How RTEMS startup handles static C++ constructors

The C User's Manual has a nice picture here of what happens when RTEMS is just getting started.

One thing it doesn't say is just how static constructors get called. This RTEMS wiki page fills things in nicely though it's a bit hard to understand, in part because the author seems to have had some trouble with wiki markup involving underscores.

It turns out that when multitasking is turned on and the first task is created and started, control passes from rtems_initialize_start_multitasking(), to _Thread_Start_multitasking(), to _CPU_Context_switch() and then to a function named _Thread_Handler(). This function is the true starting point for all tasks; it performs some housekeeping and then calls the function named as the task entry point. In our case the task entry point for the very first task is named Init(), which is defined in release/rce/init/src/Init.cc.

_Thread_Handler() checks to see whether it's being executed for the first time and if so it calls _init() which eventually calls __do_global_ctors_aux(). The _init() function resides in a code section named .init which in turn is the concatenation of input sections whose names match the pattern ".init*". That in itself is nothing special; when linking an executable it's typical for an output section to be made from a concatenation of input sections with similar names. What makes .init special is the way it's used by start-up code: .init is expected to contain one big function to be executed at start-up.

To make the contents of .init into a single function we have to make sure that the first object code file containing an ".init*" section starts with function entrance code and that the last-linked ".init*" section ends the call to __do_global_ctors_aux() followed by function exit code. This is why the start-up file ecrti.o is linked first; it contributes the function entrance code. crtend.o is linked next to last; it contributes the call to __do_global_ctors_aux(). ecrtn.o is linked last; it contributes the function exit code.

Contents of .init (non-PowerPC):

- Function entry (ecrti.o) (label __init)
- Other code bits.
- Call to __do_global_ctors_aux() (crtend.o)
- Function exit (ecrtn.o)

For PowerPC there's a fly in the ointment: early in the RTEMS start-up the function __eabi() is called to set up the PowerPC EABI environment. __eabi() in turn calls the .init function, which it assumes has been labeled __init. The trouble is that when __eabi() is running it's too early to run static constructors since RTEMS initialization has barely begun. The solution adopted for RTEMS, rather than modifying the start-up files, is inserting a file called rtems_crti.o in between ecrti.o and crtbegin.o; it contributes function exit code followed by function entrance code:

Contents of .init (PowerPC):

- Function entry (ecrti.o) (label __init)
- Function exit (rtems_crti.o)
- Function entry (rtems_crti.o) (label _init)
- Other code bits.
- Call to __do_global_ctors_aux() (crtend.o)
- Function exit (ecrtn.o)

Now __init() is a do-nothing function followed by another function that does the real work; this second function, named _init(), is called by _Thread_Handler().