Frequency Meter

SLG47921V

This application note explains how to design a frequency meter in SLG47921V. Simulation waveforms generated by GTKWave software and Logic Analyzer in FPGA workshop can be used to verify the functionality of the design.

This document is supported by design files listed in the Reference section.

Contents

1. Introduction

This document shows how to design a frequency meter, as illustrated in Figure 1. The frequency meter supports test clock frequencies ranging from 2kHz to 100MHz. Two user-defined parameters are available, as described by the signal names listed below.

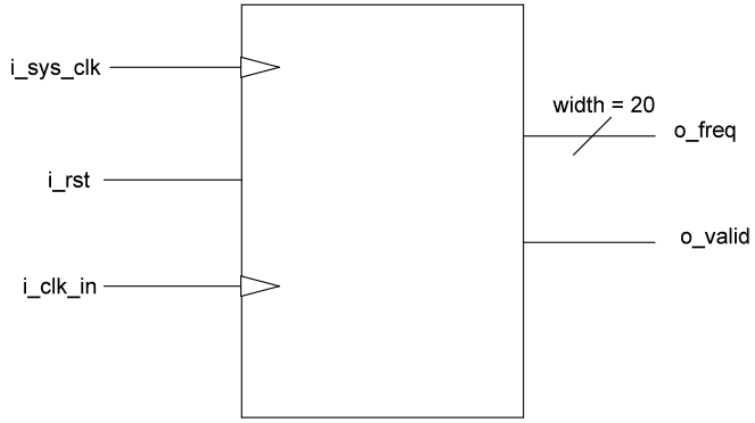

Figure 1. Frequency Meter

The following signal names are the parameters and pins that are used in the design.

- CLK_FREQ – input system clock frequency
- GATE_CYCS – gate clock cycles used to calculate the clock frequency
- i_sys_clk – input system clock
- i_rst – input reset signal, active high
- i_clk_in – input clock, the frequency to be metered
- o_freq – output clock frequency, unit is kHz
- o_valid – output frequency valid signal

The functional verification results are illustrated in the following waveform, where the output frequency (o_freq) is measured in kHz. In this simulation, the testbench generates input clock signals at 100 MHz, 50 MHz, 25 MHz, 10 MHz, 5 MHz, and 100 kHz to validate the frequency measurement accuracy across a wide range of operating conditions. As depicted in the waveform (see Figure 2), the o_freq output correctly captures the frequency of the input clock (to be tested), demonstrating the design's capability of handling both high-frequency and low-frequency signals effectively. In this document, we use FPGA Oscillator output 50MHz as the system clock, so the CLK_FREQ parameter here is 50000000.

Figure 2. Functional Waveform in GTKWave

2. Frequency Meter Description

In frequency meter design, two typical scenarios must be considered: the target clock may have either a lower or higher frequency than the system clock. For low-frequency target clocks, the period measurement method is commonly employed, wherein the period is measured using the system clock as a reference. Conversely, for high-frequency target clocks, the gated counting method is typically used, where the number of input clock pulses is counted within a fixed time window defined by the system clock.

In this design, dual-clock domain architecture is implemented to enable accurate frequency measurement across both cases. By isolating the system clock and the target clock into separate clock domains and synchronizing data across them appropriately, this approach allows reliable frequency estimation regardless of whether the target clock is slower or faster than the system clock.

Two critical parameters must be configured by the user: CLK_FREQ and GATE_CYCS. The CLK_FREQ parameter specifies the frequency of the input system clock and should be set according to the actual clock used in the target system. The GATE_CYCS parameter defines the gate time used for frequency measurement, expressed in terms of system clock cycles. This gate determines the duration over which the input (metered) clock is sampled.

By selecting appropriate gate times for different frequency ranges, the user can achieve a balance between measurement resolution and resource utilization. In general, increasing the gate time (i.e., using a larger value for GATE_CYCS) improves measurement accuracy, as it allows more input clock cycles to be counted. However, caution must be exercised when the frequency of the clock to be measured is significantly higher than the system clock frequency. In such cases, an excessively large gate time may lead to counter overflow due to insufficient counter width, potentially resulting in inaccurate measurements. As a practical guideline, setting GATE_CYCS to approximately one-thousandth of CLK_FREQ typically yields optimal performance.

