Introduction

With the application described in this application note, a voltmeter is implemented by using the A/D converter incorporated in the RL78/G11.

Target Device

RL78/G11

When applying this application note to another microcontroller, careful evaluation is recommended after making modifications to comply with the microcontroller used.
Contents

1. Specifications .................................................................................................................. 4
   1.1 Voltage Measurement Method .................................................................................. 4
   1.2 Automatic Measurement Range Switching .............................................................. 5
   1.3 Determining Resistances and Measurement Ranges .................................................... 6
   1.4 Operation Processing of Measurement Results ......................................................... 7
   1.5 Correction based on Internal Reference Voltage ....................................................... 8
   1.6 Measurement Range Switching ................................................................................. 8
   1.7 AC Source Support ................................................................................................. 9
   1.8 Display Period ........................................................................................................... 9

2. Operation Confirmation Conditions .................................................................................. 10

3. Related Application Notes .............................................................................................. 10

4. Hardware Descriptions .................................................................................................. 11
   4.1 Hardware Configuration Example ............................................................................. 11
   4.2 List of Pins Used ....................................................................................................... 12

5. Software Descriptions .................................................................................................... 12
   5.1 Operation Summary ............................................................................................... 12
   5.2 List of Option Byte Settings ..................................................................................... 13
   5.3 List of Constants ...................................................................................................... 14
   5.4 List of Variables ...................................................................................................... 15
   5.5 List of Functions ...................................................................................................... 16
   5.6 Function Specifications ............................................................................................ 17
   5.7 Flowchart ................................................................................................................ 23
   5.7.1 Initial Setting Function ......................................................................................... 23
   5.7.2 System Function .................................................................................................. 24
   5.7.3 Setting I/O Ports .................................................................................................. 25
   5.7.4 Setting CPU Clocks ............................................................................................. 26
   5.7.5 Setting Timer Array Unit ..................................................................................... 27
   5.7.6 Setting 8-Bit Interval Timer ................................................................................ 28
   5.7.7 Setting I2C .......................................................................................................... 29
   5.7.8 Setting A/D Converter ......................................................................................... 30
   5.7.9 Setting DTC ........................................................................................................ 31
   5.7.10 Setting INTPT0 .................................................................................................. 31
   5.7.11 Main Processing ................................................................................................ 32
   5.7.12 R_MAIN_UserInit Processing ............................................................................ 35
   5.7.13 Initializing LCD ................................................................................................. 36
   5.7.14 Setting LCD Commands ..................................................................................... 37
   5.7.15 Wait Time Processing ........................................................................................ 38
   5.7.16 60-us Wait Processing ....................................................................................... 38
   5.7.17 Displaying Character Strings on LCD ................................................................. 39
   5.7.18 Setting Display Data .......................................................................................... 40
   5.7.19 Starting I2C Transmission ............................................................................... 40
   5.7.20 Waiting for Completion of I2C Communication ................................................ 41
   5.7.21 Issuing Stop Condition ....................................................................................... 41
   5.7.22 Checking I2C Bus Status ................................................................................... 42
   5.7.23 INTPT0 Interrupt Processing ............................................................................. 43
   5.7.24 INTPT0 Interrupt Processing ............................................................................. 43
   5.7.25 Checking Internal Reference Voltage ............................................................... 44
   5.7.26 1-us Wait Processing ......................................................................................... 45
   5.7.27 Setting ADC and DTC ....................................................................................... 46
   5.7.28 Starting TM01 .................................................................................................. 46
   5.7.29 A/D Conversion End Interrupt Processing ......................................................... 47
   5.7.30 I2C Communication End Interrupt Processing .................................................. 49
5.7.31 16-Bit Multiplication ..................................................................................................................... 51
5.7.32 32-Bit Division .............................................................................................................................. 51
5.7.33 16-Bit Division .............................................................................................................................. 51
5.7.34 16-Bit Modulo Operation .............................................................................................................. 52
5.7.35 Root-Mean-Square Calculation for 16-Bit Data ........................................................................... 53

6. Sample Code ................................................................................................................................... 54

7. Documents for Reference ............................................................................................................... 54
1. Specifications

1.1 Voltage Measurement Method

With this application, the analog input voltage is measured by the A/D converter incorporated in the RL78/G11. Three ranges of voltages are measured: 0 to ±1 V (±1 V range), 0 to ±10 V (±10 V range), and 0 to ±100 V (±100 V range).

First, the analog voltage input circuit (Figure 1.1) is considered. The ANI0 pin is connected to GND via resistor R0. A switch is inserted between the ANI0 pin and the voltage to be measured, and either 0 Ω, R1, or R2 is used to connect the pin and the voltage.

When 0 Ω is used, the voltage is measured in the ±1 V range. When resistor R1, whose resistance is nine times the R0 resistance, is used, the voltage is measured in the ±10 V range, and when resistor R2, whose resistance is 99 times the R0 resistance, is used, the voltage is measured in the ±100 V range.

By using a resistor divider, the voltage applied to the ANI0 pin is controlled to 1 V or smaller. Note that it is not allowed to apply voltage equal to or larger than VDD to any pin of the RL78/G11.

Figure 1.1 Analog Voltage Input Circuit

Next, the case is considered in which a negative voltage as well as a positive voltage is accepted as the voltage to be measured. As shown in Figure 1.2, two ports are added; the ports are connected to each other via two resistors; and the midpoint is connected both to the analog input pin and the minus (−) side voltage to be measured. By setting these ports to the high level output and low level output, the minus (−) side voltage to be measured is controlled to the intermediate potential (approximately 1/2 VDD based on the RL78/G11). With these arrangements, a voltage applied to the ANI0 pin is always positive. By measuring the plus (+) side and minus (−) side voltages and calculating the difference between the two voltages, the voltage of the target can be obtained.

![Figure 1.1 Analog Voltage Input Circuit](image-url)
1.2 Automatic Measurement Range Switching

In manual range switching, the resistors between the analog input pin and voltage to be measured are switched over, whereas in automatic range switching, the resistors that are connected to the GND side in manual range switching are switched over.

The R1 resistance is 1/9 the R0 resistance and the R2 resistance is 1/99 the R0 resistance. When port 1 is set to the high-impedance state and port 2 to low level output, R2 is selected and the voltage applied to the ANI0 pin is 1/100 the voltage to be measured. Measure the voltage applied to the ANI0 pin in this condition. If the measurement result falls between VDD/10 and VDD/100, R1 is selected; if the measurement result falls between VDD/100 and VDD/1000, neither R1 nor R2 is selected (both port 1 and port 2 are set to the high-impedance state). Measuring again the voltage applied to the ANI0 pin enables more accurate measurement of the voltage to be measured.

Figure 1.3 Automatic Measurement Range Switching
### 1.3 Determining Resistances and Measurement Ranges

The GND level for the resistor divider is generated for each range. The GND level is set to the intermediate potential (approximately 1/2 VDD based on the RL78/G11) by dividing the high-level output and low-level output of the ports by the resistors.

The GND level for the resistor divider is actually measured and the actually measured value is used for calculating the voltage to be measured.

The maximum voltage measurement range is ±100 V (the peak value of the alternating current is approximately ±150 V). The input impedance is approximately 1 MΩ, and R0, R1, and R2 are 910 kΩ, 82 kΩ, and 8.2 kΩ, respectively.

![Measurement Circuit Diagram](image)

Figure 1.4 Measurement Circuit

---

All the protection diodes are schottky barrier diodes (SBD).
1.4 Operation Processing of Measurement Results

Although using the floating-point operation for operation processing of voltage measurement enables easy programming, it takes longer operation time and also increases the program size because it uses the floating-point library. In this application, therefore, the floating-point operation is not used but the number of data bits is considered instead. The multiply, divide, and multiply-accumulate instructions incorporated in the RL78/G11 are used for multiplication and division according to the length of the bits to be operated.

When voltage-dividing resistor Rn (R1 or R2) is active, the following relationship holds between voltage V0 and the actually measured voltage (V1).

\[
V1 = \frac{(V0 - Gn) \times Rn}{R0 + Rn} + Gn \\
= \frac{V0 \times Rn + Gn \times R0}{R0 + Rn}
\]

When this equation is rearranged,

\[
V0 = \frac{V1 \times (R0 + Rn) - Gn \times R0}{Rn}
\]

Therefore, voltage VIN to be measured is calculated as follows since it can be obtained by subtracting G0 from V0.

\[
VIN = V0 - G0 \\
= \frac{V1 \times (R0 + Rn) - Gn \times R0}{Rn} - G0 \\
= \frac{V1 \times (R0 + Rn) - (Gn \times R0 + G0 \times Rn)}{Rn}
\]

By comparing the first and second terms in the curly braces in the above equation, the voltage to be measured can be determined whether it is positive or negative. If the first term is larger, the voltage is positive, and if smaller, the voltage is negative.
1.5 Correction based on Internal Reference Voltage

VDD is used as the reference voltage of the A/D converter. When three alkaline cells are used for a VDD power source, the output voltage from the battery fluctuates between approximately 4.8 V (1.6 V × 3) to approximately 3 V (1.0 V × 3). This means that the fluctuation rate of the reference voltage of the A/D converter is more than 30%, resulting in poor A/D conversion accuracy. To avoid this, the measurement result is corrected by A/D-converting the internal reference voltage incorporated in the RL78/G11 (TYP. 1.45 V). However, the internal reference voltage varies by 5%.

