Keysight Clock Glitcher

DS1150A Clock Glitcher

KEYSIGHT

USER MANUAL

Notices

Trademark Acknowledgments

Manual Part Number

DS1150-90002

Edition

Edition 1, October 2024

Published by:
Keysight Technologies
1400 Fountain Grove Parkway
Santa Rosa, CA 95403

Warranty

Technology Licenses

U.S. Government Rights

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION notice until the indicated conditions are fully understood and met.

WARNING

A **WARNING** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Where to Find the Latest Information

Documentation is updated periodically. For the latest information about these products, including instrument software upgrades, application information, and product information, browse to one of the following URLs, according to the name of your product:

https://www.keysight.com/us/en/product/DS1150A/clock-glitcher.html

To receive the latest updates by email, subscribe to Keysight Email Updates at the following URL:

https://support.keysight.com

Information on preventing instrument damage can be found at:

https://www.keysight.com/find/PreventingInstrumentDamage

Is your product software up-to-date?

Periodically, Keysight releases software updates to fix known defects and incorporate product enhancements. To search for software updates for your product, go to the Keysight Technical Support website at:

https://www.keysight.com/find/techsupport

Product and Solution Cybersecurity

Keysight complies with multinational regulations for the cybersecurity of its own products and is committed to providing information to assist you in protecting your products and solutions from external cyber threats. For more information, see:

https://www.keysight.com/us/en/about/quality-and-security/security/product-and-solution-cyber-security.html

Keysight also recommends that you secure your IT environments using appropriate third-party tools. For instruments that run the Microsoft Windows operating system, Keysight concurs with Microsoft's recommendations for ensuring that the instrument is protected:

- Get the latest critical Windows updates
- For network-connected instruments, use an Internet firewall (in Keysight instruments, Windows Firewalls enabled by default)
- For network-connected instruments, use up-to-date antivirus and anti-spyware software

Responsible Disclosure Program

Keysight recommends that security researchers share the details of any suspected vulnerabilities across any asset owned, controlled, or operated by Keysight (or that would reasonably impact the security of Keysight and our users) using this form:

https://www.keysight.com/us/en/contact/responsible-disclosure-program.html

Report a Product Cybersecurity Issue

If you discover a cybersecurity issue that you suspect may involve Keysight's proprietary software, or third-party software supplied by Keysight as part of a product, or that may affect the operation of Keysight products, we encourage you to report it to us using this form:

https://www.keysight.com/us/en/about/quality-and-security/security/product-and-solution-cyber-security.html

Contents

What’s in the Box?

The box contains the Clock Glitcher and all accessories.

Box content checklist

Quantity[1]	Description		Identifier[2]
1	Clock Glitcher		CG
1	Banana plug cables — Black — Red		
2	Signal cable: SMB – SMB, 50 Ω, coaxial, 3 ft		SMB2SMB
1	Target clock connector cable: custom 2-pin connector		
1	Signal cable: BNC – SMB, 50 Ω, coaxial , 3 ft		BNC2SMB
1	USB-A – USB-C cable 3 ft		
1	Crypto Training Target test board (CG version)		
1	Lab power supply (Optional): Lab power supply to power the Clock Glitcher	PSU for CG	
	This “DS1150A Clock Glitcher User Manual”		

1. Quantity of items registered in the package
2. Identifier used in references in this document.

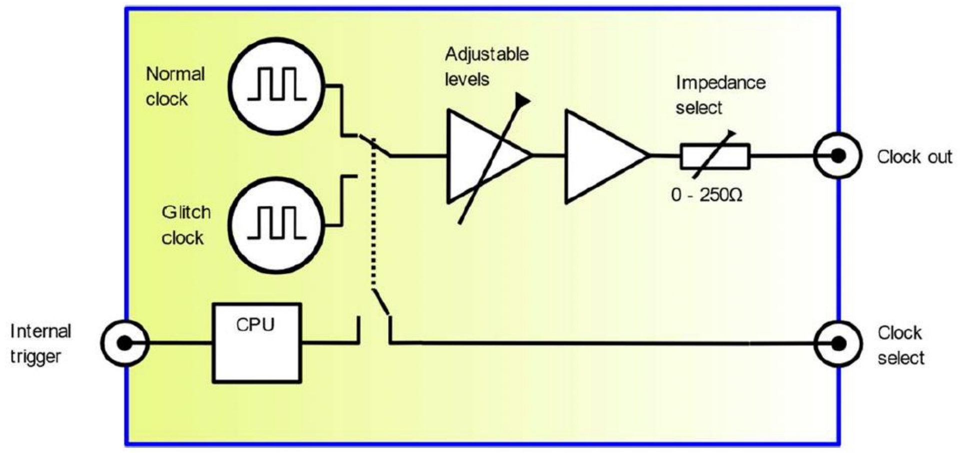
What It Does

The Clock Glitcher

The Clock Glitcher (CG) is a device with which the clock of a processor can be changed.

The CG has two internal clock generators, and these can produce a clock signal between 1KHz and 300MHz. Both generators can be configured independently. Via the clock select input, it's possible to switch between the normal clock and glitch clock.

Figure 1 **Functional overview of Clock Glitcher**



The high and low voltage levels of both clock signals can be changed to any value between -1V and +4V.

The glitch clock has three clock modes.

- Fixed frequency digital
- Random frequency digital
- Random frequency analog

The internal processor can also control the glitch clock pulse.

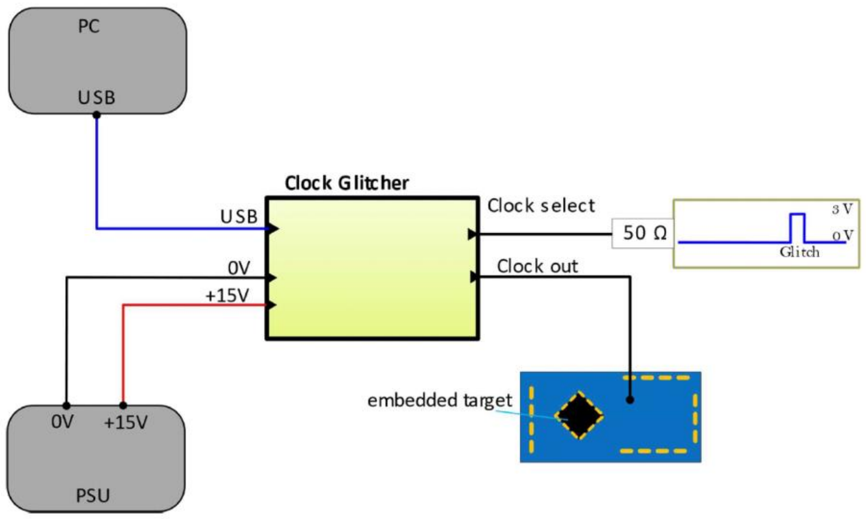
Basic setup

Connect the unit to a power supply at 15V with a minimum current of 1A.

For the adjustment and controlling the Clock Glitcher (CG) the unit is connected to a PC using a USB-C cable. The unit acts as a serial port. No baud rate setting is necessary.

The clock output is connected to the “clock in” pin of the processor and ground of the system. If a crystal or oscillator is present, disconnect these first.

Figure 2 Basic setup of Glitch Amplifier

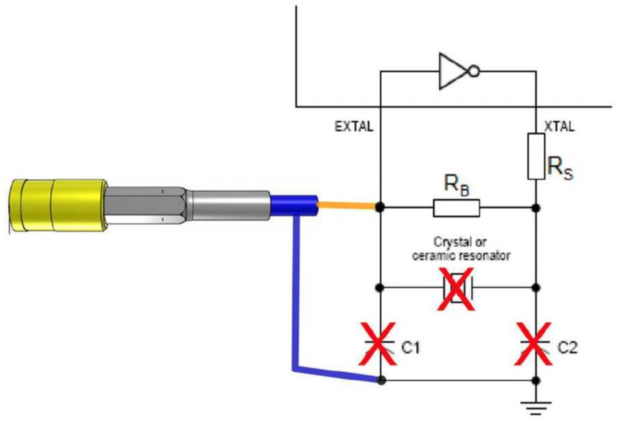


When “clock select” input is low the normal clock is on the clock output. When select input goes high the glitch clock is on the clock output. There is also a reference clock output that can be connected to an oscilloscope to see the actual clock signal.

Some remarks on connecting to a target

When connecting to a crystal circuit of the processor, make sure that the crystal and both series capacitors are removed from the circuitry. For the normal clock, do not exceed the voltage levels as mentioned in the datasheet of the processor. Experiment with the output impedance for the best result. 50 Ω – 250 Ω should work in most circumstances.

