RL78/G14

7-Segment LED Lighting Control (Arduino API)

Introduction

This application note describes a method to control dynamically lighting of 7-segment LEDs using a program written in an Arduino language using the RL78/G14 Fast Prototyping Board (FPB).

Target Device

RL78/G14

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.
Contents

1. Specifications ............................................................................................................................ 3
  1.1 Program Execution Environment .............................................................................................. 4
  1.2 Program (Sketch) Configuration .................................................................................................. 5
  1.3 Preparations for Project Startup .................................................................................................. 5
  1.4 Definitions in the Program (sketch) ............................................................................................ 6
  1.5 Initial Setting Processing ............................................................................................................. 8
  1.6 Main Processing Part ................................................................................................................... 8

2. Operating Conditions ................................................................................................................. 9

3. Related Application Notes ......................................................................................................... 9

4. Hardware ................................................................................................................................... 10
  4.1 Example of Hardware Configuration ........................................................................................... 10
  4.2 List of Pins Used ........................................................................................................................ 10

5. Software .................................................................................................................................. 11
  5.1 Summary of Operation ................................................................................................................. 11
    (1) Pins to be used are set in the setup function. ......................................................................... 11
    (2) The main processing is performed in the loop function. ............................................................ 11
    (3) In the poll_sw function, the state of the SW_USR switch is confirmed every 10 milliseconds, and the number of times the switch is pressed is counted. .............................................. 12
  5.2 List of Constants ....................................................................................................................... 13
  5.3 List of Variables ........................................................................................................................ 14
  5.4 List of Functions ....................................................................................................................... 15
  5.5 Specification of Functions .......................................................................................................... 16
  5.6 Flowcharts ............................................................................................................................... 18
    5.6.1 Initial setting function ........................................................................................................... 18
    5.6.2 Main Processing Function ................................................................................................... 20
    5.6.3 Function to set display data for 7-segment LEDs ................................................................. 24

6. Sample Code ........................................................................................................................... 26

7. Reference Documents .............................................................................................................. 26

Revision History .............................................................................................................................. 27
1. Specifications

In this application note, an FPB is used with a program written in an Arduino language to dynamically control lighting of three 7-segment LEDs (cathode-common) and eight LEDs that are externally connected to the FPB.

The 7-segment LEDs realize a one-minute timer that displays an elapsed time in 100-millisecond (0.1 second) units. Upon the passing of each minute, the eight LEDs are lit in sequence, to realize a timer that can measure times up to nine minutes. Moreover, the timer counting operation can be controlled using a switch (SW_USR) mounted on the FPB or using an externally connected switch (ex_SW).

Table 1.1 shows peripheral functions used in this program and their uses.

<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital input</td>
<td>Reading the state of the switch (SW_USR) or external switch (ex_SW)</td>
</tr>
<tr>
<td>Digital output</td>
<td>Control of the LEDs (three 7-segment LEDs and eight separate LEDs)</td>
</tr>
<tr>
<td>Timer array unit</td>
<td>Measurement of the elapsed time</td>
</tr>
</tbody>
</table>
1.1 Program Execution Environment

In this application note, a program in an Arduino language is executed in a development environment specific to the RL78 family. A conceptual diagram of the program execution environment is shown in Figure 1.1.

![Figure 1.1 Program execution environment](image)

Library functions that can be used in this application note are shown in Table 1.2.

<table>
<thead>
<tr>
<th>Item</th>
<th>Library Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital I/O</td>
<td>pinMode(pin, mode)</td>
<td>Specifies the operation mode (input mode/output mode/input mode with internal pull-up resistor enabled) for the pin specified by “pin”.</td>
</tr>
<tr>
<td></td>
<td>digitalWrite(pin, value)</td>
<td>Sets the pin specified by “pin” to the state specified by “value” (high level/low level).</td>
</tr>
<tr>
<td></td>
<td>digitalRead(pin)</td>
<td>Reads out the state of the pin specified by “pin”.</td>
</tr>
<tr>
<td>Time control</td>
<td>millis()</td>
<td>Returns, in millisecond units, the time from the start of program execution to the present time.</td>
</tr>
<tr>
<td></td>
<td>micros()</td>
<td>Returns, in microsecond units, the time from the start of program execution to the present time.</td>
</tr>
<tr>
<td></td>
<td>delay (ms)</td>
<td>Stops the program for the specified time in millisecond units.</td>
</tr>
<tr>
<td></td>
<td>delayMicroseconds (us)</td>
<td>Stops the program for the specified time in microsecond units.</td>
</tr>
</tbody>
</table>
1.2 Program (Sketch) Configuration

Subfolders are prepared for each integrated development environment below the folder (workspace) in which the project is stored. In the folders for each of the integrated development environments the files are stored that are used in the RL78 family development environment.

