RX21A Group
Using the Temperature Sensor to Calculate the Ambient Temperature

Abstract
This document describes a method of using the RX21A Group temperature sensor to calculate the ambient temperature.

Products
RX21A Group, 64-Pin Package, ROM Capacity: 256 Kbytes to 512 Kbytes
RX21A Group, 80-Pin Package, ROM Capacity: 256 Kbytes to 512 Kbytes
RX21A Group, 100-Pin Package, ROM Capacity: 256 Kbytes to 512 Kbytes

Note: Only the G version (operating temperature: −40°C to +105°C) of the products are the target products.
1. Specifications

This document describes using the temperature sensor to measure the ambient temperature of the MCU. The ambient temperature is measured and the result is displayed on a 7-segment LED (hereinafter referred to as 7SEG).

In order to measure the ambient temperature of the MCU, the temperature sensor is calibrated beforehand. The calibration performed in this application note calculates the temperature slope necessary for the formula for the temperature characteristic.

In the G version of the RX21A Group MCU, the calibration data for the temperature sensor that is measured for every chip is stored when shipped. The temperature slope can be calculated using the data stored on the chip and a temperature obtained by the user in the trial measurement.

In the accompanying sample code, an ambient temperature of 25°C (hereinafter referred to as normal reference temperature) is assumed as the temperature obtained in the user trial measurement and used to calculate the ambient temperature. Refer to section 5.1.1 for details on calibration.

Table 1.1 lists the Peripheral Functions and Their Applications.

<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-bit A/D converter (hereinafter referred to as AD)</td>
<td>The AD measures temperature sensor output.</td>
</tr>
<tr>
<td>Temperature sensor</td>
<td>The temperature sensor measures the ambient temperature of the MCU.</td>
</tr>
<tr>
<td>Compare match timer (CMT0) (hereinafter referred to as CMT)</td>
<td>The CMT is used as a timer for the temperature measurement cycle.</td>
</tr>
<tr>
<td>External pin interrupt (IRQ2) (hereinafter referred to as IRQ)</td>
<td>Switch input for calibrating with the normal reference temperature.</td>
</tr>
<tr>
<td>I/O ports</td>
<td>I/O ports are used to display the result of the temperature measurement on the 7SEG.</td>
</tr>
</tbody>
</table>
Figure 1.1 shows the Transitioning States and Patterns Displayed on the 7SEG.

<table>
<thead>
<tr>
<th>Status</th>
<th>7SEG display</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset state</td>
<td>All segments are off</td>
</tr>
<tr>
<td>Release from the reset state</td>
<td>Dash displayed</td>
</tr>
<tr>
<td>Waiting for calibration to start</td>
<td></td>
</tr>
<tr>
<td>Switch is pushed</td>
<td>Temperature measurement result displayed as a decimal number</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Display when the temperature measurement result is less than 0°C</td>
</tr>
<tr>
<td></td>
<td>Display when the temperature measurement result is 100°C or higher</td>
</tr>
</tbody>
</table>

Figure 1.1 Transitioning States and Patterns Displayed on the 7SEG
2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU used</td>
<td>R5F521A8BDFP (RX21A Group)</td>
</tr>
<tr>
<td>Operating frequencies</td>
<td>Main clock: 20 MHz System clock (iCLK): 20 MHz Peripherals module clock B (PCLKB): 20 MHz Peripherals module clock D (PCLKD): 2.5 MHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Integrated development environment</td>
<td>Renesas Electronics Corporation High-performance Embedded Workshop Version 4.09.01</td>
</tr>
<tr>
<td>C compiler</td>
<td>Renesas Electronics Corporation C/C++ Compiler Package for RX Family V.1.02 Release 01</td>
</tr>
<tr>
<td>Compile options</td>
<td>~cpu=rx200 –output=obj=&quot;$(CONFIGDIR)$(FILELEAF).obj&quot; –debug –nologo</td>
</tr>
<tr>
<td>The integrated development environment default settings are used.</td>
<td></td>
</tr>
<tr>
<td>iofdefine.h version</td>
<td>Version 1.1</td>
</tr>
<tr>
<td>Endian</td>
<td>Little endian</td>
</tr>
<tr>
<td>Operating mode</td>
<td>Single-chip mode</td>
</tr>
<tr>
<td>Processor mode</td>
<td>Supervisor mode</td>
</tr>
<tr>
<td>Sample code version</td>
<td>Version 1.00</td>
</tr>
</tbody>
</table>

3. Reference Application Notes

For additional information associated with this document, refer to the following application notes.

- RX21A Group Initial Setting Rev. 1.00 (R01AN1486EJ)
- RX Family Coding Example of Wait Processing by Software Rev. 1.00 (R01AN1852EJ)

The initial setting functions and wait processing by software in the reference application notes are used in the sample code in this application note. The revision numbers of the reference application notes are current as of the issue date of this application note. However, the latest versions are always recommended. Visit the Renesas Electronics Corporation website to check and download the latest versions.
4. Hardware

4.1 Hardware Configuration

Figure 4.1 shows the Connection Example.

![Connection Example Diagram]

Note 1. This example assumes 7SEG dynamic common anodes are used.

