RX63N Group, RX631 Group
Using the Temperature Sensor to Calculate the Ambient Temperature

Abstract
This document describes a method of using the RX63N Group, RX631 Group temperature sensor to calculate the ambient temperature.

Products
RX63N Group 176-pin package with a ROM size between 768 Kbytes to 2 Mbytes
RX63N Group 144-pin package with a ROM size between 768 Kbytes to 2 Mbytes
RX63N Group 100-pin package with a ROM size between 768 Kbytes to 2 Mbytes
RX631 Group 176-pin package with a ROM size between 256 Kbytes to 2 Mbytes
RX631 Group 144-pin package with a ROM size between 256 Kbytes to 2 Mbytes
RX631 Group 100-pin package with a ROM size between 256 Kbytes to 2 Mbytes
RX631 Group 64-pin and 48-pin packages with a ROM size between 256 Kbytes to 512 Kbytes

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

1. Specifications ........................................................................................................................................ 3

2. Operation Confirmation Conditions .................................................................................................... 5

3. Reference Application Notes ............................................................................................................. 5

4. Hardware ........................................................................................................................................... 6
   4.1 Hardware Configuration .................................................................................................................. 6
   4.2 Note on Using the RSK Board ..................................................................................................... 6
   4.3 Pins Used ..................................................................................................................................... 6

5. Software ............................................................................................................................................. 7
   5.1 Operation Overview ...................................................................................................................... 7
      5.1.1 Formula for the Temperature Characteristic ........................................................................... 9
   5.2 File Composition .......................................................................................................................... 11
   5.3 Option-Setting Memory ............................................................................................................... 12
   5.4 Constants ................................................................................................................................... 12
   5.5 Variables .................................................................................................................................... 14
   5.6 Functions .................................................................................................................................... 15
   5.7 Function Specifications ................................................................................................................ 16
   5.8 Flowcharts .................................................................................................................................. 21
      5.8.1 Main Processing .................................................................................................................... 21
      5.8.2 Port Initialization .................................................................................................................... 22
      5.8.3 Peripheral Function Initialization ........................................................................................... 22
      5.8.4 CMT Initialization .................................................................................................................. 23
      5.8.5 IRQ Initialization .................................................................................................................... 24
      5.8.6 Processing to Update the 7SEG Display Data ....................................................................... 25
      5.8.7 Processing to Switch the 7SEG Select Output ...................................................................... 25
      5.8.8 Processing to Display a Dash on the 7SEG ........................................................................ 26
      5.8.9 Compare Match Interrupt Handling ....................................................................................... 26
      5.8.10 AD and Temperature Sensor Initialization .......................................................................... 27
      5.8.11 Obtain the Temperature Sensor Measurement Result ....................................................... 28
      5.8.12 Processing to Calculate the Current Temperature ............................................................... 28
      5.8.13 Release the AD and the Temperature Sensor from the Module Stop State......................... 28
      5.8.14 Obtain the A/D Conversion Status ....................................................................................... 29
      5.8.15 Obtain the Current Temperature ............................................................................................ 29
      5.8.16 Processing for Temperature Sensor Calibration ................................................................. 30
      5.8.17 Processing for Temperature Sensor Measurement .............................................................. 30
      5.8.18 A/D Conversion Complete Interrupt Handling .................................................................. 31

6. Appendix (A/D Converted Value and Measured Temperature of the Temperature Sensor) ........... 32

7. Sample Code ....................................................................................................................................... 33