In each sketch subfolder, AR_SKETCH.c is stored which is the Arduino language program (sketch). When viewing or modifying sketch, the "AR_SKETCH.c" file in the sketch subfolder is used.

1.3 Preparations for Project Startup

Preparations for project startup are different depending on the integrated development environment used. For details, refer to the following application note.

RL78 Family Arduino API Introduction Guide (R01AN5413)
1.4 Definitions in the Program (sketch)
Definitions in the program (sketch) are indicated in Figure 1.2.

Figure 1.2 Program definition details

```c
1) int segPinA = 0; // assign D0 pin to segPinA for 7SEG_LED.
   int segPinB = 1; // assign D1 pin to segPinB for 7SEG_LED.
   int segPinC = 2; // assign D2 pin to segPinC for 7SEG_LED.
   int segPinD = 3; // assign D3 pin to segPinD for 7SEG_LED.
   int segPinE = 4; // assign D4 pin to segPinE for 7SEG_LED.
   int segPinF = 5; // assign D5 pin to segPinF for 7SEG_LED.
   int segPinG = 6; // assign D6 pin to segPinG for 7SEG_LED.
   int segPinDP = 7; // assign D7 pin to segPinDP for 7SEG_LED.
   int comPin0 = 8; // assign D8 pin to comPin0 for 7SEG_LED.
   int comPin1 = 9; // assign D9 pin to comPin1 for 7SEG_LED.
   int comPin2 = 10; // assign D10 pin to comPin2 for 7SEG_LED.
   int comPin3 = 11; // assign D11 pin to comPin3 for DOT_LEDS.
   int ex_swPin = 13; // assign D13 pin to ex_swPin for external SW.
   int swPin = 18; // assign D18 pin to swPin for SW_USER.

2) const int SEG_TABLE[] = {
   0x3F, // segment data table for 7seg_LED
   0x06,
   0x5B,
   0x4F,
   0x66,
   0x6D,
   0x7D,
   0x27,
   0x7F,
   0x6F,
   0x00
};

3) int old_time = 0x0000; // previous time(milli sec.)
   char mini_data = 0; // minute data 0:1:3:7:F:1F:3F:7F:FF
   char sec_data = 0; // second count data(0 to 59)
   char prescount1 = 0; // prescaler for 100milli seconds. (25)
   char prescount2 = 0; // prescaler(10) for 1second
   char digi_sel = 0; // digit select
   char prescale4sw = 0; // prescaler for SW check (up to 5:20milli sec.)
   int sw_data = 0xFFFF; // SW data image(history of 20 milli sec. interval)
   int time_mode = 0; // o:stop/1:run
```
1) The digital output pins used to drive segments of the 7-segment LEDs are defined.
   0 is specified for the segPinA pin, which controls segment A, allocating this pin to D0.
   Similarly, segments B, C, D, E, F, G, and DP are defined.

   The digital output pins used to drive commons (for selecting the LED display digits) of the 7-segment
   LEDs are defined.
   8 is specified for the comPin0 pin, which controls the 10-second digit, allocating this pin to D8.
   9 is specified for the comPin1 pin, which controls the 1-second digit, allocating this pin to D9.
   10 is specified for the comPin2 pin, which controls the 0.1-second digit, allocating this pin to D10.
   11 is specified for the comPin3 pin, which controls eight separate LEDs, allocating this pin to D11.

   The digital input pins used to read the states of the switches are defined.
   13 is specified for the ex_swPin pin, which controls the external switch, allocating this pin to D13.
   18 is specified for the swPin pin, which controls the on-board switch (SW_USR), allocating this pin to
   D18.

2) Control data is defined using the array SEG_TABLE, which specifies the numerals (0 to 9) to be
   displayed by the 7-segment LEDs and specifies where there is no display (LED extinguished).

3) A number of variables used in the program (sketch) are defined.
   - old_time : Variable to confirm the elapsed time (in milliseconds)
   - sec_data : Variable to count seconds
   - mini_data : Variable to count minutes
   - precount1 : Counter variable to obtain 100 milliseconds from the timing of switching of the 4-
     millisecond display digit
   - precount2 : Variable to obtain 1 second by counting 100 milliseconds
   - digi_sel : Variable to control display dynamically
   - prescale4sw : Variable to count timing to confirm the switch state
   - sw_data : Variable to store change of the switch state
   - time_mode : Variable to start/stop counting time
1.5 Initial Setting Processing
The initial settings section of the program (sketch) is shown in Figure 1.3.