3. Verilog Code

The Frequency Meter design is available for download (AN-FG-022 Frequency Meter.ffpga). It contains the complete frequency meter design with the SLG47921V device.

Shown below is the (*top*) module of frequency meter. It is available for download. (AN-FG-022 Frequency Meter.ffpga)

```
(* top *) module frequency_meter_top
(
 (* iopad_external_pin, clkbuf_inhibit *) input          i_sys_clk,
 (* iopad_external_pin, clkbuf_inhibit *) input          i_clk_in,

 (* iopad_external_pin *)  input                         i_rst_n,

 (* iopad_external_pin *)  output                        o_gpio_clk,
 (* iopad_external_pin *)  output                        o_gpio_clk_oe,

 (* iopad_external_pin *)  output reg                    o_trig,
 (* iopad_external_pin *)  output reg                    o_trig_oe,

 (* iopad_external_pin *)  input [1:0]                   i_pll_div,

 (* iopad_external_pin *)  output                        o_osc_ctrl_en,
 (* iopad_external_pin *)  output                        o_osc_ctrl_mode,

 (* iopad_external_pin *)  output reg [19:0]             o_freq,
 (* iopad_external_pin *)  output [19:0]                 o_freq_oe,

 (* iopad_external_pin *)  output                        o_postdiv0_ctrl_en,
 (* iopad_external_pin *)  output [2:0]                  o_postdiv0_ctrl_sel,

 (* iopad_external_pin *)  output                        pll1_en,
 (* iopad_external_pin *)  output [5:0]                  pll1_refdiv,
 (* iopad_external_pin *)  output [11:0]                 pll1_fbdiv,
 (* iopad_external_pin *)  output [2:0]                  pll1_postdiv0_out0,
 (* iopad_external_pin *)  output [2:0]                  pll1_postdiv1_out0,
 (* iopad_external_pin *)  output [1:0]                  pll1_ref_clk_sel,
 (* iopad_external_pin *)  output                        pll1_bypass
```

```
);
    localparam  IN_CLK_HZ = 50_000_000;
    localparam  TIME_250ms = 12_500_000;
    localparam   GATE_CYCS = 50_000;

    reg [1:0]    r_pll_div;
    reg [2:0]    r_rst;
    wire         sys_rst;
    reg [5:0]    pll_div_t;
    reg [23:0]  clk_cnt;
    wire          clk_valid;
    wire [19:0] freq_t;

    assign o_osc_ctrl_en = 1'b1;
    assign o_osc_ctrl_mode = 1'b1;
    assign o_postdiv0_ctrl_en = 1'b1;
    assign o_postdiv0_ctrl_sel = 3'b000;

    // Assigned signals
    assign pll1_en = 1'b1;
    assign pll1_refdiv = pll_div_t;                 // Equivalent value in decimal form 6'd1
    assign pll1_fbdiv = 12'b0000_0001_1000;  // Equivalent value in decimal form 12'd24
    assign pll1_postdiv0_out0 = 3'b110;         // Equivalent value in decimal form 3'd6
    assign pll1_postdiv1_out0 = 3'b010;         // Equivalent value in decimal form 3'd2
    assign pll1_bypass = 1'b0;
    assign pll1_ref_clk_sel = 2'b00;               // OSC
    assign o_freq_oe = 20'hFFFFF;

  // This always block synchronizes and inverts input i_reset signal
    always @(posedge i_sys_clk) begin
            r_rst[0] <= i_rst_n;
            r_rst[1] <= r_rst[0];
            r_rst[2] <= ~r_rst[1];
    end
    assign sys_rst = r_rst[2];
```

```
always @(posedge i_sys_clk) begin
  if(sys_rst)
        r_pll_div <= 2'd0;
  else
        r_pll_div <= i_pll_div;
end

always @(posedge i_sys_clk) begin
  if(sys_rst)
        pll_div_t <= 6'd1;
  else
        pll_div_t <= r_pll_div + 1;
end

frequency_meter #
(
        .CLK_FREQ    (IN_CLK_HZ     ),
        .GATE_CYCS (GATE_CYCS  )
)
uut
(
        .i_sys_clk       (i_sys_clk       ),
        .i_rst           (sys_rst         ),
        .i_clk_in        (i_clk_in        ),
        .o_freq          (freq_t          ),
        .o_valid         (clk_valid       )
);

assign o_gpio_clk = i_clk_in;
assign o_gpio_clk_oe = 1'b1;
assign o_trig_oe = 1'b1;

always @(posedge i_sys_clk) begin
  if(sys_rst)
        clk_cnt <= 24'd0;
  else if(clk_cnt == TIME_250ms - 1)
        clk_cnt <= 24'd0;
```

```
    else
        clk_cnt <= clk_cnt + 1;
  end

  always @(posedge i_sys_clk) begin
    if(sys_rst)
        o_trig <= 1'b0;
    else if(clk_cnt < TIME_250ms/2)
        o_trig <= 1'b1;
    else
        o_trig <= 1'b0;
  end

  always @(posedge i_sys_clk) begin
    if(sys_rst)
        o_freq <= 20'd0;
    else if(clk_valid)
        o_freq <= freq_t;
  end

endmodule
```

