Writing Ada code for M2OS

There are two ways of writing multitasking Ada applications:

- Using the M2OS API to create one-shot tasks.
- Writing standard Ada tasks (with some restrictions to behave as one-shot tasks) and processing the code with the "code transformation tool".

Creating Ada tasks using the M2OS API

If you chose to use the M2OS API (not use the "code transformation tool") the main procedure acts as the initialization code of your application and must fulfil the following structure:

```
procedure Main is
begin
   ... application initialization code (without any blocking operation)
end Main;
```

Tasks must be created using `package AdaX_Dispatching_Stack_Sharing`. For example, in order to create a periodic task in the package My_Package:

```
with Ada.Real_Time;
with AdaX_Dispatching_Stack_Sharing;

package body My_Package is
   package RT renames Ada.Real_Time;

   use type RT.Time_Span;

   Period : constant RT.Time_Span := RT.Milliseconds (1_000);
   Next_Time : RT.Time;

   procedure Task_Init is
   begin
      ... Initialization code

      Next_Time := Ada.Real_Time.Clock;
   end Task_Init;

   procedure Task_Body is
   begin
      ...  Task's Job

      Next_Time := Next_Time + Period;
      delay until Next_Time;
   end Task_Body;

   Periodic_Task : AdaX_Dispatching_Stack_Sharing.One_Shot_Task
     (Init_Ac  => Task_Init'Access,
      Body_Ac  => Task_Body'Access,
      Priority => 4);

end My_Package;
```

The first time the task is dispatched it executes the initialization procedure (called `Task_Init` in the example). Subsequent releases of the task will execute the body procedure (called `Task_Body` in the example). The task body must end in a potentially blocking operation (`delay until`, protected object `entry` or suspension object `Suspend_Until_True` procedure) to wait for the next release event.

Code in this example is equivalent to the one shown in the next section.

Examples of applications using the M2OS API can be found in `M2OS/examples/api_m2os/` and `M2OS/tests/api_m2os/`. To study and run them use:

```
M2OS$ gps -P examples/api_m2os/examples_api_m2os_arduino_uno.gpr &
  or
```

```
M2OS$ gps -P tests/api_m2os/tests_api_m2os_arduino_uno.gpr &
```

Using standard Ada tasks

If you prefer to write code with standard Ada tasks you can do it, but tasks must fulfil a pattern:

- Task must have an infinite loop.
- The loop must end in a potentially blocking operation.

```
with Ada.Real_Time;
with AdaX_Dispatching_Stack_Sharing;

package body My_Package is
   package RT renames Ada.Real_Time;

   use type RT.Time_Span;

   Period : constant RT.Time_Span := RT.Milliseconds (1_000);

   task Main_Task with Priority => 4;

   task body Periodic_Task is
      Next_Time : RT.Time;
   begin
      ... Initialization code

      Next_Time := Ada.Real_Time.Clock;
      loop
         ...  Task's Job

         Next_Time := Next_Time + Period;
         delay until Next_Time;
      end loop;
   end Periodic_Task;
end My_Package;
```

Code in this example is equivalent to the one shown in the previous section. The "code transformation tool" is used to perform the automatic transformation.

Examples of applications using Ada tasks can be found in `M2OS/examples/ada_tasks/` and `M2OS/tests/ada_tasks/`. To study and run them use:

```
M2OS$ gps -P examples/ada_tasks/examples_arduino_uno.gpr &
  or
M2OS$ gps -P tests/ada_tasks/tests_arduino_uno.gpr &
```