In the setup function, input or output mode is specified for each pin.

```c
void setup(void)
{
    // put your setup code here, to run once:
    pinMode(segPinA, OUTPUT);  // set D0pin to output mode
    pinMode(segPinB, OUTPUT);  // set D1pin to output mode
    pinMode(segPinC, OUTPUT);  // set D2pin to output mode
    pinMode(segPinD, OUTPUT);  // set D3pin to output mode
    pinMode(segPinE, OUTPUT);  // set D4pin to output mode
    pinMode(segPinF, OUTPUT);  // set D5pin to output mode
    pinMode(segPinG, OUTPUT);  // set D6pin to output mode
    pinMode(segPinDP, OUTPUT); // set D7pin to output mode
    pinMode(comPin0, OUTPUT);  // set D10pin to output mode
    pinMode(comPin1, OUTPUT);  // set D9pin to output mode
    pinMode(comPin2, OUTPUT);  // set D10pin to output mode
    pinMode(comPin3, OUTPUT);  // set D11pin to output mode
    pinMode(ex_swPin, INPUT_PULLUP); // set D13pin to input mode
    pinMode(swPin, INPUT); // set D18pin to input mode
}
```

Figure 1.3 Initial setting processing section

1.6 Main Processing Part
The leading section of the main processing, which is executed repeatedly, is shown in Figure 1.4. When preparations for project startup have been set correctly, the startup screen is as in Figure 1.4.

```c
void loop(void)
{
    // put your main code here, to run repeatedly:
    int time_work;  // present time buffer
    char com_sel;   // common select signal
    char seg_data; // segment data
    char sw_work;  // work for switch check

    wait for 4mili seconds interval.
    time_work = (int)(millis() & 0xFFC); // read milli sec data

    if (old_time != time_work) // check timing of change LED
    {
    }
```
2. Operating Conditions

The operation of the sample code provided with this application note has been tested under the following conditions.

Table 2.1 Operating conditions

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller used</td>
<td>RL78/G14 (R5F104MLAFB)</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>• High-speed on-chip oscillator clock ($f_{\text{IH}}$): 32 MHz</td>
</tr>
<tr>
<td></td>
<td>• CPU/peripheral hardware clock: 32 MHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>3.3 V (can be operated at 2.75 V to 5.5 V)</td>
</tr>
<tr>
<td></td>
<td>LVD operation: Reset mode</td>
</tr>
<tr>
<td></td>
<td>LVD detection voltage ($V_{\text{LVD}}$)</td>
</tr>
<tr>
<td></td>
<td>• At rising edge: 2.81 V typ. (2.76 V to 2.87 V)</td>
</tr>
<tr>
<td></td>
<td>• At falling edge: 2.75 V typ. (2.70 V to 2.81 V)</td>
</tr>
<tr>
<td>Integrated development environment</td>
<td>Renesas Electronics</td>
</tr>
<tr>
<td></td>
<td>• CS+ for CC V8.03.00</td>
</tr>
<tr>
<td></td>
<td>Renesas Electronics</td>
</tr>
<tr>
<td></td>
<td>• e² studio V7.7.0</td>
</tr>
<tr>
<td></td>
<td>IAR Systems</td>
</tr>
<tr>
<td></td>
<td>• Embedded Workbench for RL78</td>
</tr>
<tr>
<td>C compiler</td>
<td>Renesas Electronics</td>
</tr>
<tr>
<td></td>
<td>• CC-RL V1.09.00</td>
</tr>
<tr>
<td></td>
<td>IAR Systems</td>
</tr>
<tr>
<td></td>
<td>• IAR C/C++ Compiler v4.20.1 for RL78</td>
</tr>
</tbody>
</table>

3. Related Application Notes

The application notes related to this application note are shown below. Refer to these together with this application note.

RL78 Family Arduino API Introduction Guide (R01AN5413)
RL78/G14 Onboard LED Flashing Control (Arduino API) (R01AN5384)
4. Hardware

4.1 Example of Hardware Configuration

Figure 4.1 shows the hardware (FPB) that is used in this application note.

![Figure 4.1 Hardware configuration example](image)

Note: This conceptual diagram is simplified in order to summarize the connections.

As the power supply voltage, 3.3 V is supplied via USB.

4.2 List of Pins Used

Table 4.1 shows the pins used and their functions.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Port Name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>D16</td>
<td>P43</td>
<td>Output</td>
<td>Control of LED0</td>
</tr>
<tr>
<td>D17</td>
<td>P44</td>
<td>Output</td>
<td>Control of LED1</td>
</tr>
<tr>
<td>D18</td>
<td>P137</td>
<td>Input</td>
<td>Switch (SW_USR) input</td>
</tr>
</tbody>
</table>
5. Software

5.1 Summary of Operation

In this application note, when the initial settings (pin settings) are completed and the main processing (loop) is started, 7-segment LEDs display "00.0".