The trig here is used to trig the logic analyzer to read out the o_freq, and the gpio_clk output is a duplicate output for measure with a scope.

Note: The unit of o_freq is **kHz**

4. Floorplan: CLB Utilization

The resources that are used when designing the frequency meter can be seen under the Floorplan and the Resources Tab in the toolbar on the top of the window (Figure 3). The Floorplan highlights the CLB that are been used in this design. When the user clicks the CLB box, the software shows the wires (blue/yellow) connecting multiple CLB blocks.

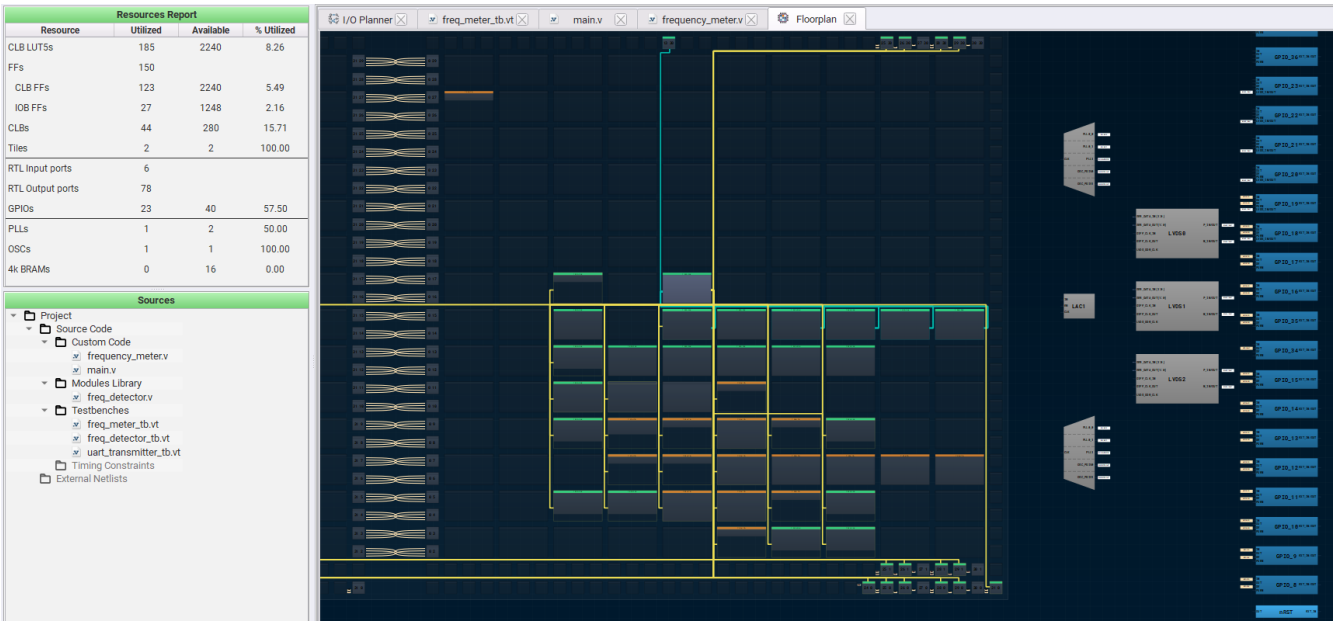

Figure 3. Distributed RAM CLB Utilization

5. Logic Analyzer: Frequency Being Measured

In this design, the i_pll_div signal is used to control the pll1_refdiv parameter, enabling the generation of different clock frequency via PLL1. The output of PPL1 serves as the input clock to be measured by the frequency meter. Additionally, the output of the frequency meter module, o_freq, is connected to gpio0~gpio19, allowing the measured frequency to be assessed externally. The trig signal can be used as a trigger source for the Logic Analyzer, facilitating the capture and observation of the frequency output (Figure 4, Figure 5, Figure 6)