When the internal reference voltage is A/D-converted, the result (SAR) is expressed as follows.

\[ \text{SAR} = \frac{1.45}{\text{VDD}} \times 1024 \]

When this equation is rearranged,

\[ \text{VDD} = \frac{1.45}{\text{SAR}} \times 1024 \]

From a slightly different point of view, the voltage of a single bit can be expressed as 1.45/SAR.

Therefore, the measurement result voltage is expressed as follows in this application note.

\[ \text{VIN} = \left\lfloor \frac{\text{V1} \times (\text{R0} + \text{Rn}) - (\text{Gn} \times \text{R0} + \text{G0} \times \text{Rn})}{\text{Rn}} \right\rfloor \times \frac{1.45}{\text{SAR}} \]

Since the above equation is solved by integer operation not by floating-point operation, the following corrections are made.

1. Resistance is calculated in 0.1 kΩ units.
2. 1.45 is assumed as “145/100”; and the result is not divided by 100 but obtained in 10 mV units.

Although the voltage to be measured is calculated in 10 mV units, it can be handled with 16 bits as long as the voltage to be measured is alternating 100 V.

Note that, in this application, division is performed to enable 32-bit operation on the voltage to be measured (effective value). Since division is performed using a shift instruction, divisors should be 16 and 4 for the ±100 V range and ±10 V range, respectively. This controls the number of bits of the calculation result (g_vin_data[32]) of the voltage to be measured to 12 bits or fewer, which enables 32-bit operation to handle addition (calculation of the effective value) of the 32 squares of the calculation result (g_vin_data[32]).

1.6 Measurement Range Switching

Be sure to start measuring the voltage to be measured when it is connected to the intermediate potential (G2) via R2 (8.2 kΩ) (±100 V range). Also be sure to measure the voltage to be measured 32 times with the above settings. Determine the appropriate measurement range according to the maximum absolute value of the measurement result.

If the calculation result (g_vin_data[32]) of the voltage to be measured is 9 or smaller (equivalent to ±1.5 V) when R2 is selected, select the ±1 V range. If the calculation result is 10 to 93 (equivalent to ±15 V), select the ±10 V range. If the calculation result (g_vin_data[32]) exceeds 937 (equivalent to ±150 V), over range is indicated. After measurement is completed, select R2 (8.2 kΩ) to assure safety.

The absolute value of the voltage to be measured can be obtained by multiplying the calculation result (g_vin_data[32]) by the constant specific to the measurement range (16 for the ±100 V range and 4 for the ±10 V range) and changing the 10 mV units to 1 V units.
1.7 AC Source Support

This application supports 50- to 60-Hz sine waves.

A sine wave is sampled 32 times in a 1-ms period to measure the interval between the points (zero crosses) at which the polarity is reversed. In calculation of the effective value, only the voltage to be measured during the interval is included. If there are no zero crosses, the last 20 samples are used to calculate the effective value.

Note that this application does not support measurement of the pulsating current. To measure the pulsating current, the increased number of samples are required to calculate the effective values.

1.8 Display Period

The display switching period is 1 second. After the internal reference voltage is measured, the voltage to be measured is calculated each time an interrupt of the 1-ms period interval timer occurs to determine the voltage to be measured (effective value). Then, the 1-ms period interval timer is stopped; the voltage to be measured (effective value) is displayed; and the standby state (STOP mode) is set. After this, the above operations are repeated every second.

If the voltage to be measured (effective value) is equal to or smaller than ±10 mV for 20 consecutive times, the display is turned off; the 1-s period interval timer is stopped; and the standby state (STOP mode) is set. When the switch is pressed (INTP0 occurs), the voltage measurement and display are started again.
2. Operation Confirmation Conditions

The sample code covered in 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>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller used</td>
<td>RL78/G11 (R5F1056)</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>• High-speed on-chip oscillator (HOCO) clock: 16 MHz</td>
</tr>
<tr>
<td></td>
<td>• CPU/peripheral hardware clock: 16 MHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>3.0 V to 5.5 V</td>
</tr>
<tr>
<td></td>
<td>LVD operation mode: reset mode; voltage: VLVD2 = Min. 3.00 V</td>
</tr>
<tr>
<td>Integrated development environment(CS+)</td>
<td>CS+ V5.00.00 from Renesas Electronics</td>
</tr>
<tr>
<td>Integrated development environment(e2studio)</td>
<td>e2 studio 5.3.0.023 from Renesas Electronics</td>
</tr>
<tr>
<td>Compiler</td>
<td>CC-RL V1.04.00 from Renesas Electronics</td>
</tr>
<tr>
<td>Board used</td>
<td>RL78/G11 target board + LCD</td>
</tr>
</tbody>
</table>

3. Related Application Notes

Refer to the following related application notes for additional information.

RL78/G13 Initialization CC-RL(R01AN2575E) Application Note
RL78/G10 Square Root Program CC-RL (R01AN3079E) Application Note
4. Hardware Descriptions

4.1 Hardware Configuration Example

Figure 4.1 shows the hardware configuration example covered in this application note.

Notes: 1. The above figure is a simplified circuit image for showing the outline of the connections. The actual circuit should be designed so that the pins are handled appropriately and that the electrical characteristics are satisfied (input-only ports should be each connected to \( V_{DD} \) or \( V_{SS} \) via a resistor).

2. \( V_{DD} \) must be equal to or greater than the reset release voltage \( (V_{LVD}) \) specified with LVD.

All the protection diodes are schottky barrier diodes (SBD).

Figure 4.1 Hardware Configuration
4.2 List of Pins Used

Table 4.1 lists the pins used and their functions.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANI0 to ANI3</td>
<td>Input</td>
<td>Analog signal input</td>
</tr>
<tr>
<td>P00, P01, P30, P31, P33, P54</td>
<td>Output</td>
<td>Intermediate voltage generation for each measurement range</td>
</tr>
<tr>
<td>P55/SDAA1, P56/SCLA1</td>
<td>I/O</td>
<td>I2C bus control for transferring display data to LCD</td>
</tr>
<tr>
<td>INTP0/P137</td>
<td>Input</td>
<td>SW input</td>
</tr>
</tbody>
</table>

5. Software Descriptions

5.1 Operation Summary

With this application, a voltmeter is implemented by using the A/D converter incorporated in the RL78/G11.

(1) After the initial settings, “Renesas RL78/G11” is displayed on the LCD and the standby state (STOP mode) is set.

(2) When the switch is pressed, INTP0 is generated and the standby state is released.

(3) The 1-s timer is activated.

(4) The internal reference voltage is measured. The voltage to be measured is measured each time an interrupt of the 1-ms period interval timer occurs. The analog input voltages of ANI0 to ANI3 are measured in this order. The A/D conversion results are saved in the internal RAM by using the DTC.

(5) The voltage to be measured is sampled 32 times in a 1-ms period to measure the maximum absolute value of the voltage to be measured and the interval between the points (zero crosses) at which the polarity of the voltage to be measured is reversed.

(6) The maximum absolute value of the voltage to be measured is judged, the measurement ranges are switched when necessary, and the voltage is measured every 1 ms 32 times again. Also, the interval is measured between the points (zero crosses) at which the polarity is reversed.

(7) After 32nd voltage measurement, the voltage to be measured (effective value) between the points at which the polarity is reversed is calculated. When there are no zero crosses, the last 20 samples are used to calculate the effective value.

(8) The 1-ms period interval timer is stopped; the voltage to be measured (effective value) that has been calculated is displayed on the LCD; and the standby state (STOP mode) is set. After this, (4) to (8) above are repeated every second.

If the voltage to be measured (effective value) is equal to or smaller than ±10 mV for 20 consecutive times, the display is turned off; the 1-s period interval timer is stopped; and the standby state (STOP mode) is set. When the switch is pressed (INTP0 occurs), the operation is restarted at (2).
5.2 List of Option Byte Settings

Table 5.1 shows an example of the option byte settings.

<table>
<thead>
<tr>
<th>Address</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000C0</td>
<td>0b11101110</td>
<td>Disables the watchdog timer. (Stops counting after the release from the reset state.)</td>
</tr>
<tr>
<td>0x000C1</td>
<td>0b01111111</td>
<td>LVD reset mode: 2.45 V (2.40 V to 2.50 V)</td>
</tr>
<tr>
<td>0x000C2</td>
<td>0b11101001</td>
<td>HS mode; HOCO: 16 MHz</td>
</tr>
<tr>
<td>0x000C3</td>
<td>0b10000100</td>
<td>Enables the on-chip debugger.</td>
</tr>
</tbody>
</table>
5.3 List of Constants

Table 5.2 lists the constants used in the sample code.