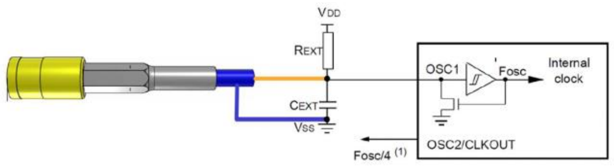
Figure 0-3 Connecting to a crystal circuit



In an RC configuration, do not remove the RC network, but connect the clock output over Cext (see figure 4). The clock output levels must be adjusted to the oscillator Schmitt trigger levels.

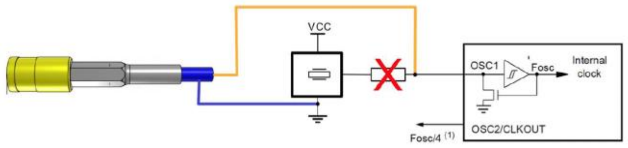
Experiment with the output impedance for better results. Impedances of 50 Ω – 250 Ω should work in most circumstances.

Figure 4 Connecting to a RC oscillator circuit



When the processor is connected to a crystal oscillator, remove the connection between oscillator and processor and connect the clock output to the processor.

Figure 5 Connection to crystal oscillator circuit

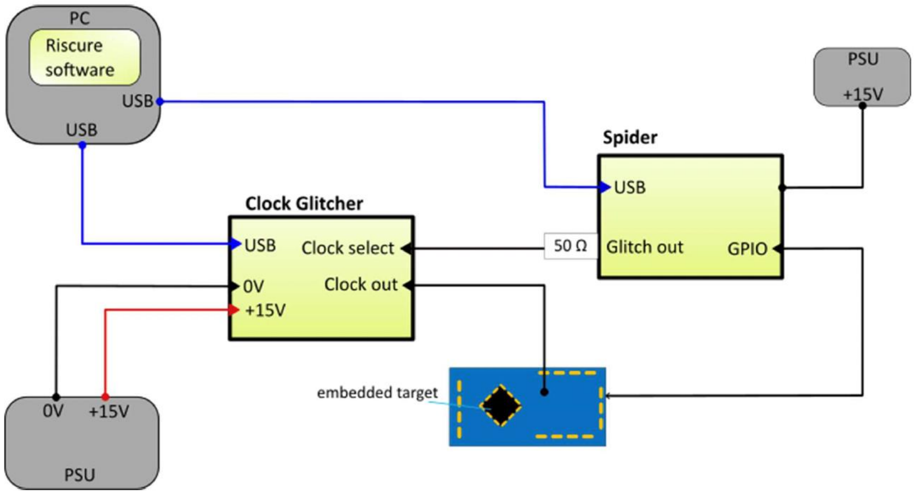


How to Build a Setup

For the best and adequate solution connect the Clock Glitcher to a Glitch Pattern

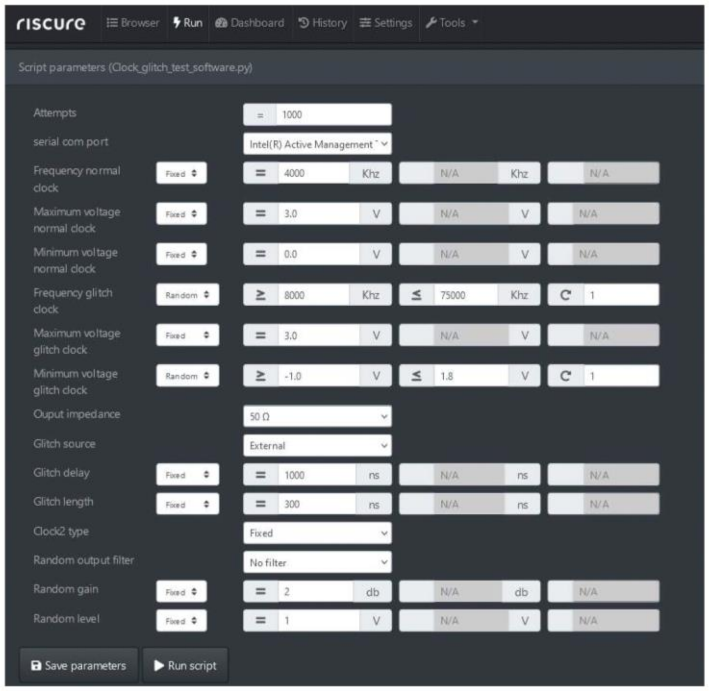
Generator and Keysight software. See diagram below.

Figure 6 **Glitch Pattern Generator setup**



Use a 50 Ω adapter in the glitch output for a proper signal on the clock select input.

Figure 7 **Clock Glitcher configuration page**



Script parameters (Clock_glitch_test_software.py)

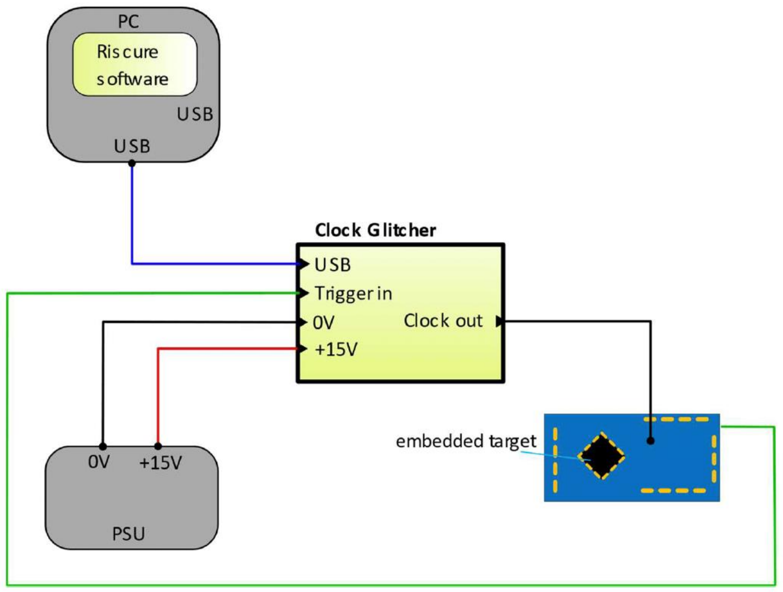
Attempts		= 1000		
serial com port		Intel(R) Active Management ˜		
Frequency normal clock	Fixed	= 4000 Khz	N/A Khz	N/A
Maximum voltage normal clock	Fixed	= 3.0 V	N/A V	N/A
Minimum voltage normal clock	Fixed	= 0.0 V	N/A V	N/A
Frequency glitch clock	Random	≥ 8000 Khz	≤ 75000 Khz	C 1
Maximum voltage glitch clock	Fixed	= 3.0 V	N/A V	N/A
Minimum voltage glitch clock	Random	≥ -1.0 V	≤ 1.8 V	C 1
Ouput impedance		50 Ω		
Glitch source		External		
Glitch delay	Fixed	= 1000 ns	N/A ns	N/A
Glitch length	Fixed	= 300 ns	N/A ns	N/A
Clock2 type		Fixed		
Random output filter		No filter		
Random gain	Fixed	= 2 db	N/A db	N/A
Random level	Fixed	= 1 V	N/A V	N/A

It is also possible to connect the trigger of the target directly to the CG.

This configuration has some disadvantages. The delay timing of glitch clock must be at least 300ns, and glitch length must be minimal 300ns.

The resolution of these timings is in 4ns steps. Software steps of 1ns can be given but are rounded to 4ns steps. When the total time of delay and glitch is above de 250µS the resolution is changed to 8ns or more.

Figure 8 Minimal setup



Software

The unit is connected through a USB serial port. There is no need to configure baud rate number of bits and stop bits. All these parameters are not used. For the maximum transfer rate they are not to be programmed.

There are two ways to communicate to the unit.

– A block transfer of all the data.

– Single line programming.

A block transfer looks like:

```
[2000, 3000, 0, 10000, 3000, -1000, 1, 0, 400, 400, 0, 0, 0,
0, 0, 0]<LF>
```

A block always starts with a '[' bracket and ends with a ']' bracket.

– Between the variables a comma ',' and a space ' ' character must be used.

– All numbers are integers. Negative numbers must be written with a minus sign.

– The line must always be closed with <LF> or <CR> character.

– Values are 0x0D of 0x0A.

– Block must be in a single transmission.

A single line transfer looks like:

```
FRE1->2000<LF>
```

– All numbers are integers. Negative numbers must be written with a minus sign.

– The line must always be closed with <LF> or <CR> character. Values are 0x0D or 0x0A.

Both transfers can work at the same time, but a block transfer is preferred because all the configurations are done in the correct order and rules.

When only in single line transfer there are some rules in sending the correct sequence of lines. Otherwise, the programming does not have any effect.