8. Reference Documents ........................................................................................................................ 33
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 RX63N Group and RX631 Group MCUs, 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 Formula for the Temperature Characteristic 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>12-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 (IRQ15) (hereinafter referred to as IRQ)</td>
<td>Switch input (SW3 on the RSK board) for calibrating with the normal reference temperature (25°C).</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></td>
<td><img src="image" alt="7SEG display" /></td>
</tr>
<tr>
<td>Release from the reset state</td>
<td>Dash displayed</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Dash displayed" /></td>
</tr>
<tr>
<td>Waiting for calibration to start</td>
<td>Temperature measurement result displayed as a decimal number</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Temperature measurement result" /></td>
</tr>
<tr>
<td>Switch is pushed</td>
<td>Displayed temperature is updated in 600 ms cycles</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Temperature measurement result" /></td>
</tr>
<tr>
<td></td>
<td>Display when the temperature measurement result is less than 0°C</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Temperature measurement result" /></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>R5F563NBDGFC (RX63N Group)</td>
</tr>
<tr>
<td>Operating frequencies</td>
<td>Main clock: 12 MHz&lt;br&gt; PLL: 192 MHz (main clock divided by 1 and multiplied by 16)&lt;br&gt; System clock (ICLK): 96 MHz (PLL divided by 2)&lt;br&gt; Peripheral module clock B (PCLKB): 48 MHz (PLL divided by 4)</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Integrated development environment</td>
<td>Renesas Electronics Corporation&lt;br&gt; High-performance Embedded Workshop Version 4.09.01</td>
</tr>
<tr>
<td>C compiler</td>
<td>Renesas Electronics Corporation&lt;br&gt; C/C++ Compiler Package for RX Family V.1.02 Release 01</td>
</tr>
<tr>
<td>Compile options</td>
<td>–cpu=rx600 –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>iodefined.h version</td>
<td>Version 1.6A</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>
<tr>
<td>Board used</td>
<td>Renesas Starter Kit+ for RX63N (part number: R0K50563NSxxxBE)</td>
</tr>
</tbody>
</table>