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUTH</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>FALSE</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>COMBYTE</td>
<td>0x00</td>
<td>Command write mode to LCD</td>
</tr>
<tr>
<td>DATABYTE</td>
<td>0x80</td>
<td>Data write mode to LCD</td>
</tr>
<tr>
<td>CLRDISP</td>
<td>0x01</td>
<td>LCD clear instruction</td>
</tr>
<tr>
<td>HOMEPOSI</td>
<td>0x02</td>
<td>LCD home position instruction</td>
</tr>
<tr>
<td>LCD_Mode</td>
<td>0b00111000</td>
<td>LCD display mode (2-line display)</td>
</tr>
<tr>
<td>DISPON</td>
<td>0b00001111</td>
<td>LCD on</td>
</tr>
<tr>
<td>ENTRY_Mode</td>
<td>0b00000110</td>
<td>Right-shift of LCD display data</td>
</tr>
<tr>
<td>SLAVEADDR</td>
<td>0xA0</td>
<td>LCD slave address on I2C bus</td>
</tr>
<tr>
<td>R0_OPHM</td>
<td>10000</td>
<td>R0 resistance (1 MΩ)</td>
</tr>
<tr>
<td>R1_OPHM</td>
<td>820</td>
<td>R1 resistance (82 kΩ)</td>
</tr>
<tr>
<td>R2_OPHM</td>
<td>82</td>
<td>R2 resistance (8.2 kΩ)</td>
</tr>
<tr>
<td>R0_R2</td>
<td>R0_OPHM + R2_OPHM</td>
<td>Combined resistance of R0 and R2</td>
</tr>
<tr>
<td>R0_R1</td>
<td>R0_OPHM + R1_OPHM</td>
<td>Combined resistance of R0 and R1</td>
</tr>
<tr>
<td>LIMIT100</td>
<td>937</td>
<td>Maximum value for 100-V range</td>
</tr>
<tr>
<td>LIMIT10</td>
<td>93</td>
<td>Maximum value for 10-V range</td>
</tr>
<tr>
<td>LIMIT1</td>
<td>8</td>
<td>Maximum value for 1-V range</td>
</tr>
<tr>
<td>SLEEPTIME</td>
<td>20</td>
<td>Number of no-inputs before stop</td>
</tr>
</tbody>
</table>
5.4 List of Variables

Table 5.3 lists the variables used in the sample code.

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint16_t</td>
<td>ad_buff[4]</td>
<td>A/D conversion result buffer</td>
<td>R_ADC_Init(), r_adc_interrupt()</td>
</tr>
<tr>
<td>uint16_t</td>
<td>g_vin_data[32]</td>
<td>Measurement result buffer</td>
<td>R_ADC_Init(), r_adc_interrupt()</td>
</tr>
<tr>
<td>uint16_t</td>
<td>g_vref</td>
<td>Internal reference voltage measurement result</td>
<td>R_ADC_Init(), r_adc_interrupt(), main()</td>
</tr>
<tr>
<td>uint16_t</td>
<td>g_vmax</td>
<td>Maximum voltage measured</td>
<td>R_ADC_Init (), r_adc_interrupt(), main()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_range</td>
<td>Measurement range</td>
<td>main(), r_adc_interrupt(), R_ADC_inter_ref()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_times</td>
<td>Number of measurements</td>
<td>main(), R_MAIN_UserInit(), r_adc_interrupt(), R_ADC_inter_ref()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_vin_sign</td>
<td>Sign of measurement result</td>
<td>main(), r_adc_interrupt(), R_ADC_inter_ref()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_sign_chng1</td>
<td>Zero cross point 1</td>
<td>main(), r_adc_interrupt(), R_ADC_inter_ref()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_sign_chng2</td>
<td>Zero cross point 2</td>
<td>main(), r_adc_interrupt(), R_ADC_inter_ref()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_sign_chng3</td>
<td>Zero cross point 3</td>
<td>main(), r_adc_interrupt(), R_ADC_inter_ref()</td>
</tr>
<tr>
<td>uint16_t</td>
<td>g_voltage</td>
<td>Measurement result (10-mV units)</td>
<td>main()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_period</td>
<td>Number of valid measured data</td>
<td>main()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_start</td>
<td>Start position of valid data</td>
<td>main()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_stpcnt</td>
<td>Voltage not-detect counter</td>
<td>main(), R_MAIN_UserInit()</td>
</tr>
<tr>
<td>int8_t</td>
<td>g_2ndline</td>
<td>Display data buffer</td>
<td>main()</td>
</tr>
<tr>
<td>st_dtc_data_t</td>
<td>dtc_controldata_0</td>
<td>DTC parameter</td>
<td>R_DTC_Create(), R_ADC_DTC_Init(), r_adc_interrupt(), main()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_iica_status</td>
<td>IICA1 status</td>
<td>R_IICA_bus_check, r_iica_interrupt(), R_IICA_wait_cmdend</td>
</tr>
<tr>
<td>uint8_t*</td>
<td>gp_iica_tx_address</td>
<td>IICA1 transmission pointer</td>
<td>R_IICA_Master_Send(), r_iica_interrupt()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_iica_tx_cnt</td>
<td>IICA1 transmission counter</td>
<td>R_IICA_Master_Send(), r_iica_interrupt()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_write_command[2]</td>
<td>LCD command buffer</td>
<td>set_command()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_write_data[2]</td>
<td>LCD data buffer</td>
<td>set_data()</td>
</tr>
</tbody>
</table>
5.5 List of Functions

Table 5.4 lists the functions used in this application.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_MAIN_UserInit()</td>
<td>Activates the various features etc.</td>
</tr>
<tr>
<td>InitialiseDisplay</td>
<td>Initializes LCD display.</td>
</tr>
<tr>
<td>wait_time</td>
<td>Waits for time in 60us units.</td>
</tr>
<tr>
<td>wait_60us</td>
<td>Waits for 60 us.</td>
</tr>
<tr>
<td>set_command</td>
<td>Sets the LCD command.</td>
</tr>
<tr>
<td>set_data</td>
<td>Sets LCD display data.</td>
</tr>
<tr>
<td>DisplayString</td>
<td>Displays ASCII character strings on LCD.</td>
</tr>
<tr>
<td>R_IICA_Master_Send</td>
<td>Transmits data to LCD with IICA.</td>
</tr>
<tr>
<td>R_IICA_wait_command</td>
<td>Waits for completion of IICA transfer.</td>
</tr>
<tr>
<td>R_IICA_StopCondition</td>
<td>Issues the stop condition from IICA.</td>
</tr>
<tr>
<td>R_IICA_bus_check</td>
<td>Checks the IIC bus state and issues the start condition.</td>
</tr>
<tr>
<td>r_iica_interrupt</td>
<td>Processes the IICA interrupt.</td>
</tr>
<tr>
<td>r_intp0_interrupt</td>
<td>Processes the INTP0 interrupt.</td>
</tr>
<tr>
<td>r_it_interrupt</td>
<td>Processes the INTIT00 interrupt.</td>
</tr>
<tr>
<td>R_ADC_inter_ref</td>
<td>Measures the internal reference voltage.</td>
</tr>
<tr>
<td>R_ADC_DTC_Init</td>
<td>Sets A/D and DTC.</td>
</tr>
<tr>
<td>r_tm01_start</td>
<td>Activates TM01 (1-ms interval timer).</td>
</tr>
<tr>
<td>delay_us</td>
<td>Waits for 1 us.</td>
</tr>
<tr>
<td>r_adc_interrupt</td>
<td>Completes DTC transfer of A/D conversion results.</td>
</tr>
<tr>
<td>__ssqrt</td>
<td>Square-root operation of 32-bit data (For details, refer to R01AN3079.)</td>
</tr>
<tr>
<td>__r_mul32</td>
<td>16 bits × 16 bits</td>
</tr>
<tr>
<td>__r_div32</td>
<td>32 bits/16 bits</td>
</tr>
<tr>
<td>__r_div16</td>
<td>16 bits/8 bits</td>
</tr>
<tr>
<td>__r_mod16</td>
<td>Modulo operation of 16 bits/8 bits</td>
</tr>
<tr>
<td>__r_rms16</td>
<td>Root-mean-square operation of 16-bit data</td>
</tr>
</tbody>
</table>
5.6 Function Specifications

The following gives the specifications of the functions used in the sample code.