Data Transfer

All the parameters a user can send to the CG are in the table below.

```
//Parameter structure to inspector
typedef struct {
    /* data */
    int32_t normal_clk;             // Normal clock frequency in Khz.
                                    // 0 - 500.000 -> 0 .. 500Mhz
    int32_t normal_clk_top_level;   // High voltage of normal clock.
                                    // -1000 - 4000 -> -1V .. 4V
    int32_t normal_clk_low_level;   // Low voltage of normal clock.
                                    // -1000 - 4000 -> -1V .. 4V
    int32_t glitch_clk;             // Glitch clock frequencyin Khz.
                                    // 0 - 500.000 -> 0 .. 500Mhz
    int32_t glitch_clk_high_level;  // High voltage of glitch     clock.
                                    // -1000 - 4000 -> -1V .. 4V
    int32_t glitch_clk_low_level;   // Low voltage of glitch clock.
                                    // -1000 - 4000 -> -1V .. 4V
    int32_t output_impedance;       // Output impedance
                                    // 0 = 0Ohm
                                    // 1 = 50Ohm
                                    // 2 = 200Ohm
                                    // 3 = 250Ohm
    int32_t glitch_source;          // Who switches between normal
                                    or glitch clock.
                                    // 0 = External
                                    // 1 =internal
    int32_t glitch_delay;           // When internal switch this parameter
                                    // is the delay value of the glitch
    int32_t glitch_length;          // When internal switch this parameter
                                    // is the length value of the glitch
    int32_t glitch_clock_type;      // 0 = Frequency clock 2 active
                                    // 1 = Random digital clock signal
                                    // 3 = analog random noise on clock 2
    int32_t random_filter;          // 0 = No filteris applied to
                                    random signal
                                    // 1 = 1 - 10Mhz filter
                                    // 2 = 10 - 30Mhz
    int32_t random_level;           // Base level of random signal.
                                    // -1000 - 4000 -> -1V .. 4V
    int32_t random_gain;            // Random signal amplitude/gain.
                                    // -15000 - 15000 -> -15dB .. +15dB
    int32_t spare1;                 // Spare parameter for future use
    int32_t spare2;                 // Spare parameter for future use
} CG_data_t;
```

`int32_t normal_clk;`

This is the frequency setting for the normal clock. This parameter is in KHz. When the value is 0 (zero) then this clock output is held to the low voltage settings when this clock must be active.

When set to 1 or higher the clock out is starts at a frequency of 1KHz. This can be up to 500MHz. This has a value of 500000. The software in the Clock Glitcher limits this value to 500.000KHz. Frequencies above 300MHz are not within the specifications of the Clock Glitcher.

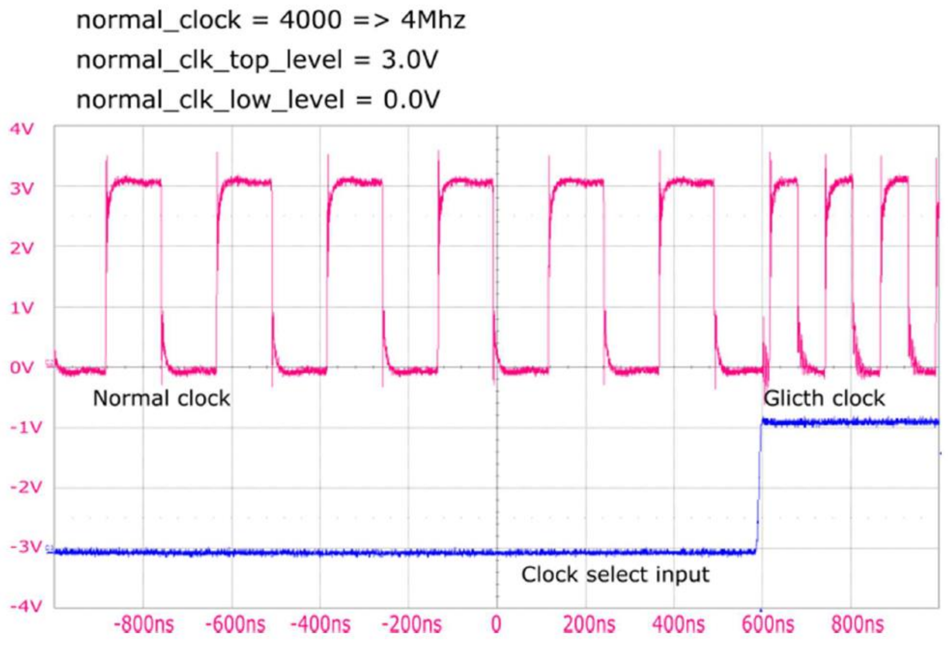
Although you can fill in every frequency, not all will be set at the exact value. Especially the higher frequencies have some offset. See Table 1.

`int32_t normal_clk_top_level;`

`int32_t normal_clk_low_level;`

These are the voltage levels of normal clocks. The voltage levels are specified in millivolts (mV), with 1000 being equivalent to 1.0 volt (V). Clocks levels can be set between -1.0V and 4.0V. These values are limited to these min and max voltages. Because there is no check that the “top_level” must be greater than “low_level”. Making “low_level” bigger than “top_level” inverts the clock.

Figure 9 Trace view normal clock



```
int32_t glitch_clk;
```

These are the frequency settings for the glitch clock. This parameter is in KHz. When the value is 0 (zero) then this clock output is held to the low voltage settings when this clock must be active.

When set to 1 or higher the clock out starts at a frequency of 1KHz. This can be up to 500MHz. This has a value of 500000. The software in the Clock Glitcher limits this value to 500.000KHz. Frequencies above 300MHz are not within the specifications of the Clock Glitcher.

Although you can fill in every frequency not all will be set at the exact value.

Especially the higher frequencies have some offset. See table 1.

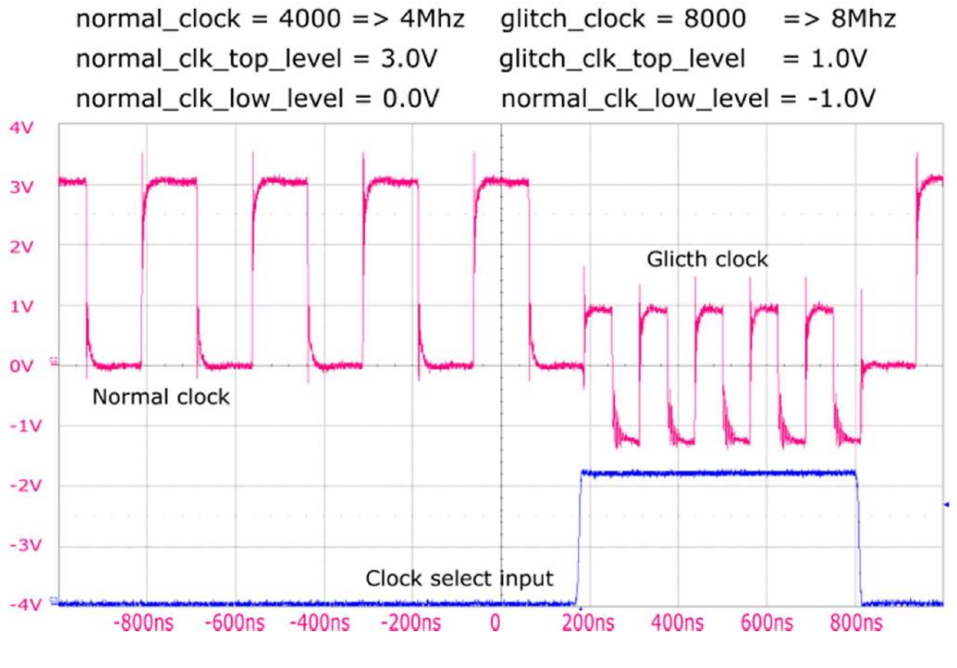
Because the glitch clock can also be a random clock signal (analog or digital) these settings are only active when the variable "glitch_clock_type" is zero.

```
int32_t glitch_clk_high_level;
```

```
int32_t glitch_clk_low_level;
```

These are the voltage levels of glitch clock. The voltage levels are specified in millivolts (mV), with 1000 being equivalent to 1.0 volt (V). Clocks levels can be set between -1.0V and 4.0V. These values are limited to these min and max voltages. Because there is no check that "top_level" must be greater than "low_level". Making "low_level" bigger than "top_level" inverts the clock.

Figure 10 Trace view glitch clock



It is possible to put a glitch in the normal clock.

