ID 021L: Model Based Control Design and Auto-Code Generation using the R8C

Christopher Myers
Director of Software Development
12 October 2010
Version: 1.2
Christopher Myers  
cmyers@simuquest.com

- Director of Software Development
  - Responsible for designing and managing development of SimuQuest product lines
  - Lead Developer of QuantiPhi, a line of code generation tools that bring a hardware interface into the Simulink modeling environment.

PREVIOUS EXPERIENCE:
- Working in the area of model-based development using MATLAB / Simulink since 2003.
- Production automotive and consumer electronics experience at Motorola since 2000
- Designed high-performance network appliances designed to provide unparalleled visibility into network usage at Arbor Networks, Inc.
- MS, BS from the University of Michigan
Renesas Technology and Solution Portfolio

Microcontrollers & Microprocessors
#1 Market share worldwide *

ASIC, ASSP & Memory
Advanced and proven technologies

Solutions for Innovation

Analog and Power Devices
#1 Market share in low-voltage MOSFET**

* MCU: 31% revenue basis from Gartner

** Power MOSFET: 17.1% on unit basis from Marketing Eye 2009 (17.1% on unit basis).
Renesas Technology and Solution Portfolio

Microcontrollers & Microprocessors
#1 Market share worldwide *

Solutions for Innovation

ASIC, ASSP & Memory
Advanced and proven technologies

Analog and Power Devices
#1 Market share in low-voltage MOSFET**

* MCU: 31% revenue basis from Gartner
** Power MOSFET: 17.1% on unit basis from Marketing Eye 2009 (17.1% on unit basis).
Microcontroller and Microprocessor Line-up

Superscalar, MMU, Multimedia
- Up to 1200 DMIPS, 45, 65 & 90nm process
- Video and audio processing on Linux
- Server, Industrial & Automotive

High Performance CPU, Low Power
- Up to 500 DMIPS, 150 & 90nm process
- 600uA/MHz, 1.5 uA standby
- Medical, Automotive & Industrial

High Performance CPU, FPU, DSC
- Up to 165 DMIPS, 90nm process
- 500uA/MHz, 2.5 uA standby
- Ethernet, CAN, USB, Motor Control, TFT Display
- Legacy Cores
- Next-generation migration to RX

8/16 bit
- General Purpose
  - Up to 10 DMIPS, 130nm process
  - 350 uA/MHz, 1uA standby
  - Capacitive touch

- Ultra Low Power
  - Up to 25 DMIPS, 150nm process
  - 190 uA/MHz, 0.3uA standby
  - Application-specific integration

- Embedded Security
  - Up to 25 DMIPS, 180, 90nm process
  - 1mA/MHz, 100uA standby
  - Crypto engine, Hardware security
Microcontroller and Microprocessor Line-up

R8C

- 16 Bit CISC
- Superb Noise Performance
- Low Power Consumption
- Higher Functionality
- ASSP Lineup
- Low Pin Count Lineup

General Purpose

- Up to 10 DMIPS, 130nm process
- 350 uA/MHz, 1uA standby
- Capacitive touch

Ultra Low Power

- Up to 25 DMIPS, 150nm process
- 190 uA/MHz, 0.3uA standby
- Application-specific integration

Embedded Security

- Up to 25 DMIPS, 180, 90nm process
- 1mA/MHz, 100uA standby
- Crypto engine, Hardware security
Innovation: Innovate Faster

Innovate Faster!
Model-Based Design

Model-Based Design and Automatic Code Generation is the future of embedded controls. SimuQuest believes that as companies across all industries realize the innovation advantage that it provides, it will play a part in the design of nearly all embedded systems worldwide.
Agenda

- Introduction to modeling and auto-code generation
- Introduction to MATLAB / Simulink / Stateflow
- Lab: Modeling controls in Simulink / Stateflow
- Discussion of auto-code generation for a target
- Lab: Generating code for an R8C controller
- Q&A
Key Takeaways

- Models helps define requirements
- Models can be simulated
- Models can be auto-coded
- Auto-coding saves you time
- Auto-coding is powerful!
Idea #1
Idea #2

Model

C Code

Target

QuantiPhi RE for R8C/3X
What is a Model?