Figure 4.1 Connection Example

4.2 Pins Used

Table 4.1 lists the Pins Used and Their Functions. The pins used assume that the target product is a 100-pin MCU. When using products with less than 100 pins, select pins appropriate to the product used.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P32/IRQ2</td>
<td>Input</td>
<td>Switch input for executing calibration</td>
</tr>
<tr>
<td>P20</td>
<td>Output</td>
<td>Outputs segment a of the 7SEG</td>
</tr>
<tr>
<td>P21</td>
<td>Output</td>
<td>Outputs segment b of the 7SEG</td>
</tr>
<tr>
<td>P22</td>
<td>Output</td>
<td>Outputs segment c of the 7SEG</td>
</tr>
<tr>
<td>P23</td>
<td>Output</td>
<td>Outputs segment d of the 7SEG</td>
</tr>
<tr>
<td>P24</td>
<td>Output</td>
<td>Outputs segment e of the 7SEG</td>
</tr>
<tr>
<td>P25</td>
<td>Output</td>
<td>Outputs segment f of the 7SEG</td>
</tr>
<tr>
<td>P26</td>
<td>Output</td>
<td>Outputs segment g of the 7SEG</td>
</tr>
<tr>
<td>PA0</td>
<td>Output</td>
<td>Outputs the first digit of the 7SEG</td>
</tr>
<tr>
<td>PA1</td>
<td>Output</td>
<td>Outputs the second digit of the 7SEG</td>
</tr>
</tbody>
</table>
5. Software

5.1 Operation Overview

After the MCU is released from the reset state, the I/O ports and peripheral functions are initialized, and the MCU enters the waiting for calibration state. If the IRQ2 interrupt request is generated in this state, calibration is performed. The normal reference temperature is A/D converted in the calibration. The A/D converted value and the temperature sensor calibration data are used to calculate the temperature slope.

When calibration is complete, A/D conversion continues. The A/D converted value and temperate slope are used to calculate the ambient temperature, and the calculated value is displayed on the 7SEG.

In this application note, A/D conversion is performed every 100 ms. Also, in order to calculate the average A/D converted value, six A/D converted values are stored to the RAM, the highest and lowest values are eliminated, and the average of the remaining four values is calculated as the ambient temperature.

The CMT CMI0 interrupt is used to start A/D conversion every 100 ms. The CMT is set to generate a compare match interrupt request in 1 ms cycles, and for each compare match interrupt request generated, the A/D converter cycle counter variable (cnt_cycle) is incremented up to 100 ms.

Settings for the CMT, AD, and temperature sensor are listed below.

CMT0
- Count clock: PCLKB divided by 8
- Compare match interrupt cycle: 1 ms

AD
- Operating mode: Single scan mode
- A/D conversion start condition: Synchronous trigger (trigger from the temperature sensor)
- Number of sampling states: 180 states (sampling time is 72 µs)
- Analog input disconnection detection assist: Not used
- A/D-converted value addition mode: Not used
- Self-diagnosis of 10-bit A/D converter: Not used

Temperature sensor
- PGA gain \(^{1}\): \(2.7 \text{ V} \leq \text{AVCC0} \leq 3.6 \text{ V} \) \(^{2}\)
  
  Note 1. PGA: Programmable gain amplifier
  
  Note 2. Change the constant settings as needed for the user system.
Figure 5.1 shows the Temperature Measurement Timing Diagram.

(1) After the MCU is released from the reset state, the AD and temperature sensor are initialized.
(2) After the AD is released from the module stop state, the MCU waits 1 µs, and then enters the calibration wait state. At this time, a dash is displayed on the 7SEG.
(3) When a falling edge is detected on the switch (IRQ2), the CMT count starts.
(4) The CMT is set to generate a compare match interrupt request in 1 ms cycles, and for each compare match interrupt request generated, the A/D converter cycle counter variable (cnt_cycle) is incremented.
(5) When the A/D converter cycle counter variable reaches 100 (100 ms), the TSCR.PGAEN bit is set to 1 (starts PGA), and A/D conversion starts.
(6) A/D conversion is performed six times. Their average becomes the A/D converted value of the normal reference temperature, the temperature slope is calculated, and calibration is done.
(7) When the A/D converter cycle counter variable reaches 100 (100 ms), the TSCR.PGAEN bit is set to 1, and A/D conversion starts.
(8) After performing A/D conversion six times, the current temperature is calculated using the average and the temperature slope, and then displayed on the 7SEG.

Note 1. After the AD is released from the module stop state, wait at least 1 µs before starting A/D conversion.
5.1.1 Formula for the Temperature Characteristic

In this application note, the slope necessary for the temperature characteristic formula is calculated using the following items:

- An ambient temperature of 125°C (hereinafter referred to as high reference temperature) stored in the temperature sensor calibration data registers (TSCDRn (n=0,1,3))
- The A/D converted value of the normal reference temperature measured after the MCU is released from the reset state.

Refer to the RX21A User's Manual: Hardware (hereinafter referred to as UMH) for details on the TSCDRn register (n=0,1,3).

Table 5.1 lists the Conditions for Measuring the A/D Converted Values of the Temperature Sensor Output Values Stored in the TSCDRn Register (n=0,1,3).

Table 5.1 Conditions for Measuring the A/D Converted Values of the Temperature Sensor Output Values Stored in the TSCDRn Register (n=0,1,3)

<table>
<thead>
<tr>
<th>Register Symbol</th>
<th>Conditions for Measuring A/D Converted Values</th>
<th>Voltage applied to AVCC0 and VREFH0</th>
<th>TSCR.PGAGAIN[1:0] bits</th>
<th>Temperature for measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSCDR0</td>
<td></td>
<td>1.8 V</td>
<td>00b</td>
<td>125°C</td>
</tr>
<tr>
<td>TSCDR1</td>
<td></td>
<td>2.7 V</td>
<td>01b</td>
<td></td>
</tr>
<tr>
<td>TSCDR3</td>
<td></td>
<td>3.3 V</td>
<td>01b</td>
<td></td>
</tr>
</tbody>
</table>

When applying voltage not listed in Table 5.1 to AVCC0 and VREFH, the A/D converted value must be calculated according to the applied voltage. The A/D conversion value to be calculated is defined as CAL125 here.

When AVCC0 is 1.8 V ≤ AVCC0 < 2.7 V, then formula [1] below is used to calculate CAL125; when AVCC0 is 2.7 V ≤ AVCC0 ≤ 3.6 V, then formula [2] below is used to calculate CAL125.