Figure 11 Glitch in top level

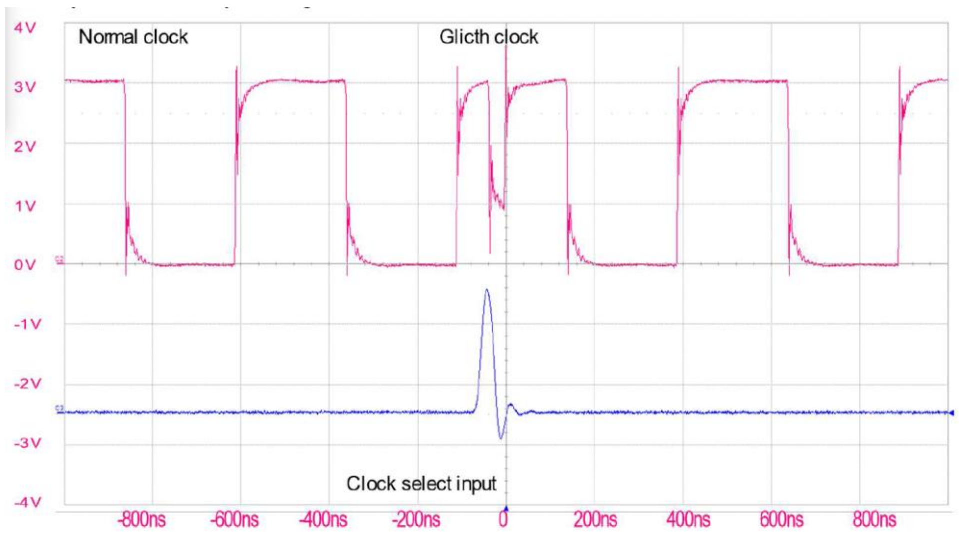
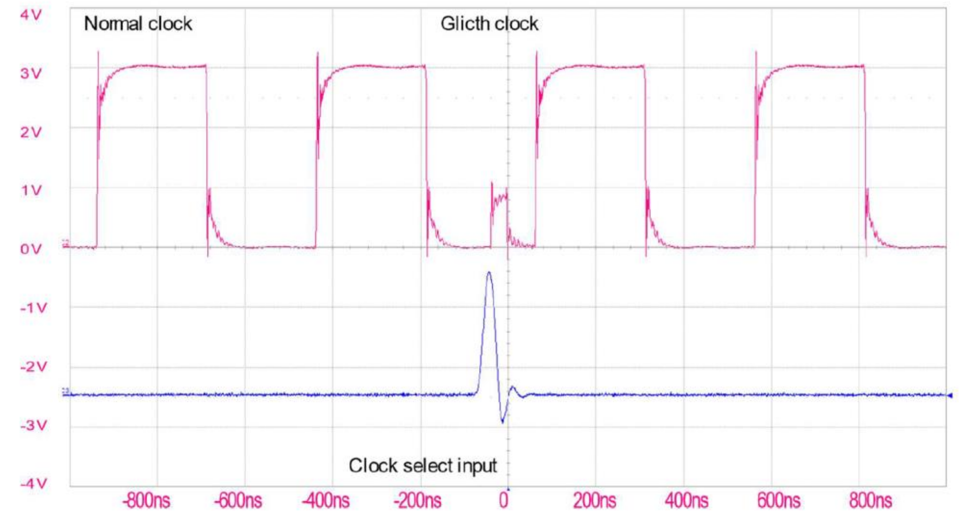


Figure 12 Glitch in low level



```
int32_t output_impedance;
```

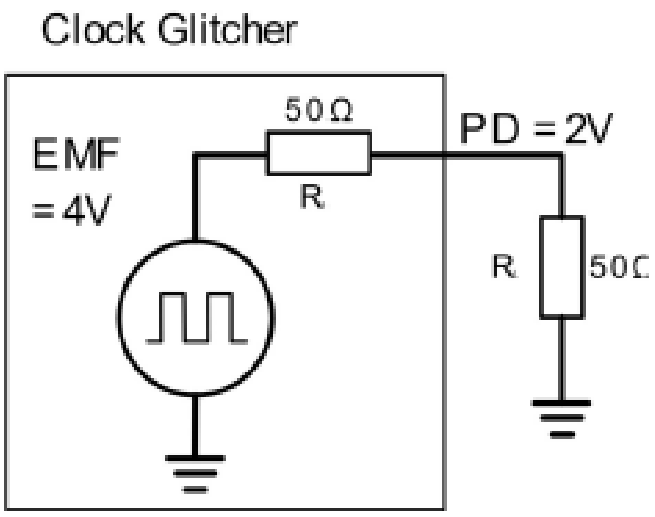
This parameter controls the output impedance of the clock output. There are three impedances possible.

Value	Impedance
0	0Ω
1	50Ω
2	200Ω
3	250Ω (50+200)

Remark

The real output levels "top" and "low" are dependent on this impedance. When choosing 50 **Ω** and the other side is also terminated with 50 **Ω** the output voltage is divided by two. When on 200 **Ω** this would be even more.

Figure 13 Output signal impedance behavior



```
int32_t glitch_source;
```

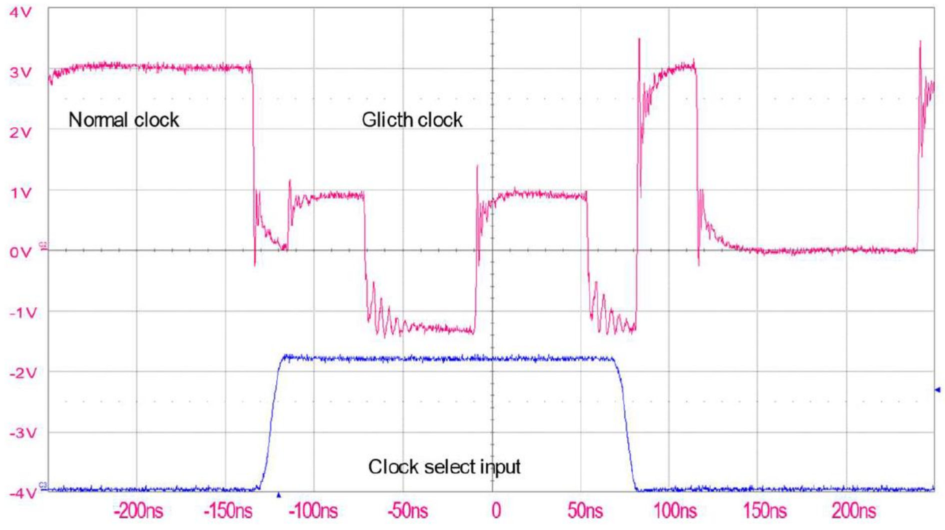
The switching between normal_clock and glitch_clock can be controlled by an external source like Keysight's Glitch Pattern Generator unit or by the internal processor.

Value	Impedance
0	External
1	Internal

When external is selected, the clock output changes when clock select input values changes. When low normal clock is on the clock output and when high the glitch clock or random clock is on the output. The propagation delay of this input is 20nS.

This input is a digital type of input.

Figure 14 External clock switch



When glitch_source variable is set to 1 the Clock Glitcher unit switches over to an internal clock switch sequence. When internal is selected the internal processor uses the values of “Glitch_delay” and “Glitch_length” to control glitch clock.

```
int32_t glitch_delay;
```

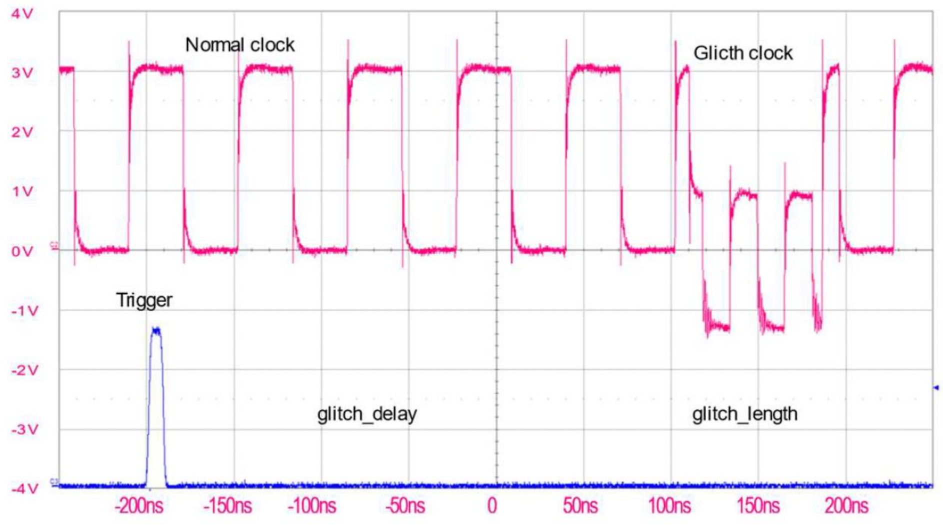
```
int32_t glitch_length;
```

When glitch_source is set to 1, the glitch sequence is started on the rising edge of the internal trigger input. The delay starts, when glitch_delay is finished the glitch clock becomes active on de output and the glitch_length timer is started. When this timer is finished, the normal clock is switch back on the output.

Due to the limitations of the internal processor hardware the minimum duration of delay and length must be at least 300ns. The resolution is in 4ns steps. When delay and length together are longer then 250us resolution decreases to 8ns.

Values are in ns resolution.

Figure 15 Internal glitch activ.



```
int32_t glitch_clock_type;
```

It's possible to change the type of glitch clock. There are three possibilities.