[Function Name] R_MAIN_UserInit

<table>
<thead>
<tr>
<th>Outline</th>
<th>Initializes the various features used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h, r_cg_userdefine.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_MAIN_UserInit (void);</td>
</tr>
<tr>
<td>Explanation</td>
<td>Initializes the variables and others used.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

[Function Name] InitialiseDisplay

<table>
<thead>
<tr>
<th>Outline</th>
<th>Initializes LCD display.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>lcd.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void InitialiseDisplay (void);</td>
</tr>
<tr>
<td>Explanation</td>
<td>Initializes the display controller and LCD display.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

[Function Name] wait_time

<table>
<thead>
<tr>
<th>Outline</th>
<th>Waits for time in 60 us units.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void wait_time(uint16_t time);</td>
</tr>
<tr>
<td>Explanation</td>
<td>Waits for a time represented by 60 us × the number specified with the argument.</td>
</tr>
<tr>
<td>Arguments</td>
<td>Wait time in 60 us units</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

[Function Name] wait_60us

<table>
<thead>
<tr>
<th>Outline</th>
<th>Waits for 60 us using TM03.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void wait_60us(void);</td>
</tr>
<tr>
<td>Explanation</td>
<td>Waits for 60 us using TM03 (delay count).</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

[Function Name] set_command

<table>
<thead>
<tr>
<th>Outline</th>
<th>Transmits the command to LCD via I2C.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h, lcd.h, r_iica_user.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void set_command(uint8_t comcode);</td>
</tr>
<tr>
<td>Explanation</td>
<td>Transmits the command specified with the argument to the LCD controller via I2C.</td>
</tr>
<tr>
<td>Arguments</td>
<td>Command to LCD</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>
[Function Name] set_data
Outline Transmits data to LCD via I2C.
Header r_cg_macrodriver.h, lcd.h, r_iica_user.h
Declaration void set_data(uint8_t datacode);
Explanation Transmits the display data specified with the argument to the LCD controller via I2C.
Arguments Display data to LCD
Return value None
Remarks None

[Function Name] DisplayString
Outline Displays ASCII character strings on LCD.
Header r_cg_macrodriver.h, lcd.h, r_iica_user.h
Declaration void DisplayString(uint8_t position, __far int8_t * string);
Explanation Displays the ASCII character strings specified with the second argument on the LCD position specified with the first argument via I2C.
Arguments First argument Display start position on LCD
Second argument Pointer to the character strings to be displayed
Return value None
Remarks None

[Function Name] R_IICA_Master_Send
Outline Starts data transmission to the I2C slave.
Header r_cg_macrodriver.h, r_iica_user.h
Declaration MD_STATUS R_IICA_Master_Send(uint8_t addr, uint8_t * const tx_buf, uint16_t tx_num);
Explanation Starts transmitting the specified data to the slave specified with the argument by using IICA1. Returns an error as the transmission status if the I2C bus is currently used, and starts transmitting the slave address when the I2C bus is available.
Arguments First argument Slave address
Second argument Pointer to data to be transmitted
Third argument Number of data to be transmitted
Return value Transmission status:
MD_OK Successfully started
MD_ERROR1 I2C bus is currently used.
Remarks None

[Function Name] R_IICA_wait_command
Outline Waits for completion of I2C transfer.
Header r_cg_macrodriver.h, r_iica_user.h
Declaration MD_STATUS R_IICA_wait_command(void);
Explanation Checks the IICA1 status and waits for completion of transfer (communication).
Arguments None
Return value Communication result status:
MD_OK Communication successfully completed
MD_ERROR3 No ACK response from the slave
Remarks None
[Function Name] R_IICA_StopCondition  
Outline: Issues the stop condition from IICA to the I2C bus.  
Header: r_cg_macrodriver.h, r_iica_user.h  
Declaration: void R_IICA_StopCondition(void);  
Explanation: Issues the stop condition from IICA to the I2C bus and waits for the stop condition to be detected.  
Arguments: None  
Return value: None  
Remarks: None  

[Function Name] R_IICA_bus_check  
Outline: Checks the I2C bus state.  
Header: r_cg_macrodriver.h, r_iica_user.h  
Declaration: MD_STATUS R_IICA_bus_check(void);  
Explanation: Checks the I2C bus state and issues the start condition if the bus is available. Successfully ends if the start condition is detected by the 100th checking.  
Arguments: None  
Return value: Transmission status:  
- MD_OK: Successfully started.  
- MD_ERROR1: I2C bus is currently used.  
Remarks: None  

[Function Name] r_iica_interrupt  
Outline: Processes the IICA1 transfer end interrupt.  
Header: r_cg_macrodriver.h, r_iica_user.h  
Declaration: #pragma interrupt r_iica_interrupt(vect=INTIICA1,BANK=rb1) static void r_iica_interrupt(void)  
Explanation: Processes the IICA1 transfer end interrupt. When transfer of the specified data is completed, sets the pertinent status to “communication successfully completed”.  
Arguments: None  
Return value: None  
Remarks: None  

[Function Name] r_intp0_interrupt  
Outline: Processes the INTP0 interrupt.  
Header: r_cg_macrodriver.h, r_timer_user.h  
Declaration: #pragma interrupt r_intp0_interrupt(vect=INTP0) static void __near r_intp0_interrupt(void)  
Explanation: Initiated when SW is pressed; activates the 8-bit interval timer.  
Arguments: None  
Return value: None  
Remarks: None
### Function Name: r_it_interrupt

**Outline:** Processes the interval timer interrupt.

**Header:**
```
r_cg_macrodriver.h, r_timer_user.h
```

**Declaration:**
```
#define interrupt r_it_interrupt(vect=INTIT00)
static void __near r_it_interrupt(void)
```

**Explanation:** Initiated by the INTIT00 every second; measures the internal reference voltage by using the A/D, sets DTC and TM01, and activates the A/D hardware trigger every millisecond.

**Arguments:** None

**Return value:** None

**Remarks:** None

### Function Name: R_ADC_inter_ref

**Outline:** Measures the internal reference voltage.

**Header:**
```
r_cg_macrodriver.h, r_timer_user.h, r_cg_adc.h
```

**Declaration:**
```
void R_ADC_inter_ref(void);
```

**Explanation:** Measures the internal reference voltage, stores it to variable g_vref, sets the measurement range to 2 (100 V), and initializes the parameters for measurement.

**Arguments:** None

**Return value:** None

**Remarks:** None

### Function Name: R_ADC_DTC_Init

**Outline:** Makes the initial settings for A/D and DTC.

**Header:**
```
r_cg_macrodriver.h, r_timer_user.h, r_cg_adc.h, r_cg_dtc.h
```

**Declaration:**
```
void R_ADC_DTC_Init(void);
```

**Explanation:** Sets ADC to hardware trigger and single scan mode of ANI3-ANI0, and sets the parameters for DTC transfer again.

**Arguments:** None

**Return value:** None

**Remarks:** None

### Function Name: r_tm01_start

**Outline:** Activates TM01.

**Header:**
```
r_cg_macrodriver.h, r_timer_user.h
```

**Declaration:**
```
void r_tm01_start(void);
```

**Explanation:** Activates TM01 for A/D conversion trigger with interrupt disabled.

**Arguments:** None

**Return value:** None

**Remarks:** None

### Function Name: delay_1u

**Outline:** Waits for 1 us.

**Header:**
```
r_cg_macrodriver.h, r_timer_user.h
```

**Declaration:**
```
void delay_1us(void);
```

**Explanation:** Waits for 1 us after ADCE setting until ADCS setting.

**Arguments:** None

**Return value:** None

**Remarks:** None
### Function Name: r_adc_interrupt

**Outline:** Processes the INTAD interrupt.

**Header:** r_cg_macrodriver.h, r_timer_user.h

**Declaration:**
```
#pragma interrupt r_adc_interrupt(vect=INTAD)
static void __near r_adc_interrupt(void)
```

**Explanation:**
Initiated by the DTC transfer end interrupt of 4-channel A/D conversion data.
Performs the appropriate operation on the A/D converted data according to the measurement range, and records the maximum value and sign change points of the result. After the 32nd measurement, stops TM01 and completes A/D conversion. Before the 32nd measurement, sets the parameters for DTC transfer again.