[1] CAL125 = 1.8 ÷ VREFH0 × TSCDR0

[2] CAL125 = 2.7 ÷ VREFH0 × TSCDR1 + {3.3 ÷ VREFH0 × TSCDR3} - (AVCC0 - 2.7) ÷ 0.6

TSCDR(n): TSCDRn.TSCD[9:0] bit value (n = 0, 1, 3)

To calculate the ambient temperature, the temperature slope must be calculated first. Here, the temperature slope become is defined as the increment value of the A/D converted value to the temperature. Note that the UMH describes the method to calculate the temperature slope and temperature after converting the A/D converted value to voltage, but this application note calculates the temperature slope and temperature using the A/D converted value with no conversion to voltage.

The formula for calculating the temperature slope is below.

Temperature slope: Slope

High reference temperature (125°C): T1

Normal reference temperature (25°C): T2

A/D converted value of the high reference temperature (125°C): CAL125

A/D converted value of the normal reference temperature (25°C): CAL25 (value measured using the normal reference temperature after the MCU is released from the reset state)

Temperature slope: Slope = (CAL125 - CAL25) ÷ (T1 - T2)

Since T1 = 125(°C) and T2 = 25(°C), the slope becomes the following:

Slope = (CAL125 - CAL25) ÷ (125 - 25) = (CAL125 - CAL25) ÷ 100
The formula for calculating the ambient temperature is below.

Measured temperature: $T\ (°C)$

A/D converted value of the temperature sensor when the temperature was measured: $CALS$

$$T = T_2 + \frac{(CALS - CAL_{25})}{\text{Slope}}$$

$$= T_2 + \frac{(CALS - CAL_{25})}{((CAL_{125} - CAL_{25}) \div (T_1 - T_2))}$$

$$= T_2 + \frac{(T_1 - T_2)}{(CAL_{125} - CAL_{25})} ((CALS - CAL_{25}))$$

$$= 25 + 100((CALS - CAL_{25}) \div (CAL_{125} - CAL_{25}))$$

When measuring the temperature to the tenths place, temperature data $(T_1, T_2)$ is multiplied by 10.

Measured temperature: $Ts\ (°C)$

$$Ts = T \times 10 = (25 + 100((CALS - CAL_{25}) \div (CAL_{125} - CAL_{25}))) \times 10$$

$$= (25 \times 10) + (100((CALS - CAL_{25}) \div (CAL_{125} - CAL_{25}))) \times 10$$

$$= 250 + 1000((CALS - CAL_{25}) \div (CAL_{125} - CAL_{25}))$$

Refer to the UMH for basic information.
### 5.2 File Composition

Table 5.2 lists the Files Used in the Sample Code, Table 5.3 lists the Standard Include Files, and Table 5.4 lists Functions and Setting Values for the Reference Application Notes. Files generated by the integrated development environment are not included in this table.

**Table 5.2 Files Used in the Sample Code**

<table>
<thead>
<tr>
<th>File Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.c</td>
<td>Main processing</td>
</tr>
<tr>
<td>temps.c</td>
<td>Temperature sensor processing</td>
</tr>
<tr>
<td>temps.h</td>
<td>Header file for temps.c</td>
</tr>
</tbody>
</table>

**Table 5.3 Standard Include Files**

<table>
<thead>
<tr>
<th>File Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>stdbool.h</td>
<td>This file defines the macros associated with the Boolean and its value.</td>
</tr>
<tr>
<td>stdint.h</td>
<td>This file defines the macros declaring the integer type with the specified width.</td>
</tr>
<tr>
<td>machine.h</td>
<td>This file defines the types of intrinsic functions for the RX Family.</td>
</tr>
</tbody>
</table>

**Table 5.4 Functions and Setting Values for the Reference Application Notes (RX21A Group Initial Setting, RX Family Coding Example of Wait Processing by Software)**

<table>
<thead>
<tr>
<th>File Name</th>
<th>Function</th>
<th>Setting Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_init_stop_module.c</td>
<td>R_INIT_StopModule()</td>
<td></td>
</tr>
<tr>
<td>r_init_stop_module.h</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>r_init_non_existent_port.c</td>
<td>R_INIT_NonExistentPort()</td>
<td></td>
</tr>
<tr>
<td>r_init_non_existent_port.h</td>
<td>—</td>
<td>Set to 100-pin package</td>
</tr>
<tr>
<td>r_init_clock.c</td>
<td>R_INIT_Clock()</td>
<td></td>
</tr>
<tr>
<td>r_init_clock.h</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>r_delay.c</td>
<td>R_DELAY_Us(unsigned long us, unsigned long khz)</td>
<td></td>
</tr>
<tr>
<td>r_delay.h</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>
5.3 Option-Setting Memory

Table 5.5 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFS0</td>
<td>FFFF FF8Fh to FFFF FF8Ch</td>
<td>FFFF FFFFh</td>
<td>The IWDT is stopped after a reset. The WDT is stopped after a reset.</td>
</tr>
<tr>
<td>OFS1</td>
<td>FFFF FF88h to FFFF FF8Ah</td>
<td>FFFF FFFFh</td>
<td>The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.</td>
</tr>
<tr>
<td>MDES</td>
<td>FFFF FF83h to FFFF FF80h</td>
<td>FFFF FFFFh</td>
<td>Little endian</td>
</tr>
</tbody>
</table>

5.4 Constants

Table 5.6 to Table 5.9 list the constants used in the sample code.