3. Reference Application Notes

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

- RX63N Group, RX631 Group Initial Setting Rev. 1.10 (R01AN1245EJ)
- 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. Note that iodefined.h is partially modified according to the specification of this application note. 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.

```
        Renesas Starter Kit+ for RX63N
        RX63N Group
        RX631 Group
        VCC
        AVCC0
        VREFH0
        P07/IRQ15
        PA0 to PA6
        SW3 input

        VCC
        P20
        P21

        7SEG
        a
        b
        c
d
        e
        f
        g

        7SEG
        a
        b
        c
d
        e
        f
        g

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

Figure 4.1 Connection Example

4.2 Note on Using the RSK Board

VCC and VREFH0 are connected to Renesas Starter Kit+ for RX63N. When disconnecting VCC and VREFH0, remove the option link resistor R5 (0 Ω) on the solder side of the RSK board and connect an option link resistor R6 (0 Ω). VREFH0 can be applied to CON_VREFH0 (header JA1, pin 7). Refer to Renesas Starter Kit+ for RX63N User’s Manual and circuit diagrams for details.

4.3 Pins Used

Table 4.1 lists the Pins Used and Their Functions. The pins used assume that the target product is a 176-pin MCU. When using products with less than 176 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>P07/IRQ15</td>
<td>Input</td>
<td>Switch input for executing calibration</td>
</tr>
<tr>
<td>PA0</td>
<td>Output</td>
<td>Outputs segment a of the 7SEG</td>
</tr>
<tr>
<td>PA1</td>
<td>Output</td>
<td>Outputs segment b of the 7SEG</td>
</tr>
<tr>
<td>PA2</td>
<td>Output</td>
<td>Outputs segment c of the 7SEG</td>
</tr>
<tr>
<td>PA3</td>
<td>Output</td>
<td>Outputs segment d of the 7SEG</td>
</tr>
<tr>
<td>PA4</td>
<td>Output</td>
<td>Outputs segment e of the 7SEG</td>
</tr>
<tr>
<td>PA5</td>
<td>Output</td>
<td>Outputs segment f of the 7SEG</td>
</tr>
<tr>
<td>PA6</td>
<td>Output</td>
<td>Outputs segment g of the 7SEG</td>
</tr>
<tr>
<td>P20</td>
<td>Output</td>
<td>Outputs the first digit of the 7SEG</td>
</tr>
<tr>
<td>P21</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 IRQ15 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 32
- Compare match interrupt cycle: 1 ms

AD
- Operating mode: Single scan mode
- A/D conversion start condition: Software trigger
- Number of sampling states: 240 states (sampling time is 5 µs)
- A/D-converted value addition mode: Not used
- A/D conversion clock: PCLK

Temperature sensor
- Temperature sensor output enable: Enable output from the temperature sensor to the 12-bit A/D converter.
Figure 5.1 shows the Temperature Measurement Timing Diagram.

1. After the MCU is released from the reset state, the CMT and IRQ15 are initialized, and the AD and the temperature sensor are released from the module stop state.
2. After the AD and the temperature sensor are released from the module stop state, the MCU waits 10 ms \( ^{\ast 1} \), 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 (IRQ15), 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 count variable (cnt_cycle) is incremented.
5. When the A/D converter cycle counter variable reaches 100 (100 ms), the AD and the temperature sensor are initialized, and the ADST bit is set to 1 (starts PGA) to start A/D conversion.
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 AD and the temperature sensor are initialized, and the ADST bit is set to 1 to start A/D conversion.
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 10 ms 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 128°C (hereinafter referred to as high reference temperature) stored in the temperature sensor calibration data registers (TSCDR)
- The A/D converted value of the normal reference temperature measured after the MCU is released from the reset state.

Refer to the RX63N Group, RX631 Group User's Manual: Hardware (hereinafter referred to as UMH) for details on the TSCDR register.

Table 5.1 lists the Conditions for Measuring the A/D Converted Values of the Temperature Sensor Output Values Stored in the TSCDR Register.

<table>
<thead>
<tr>
<th>Register Symbol</th>
<th>Conditions for Measuring A/D Converted Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSCDR</td>
<td>Voltage applied to AVCC0 and VREFH0</td>
</tr>
<tr>
<td></td>
<td>Temperature for measurement</td>
</tr>
<tr>
<td></td>
<td>3.3 V</td>
</tr>
<tr>
<td></td>
<td>128°C</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 CAL128 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.

\[
CAL_{128} = \frac{3.3}{VREFH0} \times TSCDR \quad (TSCDR: \text{TSCDR.TSCD}[11:0 \text{ bit value})
\]

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 (128°C): T1
Normal reference temperature (25°C): T2
A/D converted value of the high reference temperature (128°C): CAL128
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 = \((CAL_{128} – CAL_{25}) \div (T1 – T2)\)

Since T1 = 128(°C) and T2 = 25(°C), the slope becomes the following:

Slope = \((CAL_{128} – CAL_{25}) \div (128 – 25) = (CAL_{128} – CAL_{25}) \div 103\)
The formula for calculating the ambient temperature is below.

Measured temperature: $T \, (^{\circ}C)$

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

$$T = T2 + (CALS - CAL_{25}) \times \text{Slope}$$

$$= 25 + (CALS - CAL_{25}) \times ((CAL_{128} - CAL_{25}) + 103)$$

$$= 25 + 103((CALS - CAL_{25}) \times (CAL_{128} - CAL_{25}))$$

When measuring the temperature to the tenths place, temperature data ($T1$, $T2$) is multiplied by 10.

Measured temperature: $Ts \, (^{\circ}C)$

$$Ts = T \times 10 = (25 + 103((CALS - CAL_{25}) \times (CAL_{128} - CAL_{25}))) \times 10$$

$$= (25 \times 10) + (103((CALS - CAL_{25}) \times (CAL_{128} - CAL_{25}))) \times 10$$

$$= 250 + 1030((CALS - CAL_{25}) \times (CAL_{128} - CAL_{25}))$$

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>r_temps.c</td>
<td>Temperature sensor processing</td>
</tr>
<tr>
<td>r_temps.h</td>
<td>Header file for r_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 (RX63N 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 the 176-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>Set the wait time.</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 5.5  Option-Setting Memory Configured in the Sample Code

<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 FF8Bh to FFFF FF88h</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 (r_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>VREF_VOLTAGE</td>
<td>3.3f</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. Do not set 2 or less for this constant.</td>
</tr>
</tbody>
</table>
