Gem #115: Среда языка Ada для работы с Lego Mindstorms – Часть 2

Автор: Pat Rogers, AdaCore

Краткое содержание: В данной серии публикаций будет описана среда программирования языка Ада 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 данном Gem представляются базовые шаги инициализации и отключения аппаратного обеспечения Mindstorms.

Давайте начнём…

Mindstorms NXT brick – аппаратный модуль – содержит как 32-битный ARM-процессор, так и 8-битный AVR-процессор. ARM выполняет код приложения, в то время как AVR отвечает за функционирование устройства более низкого уровня, включая управление выходами, сбор входных сигналов датчиков и даже выключение самого аппаратного модуля.

Оба процессора координируют свои действия, посылая нескончаемый поток сообщений друг другу. При инициализации AVR начинает отправлять сообщения, содержащие информацию, среди других данных, такую как нажатые текущие кнопки, если таковые имеются, и состояние батареи. ARM отправляет сообщения обратно в AVR, например, для установки уровней мощности на выходах для двигателей или для выполнения аналого-цифрового преобразования. Сообщения постоянно обновляются с фиксированной скоростью. Другими словами, они не управляются событиями, хотя их контент, безусловно, отражает внешние события и внутренние команды. Посмотрите в каталоге «drivers» для тела пакета `NXT.AVR`, если вы хотите посмотреть, как эти сообщения отправляются и принимаются. Одна задача, объявленная во всем API, находится в этом теле пакета для этой цели.

Временные ограничения для обработки сообщений фиксируются аппаратными средствами и очень строги. Следовательно, возможно, что сообщения могут быть потеряны или искажены изредка. Контрольная сумма рассчитывается для обнаружения этих ошибок, что приводит к отбрасыванию проблемного сообщения.

API Ada скрывает все эти детали за абстрактными типами данных высокого уровня для датчиков и двигателей и процедурные интерфейсы для других устройств нижнего уровня, таких как аналого-цифровой преобразователь и возможности дискретного ввода-вывода портов. Тем не менее, код приложения должен быть написан с учётом упомянутой выше архитектуры. В частности, входящие данные недоступны до тех пор, пока AVR не будет инициализирован и отправил хотя бы одно сообщение в ARM. Хотя AVR автоматически инициализируется драйверами Ada, программист приложения все равно должен ждать получения этого сообщения. Например, текущее напряжение батареи сохраняется в переменной, указанной в пакете `NXT.AVR`. Значение этой переменной зависит от поступления сообщения из AVR и не определено заранее. Функции доступа, которые декодируют представление напряжения, просто обрабатывают переменную, как будто значение уже определено. (Конечно, есть способы смягчить использование неопределённого значения, но ни один из них не идеален.) Другие значения, такие как необработанные показания кнопок, также сохраняются как переменные, которые задаются декодированными сообщениями AVR.

Таким образом, пакет `NXT.AVR` предоставляет процедуру, которая ожидает инициализацию AVR и получение первоначального сообщения. Эта процедура называется `Await_Data_Available`. Вызов этой процедуры обычно должен быть первым, что было сделано приложением.

Другая процедура предоставляется для отключения аппаратного блока для тех приложений, которые предназначены для завершения (в отличие, скажем, от встроенных систем управления, которые останавливаются только при отключении внешнего питания). Это процедура `Power_Down`,

также найденная в объявлении пакета `NXT.AVR`. AVR отвечает за фактическое выключение питания, поэтому процедура вводит соответствующее сообщение в AVR в поток сообщений. Однако, как уже упоминалось, сообщения могут быть потеряны или искажены, поэтому запрос на подачу питания может быть потерян. В результате лучше использовать бесконечный цикл, который вызывает `Power_Down` несколько раз. Фактически, цикл «выходит», когда AVR отключает питание от аппаратного блока. Например:

```
loop
   NXT.AVR.Power_Down;
   delay until Clock + Seconds (1);
end loop;
```

Оператор абсолютной задержки эмулирует относительную задержку (поскольку подмножество Ravenscar не включает относительные задержки), вызывая функцию `Ada.Real_Time.Clock` и добавляя к ней интервал. Продолжительность интервала произвольная и не должна быть только на одну секунду.

Обратите внимание, что пакет `NXT.AVR` также обнаруживает ситуацию, когда кнопка Power на аппаратном модуле удерживается на некоторый промежуток времени и автоматически отключает аппаратный модуль в ответ.

Таким образом, хотя API разделён на несколько уровней, чтобы скрыть подробности о взаимодействиях ARM/AVR, последствия архитектуры сообщений остаются видимыми.

В следующем Gem в этой серии мы рассмотрим абстрактные типы данных высокого уровня, представляющие датчики и двигатели.

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: 10/13/2017
Posted on: 2/1/2012

Обсуждение…