Gem #112: Среда языка Ada для работы с Lego Mindstorms – Часть 1

Автор: Pat Rogers, AdaCore

Краткое содержание: В данной серии публикаций будет описана среда программирования языка Ada GNAT в контексте работы с набором робототехнических инструментов Lego Mindstorms. В серии будут исследованы высоко- и низкоуровневые интерфейсы для работы с аппаратным обеспечением, подмножество языка, поддерживаемое библиотекой программ этапа выполнения, и наиболее эффективные способы работы со средой. Также будут рассматриваться другие вопросы. В первом Gem серии будут представлены аспекты доступного разработчикам языкового подмножества Ada общего характера и показано, как интерфейсы Ada взаимодействую с одним таким аспектом.

Давайте начнём…

Процессор Lego Mindstorms имеет ограниченное по размеру хранилище для кода, которого, безусловно, недостаточно для нетривиального приложения и полной библиотеки времени выполнения. Поэтому цепочка инструментов GNAT не реализует полный язык Ada для этой платформы. Значительное количество языковых возможностей все ещё включено, особенно профиль задач Ravenscar, но весь язык недоступен.

Например, семантика полного исключения не реализована. Программисты могут создавать пользовательские и языковые исключения и могут обрабатывать их локально, но необработанные (и перезаписанные) исключения не распространяются динамически по цепочке вызовов, как это происходит семантически полной Ada. Вместо этого необработанное исключение приводит к вызову одной глобальной подпрограммы, называемой "обработчиком последнего шанса", так называемой, поскольку она не возвращается в приложение. Разработчики могут определить зависящие от приложения замены обработчика последнего шанса для реализации по умолчанию.

Реализация по умолчанию, поставляемая с этим компилятором, просто завершает работу процессора Mindstorms. Однако интерфейсы Ada определяют альтернативу, которая отображает Имя файла и номер строки, соответствующее местоположению исключения, используя ЖК-дисплей на аппаратной платформе Mindstorms. Процедура также включает на небольшое время гудок в динамик, чтобы привлечь внимание пользователя. Затем он переходит в бесконечный цикл, чтобы пользователь мог отметить местоположение исключения. Эта реализация объявлена в NXT.Last_Chance пакет, показанный ниже.

```
with System;
package NXT.Last_Chance is
   procedure Last_Chance_Handler
     (Source_Location : System.Address;
      Line            : Integer);
   pragma Export (C, Last_Chance_Handler, "__gnat_last_chance_handler");
end NXT.Last_Chance;
```

Обратите внимание на директиву `pragma`, экспортирующую имя как "`__gnat_Last_chance_handler`", как стандартное имя подпрограммы, вызываемой библиотекой среды выполнения при возникновении необработанного исключения. Приложения не вызывают эту процедуру сами, так как она не возвращает управление вызывающему приложению и завершает работу системы. Вместо этого они переопределяют символ, а компоновщик включает свою версию подпрограммы в исполняемый файл, чтобы библиотека времени выполнения могла вызвать её при необходимости. В случае интерфейсов Mindstorms пользователь указывает пакет `NXT.Last_Chance` вместе с директивой, как правило, в основной программе.

```
with NXT.Last_Chance;
...
procedure ...
```

Вы можете увидеть исходный текст кода для тела процедуры, действующие коды для всех интерфейсов, в каталоге "drivers" в корневом каталоге установки. По умолчанию это будет:

```
C:\GNAT\2011\lib\mindstorms-nxt\drivers
```

Несмотря на то, что эта процедура не очень большая или сложная, в такой ограниченной памятью среде каждый байт может быть драгоценным, таким образом, Вы должны принять во внимание размер кода. Например, процедура использует аудиоинтерфейс, чтобы динамик подавал сигнал, и пакет жидкокристаллического дисплея также импортируется. Если бы Вы решили, не использовали эти пакеты (и их потомков), это могло иметь значение на размер кода. Однако нет никакого требования для Вас, чтобы использовать, именно эту версию обработчика, Вы можете просто закомментировать или удалить `with` с именем пакета обработчика fancier и код которого Вы не хотите более включать.

Связанный со статьёй текст программы

Attached Files отсутствуют

Файлы примеров Ada Gems распространяются AdaCore и могут быть использованы или изменены для любых целей без ограничений.

Об авторе

Пэт Роджерс (Pat Rogers) был профессионалом в области вычислительной техники с 1975 года, в основном работая над приложениями на основе микропроцессора в режиме реального времени на языках Ada, C, C ++ и других языках, включая высокопроизводительные имитаторы полета и системы контроля и сбора данных (SCADA), контролирующие опасные материалы. Впервые узнав язык программирования Ada в 1980 году, он был директором лаборатории Ada9X для совместной программы Advanced Strike Technology для ВВС США, исследователя принципов в распределённых системах и исследовательских проектов отказоустойчивости с использованием Ada для ВВС США и армии, а также помощника директора по исследованиям в NASA Software Engineering Research Center. У него есть B.S. и M.S. степени в области проектирования компьютерных систем и компьютерных наук Университета Хьюстона и доктора философии. в информатике из Университета Йорка, Англия. Являясь членом старшего технического персонала AdaCore, он специализируется на поддержке разработчиков в режиме реального времени / встроенных систем, создает и предоставляет учебные курсы, а также является руководителем проекта и разработчиком плагина GNATbench Eclipse для Ada. Он также имеет чёрный пояс 3-го Дан в Тэ Квон До и является основателем клуба AdaCore «The Wicked Uncles».

Last Updated: 11/24/2017
Posted on: 10/17/2011

Обсуждение…