Using a Renesas Code Generation Tool for RL78 Devices
Renesas Technology & Solution Portfolio
## Microcontroller and Microprocessor Line-up

<table>
<thead>
<tr>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1200 DMIPS, Superscalar</strong></td>
</tr>
<tr>
<td>Industrial &amp; Automotive, 65nm</td>
</tr>
<tr>
<td>600µA/MHz, 1.5µA standby</td>
</tr>
<tr>
<td><strong>500 DMIPS, Low Power</strong></td>
</tr>
<tr>
<td>Industrial &amp; Automotive, 90nm</td>
</tr>
<tr>
<td>600µA/MHz, 1.5µA standby</td>
</tr>
<tr>
<td><strong>165 DMIPS, FPU, DSC</strong></td>
</tr>
<tr>
<td>Industrial, 90nm</td>
</tr>
<tr>
<td>242µA/MHz, 0.2µA standby</td>
</tr>
<tr>
<td><strong>25 DMIPS, Low Power</strong></td>
</tr>
<tr>
<td>Industrial &amp; Automotive, 150nm</td>
</tr>
<tr>
<td>190µA/MHz, 0.3µA standby</td>
</tr>
<tr>
<td><strong>10 DMIPS, Capacitive Touch</strong></td>
</tr>
<tr>
<td>Industrial &amp; Automotive, 130nm</td>
</tr>
<tr>
<td>350µA/MHz, 1µA standby</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1200 DMIPS, Performance</strong></td>
</tr>
<tr>
<td>Automotive, 40nm</td>
</tr>
<tr>
<td>500µA/MHz, 35µA deep standby</td>
</tr>
<tr>
<td><strong>165 DMIPS, FPU, DSC</strong></td>
</tr>
<tr>
<td>Industrial, 40nm</td>
</tr>
<tr>
<td>242µA/MHz, 0.2µA standby</td>
</tr>
<tr>
<td><strong>Embedded Security, ASSP</strong></td>
</tr>
<tr>
<td>Industrial, 90nm</td>
</tr>
<tr>
<td>1mA/MHz, 100µA standby</td>
</tr>
<tr>
<td><strong>44 DMIPS, True Low Power</strong></td>
</tr>
<tr>
<td>Industrial &amp; Automotive, 130nm</td>
</tr>
<tr>
<td>144µA/MHz, 0.2µA standby</td>
</tr>
</tbody>
</table>
## Microcontroller and Microprocessor Line-up

### 8/16-Bit True Low Power
- **High Efficiency & Integration**

### 2010

<table>
<thead>
<tr>
<th>Processor</th>
<th>Speed</th>
<th>Power Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SuperH</strong></td>
<td>1200 DMIPS, Superscalar</td>
<td></td>
</tr>
<tr>
<td>- Automotive &amp; Industrial, 65nm</td>
<td>600µA/MHz</td>
<td>0.2µA standby</td>
</tr>
<tr>
<td>- Industrial, 150nm</td>
<td>190µA/MHz</td>
<td>100µA standby</td>
</tr>
<tr>
<td><strong>V850</strong></td>
<td>500 DMIPS</td>
<td></td>
</tr>
<tr>
<td>- Automotive &amp; Industrial, 242µA/MHz</td>
<td>0.2µA standby</td>
<td></td>
</tr>
<tr>
<td>- Industrial, 90nm</td>
<td>242µA/MHz</td>
<td>0.2µA standby</td>
</tr>
<tr>
<td><strong>RX</strong></td>
<td>165 DMIPS, FPU, DSC</td>
<td></td>
</tr>
<tr>
<td>- Automotive &amp; Industrial, 600µA/MHz</td>
<td>1.5µA standby</td>
<td></td>
</tr>
<tr>
<td>- Automotive &amp; Industrial, 65nm</td>
<td>600µA/MHz</td>
<td>1.5µA standby</td>
</tr>
<tr>
<td><strong>78K</strong></td>
<td>25 DMIPS</td>
<td></td>
</tr>
<tr>
<td>- Industrial &amp; Automotive, 144µA/MHz</td>
<td>0.2µA standby</td>
<td></td>
</tr>
<tr>
<td>- Industrial &amp; Automotive, 130nm</td>
<td>350µA/MHz</td>
<td>1µA standby</td>
</tr>
</tbody>
</table>