Table 5.6 Constants Used in the Sample Code (main.c)

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMT_CYCLE_MS</td>
<td>100</td>
<td>A/D conversion cycle (ms)</td>
</tr>
<tr>
<td>SEG_CYCLE_MS</td>
<td>8</td>
<td>7SEG select output switch cycles (ms)</td>
</tr>
<tr>
<td>ONES_DIGIT</td>
<td>0</td>
<td>7SEG output flag value</td>
</tr>
<tr>
<td>SEG_TBL_DASH</td>
<td>10</td>
<td>7SEG display table index: &quot;—&quot;</td>
</tr>
<tr>
<td>SEG_TBL_H</td>
<td>11</td>
<td>7SEG display table index: &quot;H&quot;</td>
</tr>
<tr>
<td>SEG_TBL_i</td>
<td>12</td>
<td>7SEG display table index: &quot;i&quot;</td>
</tr>
<tr>
<td>SEG_TBL_L</td>
<td>13</td>
<td>7SEG display table index: &quot;L&quot;</td>
</tr>
<tr>
<td>SEG_TBL_o</td>
<td>14</td>
<td>7SEG display table index: &quot;o&quot;</td>
</tr>
<tr>
<td>SEG_TBL_BLANK</td>
<td>15</td>
<td>7SEG display table index: Blank</td>
</tr>
</tbody>
</table>

Table 5.7 Constants Used in the Sample Code (temps.c)

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGH_REF_TEMP</td>
<td>125</td>
<td>High reference temperature (°C)</td>
</tr>
<tr>
<td>ADCONV_IN_OPERATION</td>
<td>0xFFFF</td>
<td>A/D converted value during A/D conversion (invalid value)</td>
</tr>
<tr>
<td>SLOPE_COEFFICIENT_TEMP</td>
<td>(HIGH_REF_TEMP — ORDINARY_REF_TEMP) * TEMP_ACCURACY</td>
<td>Temperature slope</td>
</tr>
<tr>
<td>ORDINARY_REF_TEMP_IN_ACC</td>
<td>ORDINARY_REF_TEMP * TEMP_ACCURACY</td>
<td>Value of the normal reference temperature multiplied by the temperature calculation accuracy</td>
</tr>
</tbody>
</table>
### Table 5.8  Constants Used in the Sample Code (temps.h) (Changeable by the User)

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEL_PGAGAIN</td>
<td>GAIN_RANGE1</td>
<td>Select PGA gain&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GAIN_RANGE0: 1.8 V ≤ AVCC0 &lt; 2.7 V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GAIN_RANGE1: 2.7 V ≤ AVCC0 ≤ 3.6 V</td>
</tr>
<tr>
<td>AVCC_VOLTAGE</td>
<td>3.3</td>
<td>Voltage applied to the AVCC0 pin (in units of V)&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>VREF_VOLTAGE</td>
<td>3.3</td>
<td>Voltage applied to the VREFH0 pin (in units of V)</td>
</tr>
<tr>
<td>ORDINARY_REF_TEMP</td>
<td>25</td>
<td>Normal reference temperature (°C): If the value set is 25, then the normal reference temperature is assumed to be 25°C.</td>
</tr>
<tr>
<td>TEMP_ACCURACY</td>
<td>10</td>
<td>Temperature calculation accuracy: The multiplication rate is set. When the value set is &quot;10&quot;, the value is calculated to the tenths place. When the value set is &quot;100&quot;, the value is calculated to the hundreds place. Do not set a multiplier other than a multiple of 10, and do not set a negative value.</td>
</tr>
<tr>
<td>CNV_CNT_MAX</td>
<td>6</td>
<td>Number of average value samplings: If the set value is 6, when six A/D converted values have been accumulated, the highest and lowest values are excluded, and the average of the remaining four becomes the A/D converted value.</td>
</tr>
</tbody>
</table>

Note 1. Specify the value according to the voltage applied. If the value specified is inappropriate, the calculated result will be incorrect.
Using the Temperature Sensor to Calculate the Ambient Temperature

### Table 5.9 Constants Used in the Sample Code (temps.h) (Not Changeable by the User)

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAIN_RANGE0</td>
<td>00h</td>
<td>PGA gain: $1.8 \leq AVCC0 &lt; 2.7$ V</td>
</tr>
<tr>
<td>GAIN_RANGE1</td>
<td>01h</td>
<td>PGA gain: $2.7 \leq AVCC0 \leq 3.6$ V</td>
</tr>
<tr>
<td>STA_AD_IDLE</td>
<td>0</td>
<td>A/D conversion status: Not performed</td>
</tr>
<tr>
<td>STA_AD_WAIT</td>
<td>1</td>
<td>A/D conversion status: Waiting for A/D conversion to be completed</td>
</tr>
<tr>
<td>STA_AD_FINISH</td>
<td>2</td>
<td>A/D conversion status: A/D conversion completed</td>
</tr>
<tr>
<td>TSCDR0_VALUE</td>
<td>(TEMPSCONST.TSCDR0.BIT.TSCD)</td>
<td>TSCDR0 register value</td>
</tr>
<tr>
<td>TSCDR1_VALUE</td>
<td>(TEMPSCONST.TSCDR1.BIT.TSCD)</td>
<td>TSCDR1 register value</td>
</tr>
<tr>
<td>TSCDR3_VALUE</td>
<td>(TEMPSCONST.TSCDR3.BIT.TSCD)</td>
<td>TSCDR3 register value</td>
</tr>
<tr>
<td>HIGH_REF_POTENTIAL_VAL</td>
<td>See Note 1</td>
<td>A/D converted value of the high reference temperature</td>
</tr>
</tbody>
</table>

**Note 1.** The setting value varies according to the PGA gain selected. The following shows the setting value for each PGA gain.

When GAIN_RANGE0 is selected:

\[
(\text{uint16_t})(1.8 \div \text{VREF_VOLTAGE} \times \text{TSCDR0\_VALUE})
\]

When GAIN_RANGE1 is selected:

\[
(\text{uint16_t})(2.7 \div \text{VREF_VOLTAGE} \times \text{TSCDR1\_VALUE}) + ((3.3 \div \text{VREF_VOLTAGE} \times \text{TSCDR3\_VALUE}) - (2.7 \div \text{VREF_VOLTAGE} \times \text{TSCDR1\_VALUE})) \times (\text{AVCC\_VOLTAGE} - 2.7) \div 0.6
\]
5.5 Variables

Table 5.10 and Table 5.11 list the static variables, and Table 5.12 lists the const Variable.