**Arguments:** None

**Return value:** None

**Remarks:** None

### Function Name: __ssqrt

**Outline:** Performs square-root operation of 32-bit data.

**Header:** arith_lib.h

**Declaration:**
```
uint16_t __ssqrt(uint32_t);
```

**Explanation:**
Calculates the square root of the 32-bit data passed with the argument.

**Arguments:**
- 32-bit data
  - BC: Upper 16 bits, AX: lower 16 bits

**Return value:** Square root
  - AX (= BC)

**Remarks:** For details, refer to R01AN3079.

### Function Name: __r_mul32

**Outline:** 16 bits × 16 bits

**Header:** r_cg_macrodriver.h, arith_lib.h

**Declaration:**
```
uint32_t __r_mul32( uint16_t multiplicand, uint16_t multiplier );
```

**Explanation:**
Performs multiplication on two 16-bit arguments.

**Arguments:**
- First argument: Multiplicand (AX)
- Second argument: Multiplier (BC)

**Return value:** Product
  - (BC-AX)

**Remarks:** None

### Function Name: __r_div32

**Outline:** 32 bits/16 bits

**Header:** r_cg_macrodriver.h, arith_lib.h

**Declaration:**
```
uint32_t __r_div32( uint32_t dividend, uint16_t divisor );
```

**Explanation:**
Divides the first argument (32 bits) by the second argument (16 bits).

**Arguments:**
- First argument: Dividend (BC-AX)
- Second argument: Divisor (DE)

**Return value:** Quotient
  - (BC-AX)

**Remarks:** None
## __r_div16

<table>
<thead>
<tr>
<th>Outline</th>
<th>16 bits/8 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h, arith_lib.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>uint16_t __r_div16( uint16_t dividend, uint8_t divisor );</td>
</tr>
<tr>
<td>Explanation</td>
<td>Divides the first argument (16 bits) by the second argument (8 bits).</td>
</tr>
<tr>
<td>Arguments</td>
<td>First argument Dividend (AX)</td>
</tr>
<tr>
<td></td>
<td>Second argument Divisor (C)</td>
</tr>
<tr>
<td>Return value</td>
<td>Quotient AX</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

## __r_mod16

<table>
<thead>
<tr>
<th>Outline</th>
<th>Modulo operation of 16 bits/8 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h, arith_lib.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>uint8_t __r_mod16( uint16_t dividend, uint8_t divisor );</td>
</tr>
<tr>
<td>Explanation</td>
<td>Finds the modulo after dividing the first argument (16 bits) by the second argument (8 bits).</td>
</tr>
<tr>
<td>Arguments</td>
<td>First argument Dividend (AX)</td>
</tr>
<tr>
<td></td>
<td>Second argument Divisor (C)</td>
</tr>
<tr>
<td>Return value</td>
<td>Modulo A</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

## __r_rms16

<table>
<thead>
<tr>
<th>Outline</th>
<th>Performs root-mean-square operation of 16-bit data.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h, arith_lib.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>uint16_t __r_rms16( uint8_t start, uint18_t data_no );</td>
</tr>
<tr>
<td>Explanation</td>
<td>Calculates the root-mean-square of as many data as the number specified with the second argument, starting at the data specified with the first argument of the expected result (_g_vin_data[]).</td>
</tr>
<tr>
<td>Arguments</td>
<td>First argument Starting data (A)</td>
</tr>
<tr>
<td></td>
<td>Second argument Number of data (X)</td>
</tr>
<tr>
<td>Return value</td>
<td>Root-mean-square AX</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>
5.7 Flowchart

Figure 5.1 shows the overall flow of the process described in this application note.

Figure 1.1 Analog Voltage Input Circuit

Note: This processing is performed in the start-up routines (r_cg_cstart.asm, etc.). The setting for the memory is performed between calling the initial setting function and the main processing function.

5.7.1 Initial Setting Function

Figure 5.2 shows the flowchart of the initial setting function.
5.7.2 System Function

Figure 5.3 shows the flowchart of the system function.

![System Function Flowchart](image)

- **R_Systeminit()**
  - Set peripheral I/O redirect function.
  - Set I/O ports.
    - **R_PORT_Create()**
  - Set CPU clocks.
    - **R_CGC_Create()**
  - Set TAU0.
    - **R_TAU0_Create()**
  - Set 8-bit IT.
    - **R_IT8Bit0_Channel0_Create()**
  - Set IICA.
    - **R_IICA1_Create()**
  - Set A/D converter.
    - **R_ADC_Create()**
  - Set DTC.
    - **R_DTC_Create()**
  - Set INTP0.
    - **R_INTC_Create()**
  - Disable detection of illegal memory access.

**PIOR3 to PIOR0 registers ← 0x00**

**IAWCTL register ← 0x00**

return

Figure 5.3 System Function
5.7.3 Setting I/O Ports

Figure 5.4 shows the flowchart for setting the I/O ports.

Caution: Provide proper treatment for unused pins so that their electrical specifications are observed. Connect each of any unused input-only ports to VDD or VSS via separate resistors.
5.7.4 Setting CPU Clocks

Figure 5.5 shows the flowchart for setting the CPU clocks.

![Flowchart for setting CPU clocks]

- **R_CGC_Create()**
  - Set clock oscillators.
  - Select subsystem clock.
  - Wait for clock stabilization.
  - Select IT clock.
  - Select CPU/peripheral hardware clock ($f_{CLK}$).
  - return

MIOEN bit ← 0: Stops middle-speed on-chip oscillator.
CMC register ← 0b00000000: Does not use high-speed system clock.
MSTOP bit ← 1
SELLOSC bit ← 1: Selects $f_{L}$ for $f_{SUB}$.

Waits for $f_{L}$ oscillation to stabilize.
OSMC register ← 0x10: Selects low-speed on-chip oscillator clock.
CSS bit ← 0: Selects HOCO ($f_{H}$) as CPU/peripheral hardware clock.
MCM0 bit ← 0: $f_{CLK}$.
MCM1 bit ← 0

Figure 5.5 Setting CPU Clocks
5.7.5 Setting Timer Array Unit

Figure 5.6 shows the flowchart for setting the timer array unit.

```
R_TAU0_Create()
```

- R_TAU0_Create:
  - Reset TAU0.
  - TAU0RES bit ← 1: Resets TAU0.
  - TAU0RES bit ← 0: Releases TAU0 from reset.
  - TAU0EN bit ← 1
  - Supply clocks to TAU.
  - TPS0 register ← 0x0000: CK00 and CK01: 16 MHz
  - Set TAU operating clock.
  - TT0 register ← 0x0A0F: Stops operation of all channels.
  - Disable TAU operation.
  - TMMK03 to TMMK00 bits ← all 1: Masks interrupt requests.
  - TMIF03 to TMIF00 bits ← all 0: Clears interrupt requests.
  - Set TAU interrupts.
  - TMR01 register ← 0x0000: Interval timer
  - TDR01 register ← 0x3E7F: Sets interval to 1 ms.
  - Set TM01 operation mode.
  - TO01 bit ← 0: Sets master channel output mode for TM01
  - TOL01 bit ← 0: Sets TO01 output level to active high.
  - TOE01 bit ← 0: Disables output.
  - Set TM01 output.
  - TIS05 bit ← 1: Connects TI03 to VOUT1 (dummy).
  - Set TM03 operation mode.
  - TMR03 register ← 0x0108: One–count mode
  - TDR03 register ← 0x03BF: Counts 60 us.
  - Set TO03.
  - TOM03 bit ← 0: Sets master channel output mode
  - TOL03 bit ← 0: Sets positive logic output.
  - TO03 bit ← 0: Sets low level output from TO03.
  - TOE03 bit ← 0: Disables output from TO35.
  - Set TI03 input.
  - TNFEN03 bit ← 1: Turns off noise filter (dummy).
  - return
```

Figure 5.6 Setting Timer Array Unit
5.7.6 Setting 8-Bit Interval Timer

Figure 5.7 shows the flowchart for setting the 8-bit interval timer.

Figure 5.7  Setting 8-Bit Interval Timer

- **R_IT8Bit0_Channel0_Create()**
  - Supply clocks.
  - Stop timer.
  - Set interrupts.
  - Set interrupt priority.
  - Set operation mode.
  - Set count clock.
  - Set compare value.

  **Figure 5.7** Setting 8-Bit Interval Timer
5.7.7 Setting IICA

Figure 5.8 shows the flowchart for setting IICA.

![Flowchart for setting IICA](image)

- **R_IICA1_Create()**
  - Reset IICA.
  - Supply clocks to IICA.
  - Stop IICA operation.
  - Set IICA interrupts.
  - Set multiplexed port output.
  - Set standard mode operation.
  - Set SCL width.
  - Set slave address.
  - Set communication conditions.
  - Set operating conditions.
  - Enable IICA interrupt.
  - Enable IICA operation.
  - Exit from I2C bus communications.
  - Enable output from multiplexed ports.
  - return

- IICA1RES bit ← 1: Resets IICA1.
- IICA1RES bit ← 0: Releases IICA1 from reset.
- IICA1EN bit ← 1: Supplies clocks to IICA1.
- IICE1 bit ← 0: Stops IICA1 operation.
- IICAMK1 bit ← 1: Masks interrupt requests.
- IICAF1 bit ← 0: Clears interrupt requests.
- IICAPR11 and IICAPR01 bits ← 11: Sets lowest interrupt priority
- POM56 and POM55 bits ← 1,1: Sets ports as N-ch open-drain output.
- PMC56 bit ← 0: Sets P56 as digital I/O.
- P56 and P55 bits ← 0,0: Sets output latch of P56 and P55 to 0.
- PM56 and PM55 bits ← 1,1: Leaves ports in input mode.
- SMC0 bit ← 0: Sets standard mode operation.
- IICWL0 register ← 0x4C
- IICWH0 register ← 0x55
- SVA1 register ← 0x10: Sets slave address.
- STCEN1 bit ← 1: I2C bus is released by default.
- IICRSV1 bit ← 1: Disables communication reservation.
- SPIE1 bit ← 0: Disables stop condition detection interrupt.
- WTIM1 bit ← 1: Generates interrupt at ninth clock.
- ACKE1 bit ← 1: Enables ACK response.
- IICAMK1 bit ← 0: Enables IICA1 interrupt.
- IICE1 bit ← 1: Enables IICA1 operation.
- LREL1 bit ← 1: Exits from communications.
- PM56 and PM55 bits ← 0,0: Enables output of SCL and SDA signals.

Figure 5.8 Setting IICA
5.7.8 Setting A/D Converter

Figure 5.9 shows the flowchart for setting the A/D converter.

![Flowchart for Setting A/D Converter](image)

- **R_ADC_Create()**
  - Reset A/D converter.
  - Supply clocks to A/D converter.
  - Stop A/D converter operation.
  - Set A/D converter interrupts.
  - Set multiplexed ports.
  - Set operation mode of A/D converter.
  - Set conversion result comparison value.
  - Set scan range.
  - return

**Code Explanation**

- **ADRES bit ← 1**: Resets ADC.
- **ADRES bit ← 0**: Releases ADC from reset.
- **ADCEN bit ← 1**
- **ADM0 register ← 0x00**: Stops operation of A/D converter.
- **ADMK bit ← 1**: Masks interrupt request.
- **ADIF bit ← 0**: Clears interrupt request.
- **ADPR1 and ADPR0 bits ← 11**: Sets lowest interrupt priority
- **PMC23 to PMC20 bits ← 1111**: Sets analog input.
- **PM23 to PM20 bits ← 1111**: Sets input mode.
- **ADM0 register ← 0x54**: Scan mode, low voltage mode, 19 $\mu$s
- **ADM1 register ← 0xE0**: Hardware trigger wait mode, one-shot conversion
- **ADM2 register ← 0x00**: AVREF is supplied from VDD and VSS, 10-bit resolution
- **ADUL register ← 0xFF**: Sets upper limit.
- **ADLL register ← 0x00**: Sets lower limit.
- **ADS register ← 0x00**: Scans ANI0 to ANI3.
5.7.9 Setting DTC  
Figure 5.10 shows the flowchart for setting the DTC.