When the ex_SW or the SW_USR switch is pressed, the 7-segment LED display begins counting up. When the ex_SW or SW_USR switch is again pressed, the counting stops. Thereafter, each time the ex_SW or SW_USR switch is pressed, counting is resumed or stopped.

When counting upward causes the 7-segment LED display to again return to "00.0", one LED is lit. Each time the 7-segment LED display reaches "00.0", an additional LED is lit. When all eight LEDs are lit and the 7-segment LED display again reaches "00.0", the eight LEDs are all extinguished.

Thereafter, the above operation is repeated.

Details are explained in (1) to (3) below.

(1) Pins to be used are set in the **setup** function.

- The pins (segPinA to segPinDP) for driving segments of the 7-segment LEDs are set for digital output.
- The pins (comPin0 to comPin3) for driving commons of the 7-segment LEDs are set for digital output.
- The pin (ex_swPin) for confirmation of the state of the external switch (ex_SW) is set to an input mode in which an internal pull-up resistor is enabled.
- The pin (swPin) for confirmation of the state of the on-board switch (SW_USR) is set for digital input.

(2) The main processing is performed in the **loop** function.

- 10 bits worth, bits 11 to 2, of the time elapsed from startup (in millisecond units) are acquired.
- It is confirmed that the data acquired has changed from the old data (old_time).
- If the data has not changed, processing is ended, and execution returns to the beginning of the loop function.
- If the data has changed (4 milliseconds have elapsed), old_time is changed to the acquired data.
- Processing is switched according to the values of bits 3 and 2.
  - If (bit 3, bit 2) = (0, 0), the comPin3 pin is set to Low (selection cancelled). Next, segment data for the 10-second digit is determined, and is set at segPin using the set_SEG function. The display digit specification (the comPin0 pin) is set to the com_sel variable.
  - If (bit 3, bit 2) = (0, 1), the comPin0 pin is set to Low (selection cancelled). Next, segment data for the one-second digit is determined, and is set at segPin using the set_SEG function. At this time, the DP are also set to be lit. The display digit specification (the comPin1 pin) is set to the com_sel variable.
  - If (bit 3, bit 2) = (1, 0), the comPin1 pin is set to Low (selection cancelled). Next, segment data for the 100-millisecond digit is determined, and is set at segPin using the set_SEG function. The display digit specification (the comPin2 pin) is set to the com_sel variable.
  - If (bit 3, bit 2) = (1, 1), the comPin2 pin is set to Low (selection cancelled). Next, data for the minute digit is set at segPin using the set_SEG function. The display digit specification (the comPin3 pin) is set to the com_sel variable.
- The display digit specified by the com_sel variable is selected and displayed.
When counting is permitted (the time_mode variable is 1), time counting is performed.

The precount1 variable (4-millisecond units) is counted up, and upon reaching 25 (when 100 milliseconds have elapsed), precount1 is cleared, and the precount2 variable (100-millisecond units) is counted up.

The precount2 variable (100-millisecond units) is counted up, and upon reaching 10 (when one second has elapsed), the precount2 variable is cleared, and the sec_data variable (1-second units) is counted up.

The sec_data variable (1-second units) is counted up, and upon reaching 60 (when one minute has elapsed), the sec_data variable is cleared, and the mini_data variable (1-minute units) is updated. When the value of mini_data is 0xFF, it is set to 0x00. When the value of mini_data is not 0xFF, it is shifted one bit to the left, and the LSB (least significant bit) is set to 1. The value of mini_data changes to 0x00, 0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3F, 0x7F, 0xFF.

The switch circumstances are checked every 20 milliseconds.

The prescale4sw variable is counted up, and upon reaching 5 (20 milliseconds), the prescale4sw variable is cleared. The sw_data variable is shifted one bit to the left, and bit 0 is set to the result of taking the logical sum of the states of the two switches (switches use negative logic, so the logical product is taken). When either of the switches is pressed, the time_mode variable is changed.

(3) In the poll_sw function, the state of the SW_USR switch is confirmed every 10 milliseconds, and the number of times the switch is pressed is counted.

- The argument segment data is read into the work_data working variable.
- The LSB (least significant bit) of the work_data variable is output to the segPinA pin, and the work_data variable is shifted one bit to the right.
- The LSB (least significant bit) of the work_data variable is output to the segPinB pin, and the work_data variable is shifted one bit to the right.
- The LSB (least significant bit) of the work_data variable is output to the segPinC pin, and the work_data variable is shifted one bit to the right.
- The LSB (least significant bit) of the work_data variable is output to the segPinD pin, and the work_data variable is shifted one bit to the right.
- The LSB (least significant bit) of the work_data variable is output to the segPinE pin, and the work_data variable is shifted one bit to the right.
- The LSB (least significant bit) of the work_data variable is output to the segPinF pin, and the work_data variable is shifted one bit to the right.
- The LSB (least significant bit) of the work_data variable is output to the segPinG pin, and the work_data variable is shifted one bit to the right.
- The LSB (least significant bit) of the work_data variable is output to the segPinDP pin.
### 5.2 List of Constants

Table 5.1 shows constants that are used in the sample code.