### Table 5.8  Constants Used in the Sample Code (r_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>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>HIGH_REF_VOLTAGE</td>
<td>3.3f</td>
<td>Voltage for measuring the high reference temperature (128°C)</td>
</tr>
<tr>
<td>HIGH_REF_TEMP</td>
<td>128</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>TSCDR_VALUE</td>
<td>(TEMPSCONST.TSCDR.BIT.TSCD)</td>
<td>TSCDR register value</td>
</tr>
<tr>
<td>HIGH_REF_POTENTIAL_VAL</td>
<td>(uint16_t)(HIGH_REF_VOLTAGE / VREF_VOLTAGE * TSCDR_VALUE)</td>
<td>A/D converted value of the high reference temperature (128°C)</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 (25°C) multiplied by the temperature calculation accuracy</td>
</tr>
</tbody>
</table>
### 5.5 Variables

Table 5.9 and Table 5.10 list the static variables, and Table 5.11 lists the const Variable.

#### Table 5.9 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 disp_comswitch_7seg 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 disp_comswitch_7seg disp_bar_7seg</td>
</tr>
</tbody>
</table>

#### Table 5.10 static Variables (r_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 volatile int16_t</td>
<td>slope_potential</td>
<td>Slope of the A/D converted value</td>
<td>R_TEMPS_Calibration R_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>R_TEMPS_Calibration R_TEMPS_Calc</td>
</tr>
<tr>
<td>static volatile int8_t</td>
<td>ad_status</td>
<td>A/D conversion status</td>
<td>main R_TEMPS_GetADStatus R_TEMPS_Calibration R_TEMPS_Measurement Excep_S12AD_S12ADI0</td>
</tr>
<tr>
<td>static volatile int16_t</td>
<td>now_temp</td>
<td>Calculated current temperature</td>
<td>R_TEMPS_GetNowTemp</td>
</tr>
<tr>
<td>static volatile uint16_t</td>
<td>now_potential</td>
<td>Current A/D converted value</td>
<td>R_TEMPS_Calibration Excep_S12AD_S12ADI0</td>
</tr>
<tr>
<td>static volatile int16_t</td>
<td>buf_ad_value[CNT_CNT_MAX]</td>
<td>A/D converted value buffer</td>
<td>Excep_S12AD_S12ADI0</td>
</tr>
<tr>
<td>static volatile int16_t</td>
<td>ad_max_value</td>
<td>Highest A/D converted value</td>
<td>Excep_S12AD_S12ADI0</td>
</tr>
<tr>
<td>static volatile int16_t</td>
<td>ad_min_value</td>
<td>Lowest A/D conversion value</td>
<td>Excep_S12AD_S12ADI0</td>
</tr>
<tr>
<td>static volatile uint8_t</td>
<td>ad_smp_cnt</td>
<td>Write pointer for the A/D converted value buffer</td>
<td>Excep_S12AD_S12ADI0</td>
</tr>
<tr>
<td>static volatile uint8_t</td>
<td>f_startup_wait</td>
<td>First start time wait determination Flag</td>
<td>r_temps_init</td>
</tr>
</tbody>
</table>

#### Table 5.11 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 5.12 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>r_temps_init</td>
<td>AD and temperature sensor initialization</td>
<td>r_temps.c</td>
</tr>
<tr>
<td>r_temps_getpotential</td>
<td>Obtain the temperature sensor measurement result</td>
<td>r_temps.c</td>
</tr>
<tr>
<td>r_temps_calc</td>
<td>Processing to calculate the current temperature</td>
<td>r_temps.c</td>
</tr>
<tr>
<td>R_TEMPS_ModuleStopCancel</td>
<td>Release the AD and the temperature sensor from the module stop state</td>
<td>r_temps.c</td>
</tr>
<tr>
<td>R_TEMPS_GetADStatus</td>
<td>Obtain the A/D conversion status</td>
<td>r_temps.c</td>
</tr>
<tr>
<td>R_TEMPS_GetNowTemp</td>
<td>Obtain the current temperature</td>
<td>r_temps.c</td>
</tr>
<tr>
<td>R_TEMPS_Calibration</td>
<td>Processing for temperature sensor calibration</td>
<td>r_temps.c</td>
</tr>
<tr>
<td>R_TEMPS_Measurement</td>
<td>Processing for temperature sensor measurement</td>
<td>r_temps.c</td>
</tr>
<tr>
<td>Excep_S12AD_S12ADI0</td>
<td>A/D conversion complete interrupt handling</td>
<td>r_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 performs temperature sensor calibration once. After that, it converts the temperature sensor output from digital to analog 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>
<tr>
<td>irq_init</td>
<td>IRQ initialization</td>
<td>None</td>
<td>static void irq_init(void)</td>
<td>This function initializes IRQ15.</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

R01AN2463EJ0100  Rev. 1.00  Page 16 of 33
Mar. 9, 2015
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
Return value None

Less than 0 (negative value): "Lo" is displayed
100 or higher: "Hi" is displayed
Other than above: Temperature is displayed

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
r_temps_init
Outline  AD and temperature sensor initialization
Header   None
Declaration static void r_temps_init(void)
Description This function initializes the AD and the temperature sensor.
Arguments None
Return value None

r_temps_getpotential
Outline  Obtain the temperature sensor measurement result
Header   None
Declaration static uint16_t r_temps_getpotential (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

r_temps_calc
Outline  Processing to calculate the current temperature
Header   None
Declaration static int16_t r_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)

R_TEMPS_ModuleStopCancel
Outline  Release the AD and the temperature sensor from the module stop state
Header   r_temps.h
Declaration void R_TEMPS_ModuleStopCancel (void)
Description This function releases the AD and the temperature sensor from the module stop state.
Arguments None
Return value None
### R_TEMPS_GetADStatus

**Outline:** Obtain the A/D conversion status  
**Header:** r_temps.h  
**Declaration:**  
```c
uint8_t R_TEMPS_GetADStatus(void)
```

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

**Arguments:** None

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

### R_TEMPS_GetNowTemp

**Outline:** Obtain the current temperature  
**Header:** r_temps.h  
**Declaration:**  
```c
int16_t R_TEMPS_GetNowTemp (void)
```

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

**Arguments:** None

**Return value:** int16_t: Current temperature

### R_TEMPS_Calibration

**Outline:** Processing for temperature sensor calibration  
**Header:** r_temps.h  
**Declaration:**  
```c
void R_TEMPS_Calibration(void)
```

**Description:** This function obtains the A/D converted value of the normal reference temperature and calculates the temperature slope.

**Arguments:** None

**Return value:** None

### R_TEMPS_Measurement

**Outline:** Processing for temperature sensor measurement  
**Header:** r_temps.h  
**Declaration:**  
```c
void R_TEMPS_Measurement(void)
```

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

**Arguments:** None

**Return value:** None
### Excep_S12AD_S12ADI0

<table>
<thead>
<tr>
<th>Outline</th>
<th>A/D conversion complete interrupt handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>static void Excep_S12AD_S12ADI0(void)</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
</tbody>
</table>
5.8 Flowcharts

5.8.1 Main Processing

Figure 5.2 shows the Main Processing.