### Table 5.10 static Variables (main.c)

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>static volatile uint16_t</td>
<td>cnt_cycle</td>
<td>A/D conversion cycle counter</td>
<td>Excep_CMT0_CMI0</td>
</tr>
<tr>
<td>static volatile uint16_t</td>
<td>cnt_led_cycle</td>
<td>7SEG select output switch cycle counter</td>
<td>Excep_CMT0_CMI0</td>
</tr>
<tr>
<td>static uint8_t</td>
<td>digit_10</td>
<td>7SEG second digit display data</td>
<td>disp_7seg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>disp_comswitch_7seg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>disp_bar_7seg</td>
</tr>
<tr>
<td>static uint8_t</td>
<td>digit_1</td>
<td>7SEG first digit display data</td>
<td>disp_7seg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>disp_comswitch_7seg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>disp_bar_7seg</td>
</tr>
</tbody>
</table>

### Table 5.11 static Variables (temps.c)

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int16_t</td>
<td>high_ref_potential</td>
<td>A/D converted value of the high reference temperature (= CAL125)</td>
<td>temps_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>temps_calibration</td>
</tr>
<tr>
<td>static volatile int16_t</td>
<td>slope_potential</td>
<td>Slope of the A/D converted value</td>
<td>temps_calibration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>temps_calc</td>
</tr>
<tr>
<td>static volatile int16_t</td>
<td>ordinary_potential</td>
<td>A/D converted value of the normal reference temperature (= CAL25)</td>
<td>temps_calibration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>temps_calc</td>
</tr>
<tr>
<td>static volatile int8_t</td>
<td>ad_status</td>
<td>A/D conversion status</td>
<td>main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>temps_get_ad_status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>temps_calibration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>temps_measurement</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Excep_AD_ADI</td>
</tr>
<tr>
<td>static volatile int16_t</td>
<td>now_temp</td>
<td>Calculated current temperature</td>
<td>temps_get_now_temp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Excep_AD_ADI</td>
</tr>
<tr>
<td>static volatile uint16_t</td>
<td>now_potential</td>
<td>Current A/D converted value</td>
<td>temps_calibration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Excep_AD_ADI</td>
</tr>
<tr>
<td>static volatile int16_t</td>
<td>buf_ad_value[CNT_MAX]</td>
<td>A/D converted value buffer</td>
<td>Excep_AD_ADI</td>
</tr>
<tr>
<td>static volatile int16_t</td>
<td>ad_max_value</td>
<td>Highest A/D converted value</td>
<td>Excep_AD_ADI</td>
</tr>
<tr>
<td>static volatile int16_t</td>
<td>ad_min_value</td>
<td>Lowest A/D conversion value</td>
<td>Excep_AD_ADI</td>
</tr>
<tr>
<td>static volatile int8_t</td>
<td>ad_smp_cnt</td>
<td>Write pointer for the A/D converted value buffer</td>
<td>Excep_AD_ADI</td>
</tr>
</tbody>
</table>

### Table 5.12 const Variable

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>static const uint8_t</td>
<td>seg_pattern_table</td>
<td>7SEG display table</td>
<td>disp_comswitch_7seg</td>
</tr>
</tbody>
</table>
5.6 Functions

Table 5.13 lists the Functions.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>Main processing</td>
<td>main.c</td>
</tr>
<tr>
<td>port_init</td>
<td>Port initialization</td>
<td>main.c</td>
</tr>
<tr>
<td>peripheral_init</td>
<td>Peripheral function initialization</td>
<td>main.c</td>
</tr>
<tr>
<td>cmt_init</td>
<td>CMT initialization</td>
<td>main.c</td>
</tr>
<tr>
<td>irq_init</td>
<td>IRQ initialization</td>
<td>main.c</td>
</tr>
<tr>
<td>disp_7seg</td>
<td>Processing to update the 7SEG display data</td>
<td>main.c</td>
</tr>
<tr>
<td>disp_comswitch_7seg</td>
<td>Processing to switch the 7SEG select output</td>
<td>main.c</td>
</tr>
<tr>
<td>disp_bar_7seg</td>
<td>Processing to display a dash on the 7SEG</td>
<td>main.c</td>
</tr>
<tr>
<td>Excep_CMT0_CMI0</td>
<td>Compare match interrupt handling</td>
<td>main.c</td>
</tr>
<tr>
<td>temps_init</td>
<td>AD and temperature sensor initialization</td>
<td>temps.c</td>
</tr>
<tr>
<td>temps_get_ad_status</td>
<td>Obtain the A/D conversion status</td>
<td>temps.c</td>
</tr>
<tr>
<td>temps_get_potential</td>
<td>Obtain the temperature sensor measurement result</td>
<td>temps.c</td>
</tr>
<tr>
<td>temps_get_now_temp</td>
<td>Obtain the current temperature</td>
<td>temps.c</td>
</tr>
<tr>
<td>temps_calibration</td>
<td>Processing for temperature sensor calibration</td>
<td>temps.c</td>
</tr>
<tr>
<td>temps_measurement</td>
<td>Processing for temperature sensor measurement</td>
<td>temps.c</td>
</tr>
<tr>
<td>temps_calc</td>
<td>Processing to calculate the current temperature</td>
<td>temps.c</td>
</tr>
<tr>
<td>Excep_AD_ADI</td>
<td>A/D conversion complete interrupt handling</td>
<td>temps.c</td>
</tr>
</tbody>
</table>
5.7 Function Specifications

The following tables list the sample code function specifications.