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>segPinA</td>
<td>0</td>
<td>Number of the pin to control segment A</td>
</tr>
<tr>
<td>segPinB</td>
<td>1</td>
<td>Number of the pin to control segment B</td>
</tr>
<tr>
<td>segPinC</td>
<td>2</td>
<td>Number of the pin to control segment C</td>
</tr>
<tr>
<td>segPinD</td>
<td>3</td>
<td>Number of the pin to control segment D</td>
</tr>
<tr>
<td>segPinE</td>
<td>4</td>
<td>Number of the pin to control segment E</td>
</tr>
<tr>
<td>segPinF</td>
<td>5</td>
<td>Number of the pin to control segment F</td>
</tr>
<tr>
<td>segPinG</td>
<td>6</td>
<td>Number of the pin to control segment G</td>
</tr>
<tr>
<td>segPinDP</td>
<td>7</td>
<td>Number of the pin to control segment DP</td>
</tr>
<tr>
<td>comPin0</td>
<td>8</td>
<td>Number of the pin to select the 10-second digit</td>
</tr>
<tr>
<td>comPin1</td>
<td>9</td>
<td>Number of the pin to select the 1-second digit</td>
</tr>
<tr>
<td>comPin2</td>
<td>10</td>
<td>Number of the pin to select the 100-millisecond digit</td>
</tr>
<tr>
<td>comPin3</td>
<td>11</td>
<td>Number of the pin to select separate LED</td>
</tr>
<tr>
<td>ex_swPin</td>
<td>13</td>
<td>Number of the pin that reads external switch (ex_SW)</td>
</tr>
<tr>
<td>swPin</td>
<td>18</td>
<td>Number of the pin that reads SW_USR</td>
</tr>
<tr>
<td>SEG_TABLE[]</td>
<td>0x3F</td>
<td>Data to display 0 in a 7-segment LED</td>
</tr>
<tr>
<td></td>
<td>0x06</td>
<td>Data to display 1 in a 7-segment LED</td>
</tr>
<tr>
<td></td>
<td>0x5B</td>
<td>Data to display 2 in a 7-segment LED</td>
</tr>
<tr>
<td></td>
<td>0x4F</td>
<td>Data to display 3 in a 7-segment LED</td>
</tr>
<tr>
<td></td>
<td>0x66</td>
<td>Data to display 4 in a 7-segment LED</td>
</tr>
<tr>
<td></td>
<td>0x6D</td>
<td>Data to display 5 in a 7-segment LED</td>
</tr>
<tr>
<td></td>
<td>0x7D</td>
<td>Data to display 6 in a 7-segment LED</td>
</tr>
<tr>
<td></td>
<td>0x27</td>
<td>Data to display 7 in a 7-segment LED</td>
</tr>
<tr>
<td></td>
<td>0x7F</td>
<td>Data to display 8 in a 7-segment LED</td>
</tr>
<tr>
<td></td>
<td>0x6F</td>
<td>Data to display 9 in a 7-segment LED</td>
</tr>
<tr>
<td></td>
<td>0x00</td>
<td>Data to extinguish a 7-segment LED</td>
</tr>
<tr>
<td>HIGH</td>
<td>0x01</td>
<td>High level</td>
</tr>
<tr>
<td>LOW</td>
<td>0x00</td>
<td>Low level</td>
</tr>
</tbody>
</table>
### 5.3 List of Variables

Table 5.1 lists global variables.

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Description</th>
<th>Function used</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>old_time</td>
<td>Time elapsed time from previous startup (in millisecond units)</td>
<td>loop()</td>
</tr>
<tr>
<td>char</td>
<td>mini_data</td>
<td>Data indicating minutes elapsed: 0x00/0x01/0x03/0x07/0x0F/0x1F/0x3F/0x7F/0xFF</td>
<td>loop()</td>
</tr>
<tr>
<td>char</td>
<td>sec_data</td>
<td>Second data</td>
<td>loop()</td>
</tr>
<tr>
<td>char</td>
<td>precount1</td>
<td>Used to generate 100-millisecond timing (updated every 4 milliseconds)</td>
<td>loop()</td>
</tr>
<tr>
<td>char</td>
<td>Prescount2</td>
<td>Used to generate 1-second timing (updated every 100 milliseconds)</td>
<td>loop()</td>
</tr>
<tr>
<td>char</td>
<td>digi_sel</td>
<td>Specifies the digit to display (pin numbers: 8 to 11).</td>
<td>loop()</td>
</tr>
<tr>
<td>char</td>
<td>prescale4sw</td>
<td>Used to generate 20-second timing</td>
<td>loop()</td>
</tr>
<tr>
<td>Int</td>
<td>sw_data</td>
<td>Stores state changes of the switch (initial value is 0xFF).</td>
<td>loop()</td>
</tr>
<tr>
<td>Int</td>
<td>time_mode</td>
<td>Flag to control time counting (0: do not count, 1: count)</td>
<td>loop()</td>
</tr>
</tbody>
</table>
### 5.4 List of Functions