- Model = Abstraction

Real World

Outside

↑

Inside

Software

Plant

Controller
Example: Engine Controller
Our Ambient Lighting System

Controller

E8a

Plant

TRI-COLOR LEDS

Tri-Color LEDS
control input
input switch state

Control Model

control output
PWMs x6

Plant Model

plant output
• momentary switch + noise
• LED colors

• Process input switches
• LED Color? Intensity?
• What frequency? Duty cycle?

• Model the input switches
• Model PWM to RGB
Lab Procedure - Phase 1

- Input Switches
- Noise
- Debounce
- Logic
- LEDs
Lab Procedure - Phase 2

- QuantiPhi Digital Inputs
- Debounce
- Logic
- QuantiPhi PWM Output

QuantiPhi Scheduling, Clock Configuration + Build
Our Auto-Coding Toolbox

- **MATLAB**
  Base computing platform

- **Simulink & Stateflow**
  Modeling platform

- **RTW Embedded Coder**
  Auto-code generation

- **QuantiPhi**
  Provides MCU configuration and device drivers.
Questions?
Start the Lab

- Keep your dice turned to the section of the lab you are on. (Instructions are provided in the lab handout)

- Please refer to the Lab Handout and let’s get started!

---

**Low Power Lab**

**MCU Family & product technology**

**Description:** The purpose of this lab is to show some of the low power operating modes of the M16C Tiny Family. This lab will demonstrate placing the device into a low power operating mode where the MCU is in Wait mode and wakes up once every second to update a real-time clock value then goes back to a low power state. The lab will also look at some common problems when entering and exiting low power mode. Finally, the lab exercise will conclude with an examination of the flexibility of the clock configuration of the processor when using low power modes.

**Lab Objectives**

1. Show the low power modes of the Renesas M16C Tiny MCUs.
2. Perform actual measurements to validate hardware manual data.
3. Show some common problems when entering and exiting a low power mode.
4. Examine potential power saving settings in a typical design.

**Lab Materials**

Please verify you have the following materials at your lab station:

- Laptop PC with Renesas tools installed
- RSK/29 Demo Platform
- Multimeter and probes
- This lab sheet
Checking Progress

- We are using the die to keep track of where everyone is in the lab. Make sure to update it as you change sections.

- When done with the lab, your die will have the 6 pointing up as shown here.
Lab Phase 2
Typical Auto-Coding Methodologies

Feature

Effort =

.C + .C

→
Typical Auto-Coding Methodologies

"Application Layer"

Feature 1

Feature n

Effort =
Typical Auto-Coding Methodologies
Interfacing With Hardware

Input Switches  Noise  Debounce  Logic  LEDs
Interfacing With Hardware

SW4 → \text{Debounce} → \text{Logic} → \text{PWM Output} → \text{PWM}

\text{QuantiPhi Digital Inputs} → \text{Debounce} → \text{Logic} → \text{PWM Output}

\text{QuantiPhi Scheduling, Clock Configuration + Build}
Another Hardware Abstraction Layer Example

- Example of analog counts to engineering units:
R8C/3X Code-Gen Considerations

- Don’t use floating point! (double, single)
- Natural “int” size is 16 bits
- Don’t run tasks too quickly (< 1ms for example)
Start the Lab – Phase 2

- Keep your dice turned to the section of the lab you are on. (Instructions are provided in the lab handout)

- Please refer to the Lab Handout and let’s get started!
Checking Progress

- We are using the die to keep track of where everyone is in the lab. Make sure to update it as you change sections.

- When done with the lab, your die will have the 6 pointing up as shown here.
Key Takeaways (Revisited)

- Models helps define requirements
- Models can be simulated
- Models can be auto-coded
- Auto-coding saves you time
- Auto-coding is powerful!
Innovation: Innovate Faster

Innovate Faster!
Questions?
Thank You!