### 2013

<table>
<thead>
<tr>
<th>Processor</th>
<th>Speed</th>
<th>Power Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RH850</strong></td>
<td>1200 DMIPS, Performance</td>
<td></td>
</tr>
<tr>
<td>- Automotive, 40nm</td>
<td>1.5µA/MHz</td>
<td>0.2µA standby</td>
</tr>
<tr>
<td>- Automotive, 90nm</td>
<td>1mA/MHz</td>
<td>100µA standby</td>
</tr>
<tr>
<td><strong>RL78</strong></td>
<td>1200 DMIPS, True Low Power</td>
<td></td>
</tr>
<tr>
<td>- Industrial &amp; Automotive, 130nm</td>
<td>144µA/MHz</td>
<td>0.2µA standby</td>
</tr>
</tbody>
</table>

---

© 2012 Renesas Electronics America Inc. All rights reserved.

DevCon
Enabling the Smart Society
‘Enabling The Smart Society’

■ Challenge:
“MCUs continue to become more complex especially when creating applications to enable the Smart Society, and therefore the Software becomes more complex as well. As a result Project schedules do not always scale accordingly and requires helps from the vendor to write the MCU low level driver or provide a tool which can help them to develop the MCU driver.”

■ Solution:
“This class will help you how to use Renesas code generator tool, Applilet for RL78 devices to quickly generate the MCU low level drivers and run the driver code on RL78/G14 target board.”
Agenda

- Code generation for RL78/G14 device, R5F104PJ using Applilet Tool
- Create e²studio Project Template
- Import Applilet generated IAR project files to e²studio
- Build and Run the project
- POP QUIZ
Lab Objectives

- Main goals
  - Learn how to use Renesas code generator tool, Applilet for RL78/G14 devices while going over the tools and peripheral configuration
  - Build and Run the Program on the Target hardware

- Additional goals
  - Become familiar with evaluation platform for RL78/G14 MCU
  - Learn common tips and tricks for e²studio
Lab Platform

- **Hardware Platform**
  - RL78/G14 RDK
    - R5F104PJ device,
      - 256KB Flash, 8KB Data Flash, 24KB SRAM.
    - USB cable for power and debug communication
    - OCD Debugger (TK debugger)

- **Software Platform**
  - e²studio version 1.1
  - RL78 IAR version 1.2
  - Applilet 3 Tool for RL78/G14 Family
RL78/G14 RDK’s User Features

- TK Debug
- RS-232
- Digital sensors: Temp & Accel
- Gainspan Wi-fi
- Application header: WiFi
- TRIAC
- FET
- Graphics LCD
- RL78/G14 MCU
- Eink display
- LED Ring
Definition of Terms

Software Elements

- e²studio – Eclipse based Integrated Development Environment for Renesas Microcontroller families RL78, RX, SH.

- Perspective – A perspective is like a page in a book which contains set of views.
Definition of Terms (Cont.)

- **Software Elements (cont.)**
  - **View:** A perspective consists of a number of sub-windows, called *views*.
  - **Applilet:** Renesas tools for code generation

- **Hardware Elements**
  - **TK interface – On-chip debug emulator for RL78 device.**
  - **RDK – Renesas Demonstration Kit**
  - **E1 - On-chip debug emulator**
Project Phase

- Code Generator allows you to concentrate to create application.
- You can initialize and drive MCU very easily.