Table 5.3 shows a list of functions.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>loop</td>
<td>Main processing (sketch)</td>
</tr>
<tr>
<td>setup</td>
<td>Initialization function (sketch)</td>
</tr>
<tr>
<td>set SEG</td>
<td>Function to set display data for 7-segment LEDs (sketch)</td>
</tr>
<tr>
<td>pinMode</td>
<td>Specifies the operation mode of a pin (input mode/output mode/input mode with internal pull-up resistor enabled)</td>
</tr>
<tr>
<td>digitalWrite</td>
<td>Outputs data to a pin.</td>
</tr>
<tr>
<td>digitalRead</td>
<td>Reads the state of a pin.</td>
</tr>
<tr>
<td>micros</td>
<td>Returns the time, in microsecond units, from the start of program execution until the present time.</td>
</tr>
<tr>
<td>millis</td>
<td>Returns the time, in millisecond units, from the start of program execution until the present time.</td>
</tr>
<tr>
<td>delay</td>
<td>Stops the program for the specified time, in millisecond units.</td>
</tr>
<tr>
<td>delayMicroseconds</td>
<td>Stops the program for the specified time, in microsecond units.</td>
</tr>
</tbody>
</table>
### 5.5 Specification of Functions

The function specifications of the sample code are shown below.

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>loop</strong></td>
<td>Upon starting, the time from startup is checked, and every 4 milliseconds the output of 7-segment LED digit data is switched. Moreover, when counting is permitted, counting in units of 100 milliseconds, seconds, and minutes is performed. Every 20 milliseconds, the switch states are checked, and if a switch is being pressed, the count permission flag is changed.</td>
</tr>
<tr>
<td><strong>setup</strong></td>
<td>Pins used by the program (sketch) are set.</td>
</tr>
<tr>
<td><strong>set_SEG</strong></td>
<td>Data passed in the argument is set in the corresponding segment control pins as lighting data for the 7-segment LEDs.</td>
</tr>
<tr>
<td><strong>pinMode</strong></td>
<td>The pin indicated by the first argument is set to the mode indicated by the second argument.</td>
</tr>
</tbody>
</table>

### [Function name] loop

<table>
<thead>
<tr>
<th>Overview</th>
<th>Main function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>AR_LIB_PORT.h, AR_LIB_TIME.h, r_cg_macerdriver.h, AR_SKETCH.h, r_cg_userdefine.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void loop(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Upon starting, the time from startup is checked, and every 4 milliseconds the output of 7-segment LED digit data is switched. Moreover, when counting is permitted, counting in units of 100 milliseconds, seconds, and minutes is performed. Every 20 milliseconds, the switch states are checked, and if a switch is being pressed, the count permission flag is changed.</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
</tbody>
</table>

### [Function name] setup

<table>
<thead>
<tr>
<th>Overview</th>
<th>Initialization function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>AR_LIB_PORT.h, r_cg_macerdriver.h, r_cg_userdefine.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void setup(void);</td>
</tr>
<tr>
<td>Description</td>
<td>Pins used by the program (sketch) are set.</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
</tbody>
</table>

### [Function name] set_SEG

<table>
<thead>
<tr>
<th>Overview</th>
<th>Function to set display data for 7-segment LEDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>AR_LIB_PORT.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void set_SEG(char data);</td>
</tr>
<tr>
<td>Description</td>
<td>Data passed in the argument is set in the corresponding segment control pins as lighting data for the 7-segment LEDs.</td>
</tr>
<tr>
<td>Argument</td>
<td>Data : Data for segment to be lit</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
</tbody>
</table>

