasks of a Resource with a single medium HIN to communicate with the unix half of the same Resource.

```
char* AutoInitCommands [] = { "loader -q -s 500000",
                               NULL } ;

char* AutoStartCommands [] = {
    "execute -p 50 router -m HIN",
    "execute -p 60 -i 10 rouRecover",
    "rallTasks -s 4096 task -q *",
    NULL } ;

char* AutoRestartCommands [] = {
    "unload -q -l resource",
    NULL } ;
```

7.7 The Tools

The tool set (see [3]) is available under the Supervisor as behaves almost identically to unix. Certain precautions must be observed however.

- No 2 tasks are permitted to write or read simultaneously from the standard IO channels. In practice this means only one tool may be active at any time.
- Certain tools call Exit() or exit() to terminate on errors, this will cause the Supervisor to terminate (§5).
- CmsEars uses the closure of stdout to terminate as no Control-C equivalent is available.
- CmsSpy quit command does not work.
- The debugger is not available as unix debugger can be used (see [4]).

8 Computrol LP-25

This section briefly describes how the Computrol LP-25 may be loaded with a Resource under REX and how the Auto and Supervisor principal tasks may be used. This is not the only way of doing this.

Assume that the Resource executable is in a file called `resource.run` in the current directory and it had been built with a map file called `resource.map`.

The moneic utility (see [2]) is used for loading and starting execution.

The Resource may be loaded as

```
moneic -k -M -Lresource.run -f > /dev/null
res_RUN='grep START resource.map | sed -e "s/.*=//1"'
export res_RUN
```

The entry point START is conserved so that the REX may be restarted without reloading.

The Resource may then be auto started

```
moneic -k -R$res_RUN -C"$* Auto" > /dev/null
```

or the Resource may be started with the Supervisor

```
moneic -k -R$res_RUN -C"$* Supervisor" > /dev/null
```

and a terminal opened to the Supervisor

```
etalk -x eid1
```

The Resource may then be restarted with either of the above. In general a Resource would be restarted with the sequence that started it, although a Resource started with Auto may be restarted with the Supervisor for post-mortem analysis.

Once the Supervisor has been started it may be left running by Control-C from the etalk terminal. Interaction may then be resumed with

```
etalk -x eid1
```

This method is particularly useful as the Supervisor is simply left dormant (in a blocking read) and a restart is not required simply to examine what is happening on the card. In fact a single command could be issued to the board

```
echo "$*" | etalk -x eid1 2> /dev/null | sed -e "s/Supervisor>//1"
```

These commands are encapsulated in the supplied shell functions :

cload [**<ResourceName>**] Load the Resource

auto [**<AutoOptions>**] Auto (re)start.

loadSuper [**<SuperOptions>**] Load and (re)start the Supervisor.

super (Re)Open a terminal to the Supervisor.

lp25 **<Command>** Issue a command to the Supervisor.