<table>
<thead>
<tr>
<th>Function</th>
<th>Outline</th>
<th>Header</th>
<th>Declaration</th>
<th>Description</th>
<th>Arguments</th>
<th>Return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>Main processing</td>
<td>None</td>
<td>void main(void)</td>
<td>After initialization, this function A/D converts the temperature sensor output every 100 ms, and the calculated temperature is displayed on the 7SEG.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>port_init</td>
<td>Port initialization</td>
<td>None</td>
<td>static void port_init(void)</td>
<td>This function initializes the ports.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>peripheral_init</td>
<td>Peripheral function initialization</td>
<td>None</td>
<td>static void peripheral_init(void)</td>
<td>This function initializes the peripheral functions.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>cmt_init</td>
<td>CMT initialization</td>
<td>None</td>
<td>static void cmt_init(void)</td>
<td>This function initializes CMT0.</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
irq_init
Outline        IRQ initialization
Header         None
Declaration    static void irq_init(void)
Description    This function initializes IRQ2.
Arguments      None
Return value   None

disp_7seg
Outline        Processing to update the 7SEG display data
Header         None
Declaration    static void disp_7seg(int16_t disp_data)
Description    This function sets the value specified in the argument as the data to be displayed in the 7SEG.
Arguments      int16_t disp_data : 7SEG display
                Less than 0 (negative value): "Lo" is displayed
                100 or higher: "Hi" is displayed
                Other than above: Temperature is displayed
Return value   None

disp_comswitch_7seg
Outline        Processing to switch the 7SEG select output
Header         None
Declaration    static void disp_comswitch_7seg(void)
Description    This function switches the 7SEG select signal to be output.
Arguments      None
Return value   None

disp_bar_7seg
Outline        Processing to display a dash on the 7SEG
Header         None
Declaration    static void disp_bar_7seg(void)
Description    This function displays a dash on the 7SEG.
Arguments      None
Return value   None
## Excep_CMT0_CMI0

**Outline**  
Compare match interrupt handling

**Header**  
None

**Declaration**  
static void Excep_CMT0_CMI0(void)

**Description**  
This function performs interrupt handling in 1 ms cycles. The counter is incremented each time an interrupt request is generated. When the counter reaches 100 (100 ms), temperature measurement is started. Also, after the counter reaches 8 (8 ms), the 7SEG select signal to be output is switched.

**Arguments**  
None

**Return value**  
None

## temps_init

**Outline**  
AD and temperature sensor initialization

**Header**  
temps.h

**Declaration**  
void temps_init(void)

**Description**  
This function initializes the AD and the temperature sensor.

**Arguments**  
None

**Return value**  
None

## temps_get_ad_status

**Outline**  
Obtain the A/D conversion status

**Header**  
temps.h

**Declaration**  
uint8_t temps_get_ad_status(void)

**Description**  
This function obtains the current status of the A/D conversion.

**Arguments**  
None

**Return value**  
uint8_t:  
- STA_AD_IDLE: Not performed
- STA_AD_WAIT: Waiting for A/D conversion to be completed
- STA_AD_FINISH: A/D conversion completed

## temps_get_potential

**Outline**  
Obtain the temperature sensor measurement result

**Header**  
None

**Declaration**  
static uint16_t temps_get_potential(void)

**Description**  
This function obtains the measured A/D converted value.

**Arguments**  
None

**Return value**  
uint16_t:  
- A/D converted value of the temperature sensor:  
  - ADCONV_IN_OPERATION: A/D conversion in process  
  - Other than ADCONV_IN_OPERATION: A/D converted value
### temps_get_now_temp

**Outline**
Obtain the current temperature

**Header**
temps.h

**Declaration**
int16_t temps_get_now_temp (void)

**Description**
This function obtains the current temperature.

**Arguments**
None

**Return value**
int16_t: Current temperature

### temps_calibration

**Outline**
Processing for temperature sensor calibration

**Header**
temps.h

**Declaration**
void temps_calibration(void)

**Description**
This function obtains the A/D converted value of the normal reference temperature, and saves it to the RAM.

**Arguments**
None

**Return value**
None

### temps_measurement

**Outline**
Processing for temperature sensor measurement

**Header**
temps.h

**Declaration**
void temps_measurement(void)

**Description**
This function starts measuring the current temperature.

**Arguments**
None

**Return value**
None

### temps_calc

**Outline**
Processing to calculate the current temperature

**Header**
None

**Declaration**
static uint16_t temps_calc(uint16_t w_now_potential)

**Description**
This function calculates the temperature from the A/D converted value in the argument.

**Arguments**
uint16_t w_now_potential : A/D converted value

**Return value**
int16_t: Current temperature (°C)

### Excep_AD_ADI

**Outline**
A/D conversion complete interrupt handling

**Header**
None

**Declaration**
static void Excep_AD_ADI(void)

**Description**
When A/D conversion is completed, the A/D converted values are saved in the RAM. After the sixth A/D conversion is completed, the highest and lowest A/D converted values are excluded, and the average of the remaining four A/D converted values is calculated.

**Arguments**
None

**Return value**
None
5.8 Flowcharts

5.8.1 Main Processing

Figure 5.2 shows the Main Processing.

![Flowchart](image)

Figure 5.2 Main Processing
## 5.8.2 Port Initialization

Figure 5.3 shows Port Initialization.

<table>
<thead>
<tr>
<th>port_init</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set the 7SEG output ports</td>
</tr>
</tbody>
</table>

- PORT2.PODR register ← 7Fh: P20 to P26: High
- PORTA.PODR register ← 03h: PA0 to PA1: High
- PORT2.PDR register ← 7Fh: P20 to P26: Output
- PORTA.PDR register ← 03h: PA0 to PA1: Output

**Figure 5.3 Port Initialization**

## 5.8.3 Peripheral Function Initialization

Figure 5.4 shows Peripheral Function Initialization.