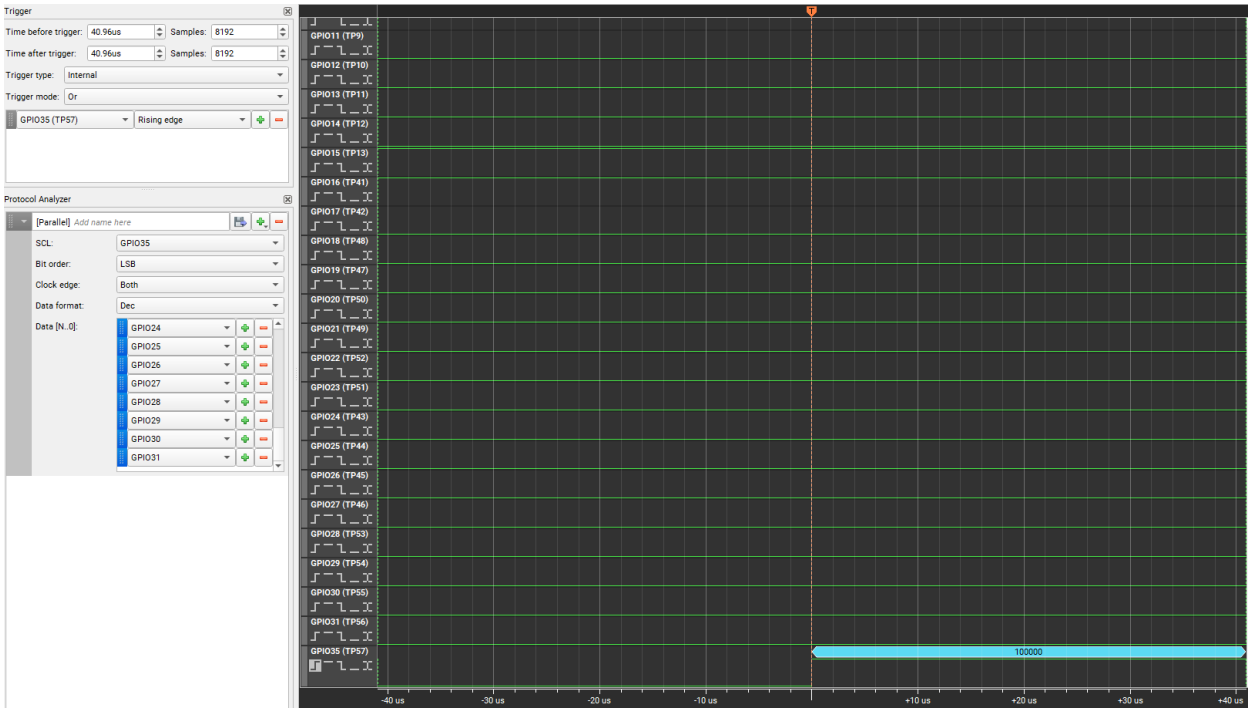

Figure 4. A frequency of 100 MHz is observed on the Logic Analyzer

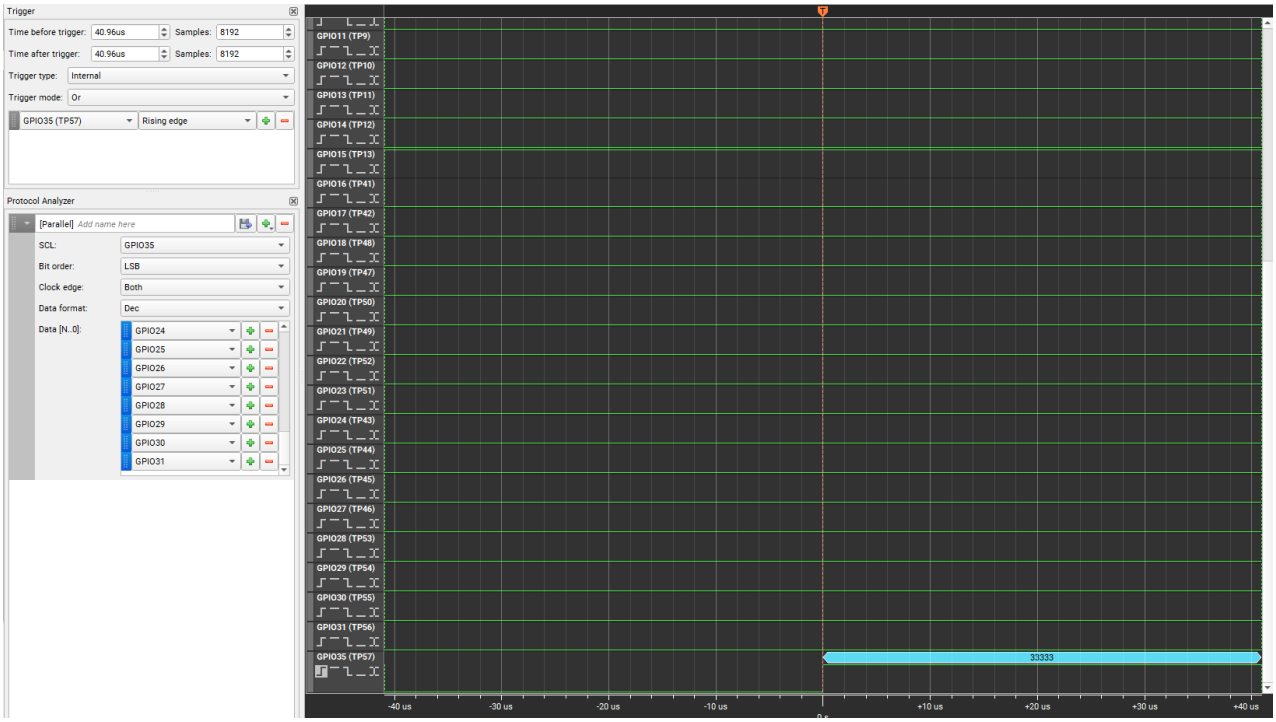

Figure 5. A frequency of 33 MHz is observed on the Logic Analyzer

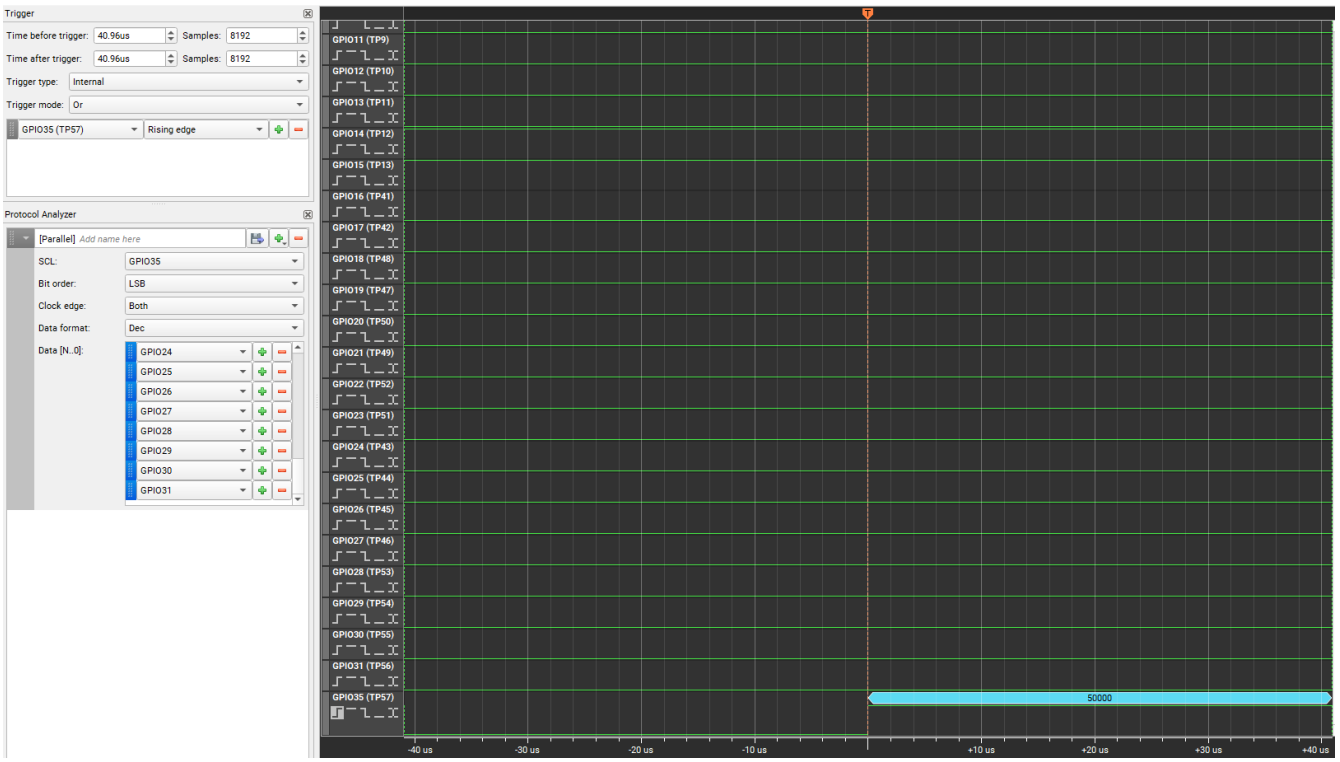

Figure 6. A frequency of 50 MHz is observed on the Logic Analyzer

6. Design Steps

1. Launch the latest version of the Go Configure Software Hub. Select SLG47921V device and the ForgeFPGA Workshop software will load.
2. Download the design example AN-FG-022 Frequency Meter.ffpga. If you are not familiar with the ForgeFPGA Workshop software, review the Frequency Meter application notes that cover the basic design steps.
3. Open the AN-FG-022 Frequency Meter.ffpga file after downloading.
4. Open the FPGA editor and review the Verilog code and the testbench code.
5. Open the IO planner tab on the FPGA editor and review the pin assignment.
6. Select the Synthesize button on the lower left side of the FPGA editor. Select the Generate Bitstream button on the lower left side of the FPGA editor. Check the Logger and Issues tabs to make sure that the bitstream was generated correctly.