```
main

Disable maskable interrupts
I flag ← 0

Port initialization
port_init()

Stop processing for active
peripheral functions after a reset
R_INIT_StopModule()

Nonexistent port initialization
R_INIT_NonExistentPort()

Clock initialization
R_INIT_Clock()

Peripheral function initialization
peripheral_init()

Enable maskable interrupts
I flag ← 1

Processing to display a dash on
the 7SEG
disp_bar_7seg()

Processing for temperature
sensor calibration
R_TEMPS_Calibration()

Obtain the A/D conversion status
R_TEMPS_GetADStatus()

A/D conversion complete?

Obtain the current temperature
now_temp ← Current temperature

Processing to update the 7SEG
display data
disp_7seg()
```

Figure 5.2 Main Processing
5.8.2 Port Initialization

Figure 5.3 shows Port Initialization.

```
port_init

Set the 7SEG output ports

PORTA.PODR register ← 7Fh: PA0 to PA6: High
PORT2.PODR register ← 03h: P20 to P21: High
PORTA.PDR register ← 7Fh: PA0 to PA6: Output
PORT2.PDR register ← 03h: P20 to P21: Output

return
```

Figure 5.3 Port Initialization

5.8.3 Peripheral Function Initialization

Figure 5.4 shows Peripheral Function Initialization.