Value	Output type
0	Stable frequency glitch clock active
1	Random digital clock signal
2	Random analog noise signal

Figure 16 Stable glitch clock

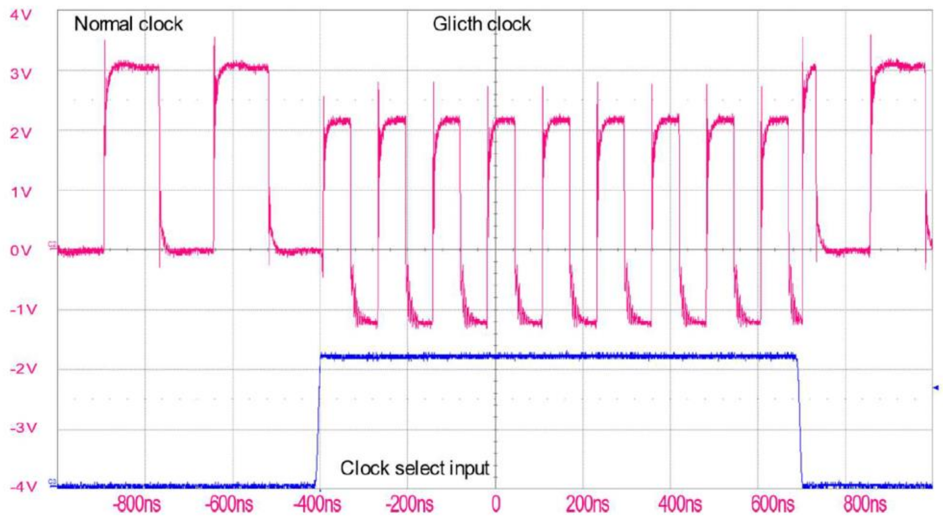


Figure 17 Random digital glitch clock

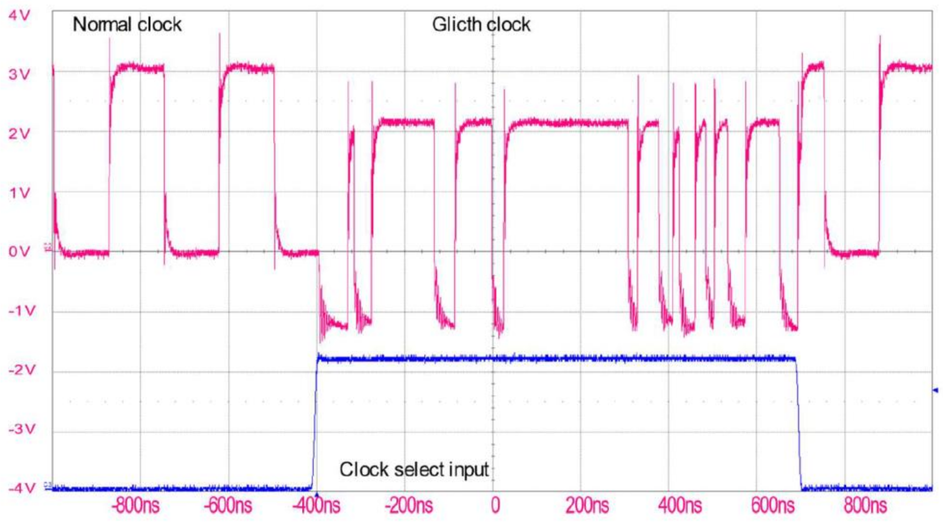
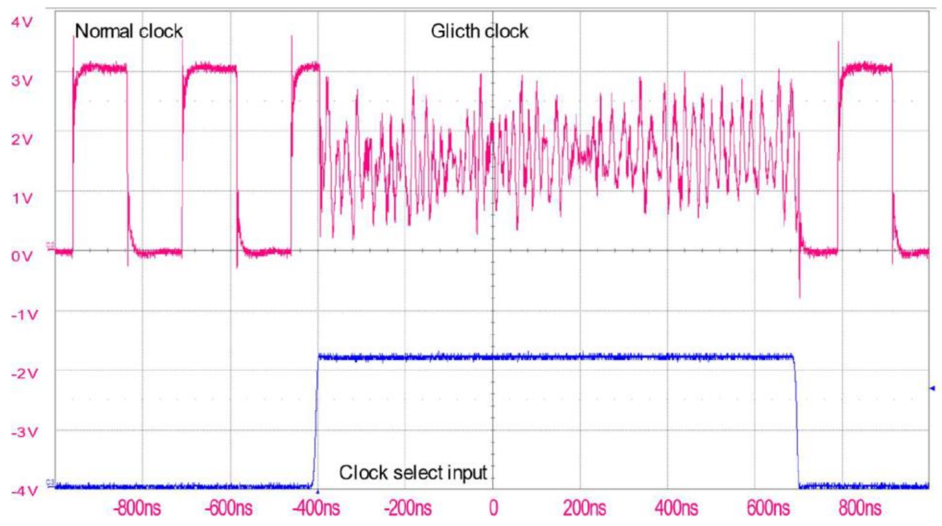


Figure 18 Random analog glitch clock



```
int32_t random_filter;
```

The random generated signal runs from 0Hz to 500MHz. It’s possible to add a frequency filter so only a small part of the frequencies can pass through.

This filter works for the random digital and random analog glitch clock signals.

Value	Frequency range
0	No filter is applied to random signal
1	1 MHz – 10MHz
2	10MHz – 30MHz
3	30MHz – 80MHz