7. Now click on the Simulate Testbench button at the upper top. The GTKWave will automatically open if there are no Syntax errors in the testbench. Check logger for errors.
8. In the GTKWave software, select the signals you want to view, and Click Insert on the left corner to insert the signals in the wave window. Once the desired signals are selected, click on Reload (Figure 2).
9. Click on the Floorplan tab and see the CLB utilization (Figure 3). Press the Ctrl and the mouse wheel to zoom-in. Confirm that the IOs selected in the IO Planner.
10. Connect the computer to the ForgeFPGA Deluxe development board, then click "Load from MCU". Wait for the Go Configure Software Hub to download the design to the board. Once the download is complete, open the Logic Analyzer. Assign different values (GND, VCC) to GPIO33 and GPIO34. In the Logic Analyzer, you will observe the clock frequency as measured by the frequency meter (see Figure 4, Figure 5, and Figure 6).

7. Conclusion

This application note shows how to design a frequency meter with SLG47921V, and the system clock frequency and gate times could be defined by users. This design file is available for download (AN-FG-022 Frequency Meter.ffpga). For more information, contact the ForgeFPGA Business Support Team.

8. Terms and Definitions

FPGA	Field Programmable Gate Array
FPGA Editor	Main FPGA design and simulation window
ForgeFPGA workshop	Main FPGA project window for debug and IO programming
Go Configure Software Hub	Main window for device selection
PLL	Phase Locked Loop
OSC	Oscillator
CLK MUX	The Clock Multiplexer

9. References

For related documents and software, visit: ForgeFPGA . Download our free ForgeFPGA™ Designer software [1] to open the. ffpga design files [2] and view the proposed circuit design.

[1] Go Configure Software Hub, Software Download and User Guide, Renesas Electronics

[2] AN-FG-022 Frequency Meter.ffpga, ForgeFPGA Design File

[3] ForgeFPGA SLG47920/21, Datasheet, Renesas Electronics

[4] ForgeFPGA Workshop user Guide, Renesas Electronics

10. Revision History

Revision	Date	Description
1.01	Jan 06, 2026	Changed Frequency Meter test clock frequency range and the simulation waveform
1.00	Nov 6, 2025	Initial release.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL

SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01 Apr 2025)

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-data version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

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