### [Function name] pinMode

<table>
<thead>
<tr>
<th>Overview</th>
<th>Function to set the pin function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>AR_LIB_PORT.h, r_cg_macerdriver.h, r_cg_userdefine.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void pinMode(uint8_t pin, uint8_t_t mode)</td>
</tr>
<tr>
<td>Description</td>
<td>The pin indicated by the first argument is set to the mode indicated by the second argument.</td>
</tr>
<tr>
<td>Argument</td>
<td>uint8_t pin : Number of the pin to be specified</td>
</tr>
<tr>
<td></td>
<td>uint8_t_t mode : Specifies the pin mode with OUTPUT/INPUT/INPUT_PULLUP</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
</tbody>
</table>
**[Function name] digitalWrite**

**Overview**
Function to output digital data to a pin

**Header**
AR_LIB_PORT.h, r_cg_macrodriver.h, r_cg_userdefine.h

**Declaration**
void digitalWrite(uint8_t pin, uint8_t value);

**Description**
The data indicated by the second argument is output to the pin indicated by the first argument.

**Argument**
- uint8_t pin : Number of the pin for data output
- uint8_t value : Data to output (HIGH/LOW)

**Return value**
None

---

**[Function name] digitalRead**

**Overview**
Function to read out digital data from a pin

**Header**
AR_LIB_PORT.h, r_cg_macrodriver.h, r_cg_userdefine.h

**Declaration**
uint8_t digitalRead(uint8_t pin);

**Description**
The state of the pin specified by the argument is read out

**Argument**
- uint8_t pin : Number of the pin to be read out

**Return value**
uint8_t : Data that was read out (HIGH/LOW)

---

**[Function name] millis**

**Overview**
Function to obtain the elapsed time in millisecond units

**Header**
AR_LIB_TIME.h, r_cg_macrodriver.h, r_cg_userdefine.h

**Declaration**
uint32_t millis (void);

**Description**
Returns the time elapsed from startup, in millisecond units.

**Argument**
None

**Return value**
uint32_t : Elapsed time in millisecond units
5.6 Flowcharts

5.6.1 Initial setting function

Figure 5.1 and Figure 5.2 show flowcharts of the initial setting.

Figure 5.1 Initial setting function (1/2)

```
setup

Set segment A control pin
pinMode()
segPinA is set to OUTPUT.

Set segment B control pin
pinMode()
segPinB is set to OUTPUT.

Set segment C control pin
pinMode()
segPinC is set to OUTPUT.

Set segment D control pin
pinMode()
segPinD is set to OUTPUT.

Set segment E control pin
pinMode()
segPinE is set to OUTPUT.

Set segment F control pin
pinMode()
segPinF is set to OUTPUT.

Set segment G control pin
pinMode()
segPinG is set to OUTPUT.

Set segment DP control pin
pinMode()
segPinDP is set to OUTPUT.
```
Figure 5.2 Initial setting function (2/2)

- Set common 0 control pin `pinMode()`
  - `comPin0` is set to OUTPUT.

- Set common 1 control pin `pinMode()`
  - `comPin1` is set to OUTPUT.

- Set common 2 control pin `pinMode()`
  - `comPin2` is set to OUTPUT.

- Set common 3 control pin `pinMode()`
  - `comPin3` is set to OUTPUT.

- Set external switch control pin `pinMode()`
  - `ex_swPin` is set to INPUT with pull-up resistor.

- Set internal switch control pin `pinMode()`
  - `swPin` is set to INPUT.

return
### 5.6.2 Main Processing Function

Figure 5.3 to Figure 5.6 show flowcharts of the main processing function.

![Flowchart of the main processing function](image)

- **Figure 5.3** Main processing function (1/4)
  - **Operation**
    - **loop**
      - Read out elapsed time.
        - `millis()`
      - Bits 11 to 2 for elapsed time in ms are read out.
        - `time_work variable ← millis() & 0x0FFC`
      - If same as previous time (including when counting has stopped), no processing is performed.
        - `old_time variable ← time_work variable`
      - Extracted time
        - 0x00
        - 0x04
        - 0x08
        - 0x0C
      - **Extracted time**
        - **A**
        - **B**
        - **C**
      - Set common 3 pin.
        - `digitalWrite()`
      - `comPin3 ← LOW : Common 3 deselected`
      - Extract segment value for 10-s digit.
        - `seg_data variable ← SEG_TABLE array [10-s value]`
      - Obtained segment data is output to pin.
        - `com_sel variable ← comPin0 : 10-s digit is set as display digit.`
      - Set segment data.
        - `set_SEG()`
      - Set display digit.
Figure 5.4 Main processing function (2/4)

A

Set common 0 pin.  
digitalWrite ()  
comPin0 ← LOW : Common 0 deselected

Extract segment value  
for 1-s digit.  
seg_data variable ← SEG_TABLE array [1-s value] + 0x80

Set segment data.  
sel_SEG ()  
Obtained segment data is output to pin.

Set display digit.  
com_sel variable ← comPin1 : 1-s digit is set as display digit.