```
  R_DTC_Create()

  Supply clocks.

  DTCEN bit ← 1: Supplies clock.

  Stop DTC channel operation.

  DTCEN2 to DTCEN0 registers ← All 0x00: Stops operation.

  Set DTC vector area.

  DTCBAR register ← 0xFD: Places the vector at 0xFFD00.

  Set A/D conversion vector.

  dtc_vector[9] ← 0x40: Uses control data 0.

  Set control data 0.

    .dtcr ← 0x48: Sets 16-bit transfer, transfer destination + 1, and fixed transfer source address.
    .dtbls ← 0x01: Sets 2-byte block size.
    .dtct ← 0x04: Sets number of DTC data transfer to 4.
    .dtrld ← 0x00: Dummy (not reloaded)
    .dtsar ← 0xFF1E: Sets ADCR as transfer source.
    .dtdar ← 0xFE00: Sets RAM area starting with FFE00 as transfer destination.

  return
```

Figure 5.10 Setting DTC

5.7.10 Setting INTP0  
Figure 5.11 shows the flowchart for setting INTP0.

```
  R_INTC_Create()

  Initialize interrupts.

  PMK11 to PMK9 bits, PMK5 to PMK0 bits ← All 1: Masks interrupts.
  PIF11 to PIF9 bits, PIF5 to PIF0 bits ← All 0: Clears interrupt requests.

  Set INTP0 interrupt priority.

  PPR10 and PPR00 bits ← 1, 1: Sets lowest priority level.

  Set detection edge.

    EGN0 bit ← 1: Enables falling edge detection.
    EGP0 bit ← 0: Disables rising edge detection.

  return
```

Figure 5.11 Setting INTP0
5.7.11 Main Processing

Figure 5.12 to Figure 5.14 show the flowcharts of the main processing.

```plaintext
5.7.11 Main Processing

Figure 5.12 to Figure 5.14 show the flowcharts of the main processing.

Figure 5.12 Main Processing (1/3)

main()

Initialization by user
InitialiseDisplay()

A

Startup screen display on LCD
DisplayString()

Display on LCD
DisplayString()

No

STOP?

Yes

Stop 1 second timer.

Stop TM03.

Enable SW interrupt.

STOP

Start TM03.

Wait for end of measurement

g times = 0, 32, +1

HALT

Wait for end of measurement.

No

Yes

g_vmax > 937?

No

Yes

Display "OVER" on second line.

B

Additional initialization

Displays first line of start-up screen on LCD.

Displays second line on LCD.

Enters STOP mode and waits for INTP0 if variable g_stpcnt is 20 or greater.

TSTART00 bit ← 0; Stops 8-bit interval timer.
ITMK00 bit ← 1: Disables 8-bit interval timer interrupt.
TT0L register ← Ox08: Stops TM03.
PIF0 bit ← 0: Clears INTP0 interrupt request.
PMK0 bit ← 0: Enables INTP0 interrupt.

Waits for SW press interrupt (INTP0) in STOP mode.

INTP0 (Cancels STOP mode.)

TS0L register ← Ox08: Starts TM03.

Waits for 32nd measurement to complete in HALT mode.

When variable g_vmax is greater than 937, voltage is above the upper limit.

Displays "OVER" to indicate overvoltage and returns.
```
When variable `g_vmax` is greater than 93, voltage is measured in 100-V range (range 2).

When variable `g_vmax` is greater than 9, voltage is measured in 10-V range (range 1). When variable `g_vmax` is 9 or smaller, it is measured in 1-V range (range 0).

Variable `g_range` ← 1: Sets range 1 for measurement.
PM01 and PM00 bits ← 0,0: Turns on 1/10 voltage-dividing resistor.
PM31 and PM30 bits ← 1,1: Turns off 1/100 voltage-dividing resistor.

Variable `g_range` ← 0: Sets range 0 for measurement.
PM31 and PM30 bits ← 1,1: Turns off 1/100 voltage-dividing resistor.

Repeats measurement for range 1 and range 0.

Variable `dtc_controldata_0.dtdar` ← 0xFE00:
Sets 0xFFE00 as transfer destination address.

Variable `dtc_controldata_0.dtcct` ← 0x04:
Sets number of transfers to 4.

DTCEN16 bit ← 1: Enables DTC activation on completion of A/D conversion.

Variable `g_times` ← 0: Sets number of measurements to 0.
Variable `g_vin_sign` ← 0x00: Initializes signs.
Variable `g_sign_chng1` ← 0x00: Clears zero cross point.
Variable `g_sign_chng2` ← 0x00: Clears zero cross point.
Variable `g_sign_chng3` ← 0x00: Clears zero cross point.
Variable `g_vmax` ← 0x00: Clears maximum value.

Starts TM01 (A/D trigger) to perform A/D conversion in target range.

Waits for 32nd measurement to complete in HALT mode.

End of 32nd measurement in target range

---

Figure 5.13  Main Processing (2/3)
Variable $g_{\text{start}} \leftarrow 12$: Sets start position of targets.
Variable $g_{\text{period}} \leftarrow 20$: Sets initial value for number of targets.

There are two or more zero crosses if variable $g_{\text{sign\_chng2}}$ is not 0x00.

Variable $g_{\text{start}} \leftarrow g_{\text{sign\_chng2}}$: Sets position.
Variable $g_{\text{period}} \leftarrow g_{\text{sign\_chng3}} - g_{\text{sign\_chng2}} + 1$: Initializes number of data indicated on display.

Variable $g_{\text{voltage}} \leftarrow _{r_{\text{rms16}}}(g_{\text{start}}, g_{\text{period}})$: Calculates root-meat-square of data within set range.

Left-shift by 4 bits for range 2
Left-shift by 2 bits for range 1

Corrects measurement result into ASCII strings and displays them.
Divides result by 10, and displays modulo in digit for 10 mV.
Stores quotient in variable $v_{\text{work}}$.
Divides variable $v_{\text{work}}$ by 10, and displays modulo in digit for 100 mV.
Divides variable $v_{\text{work}}$ by 10, and displays modulo in digit for 1 V.
Deletes unnecessary 0s displayed.

Blank in digits for 10 V and 100 V
Display in digits for 10 V and 100 V
Display in digit for 10 V
Blank in digit for 100 V

Counts down variable $g_{\text{stpcnt}}$.

Figure 5.14 Main Processing (3/3)
5.7.12  R_MAIN_UserInit Processing

Figure 5.15 shows the flowchart of the R_MAIN_UserInit processing.