```
main

CMT initialization
  cmt_init()

IRQ initialization
  irq_init()

Release the AD and temperature sensor from the module stop state
  R_TEMPS_ModuleStopCancel()

return
```

Figure 5.4 Peripheral Function Initialization
### 5.8.4 CMT Initialization

Figure 5.5 shows CMT Initialization.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>cmt_init</strong></td>
</tr>
<tr>
<td>2</td>
<td>Enable writing to relevant registers</td>
</tr>
<tr>
<td>3</td>
<td>PRCR register ← A502h</td>
</tr>
<tr>
<td>4</td>
<td>PRC1 bit = 1: Enables writing to the registers related to low power consumption</td>
</tr>
<tr>
<td>5</td>
<td>MSTPCRA register</td>
</tr>
<tr>
<td>6</td>
<td>MSTPA15 bit ← 0: The CMT0 module stop state is canceled</td>
</tr>
<tr>
<td>7</td>
<td>Cancel the CMT module stop</td>
</tr>
<tr>
<td>8</td>
<td>Disable writing to relevant registers</td>
</tr>
<tr>
<td>9</td>
<td>PRCR register ← A500h</td>
</tr>
<tr>
<td>10</td>
<td>PRC1 bit = 0: Disables writing to the registers related to low power consumption</td>
</tr>
<tr>
<td>11</td>
<td>Stop the CMT0 count</td>
</tr>
<tr>
<td>12</td>
<td>CMSTR0 register</td>
</tr>
<tr>
<td>13</td>
<td>STR0 bit ← 0: CMT0.CMCNT count is stopped</td>
</tr>
<tr>
<td>14</td>
<td>Disable writing to relevant registers</td>
</tr>
<tr>
<td>15</td>
<td>PRCR register ← A500h</td>
</tr>
<tr>
<td>16</td>
<td>PRC1 bit = 0: Disables writing to the registers related to low power consumption</td>
</tr>
<tr>
<td>17</td>
<td>Select the count source</td>
</tr>
<tr>
<td>18</td>
<td>CMT0.CMCR register ← 00C1h</td>
</tr>
<tr>
<td>19</td>
<td>CMIE bit = 1: Compare match interrupt enabled</td>
</tr>
<tr>
<td>20</td>
<td>CKS[1:0] bits = 01b: PCLK/32</td>
</tr>
<tr>
<td>21</td>
<td>Stop the CMT0 count</td>
</tr>
<tr>
<td>22</td>
<td>CMSTR0 register</td>
</tr>
<tr>
<td>23</td>
<td>STR0 bit ← 0: CMT0.CMCNT count is stopped</td>
</tr>
<tr>
<td>24</td>
<td>Select the count source</td>
</tr>
<tr>
<td>25</td>
<td>CMT0.CMCR register ← 00C1h</td>
</tr>
<tr>
<td>26</td>
<td>CMIE bit = 1: Compare match interrupt enabled</td>
</tr>
<tr>
<td>27</td>
<td>CKS[1:0] bits = 01b: PCLK/32</td>
</tr>
<tr>
<td>28</td>
<td>Set the CMT0 compare match constant register</td>
</tr>
<tr>
<td>29</td>
<td>CMT0.CMCOR register ← 1500 - 1: CMT cycle = 1 ms [(1500 - 1) × (PCLKB/32)]</td>
</tr>
<tr>
<td>30</td>
<td>Clear the CMT0 counter</td>
</tr>
<tr>
<td>31</td>
<td>CMT0.CMCNT register ← 0000h</td>
</tr>
<tr>
<td>32</td>
<td>Clear the interrupt request</td>
</tr>
<tr>
<td>33</td>
<td>IR28 register</td>
</tr>
<tr>
<td>34</td>
<td>IR bit ← 0: CMI0 interrupt request not generated</td>
</tr>
<tr>
<td>35</td>
<td>Set the interrupts</td>
</tr>
<tr>
<td>36</td>
<td>IPR004 register</td>
</tr>
<tr>
<td>37</td>
<td>IPR[3:0] bits ← 0001b: CMI0 interrupt priority level is level 1</td>
</tr>
<tr>
<td>38</td>
<td>IER03 register</td>
</tr>
<tr>
<td>39</td>
<td>IEN4 bit ← 1: CMI0 interrupt request is enabled</td>
</tr>
<tr>
<td>40</td>
<td>return</td>
</tr>
</tbody>
</table>
5.8.5 IRQ Initialization

Figure 5.6 shows IRQ Initialization.