Figure 19 Spectrum random analog with no filter

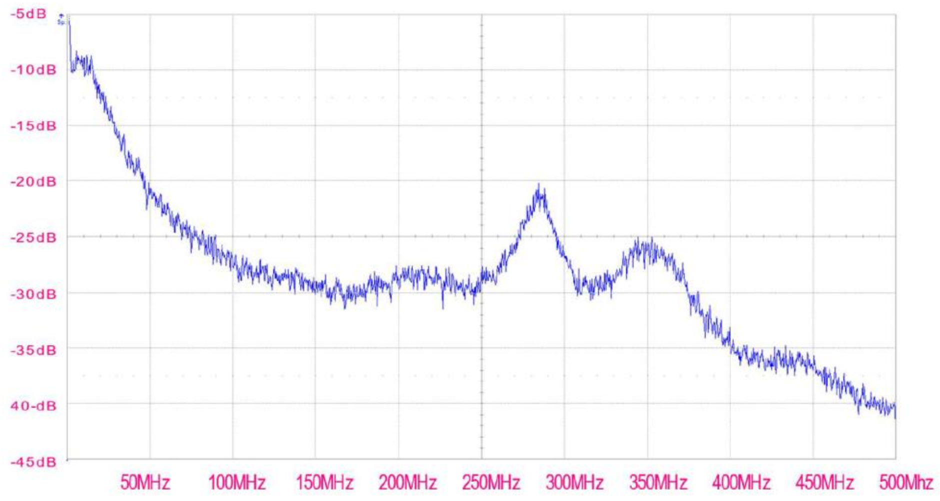


Figure 20 Spectrum random analog with 1-10MHz filter

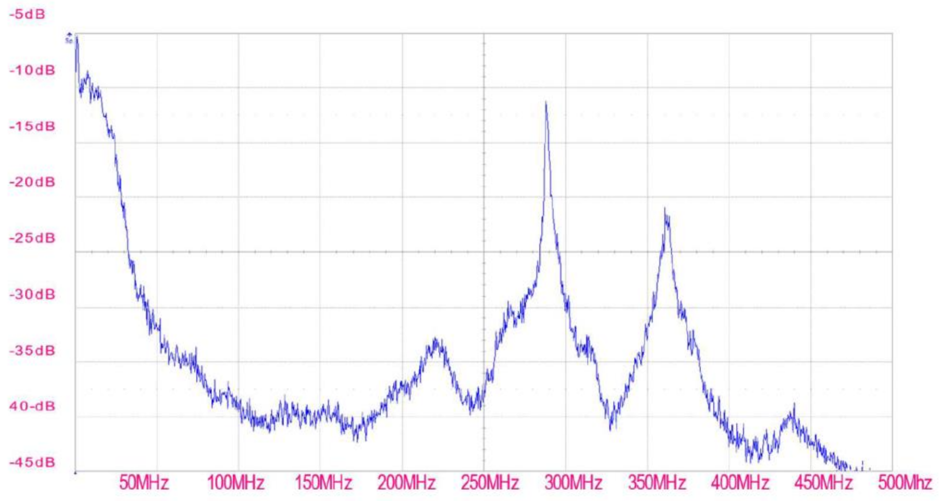


Figure 21 Spectrum random analog with 10-30MHz filter

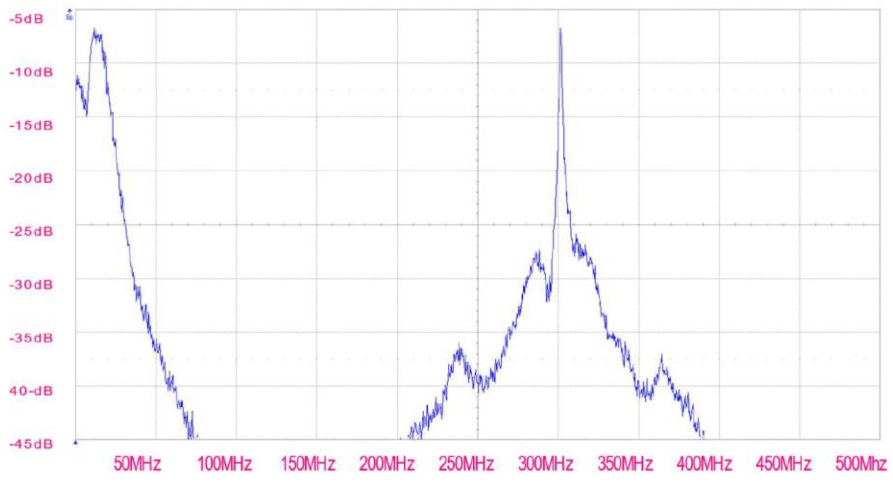
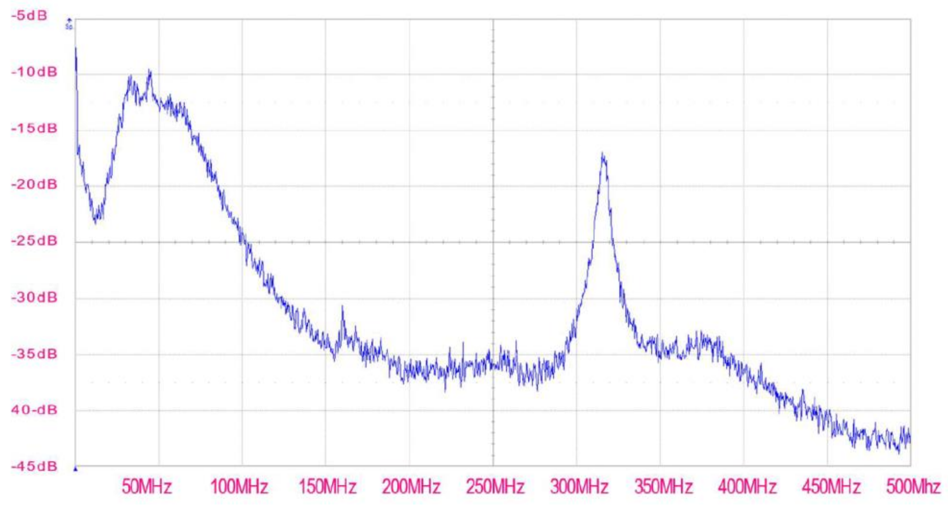


Figure 22 Spectrum random analog with 30-80MHz filter



```
int32_t random_level;
```

The base level of the analog random signal can be changed with this parameter. The level is notated in mV steps and can be controlled between -1000 - +4000 steps that translates -1.0V .. 4.0V.

Figure 23 Random analog level at -1.0V

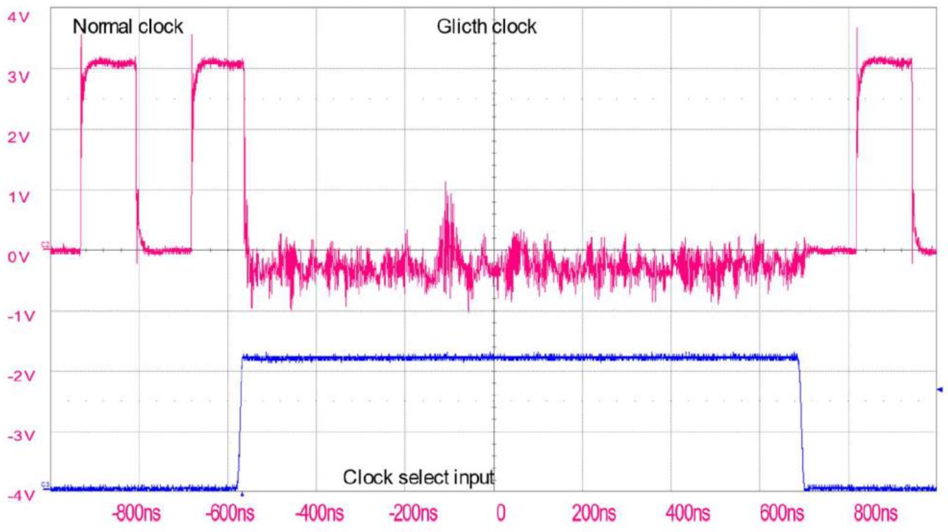
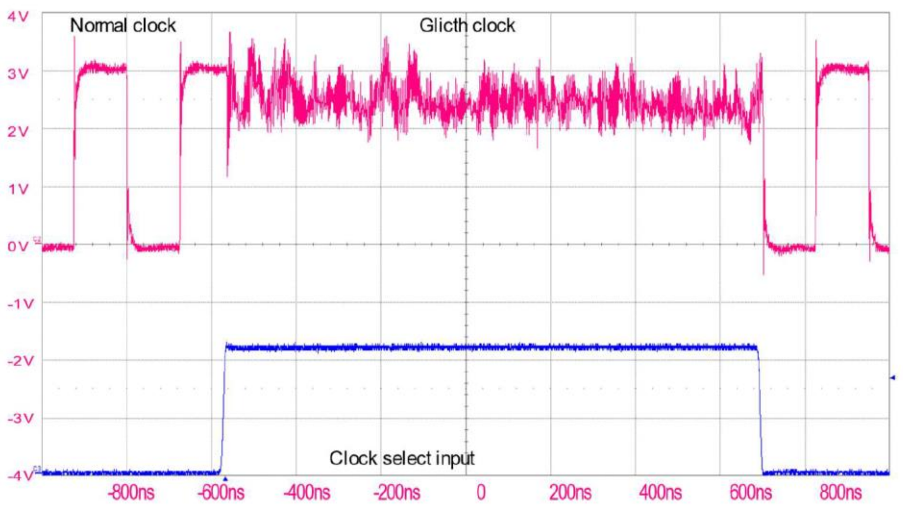
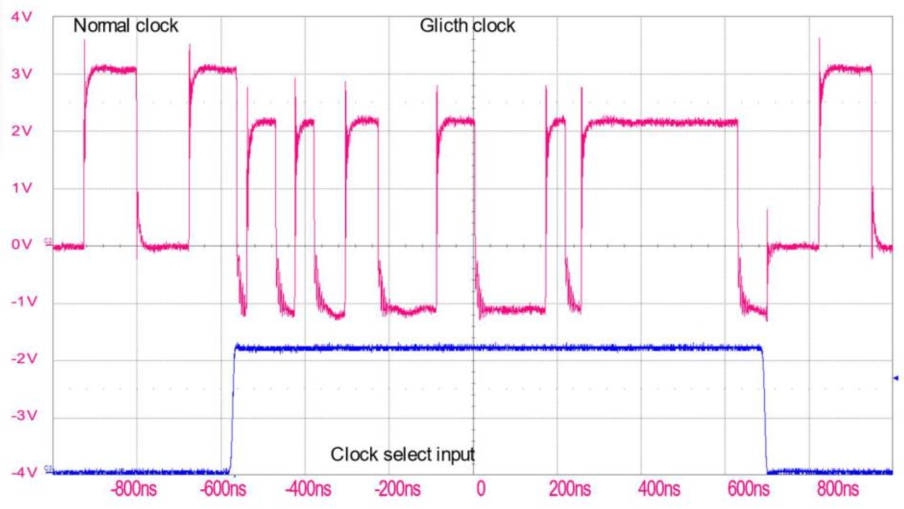


Figure 24 Random analog level at 3.0V



When in digital random mode, this value is controlled/overruled by the internal processor for optimal analog to digital conversion. When the unit is in digital random mode the digital clock level is adjusted with glitch_clock_top_level and glitch_clock_low_level values.

Figure 25 Random digital clock with level +2.0V and -1.0V



```
int32_t random_gain;
```

The amplitude of the random signal can be controlled by this parameter.

The setting is between -15000 - +15000 -> -15dB .. +15dB.

In digital random mode this value is controlled/overruled by the internal processor optimal adjustment for analog to digital conversion.