```
R_MAIN_UserInit()

Initialize parameters.
Change TM03 trigger.
Mask TM03 interrupt.
Enable TM03 operation.
Initialize LCD. InitialiseDisplay ()
Enable vector interrupt.
```

Variable g_times ← 0x00: Clears number of measurements to 0.
Variable g_stpcnt ← 20: Initializes STOP mode counter.

TMR03 register ← 0x0008: Enables software trigger only.
TMMK03 bit ← 1: Masks INTTM03.
TS0L register ← 0x08: Activates TM03.
(Places it in trigger wait state.)
Initializes LCD controller.
IE bit ← 1: Enables vector interrupt.
5.7.13 Initializing LCD

Figure 5.16 shows the flowchart for initializing the LCD.

```
InitialiseDisplay()

Wait for LCD to start.
  wait_time()

Clear LCD display.
  set_command()

Wait for LCD processing to complete.
  wait_time()

Set LCD display mode.
  set_command()

Enable LCD display.
  set_command()

Set display data entry.
  set_command()

return
```

Waits for 42 ms until LCD controller starts.

Clears LCD display.

Waits for 2.4 ms until processing of LCD controller is completed.

Sets 2-line display mode.

Enables LCD display.

Adds display data to the right.

Figure 5.16 Initializing LCD
5.7.14 Setting LCD Commands

Figure 5.17 shows the flowchart for setting the LCD commands.

```
set_command()

Set command in buffer.

Transmit command.
  R_IICA_Master_Send()

Activated normally?

No

Wait for transmission to complete.
  R_IICA_wait_command()

Yes

Issue stop condition.
  R_IICA_StopCondition()

Wait for 60 us.
  wait_60us()

return

g_write_command[0] ← COMBYTE: Specifies command.
g_write_command[1] ← Command code

Transmits command to LCD.

Waits for completion of command transmission.

Issues stop condition to I2C bus.

Waits for completion of LCD processing.
```

Figure 5.17 Setting LCD Commands
5.7.15 Wait Time Processing

Figure 5.18 shows the flowchart of the wait time processing.

wait_time()

Wait for specified time.
work=time, work=0, -1

Wait for 60 us.
wait_60us()

Wait for specified time.

return

Figure 5.18 Wait Time Processing

5.7.16 60-us Wait Processing

Figure 5.19 shows the flowchart of the 60-us wait processing.

wait_60us()

Clear TM03 interrupt.

Trigger TM03.

Wait for INTTM03.
TMIF03 = 1.

NOP()

Wait for INTTM03.

Clear TM03 interrupt.

return

Figure 5.19 60-us Wait Processing
5.7.17 Displaying Character Strings on LCD

Figure 5.20 shows the flowchart for displaying character strings on the LCD.

DisplayStrings()

No

Display data for first line?

Yes

Move cursor to first line.
set_command()

Transmits (LCD_HOME_L1 + position) to LCD.

Move cursor to second line.
set_command()

Transmits (LCD_HOME_L2 + position) – LCE_LINE2 to LCD.

Transmit display data.

Transmit display characters.
set_data()

Transmits display character strings up to the character immediately preceding terminator.

, Data = 0.
Transmit display data.

return

Figure 5.20 Displaying Character Strings on LCD
5.7.18 Setting Display Data
Figure 5.21 shows the flowchart for setting display data.

```
set_data()

Set command in buffer.
g_write_data[0] ← DATABYTE: Specifies data.
g_write_data[1] ← Display data

Wait for 60 us.
wait_60us()

Transmit display data.
R_IICA_Master_Send()

Transmits data to LCD.

Wait for transmission to complete.
R_IICA_wait_command()

Waits for completion of command transmission.

Issue stop condition.
R_IICA_StopCondition()

Issues stop condition to I2C bus.

Wait for 60 us.
wait_60us()

Waits for completion of LCD processing.

return
```

Figure 5.21 Setting Display Data

5.7.19 Starting I2C Transmission
Figure 5.22 shows the flowchart for starting I2C transmission.

```
R_IICA_Master_Send()

Check I2C bus state.
R_IICA_bus_check()

Checks to see if I2C bus is available.

Bus is available?

Yes

Set transmission parameters.
Variable g_iica_tx_cnt ← tx_num: Sets number of transmission data.
Variable gp_iica_tx_address ← tx_buf: Sets pointer.

Transmit slave address.
IICA1 register ← adr & 0xFE: Transmits slave address.

No

return
```

Figure 5.22 Starting I2C Transmission
5.7.20  Waiting for Completion of I2C Communication

Figure 5.23 shows the flowchart for waiting for completion of I2C communication.

![Flowchart for Waiting for Completion of I2C Communication]

- **R_IICA_wait_comend()**
  - Communication completed?
    - No
      - Waits in NOP loop until value of variable `g_iica_status` becomes value other than 0x01.
    - Yes
      - Sets value of variable `g_iica_status` (communication result) as return value.
  - Set communication result as return value.
  - return

**Figure 5.23  Waiting for Completion of I2C Communication**

5.7.21  Issuing Stop Condition

Figure 5.24 shows the flowchart for issuing the stop condition.

![Flowchart for Issuing I2C Stop Condition]

- **R_IICA_StopCondition()**
  - Issue stop condition.
  - SPT1 bit ← 1: Issues stop condition.
  - No
    - Waits in NOP loop until SPD1 bit is set to 1.
  - Stop condition detected?
    - Yes
    - return

**Figure 5.24  Issuing I2C Stop Condition**
5.7.22 Checking I2C Bus Status

Figure 5.25 shows the flowchart for checking the I2C bus status.

```
R_IICA_bus_check()

Set initial value of status.

Bus is available?

Yes

Issue start condition.

Wait for detection of start condition.

No

Bus use right acquired?

Yes

Set status to normal.

Set status to "under communication".

Set return value.

return

Variable status ← MD_ERROR1: Sets status to ERROR.

Bus is available if IICBSY1 bit is 0 or MSTS1 bit is 1.

STT1 bit ← 1: Issues start condition.

Waits until timeout or detection of start condition.

Variable status ← MD_OK: Sets status to normal.

Variable g_iica_status ← 0x01:
Sets status to "under communication".

Sets status as return value.
```

Figure 5.25 Checking I2C Bus Status
5.7.23 INTP0 Interrupt Processing
Figure 5.26 shows the flowchart of INTP0 interrupt processing.

![Flowchart of INTP0 Interrupt Processing]

5.7.24 INTIT00 Interrupt Processing
Figure 5.27 shows the flowchart of INTIT00 interrupt processing.

![Flowchart of INTIT00 Interrupt Processing]
5.7.25 Checking Internal Reference Voltage

Figure 5.28 and Figure 5.29 show the flowcharts for checking internal reference voltage.

**Figure 5.28 Checking Internal Reference Voltage (1/2)**

- **R_ADC_inter_ref()**
- Stop ADC.
- Disable A/D interrupt.
- Set A/D conversion mode.
- Select internal reference voltage.
- Enable operation of A/D voltage comparator.
- Wait for voltage to stabilize. delay_us()
- Start A/D conversion.
- NOP
- A/D conversion completed?
- Yes
- Clear interrupt request.
- NOP
- No
- A/D conversion completed?
- Yes
- Store A/D conversion result.
- Initialize number of measurements.
- Initialize measurement range.
- A
- ADM0 register ← 0x00: Stops ADC.
- ADMK bit ← 1: Disables A/D interrupt.
- ADM0 register ← 0x14: select mode, conversion time: 19 us
- ADM1 register ← 0x00: software trigger, sequential conversion
- ADM2 register ← 0x00: 10-bit resolution
- ADS register ← 0x81: Selects internal reference voltage.
- ADCE bit ← 1: Activates A/D voltage comparator.
- waits for 1 us or more.
- ADCS bit ← 1: Starts A/D conversion.
- ADIF bit ← 0: Clears interrupt flag.
- Executes NOP instruction to wait for completion of A/D conversion.
- waits for completion of A/D conversion while polling ADIF bit.
- ADIF bit ← 0: Clears interrupt request flag.
- Executes NOP instruction to wait for completion of A/D conversion.
- waits for completion of A/D conversion while polling ADIF bit.
- Variable g_vref ← (ADCR register >>6): Stores internal reference voltage.
- Variable g_times ← 0x00: Initializes number of measurements to 0.
- Variable g_range ← 0x02: Initializes measurement range to 2 (100-V range).
Figure 5.29  Checking Internal Reference Voltage (2/2)

5.7.26  1-us Wait Processing
Figure 5.30 shows the flowchart of the 1-us wait processing.

PM01 and PM00 bits ← 0,0: Sets voltage division by 100.
PM31 and PM30 bits ← 1,1: Cancels voltage division by 10.

Variable g_vin_sign ← 0x00:  Sign of input signal: positive (+)

Variables g_sign_chng3 to g_sign_chng1 ← 0x00:
Initializes zero cross point.

Variable g_vmax ← 0x0000: Sets maximum value to 0.

Figure 5.30  1-us Wait Processing
5.7.27 Setting ADC and DTC
Figure 5.31 shows the flowchart for setting the ADC and the DTC.