<table>
<thead>
<tr>
<th>main</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMT initialization</td>
</tr>
<tr>
<td>cmt_init()</td>
</tr>
<tr>
<td>IRQ initialization</td>
</tr>
<tr>
<td>irq_init()</td>
</tr>
</tbody>
</table>

**Figure 5.4 Peripheral Function Initialization**
5.8.4 CMT Initialization

Figure 5.5 shows CMT Initialization.

```
cmt_init

Enable writing to relevant registers
PRCR register ← A502h
PRC1 bit = 1: Enables writing to the registers related to low power consumption

Cancel the CMT module stop
MSTPCRA register
MSTPA15 bit ← 0: The CMT0 module stop state is canceled

Disable writing to relevant registers
PRCR register ← A500h
PRC1 bit = 0: Disables writing to the registers related to low power consumption

Stop the CMT0 count
CMSTR0 register
STR0 bit ← 0: CMT0.CMCNT count is stopped

Select the count source
CMT0.CMCR register ← 00C0h
CMIE bit = 1: Compare match interrupt enabled
CKS[1:0] bits = 00b: PCLK8

Clear the CMT0 counter
CMT0.CMCNT register ← 0000h

Set the CMT0 compare match constant register
CMT0.CMCOR register ← 2500 - 1: CMT cycle = 1 ms [(2500 - 1) × (PCLKB/8)]

Clear the interrupt request
IR28 register
IR bit ← 0: CMI0 interrupt request not generated

Set the interrupts
IPR004 register
IPR[3:0] bits ← 0001b: CMI0 interrupt priority level is level 1
IER03 register
IEN4 bit ← 1: CMI0 interrupt request is enabled
```

Figure 5.5 CMT Initialization
### 5.8.5 IRQ Initialization

Figure 5.6 shows IRQ Initialization.

```plaintext
ira_init

Disable interrupts
IER08 register
IEN2 bit ← 0: IRQ2 interrupt request is disabled

Disable the digital filter
IRQFLTE0 register
FLTEN2 bit ← 0: Digital filter is disabled.

Set the digital filter sampling clock
IRQFLTC0 register
FCLKSEL2[1:0] bits ← 11b: PCLK/64

Set the P32 port mode register
PORT3.PMR register
B2 bit ← 0: Use the pin as a general I/O pin.

Enable writing to the PFSWE bit
PWPR register
B0WI bit ← 0: Writing to the PFSWE bit is enabled

Enable writing to the PFS register
PWPR register
PFSWE bit ← 1: Writing to the PFS register is enabled

Set the P32PFS register
P32PFS register
ISEL bit ← 1: Used as IRQ2 input pin

Disable writing to the PFS register
PWPR register
PFSWE bit ← 0: Writing to the PFS register is disabled

Disable writing to the PFSWE bit
PWPR register
B0WI bit ← 1: Writing to the PFSWE bit is disabled

Set IRQ detection
IRQCR2 register
IRQMD[1:0] bits ← 01b: Falling edge

Clear the interrupt request
IR066 register
IR bit ← 0: IRQ2 interrupt request is not generated

Enable the digital filter
IRQFLTE0 register
FLTEN2 bit ← 1: Digital filter is enabled.

return
```

Figure 5.6 IRQ Initialization
5.8.6 Processing to Update the 7SEG Display Data

Figure 5.7 shows the Processing to Update the 7SEG Display Data.

```
Argument
disp_7seg
int16_t disp_data: Display data

Is the display
data between
0 and 99?
No

Is the display data
100 or higher?
Yes

No (digit != 0)

Turn off the 7SEG
PORTA.PODR register ← 03h

Switch the 7SEG output flag
digit ← digit ^ 1

Is the 7SEG
output flag 0?
No (digit != 0)

Yes (digit == 0)

Calculate the data to be displayed in
the first digit

Calculate the data to be displayed in
the second digit

Output data to the 7SEG
PORT2.PODR register ← data

Turn on the 7SEG
PORTA.PODR register ← data
or
PORTA.PODR register ← 02h: Turn on the first digit (digit == 0)
or
01h: Turn on the second digit (digit != 0)

return
```

Figure 5.7 Processing to Update the 7SEG Display Data

5.8.7 Processing to Switch the 7SEG Select Output

Figure 5.8 shows the Processing to Switch the 7SEG Select Output.

```
Argument
disp_comswitch_7seg

Turn off the 7SEG
PORTA.PODR register ← 03h

Switch the 7SEG output flag
digit ← digit ^ 1

Is the 7SEG
output flag 0?
No (digit != 0)

Yes (digit == 0)

Calculate the data to be displayed in
the first digit

Calculate the data to be displayed in
the second digit

Output data to the 7SEG
PORT2.PODR register ← data

Turn on the 7SEG
PORTA.PODR register ← data
or
PORTA.PODR register ← 02h: Turn on the first digit (digit == 0)
or
01h: Turn on the second digit (digit != 0)

return
```

Figure 5.8 Processing to Switch the 7SEG Select Output
5.8.8 Processing to Display a Dash on the 7SEG
Figure 5.9 shows the Processing to Display a Dash on the 7SEG.

```
5.8.8 Processing to Display a Dash on the 7SEG
```

```
Figure 5.9  Processing to Display a Dash on the 7SEG
```

```
5.8.9 Compare Match Interrupt Handling
Figure 5.10 shows the Compare Match Interrupt Handling.
```

```
5.8.9 Compare Match Interrupt Handling
```

```
Figure 5.10  Compare Match Interrupt Handling
```
5.8.10 AD and Temperature Sensor Initialization

Figure 5.11 shows the AD and Temperature Sensor Initialization.

- **temps_init**
  - Disable the A/D conversion complete interrupt
  - Enable writing to relevant registers
  - Release the AD and temperature sensor from the module stop state
  - Disable writing to relevant registers
  - Wait 1 μs
  - Select the scan mode
  - Select the A/D conversion pins
  - Select A/D conversion for the temperature sensor output
  - Set the number of sampling states
  - Select the A/D conversion start trigger
  - Select the A/D converter start trigger from the temperature sensor
  - Select the PGA gain
  - Start the temperature sensor
  - Wait the temperature sensor startup time (80 μs)
  - Enable the scan complete interrupt
  - Clear the interrupt request
  - Enable the ADI interrupt
  - Obtain the high reference temperature

Note 1. Change the constants to accommodate the user system.
Note 2. The setting value will differ according to the PGA gain selected in the constants.