Example of API function:

```
R_TAU0_Channel0_Start(); // Timer Start
R_UART0_Start(); // UART Start
```
Why use code generator? (1/2)

HW Manual: 1000~2000 Pages

C Source File

Code 10K+ Lines

Code Generator
GUI: A few mouse clicks

Code Generation

Auto add to project
Why use code generator? (2/2)

- Application source code is more portable

Application code based on Code generated functions can be ported to another hardware with limited changes or no changes at all!
Renesas Code Generator : Applilet

- Applilet: Application Leading Tool
  - RL78
  - V850
  - 78K
Applilet Features

- Generate device driver code for on-chip peripherals
- Easy to use graphical user interface
- Quick evaluation of device
- Common API to all devices
  - Example: serial module
    - `void UARTn_SendData( UCHAR* txbuf, USHORT txnum )`
    - `void UARTn_ReceiveData( UCHAR* rxbuf, USHORT rxnum )`
- Can be used in conjunction with CubeSuite+ when integrated Code Generator not available
- Code Generation for Cubesuite+, GHS e2studio* and IAR tool chain.
- Free Downloaded from the Renesas website

*:under development
Applilet for RL78 Device

- Applilet specific to the RL78 family
- Generate code for Cubesuite+, e2studio*, and IAR Tool Chain
- Same GUI as Cubesuite+ code generator
- Creates the project Framework for IAR/Cubesuite+
  - Add files
  - Change the function name

* Under development
Applilet Function Setting

- Code generator function can be customized.

- Doesn’t generate the new file
- Protect the user code
- Lose any user modification
Bring Applilet Generated IAR Code to e²studio

- Create a new project for RL78 in e²studio (a)
- Registration of IAR source files and linker files (b)
- Set project options (c)
Common Issues

- I lost my code when I re-generated the code from the Applilet.
  -> Make sure you select the “merge file option” in the option setting. Also, all user code must be written in the user area defined as below:

```c
/* Start user code. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

- I didn’t enable the watchdog timer but Applilet generated the code for it?
  -> By default, the watchdog timer is enabled in the code. It is required to disabled in the Applilet under watchdog option.

- Can’t connect e²studio to RDK board for the debug session?
  -> Check the SW5 DIP-switch setting on the board as:

```
1:ON; 2:OFF; 3:ON ; 4:ON
```
Lab Time!

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

Using a Renesas Code Generation Tool for RL78 Devices

**Description:** This hands-on lab will step through how to generate the RL78 code for IAR compiler using Renesas code generator tool, Applilet. Then bring the Applilet generated code to Renesas new Eclipse-based IDE, e²studio enviornment, Build the project in e²studio using IAR Tool chain and and then Flash program the project. This lab will work through the Applilet configuration for system and peripherals setting and E2Studio configuration to edit, build the project.

**Lab Objectives**
1. How to Generate RL78 code for IAR platform using Applilet
2. Edit/Compile/build debug the project in e²studio

**Lab Materials**
Please verify you have the following materials at your lab station.
- Laptop PC with e²studio/Applilet tools pre-installed
- RL78/G14 Target board
- USB cable
Questions?
‘Enabling The Smart Society’ in Review...

Challenge:

“MCUs continue to become more complex especially when creating applications to enable the Smart Society, and therefore the Software becomes more complex as well. As a result Project schedules do not always scale accordingly and requires helps from the vendor to write the MCU low level driver or provide a tool which can help them to develop the MCU driver.”

Solution:

“This class will help you how to use Renesas code generator tool, Applilet for RL78 devices to quickly generate the MCU low level drivers and run the driver code on RL78/G14 target board.”

Do you agree that we accomplished the above statement?
Pop Quiz

Q: What is the advantage of using Applilet?

- a) Faster time to market
- b) Code portability
- c) No hardware changes
- d) Easy to switch to another device in the family

Q: What is the name of the on chip debugger used on the RDK board?

- a) E1
- b) E20
- c) TK debugger
- d) IE Cube
Pop Quiz

Q: What MCU family is supported by Applilet?

- a) RL78
- b) V850
- c) RX
- d) 78K

Q: Where should you write the code in the Applilet generated project files to avoid overwriting of code during the code re-generation?

- a) Anywhere in the file
- b) Code protected area marked as “Start and End user code”
- c) At the beginning of the file
- d) At the end of the file