```c
void irq_init()
{
  // Disable interrupts
  IER09 register
  IEN7 bit ← 0: IRQ15 interrupt request is disabled

  // Disable the digital filter
  IROFLTE1 register
  FLTEN15 bit ← 0: Digital filter is disabled.

  // Set the digital filter sampling clock
  IROFLT1 register
  FCLKSEL15[1:0] bits ← 11b: PCLK/64

  // Disable 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 P07PFS register
  P07PFS register
  ISEL bit ← 1: Used as IRQ15 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
  IRQCR15 register
  IRQMD[1:0] bits ← 01b: Falling edge

  // Clear the interrupt request
  IR079 register
  IR flag ← 0: IRQ15 interrupt request is not generated

  // Enable the digital filter
  IROFLTE1 register
  FLTEN15 bit ← 1: Digital filter is enabled.

  // Return
  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.

```
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.

```
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.

```
disp_bar_7seg
Set the index to display a dash "—" in the first digit and second digit
digit_10 ← SEG_TBL_DASH
digit_1 ← SEG_TBL_DASH

Turn off the 7SEG
PORT2.PODR register ← 03h

Output data to the 7SEG
PORTA.PODR register ← "—" data

Turn on the 7SEG
PORT2.PODR register ← 00h
```

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.

```
Excep_CMT0_CMI0
Has the CMT_CYCLE_MS time elapsed? No
  Yes
    Clear the A/D conversion cycle counter
    cnt_cycle ← 0
    Processing for temperature sensor measurement
    R_TEMPS_Measurement()

Has the SEG_CYCLE_MS time elapsed? No
  Yes
    Clear the 7SEG select output switch cycle counter
    cnt_led_cycle ← 0
    Processing to switch the 7SEG select output
    disp_comswitch_7seg()

    Increment the A/D conversion cycle counter
cnt_cycle++

    Increment the 7SEG select output switch cycle counter
cnt_led_cycle++

    return
```

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.

![Flowchart Diagram]

---

**Figure 5.11 AD and Temperature Sensor Initialization**
5.8.11 Obtain the Temperature Sensor Measurement Result

Figure 5.12 shows Obtain the Temperature Sensor Measurement Result.

```plaintext
r_temps_getpotential

Is A/D conversion stopped?

Yes

Obtain A/D converted value

return(w_now_potential)

No

Read the ADCSR register

ADST bit 0: Stops A/D conversion process
1: Starts A/D conversion process

w_now_potential ← ADTSDR register value
```

Figure 5.12 Obtain the Temperature Sensor Measurement Result

5.8.12 Processing to Calculate the Current Temperature

Figure 5.13 shows Calculate the Current Temperature.

```plaintext
r_temps_calc

Argument

uint16_t w_now_potential: A/D converted value

Calculate the current temperature *1

w_now_temp ← 250 + 1030 * ((w_now_potential - CAL25) / (CAL128 - CAL25))

return(w_now_potential)

Note 1. Refer to 5.1.1 Formula for the Temperature Characteristic for details on calculation.
```

Figure 5.13 Calculate the Current Temperature

5.8.13 Release the AD and the Temperature Sensor from the Module Stop State

Figure 5.14 shows Release the AD and the Temperature Sensor from the Module Stop State.

```plaintext
R_TEMPS_ModuleStopCancel

Enable writing to relevant registers

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

MSTPCRA register
MSTPA17 bit ← 0 : The AD module stop state is canceled

MSTPCRB register
MSTPB8 bit ← 0 : The temperature sensor module stop state is canceled

Disable writing to relevant registers

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

Wait 10 ms
R_DELAY_Us()

After releasing the modules from the module stop state, wait 10 ms before starting A/D conversion.

return
```