Figure 5.11 AD and Temperature Sensor Initialization
5.8.11 Obtain the A/D Conversion Status
Figure 5.12 shows Obtain the A/D Conversion Status.

```
temps_get_ad_status
return (ad_status)
```

Figure 5.12 Obtain the A/D Conversion Status

5.8.12 Obtain the Temperature Sensor Measurement Result
Figure 5.13 shows Obtain the Temperature Sensor Measurement Result.

```
temps_get_potential
Is the PGA stopped?
Yes
Obtain the A/D converted value
No
Read the TSCR register
PGAEN bit = 0: Stops PGA.
PGAEN bit = 1: Starts PGA.

w_now_potential ← ADTSDR register value
return(w_now_potential)
```

Figure 5.13 Obtain the Temperature Sensor Measurement Result

5.8.13 Obtain the Current Temperature
Figure 5.14 shows Obtain the Current Temperature.

```
temps_get_now_temp
return (now_temp)
```

Figure 5.14 Obtain the Current Temperature
5.8.14 Processing for Temperature Sensor Calibration

Figure 5.15 shows the Processing for Temperature Sensor Calibration.

```
temps_calibration

1. Wait for temperature sensor calibration to be executed
2. Read the IR066.IR flag (wait until the flag becomes 1)
3. Clear the IRQ2 interrupt request
4. IR flag ← 0: IRQ2 interrupt request not generated
5. Start the CMT0 count
6. CMSTR0 register
7. STR0 bit ← 1: CMT0.CMCNT count is started
8. Wait for A/D conversion to be completed
9. Set the A/D conversion status to "Not performed"
   ad_status ← STA_AD_IDLE
10. Save the A/D converted value of the normal reference temperature (25°C)
    ordinary_potential ← now_potential
11. Save the A/D converted value slope to the RAM
    slope_potential ← high_ref_potential − ordinary_potential
12. return
```

Figure 5.15  Processing for Temperature Sensor Calibration

5.8.15 Processing for Temperature Sensor Measurement

Figure 5.16 shows the Processing for Temperature Sensor Measurement.

```
temps_measurement

1. AD and temperature sensor initialization
   temps_init()
2. Start measuring the temperature
3. TSCR register
4. PGAEN bit ← 1: Starts PGA.
5. Change the A/D conversion status to "Waiting for A/D conversion to be completed"
   ad_status ← STA_AD_WAIT
6. return
```

Figure 5.16  Processing for Temperature Sensor Measurement

5.8.16 Processing to Calculate the Current Temperature

Figure 5.17 shows the Processing to Calculate the Current Temperature.

```
temps_calc

1. Calculate the current temperature
   Argument
   uint16_t w_now_potential: A/D converted value
   (1)
2. return(w_now_temp)
```

Note 1. Refer to section 5.1.1 Formula for the Temperature Characteristic for details on the formula.

Figure 5.17  Processing to Calculate the Current Temperature
5.8.17 A/D Conversion Complete Interrupt Handling

Figure 5.18 shows the A/D Conversion Complete Interrupt Handling.

```
Excep_AD_ADI

Is the status of the A/D conversion “Waiting for A/D conversion to be completed”?  
No (ad_status != STA_AD_WAIT)

Yes (ad_status == STA_AD_WAIT)

Obtain the temperature sensor measurement result temps_get_potential()

now_potential ← A/D converted value

Is the A/D converted value obtained the lowest value?

No

Yes

Update the lowest value

ad_min_value ← A/D converted value

Is the A/D converted value obtained the highest value?

No

Yes

Update the highest value

ad_max_value ← A/D converted value

Save the A/D converted value in the buffer

Is the number of A/D converted values obtained less than CNV_CNT_MAX?

No

Increment the number of A/D converted values obtained

Stop the temperature sensor

TSCR register TSEN bit ← 0: Stops the temperature sensor.

Total the A/D conversion data for the number of times specified in CNV_CNT_MAX

Subtract the lowest and highest values from the total

Calculate the average from the result of the previous process

Save the calculated average in the RAM

now_potential ← Average value

Processing to calculate the current temperature temps_calc()

Set the status of the A/D conversion to “A/D conversion completed”

ad_status ← STA_AD_FINISH

Initialize the lowest and highest values

Initialize the number of A/D conversions

return
```

Figure 5.18 A/D Conversion Complete Interrupt Handling
6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents

User’s Manual: Hardware
   RX21A Group User’s Manual: Hardware Rev.1.00 (R01UH0025EJ)
   The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
   The latest information can be downloaded from the Renesas Electronics website.

User’s Manual: Development Tools
   RX Family C/C++ Compiler Package V.1.01 User’s Manual Rev.1.00 (R20UT0570EJ)
   The latest version can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics website
   http://www.renesas.com

Inquiries
   http://www.renesas.com/contact/
**REVISION HISTORY**

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Sep. 1, 2014</td>
<td>-- First edition issued</td>
</tr>
</tbody>
</table>

All trademarks and registered trademarks are the property of their respective owners.
General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins
   - Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.
   - The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on
   - The state of the product is undefined at the moment when power is supplied.
   - The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   - In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   - In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses
   - Access to reserved addresses is prohibited.
   - The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals
   - After applying a reset, only release the reset line after the operating clock signal has become stable.
   - When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   - When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal.
   - Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products
   - Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.
   - The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.
Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of this circuit, software, or information.

2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of others.

4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.

5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the products quality grade, as indicated below.

6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are subject to radiation resistance design. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, the control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of micromechanics software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.

8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas electronics product or technology described in this document for any purpose relating to military applications, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.

11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(y)" means any product developed or manufactured by or for Renesas Electronics.

© 2014 Renesas Electronics Corporation. All rights reserved.

Colophon 4.0