```
R_ADC_DTC_Init()
Stop ADC.
Set operation mode of A/D converter.
Set channels.
Enable operation of A/D voltage comparator.
Set ADC interrupt.
Set DTC transfer destination address.
Set number of transfers.
Enable DTC transfer to be triggered by completion of A/D conversion.
RETI
```

- ADM0 register ← 0x00: Stops ADC operation.
- ADM0 register ← 0x54: Scan mode, low-voltage, 19 us
- ADM1 register ← 0xE0: Hardware trigger, one-shot conversion
- ADM2 register ← 0x00: AVREF is VDD and VSS. 10-bit resolution
- ADS register ← 0x00: Scans AN13 to AN10.
- ADIF bit ← 0: Clears interrupt request.
- ADMK bit ← 0: Enables interrupt.
- variable dtc_controldata_0.dtdar ← 0xFE00:
  Sets transfer destination address to 0xFFE00.
- variable dtc_controldata_0.dtcct ← 0x04:
  Sets number of transfers to 4.
- DTCEN16 bit ← 1: Enables DTC activation to be triggered by completion of A/D conversion.

Figure 5.31 Setting ADC and DTC

5.7.28 Starting TM01
Figure 5.32 shows the flowchart for starting TM01 (1-ms interval) that is used for triggering ADC.

```
r_tm01_start()
Disable TM01 interrupt.
Start TM01.
return
```

- TMMK01 bit ← 1: Disables INTTM01 interrupt.
- TS0L register ← 0x02: Starts TM01 (1-ms interval).

Figure 5.32 Starting TM01
5.7.29 A/D Conversion End Interrupt Processing

Figure 5.33 and Figure 5.34 show the flowcharts of A/D conversion end interrupt processing.

```
r_adc_interrupt()
  Read probe voltage.
  g_range
  = 2
  = 1
  Other values
  Calculate first term.
  Calculate intermediate potential of divided voltage.
  Calculate second term.
  Set compression ratio to 1/16.
  Checks sign of input voltage.
  Variable work2 ← variable work2 − variable work1
  Variable sign_work ← 0x01: Result is negative.
  Variable work2 ← variable work1 − variable work2
  Variable sign_work ← 0x00: Result is positive.
  Variable AD_work ← Compresses value of variable work2.
  Variable work2 ← variable AD_work × 145
  Variable g_vin_data[g_times] ← variable work2/variable g_vref
  (Stores result of this measurement.)
```

Figure 5.33 A/D Conversion End Interrupt Processing (1/2)
Checks for zero cross point and updates zero cross point data.

- Variable g_sign_chng1 ← Variable g_sign_chng2 (last change point but one)
- Variable g_sign_chng2 ← Variable g_sign_chng3 (last change point)
- Variable g_sign_chng3 ← Variable g_times (number of measurements)
- Variable g_vin_sign ← Variable sign_work (current sign)

Obtains maximum measurement value.

- Variable g_vmax ← Variable g_vin_data[g_times] (result of this measurement)

Adds 1 to g_times.

Stops TM01 (AD conversion start trigger).

- PM01 and PM00 bits ← 0,0: Sets voltage division by 100.
- PM31 and PM30 bits ← 1,1: Cancels voltage division by 10.

Sets DTC transfer destination address to 0xFFE00.

- Variable dtc_controldata_0.dtdar ← 0xFE00: Sets transfer destination address to 0xFFE00.
- Variable dtc_controldata_0.dtcct ← 0x04: Sets number of transfers to 4.

DTCEN16 bit ← 1:
- Enables DTC activation to be triggered by completion of A/D conversion.
5.7.30  I2C Communication End Interrupt Processing

Figure 5.35 and Figure 5.36 show the flowcharts of I2C communication end interrupt processing.

**Figure 5.35  I2C Communication End Interrupt Processing (1/2)**

- **r_iica_interrupt()**
  - Checks I2C operation mode. Exits if I2C is in slave mode.
  - Communicating via I2C bus as master.
  - Checks if address (or data) transmission is completed.
  - Checks if ACK for address transmission is received.
  - Checks if data remains to be transmitted.

**Flowchart Details:**

1. **Master mode?**
   - No
   - Yes
     - **Address transmission completed?**
       - No
       - Yes
         - **ACK received?**
           - No
           - Yes
             - **Transmission mode?**
               - No
               - Yes
                 - **Any data remaining?**
                   - No
                   - Yes
                     - IICA1 register ← *gp_iica_tx_address: Transmits next data.
                     - Updates pointer (gp_iica_tx_address) and counter (g_iica_tx_cnt).
                     - Variable g_iica_status ← 0: Transmission end
                     - WTIM1 bit ← 0: Selects 8-clock wait.
                     - IICA1 register ← 0xFF: Dummy data for starting reception
                     - SPT1 bit ← 1: Issues stop condition.
                     - Variable g_iica_status ← MD_ERROR3: Communication error

2. Set status to ERROR.

**Notes:**

- Variable g_iica_status ← 0: Transmission end
- Variable g_iica_status ← MD_ERROR3: Communication error
Figure 5.36  I2C Communication End Interrupt Processing (2/2)
5.7.31  16-Bit Multiplication
Figure 5.37 shows the flowchart of 16-bit multiplication.

```
__r_mul32()
  MULHU
  RET
```

5.7.32  32-Bit Division
Figure 5.38 shows the flowchart of 32-bit division.

```
__r_div32()
  Clear upper 16 bits of divisor.
  HL register ← #0: Clears upper 16 bits of divisor.
  DIVWU
  RET
```

5.7.33  16-Bit Division
Figure 5.39 shows the flowchart of 16-bit division.

```
__r_div16()
  Set divisor.
  D register ← 0
  E register ← second argument
  DIVHU
  RET
```
5.7.34  16-Bit Modulo Operation

Figure 5.40 shows the flowchart of 16-bit modulo operation.

___r_mod16()

Set divisor.
D register ← 0
E register ← second argument
Unsigned division (AX ← AX/DE, DE ← modulo)

DIVHU

Set modulo.
A register ← E register (modulo)

RET

Figure 5.40  16-Bit Modulo Operation
5.7.35 Root-Mean-Square Calculation for 16-Bit Data

Figure 5.41 shows the flowchart for calculating the root-mean-square of 16-bit data.

```
_root_mss16()

Clear multiply-accumulate operation result.

Set data start pointer.

Set number of data.

Square accumulation

Read offset.

Set data.

MACHU

Update offset value.

Count number of data.

Number of data = 0

Square accumulation

Read accumulated value.

Mean calculation

R_DIV32

Square-root operation

_ssqrt()  

MACRH and MACRL registers ← 0x0000:
Cleans result of multiply-accumulate operation.

Variable WORK1 ← first argument << 1:
Changed to offset information

D and E registers ← second argument:
Sets number of data in working register.

B register ← variable WORK1: Sets offset.

AX register ← _g_vin_data[B]: Reads data.
BC register ← AX register

MACR register ← MACR + AX × BC: Accumulate squares.

Variable WORK1 ← WORK1 + 2: to next data.

D register ← D – 1: Counts down number of data.

BC register ← MACRH register: Reads high-order digits.
AX register ← MACRL register: Reads low-order digits.
BCAX register ← BCAX/D register: Calculates mean value.

Subsequent processing is performed in square-root operation unit.
```

Figure 5.41 Root-Mean-Square Calculation for 16-Bit Data
6. Sample Code
   The sample code is available on the Renesas Electronics Website.

7. Documents for Reference
   RL78/G11 User's Manual: Hardware (R01UH0637E)
   RL78 Family User's Manual: Software (R01US0015E)
   (The latest versions of the documents are available on the Renesas Electronics Website.)

   Technical Updates/Technical Brochures
   (The latest versions of the documents are available on the Renesas Electronics Website.)
Website and Support

Renesas Electronics Website
http://japan.renesas.com/

Inquiries
http://japan.renesas.com/contact/

All trademarks and registered trademarks are the property of their respective owners.
## Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Description</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Jan. 19, 2018</td>
<td>–</td>
<td></td>
<td>First edition issued</td>
</tr>
</tbody>
</table>
General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit 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 Microprocessing unit or Microcontroller unit products 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.
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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving 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, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.

3. 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 shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.

5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality.” The intended 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 appliances; machine tools; personal electronic equipment; and industrial robots.
   - **High Quality**: Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control 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 (space and underwater repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment etc.). Renesas Electronics disclaims any and all liability 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. When using the Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident 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 Renesas Electronics 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 ensure to implement safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including, but not limited to, redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures for your own responsibility as warranty for your products/system. Because the evaluation of microminic computer software alone is very difficult and not practical, 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 investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technologies shall 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 shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stocking, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles UVs) for delivering such weapons; (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.

10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.

11. This document shall not be reprinted, 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.

(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” means any product developed or manufactured by or for Renesas Electronics.