Figure 26 Random analog at a gain of -15dB

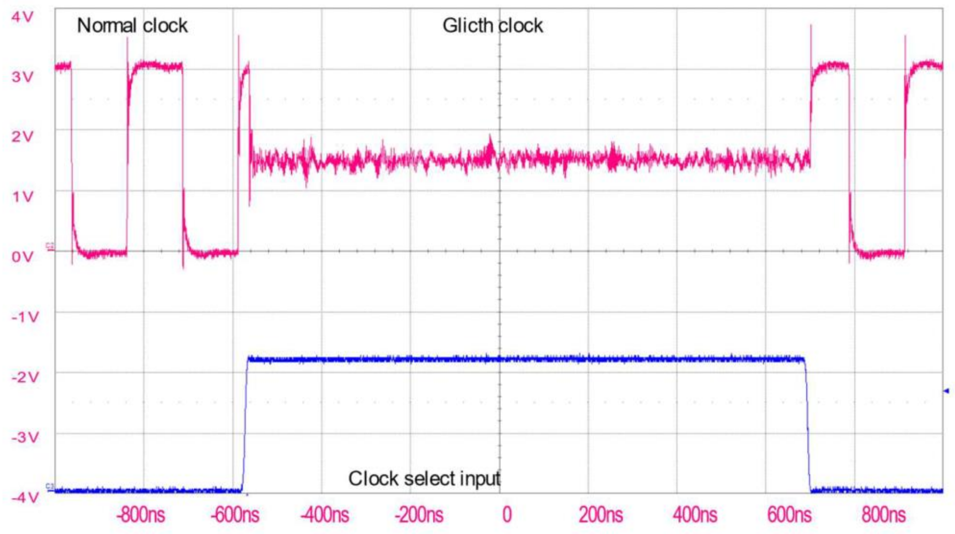
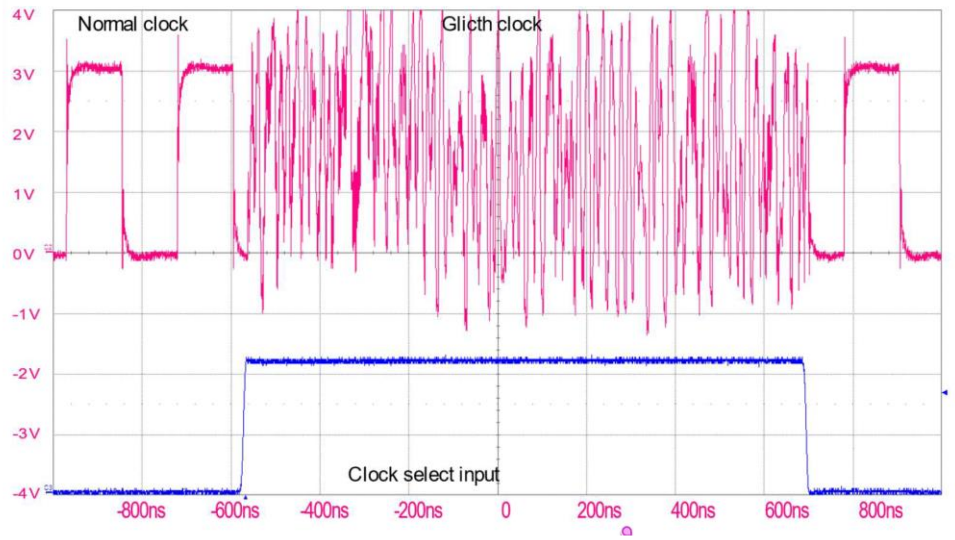


Figure 27 Random analog at a gain of +15dB



```
int32_t spare1;
```

```
int32_t spare2;
```

These are two spare values for future use.

Block Transfer

The following Python script explains how to send a block to the Clock Glitcher.

```
# Init the serial port to the Clock Glitcher
# And clear/initialiase the structure for the CG
# Send init block to the Clock Glitcher
class CGDevice:

   def __init__(self) -> object:
      # find USB serial port
      self.portCG = None
      self.ports_CG = serial.tools.list_ports.grep("USB Serial
         Device"\
         , include_links=False)
      for self.port_CG in sorted(self.ports_CG):
      print("{}:".format(self.port_CG))

      # init serial port
      self.comport_CG = str(self.port_CG)
      port_string = self.comport_CG[:5]
      print("{}:".format(port_string))
      # init serial port
      self.CG_com_port = serial.Serial(port_string)
      self.CG_com_port.port = port_string
      self.CG_com_port.close()
      self.CG_com_port.open()
      # Init CG data
      self.normal_clk = 0
      self.normal_clk_top_level = 0
      self.normal_clk_low_level = 0
      self.glitch_clk = 0
      self.glitch_clk_top_level = 0
      self.glitch_clk_low_level = 0
      self.output_impedance = 0
      self.glitch_source = 0
      self.glitch_delay = 0
      self.glitch_length = 0
      self.glitch_clk_type = 0
      self.Random_filter = 0
      self.Random_level = 0
      self.Random_gain = 0
      self.Spare1 = 0
      self.Spare2 = 0
      # send data block
      self.send_block()
```

```
# This function sends CG data block
def send_block(self) -> None:
    print("Com send block")
    self.CG_com_port.write(b"[" + str(self.normal_clk).encode())
    self.CG_com_port.write(b", " +
        str(self.normal_clk_top_level).encode())
    self.CG_com_port.write(b", " +
    str(self.normal_clk_low_level).encode())
    self.CG_com_port.write(b", " + str(self.glitch_clk).encode())
    self.CG_com_port.write(b", " +
    str(self.glitch_clk_top_level).encode())
    self.CG_com_port.write(b", " +
    str(self.glitch_clk_low_level).encode())
    self.CG_com_port.write(b", " +
    str(self.Output_impedance).encode())
    self.CG_com_port.write(b", " + str(self.Glitch_source).encode())
    self.CG_com_port.write(b", " + str(self.glitch_delay).encode())
    self.CG_com_port.write(b", " + str(self.glitch_length).encode())
    self.CG_com_port.write(b", " +
        str(self.Clock2_output_type).encode())
    self.CG_com_port.write(b", " + str(self.Random_filter).encode())
    self.CG_com_port.write(b", " + str(self.Random_level).encode())
    self.CG_com_port.write(b", " + str(self.Random_gain).encode())
    self.CG_com_port.write(b", " + str(self.Spare1).encode())
    self.CG_com_port.write(b", " + str(self.Spare2).encode() + b']'
        + b'\x0d' + b'\x0a')
```

Single Line Transfer

It’s also possible to send single commands to the Clock Glitcher.

Variable	Command	Complete string	
`int32_t normal_clk;`	`NCLK->`	`NCLK->4000<LF>`	Set normal clock on 4.0MHz
`int32_t normal_clk_top_level;`	`NTLV->`	`NTLV->3000<LF>`	Set high level normal clk to 3.0V
`int32_t normal_clk_low_level;`	`NLLV->`	`NLLV->-1000<LF>`	Set low level normal clk to -1.0V
`int32_t glitch_clk;`	`GCLK->`	`GCLK->8000<LF>`	Set glitch clock on 4.0MHz
`int32_t glitch_clk_high_level;`	`GTLV->`	`GTLV->2000<LF>`	Set high level glitch clk to 3.0V
`int32_t glitch_clk_low_level;`	`GLLV->`	`GLLV-><LF>`	Set low level glitch clk to -1.0V
`int32_t output_impedance;`	`GLOI->`	`GLOI->1<LF>`	Set output impedance to 50Ω
`int32_t glitch_source`	`CGGL->`	`CGGL->0<LF>`	Clock select input active
`int32_t glitch_delay;`	`CGDL->`	`CGDL->500<LF>`	Set glitch delay to 500ns
`int32_t glitch_length;`	`CGLE->`	`CGLE->400<LF>`	Set glitch length to 400ns
`int32_t glitch_clock_type;`	`GLTY->`	`GLTY->1<LF>`	Set glitch clock type random digital
`int32_t random_filter;`	`RAFI->`	`RAFI->0<LF>`	Set no filter in random signal

`int32_t random_level;`		`RALE->`	`RALE->0<LF>`	Set random analog level to 0V
`int32_t`	`random_gain;`	`RAGA->`	`RAGA->-5500<LF>`	Set random gain to -5.5 dB
`int32_t`	`spare1;`			
`int32_t`	`spare2;`			

A transmission in Python would look like:

```
CG_target.write(str.encode('NCL1->'))

CG_target.write(str(p['normal_clk']).encode())

CG_target.write(b'\x0d' + b'\x0a')
```

Setting Up the Power Supply

To power the Clock Glitcher (CG):

Use a lab power supply that is capable of 15VDC output and 1A initial peak current.

Connecting the power supply

Make sure the power supply is turned off.

Connect the black and red banana cables from the PSU to the CG.

Make sure the red cable is matching red connectors, and the black cable is matching black connectors.

Power supply settings

Make sure output is at 15VDC and the maximum current is 1A.