Figure 5.14 Release the AD and the Temperature Sensor from the Module Stop State
5.8.14 Obtain the A/D Conversion Status
Figure 5.15 shows Obtain the A/D Conversion Status.

```c
R_TEMPS_GetADStatus
GetADStatusreturn (ad_status)
```

Figure 5.15 Obtain the A/D Conversion Status

5.8.15 Obtain the Current Temperature
Figure 5.16 shows Obtain the Current Temperature.

```c
R_TEMPS_GetNowTemp
return (now_temp)
```

Figure 5.16 Obtain the Current Temperature
5.8.16 Processing for Temperature Sensor Calibration

Figure 5.17 shows the Processing for Temperature Sensor Calibration.

```
R_TEMPS_Calibration
Wait for IRQ15 pin input
Read the IR079.IR flag (wait until the flag becomes 1)
IR079 register
IR flag ← 0: IRQ15 interrupt request not generated
Clear the IRQ15 interrupt request
CMSTR0 register
STR0 bit ← 1: CMT0.CMCNT count is started
Start the CMT0 count
Wait for A/D conversion to be completed
Set the A/D conversion status to "Not performed"
ad_status ← STA_AD_IDLE
Save the A/D converted value of the normal reference temperature (25°C)
ordinary_potential ← now_potential
Save the A/D converted value slope to the RAM
slope_potential ← HIGH_REF_POTENTIAL_VAL - ordinary_potential
```

Figure 5.17 Processing for Temperature Sensor Calibration

5.8.17 Processing for Temperature Sensor Measurement

Figure 5.18 shows the Processing for Temperature Sensor Measurement.

```
R_TEMPS_Measurement
AD and temperature sensor initialization
r_temps_init()
Start measuring the temperature
ADCSR register
ADST bit ← 1: Starts A/D conversion
Change the A/D conversion status to "Waiting for A/D conversion to be completed"
ad_status ← STA_AD_WAIT
```

Figure 5.18 Processing for Temperature Sensor Measurement
5.8.18 A/D Conversion Complete Interrupt Handling

Figure 5.1 shows the A/D Conversion Complete Interrupt Handling.

Excep_S12AD_S12ADI0

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
r_temps_getpotential()

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

Yes

Increment the number of A/D converted values obtained
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
r_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 5.19 A/D Conversion Complete Interrupt Handling
6. Appendix  
(A/D Converted Value and Measured Temperature of the Temperature Sensor)  
The ambient temperature of the MCU rises depending on the operating frequency or operation of the peripheral functions, and it takes several tens of seconds to stabilize the temperature after a reset is released. The time necessary to stabilize the temperature also varies depending on the operating environment. Accordingly, the A/D converted value rises for a while after a reset is released as shown in Figure 6.1.

When obtaining the A/D converted value of the temperature sensor at the normal reference temperature, measure the MCU surface temperature, and perform the A/D conversion of the temperature sensor after the MCU temperature becomes stable.

![Figure 6.1 Relation Between the Temperature / A/D Converted Value and Operating Time](image-url)
7. Sample Code
Sample code can be downloaded from the Renesas Electronics website.

8. Reference Documents
User’s Manual: Hardware
RX63N Group, RX631 Group User’s Manual: Hardware Rev.1.80 (R01UH0041EJ)
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>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Mar. 9, 2015</td>
<td>First edition issued</td>
<td></td>
<td></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.
SALES OFFICES

Renesas Electronics Corporation

Renesas Electronics America Inc.
2501 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K.
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiasseestraße 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7879

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 355 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-8888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1051-1061, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852-2886-0022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-8600, Fax: +886-2-8175-8670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit 19-01/9-02 PEP Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Penyiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Slage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F, 234 Teheran-ro, Gangnam-gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

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 these circuits, 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 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 Renesas Electronics or 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 product’s quality grade, as indicated below.

   - “Standard”: Computers, office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliance; machine tools; personal electronic equipment; and industrial robots etc.

   - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

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 not 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 microcomputer 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 products or technology described in this document for any purpose relating to military applications or use by the military, 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 products” means any product developed or manufactured by or for Renesas Electronics.

Colophon 5.0

© 2015 Renesas Electronics Corporation. All rights reserved.