B

Set common 1 pin.  
digitalWrite ()  
comPin1 ← LOW: Common 1 deselected

Extract segment value  
for 100-ms digit.  
seg_data variable ← SEG_TABLE array [100-ms value]

Set segment data.  
sel_SEG ()  
Obtained segment data is output to pin.

Set display digit.  
com_sel variable ← comPin2 : 100-ms digit is set as display digit.
Figure 5.5 Main processing function (3/4)

- **C**
  - Set common 2 pin.
  - `digitalWrite()`
  - `comPin2 ← LOW`: Common 2 deselected

- **D**
  - Set display data for minute digit.
  - `seg_data variable ← min_data variable`
  - Obtained segment data is output to segment pin.

- **E**
  - Set segment data.
  - `set_SEG()`
  - `com_sel variable ← comPin3`: Separate LED is set as display digit.
  - Outputs data to specified pin.
  - `digitalWrite()`
  - `HIGH` is output to display digit pin.

- **F**
  - If counting is permitted (`time_mode = 1`) and 100 ms have elapsed (`precount1 variable = 25`), the variable is initialized.
  - `precount1 variable ← 0`: 100-ms counter is cleared.

  - If 1 second has elapsed (`precount2 variable = 10`), the variable is initialized.
  - `precount2 variable ← 0`: 1-s counter is cleared.

  - If 1 minute has elapsed (`sec_data variable = 60`), the variable is initialized.
  - `sec_data variable ← 0`: Second counter is cleared.
Figure 5.6  Main processing function (4/4)

- Change time_mode variable.
- Change time counting mode.
- Switch pressed?
  - Yes
  - Extract bits 2 and 1 of sw_data variable.
  - If bits 2 and 1 of sw_data variable are "10", processing for pressed switch is performed.
  - Switch pressed?
  - No
  - Switch checking timing?
    - Yes
    - Update switch state.
    - Check switch state.
    - Update second data.
  - Switch checking timing?
    - No
    - 9 seconds elapsed?
      - Yes
      - Initialize mini_data.
      - mini_data variable ← 0 : Minute counter is cleared.
  - 9 seconds elapsed?
    - No
    - If 20 ms have elapsed (prescale4sw variable = 5), switch state is checked.
  - Switch checked timing?
    - Yes
    - Initialize prescale4sw.
    - prescale4sw variable ← 0
    - The sw_data variable is left-shifted 1 bit, and the logical sum (logical product using negative logic) of the two switch states is set in the LSB (least significant bit).
    - Extract bits 2 and 1 of sw_data variable.
    - Change time_mode variable.
  - 9 seconds elapsed?
    - No
    - Initialize mini_data.
5.6.3 Function to set display data for 7-segment LEDs

Figure 5.7 and Figure 5.8 show flowcharts of the function to set display data for 7-segment LEDs.

```
set_SEG
  work_data variable ← argument (segment data)

Set display data.

Set segment A.
  pinMode()
  LSB of work_data variable is set to segPinA.

Update display data.

Set segment B.
  pinMode()
  LSB of work_data variable is set to segPinB.

Update display data.

Set segment C.
  pinMode()
  LSB of work_data variable is set to segPinC.

Update display data.

Set segment D.
  pinMode()
  LSB of work_data variable is set to segPinD.

Update display data.

Set segment E.
  pinMode()
  LSB of work_data variable is set to segPinE.

Update display data.
```

Figure 5.7 Function to set display data for Segment LEDs (1/2)
Figure 5.8 Function to set display data for Segment LEDs (2/2)
6. Sample Code
Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents
RL78/G14 User's Manual: Hardware (R01UH0186)
RL78 family User's Manual: Software (R01US0015)
RL78/G14 Fast Prototyping Board User's Manual (R20UT4573)
RL78 Family Arduino API Introduction Guide (R01AN5413)
(The latest versions can be downloaded from the Renesas Electronics website.)

Technical Update/Technical News
(The latest versions 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>July.10.20</td>
<td>—</td>
<td>—</td>
<td>First Edition</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. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity.
   Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on
   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. 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 the 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.

5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.

6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between \( V_{IL} \) (Max.) and \( V_{IH} \) (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between \( V_{IL} \) (Max.) and \( V_{IH} \) (Min.).

7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

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

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation 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 claims 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, drawings, charts, programs, algorithms, and 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 reverse engineer 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 any third parties arising from such alteration, modification, copying or reverse engineering.

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 or system is indicated below.
   - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

6. When using 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 dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

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. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea 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 any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

8. When using 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 dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

9. Unless specified ranges.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

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.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

Corporate Headquarters
TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information
For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/

Trademarks
Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

© 2020 Renesas Electronics Corporation. All rights reserved.