If there are overload protections and/or limits on the power supply enable them. Overload or limit:

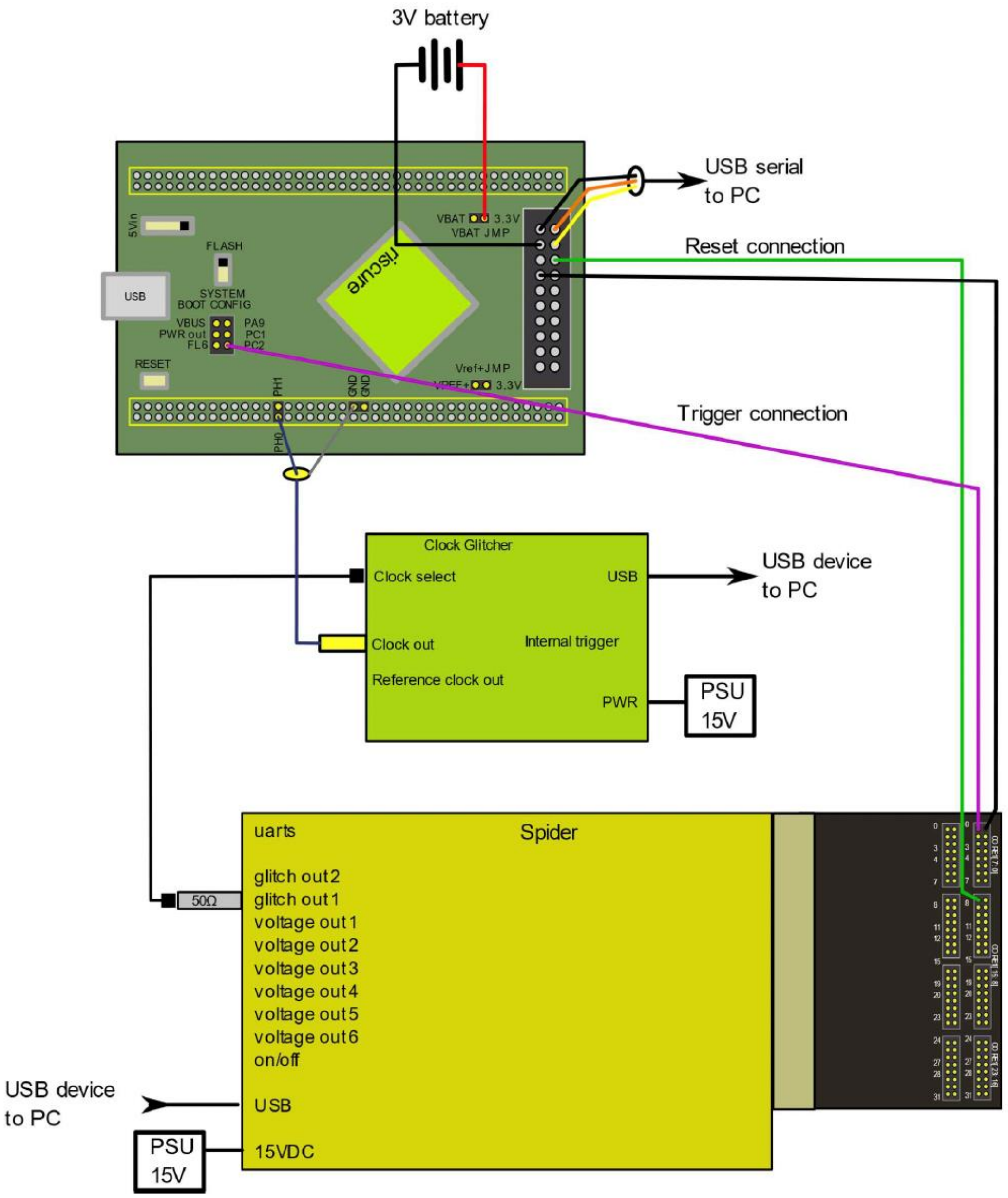
- Voltage: 24VDC
- Current: 1.5A (only on power up).

Now enable the power supply to power the CG.

Setup the CG with a Crypto Training Target

This diagram shows a setup to use a Clock Glitcher with a Crypto Training target.

Figure 28 Setup for Clock Glitcher and Crypto Training Target



The settings for this setup

Normal clock frequency	4000kHz	Fixed
Maximum voltage normal clock	3.0V	Fixed
Minimum voltage normal clock	0.0V	Fixed
Glitch clock frequency	4000kHz – 80000Khz	Random itter 1
Maximum voltage normal clock	3.0V	Fixed
Minimum voltage normal clock	0.0V	Fixed
Glitch delay	6250ns – 56250ns	Random itter 1
Glitch length	125ns – 3125ns	Random itter 1

Powerup State Clock Glitcher

At power up of the Clock Glitcher normal en glitch clock are switched off (set to 0). All levels are set to 0.

Output impedance is 0 Ω.

Random clock is not active.

No filter is selected.

Help and Troubleshooting

Still have questions?

Visit the Keysight Support Portal: http://support.keysight.com

Technical Specifications

Operational conditions

Room temperature 20 – 30 °C (68 – 86 °F).

WARNING The case of Clock Glitcher can become warm (Max 50 °C). This is normal.

NOTE Maintain stable environmental conditions (temperature, humidity, airflow etc.) in order to reliably repeat tests and compare test results.

Power supply input

- Clock Glitcher DC supply 15V 0.75A
- Power range 14 – 25V DC (ideal 15V)
- Connects with 4mm banana plugs.
- Banana plug: Red = 15V DC.
- Banana plug: Black = Ground.
- Power supply input has over- and undervoltage protection.
- Internal automatic fuse (PTC) 0.75A.

Signal input

Clock select

- Input impedance 50 Ω.
- Normal clock < 1.0V (or open)
- Glitch clock > 2.0V

Internal trigger

- Input impedance 50 Ω.
- Trigger ↑1.8V rising edge. Minimum pulse time 10ns.

Out

Clock out

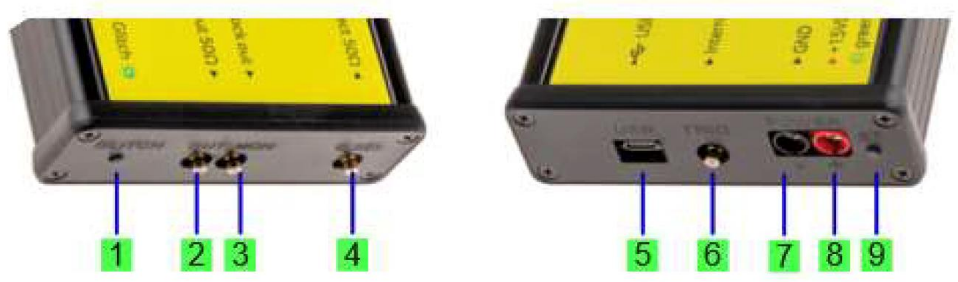
- Max. voltage range -1V... +4V
- Rise time < 1ns
- Fall time < 1ns
- Impedance selectable 0 – 250 Ω
- Signal out is NOT short circuit protected

Reference clock out

- Output impedance 50 Ω.

Control

- USB 2.0
- USB “C” connector



Port	Label	Description
1	Glitch	LED blinks when a glitch occurs
2	Monitor out	SMB connector to monitor the clock out signal. Impedance is 50 Ω
3	Clock out	SMB connector for clock output. Impedance 0Ω .. 250 Ω
4	Clock select	SMB connector for input clock select. Impedance is 50 Ω

5	USB	USB 2.0 “C” connector. Connect to PC controlling the Clock Glitcher
6	Internal trigger	SMB connector for the internal trigger signal. Impedance is 50 Ω
7	PSU	0V DC Power supply input.
8	PSU	+15V DC Power supply input. Maximum current = 1.0A Minimum current = 0.4A

9	power ok	Power status LED Green = power OK
		Red = fault (< 14V or > 25V)
		Orange = Internal power fault.

Base frequency 48000000
VCO frequency 13200000000

Frequency	Offset	Result freq	Max deviation in %
0,01 MHz	0	0,010 MHz	0,00%
0,1 MHz	0 0,100	MHz	0,00%
1 MHz	0	1,000 MHz	0,00%
2 MHz	0	2,000 MHz	0,00%
3 MHz	0	3,000 MHz	0,00%
4 MHz	0	4,000 MHz	0,00%
8 MHz	0	8,000 MHz	0,00%
10 MHz	0	10,000 MHz	0,00%
12 MHz	0	12,000 MHz	0,00%

24 MHz	0	24,000 MHz	0,00%
25 MHz	0	25,000 MHz	0,00%
33 MHz	0	33,000 MHz	0,00%
48 MHz	0	48,000 MHz	0,00%
50 MHz	0	50,000 MHz	0,00%
75 MHz	0	75,000 MHz	0,00%
100 MHz	0	100,000 MHz	0,00%
125 MHz	-714286	125,714 MHz	0,57%
200 MHz	0	200,000 MHz	0,00%
250 MHz	-3846154	253,846 MHz	1,52%
300 MHz	0	300,000 MHz	0,00%
350 MHz	-6756757	356,757 MHz	1,89%
10,5 MHz	-1193	10,501 MHz	0,01%
33,33 MHz	-3333	33,333 MHz	0,01%
10,025 MHz	-5395	10,030 MHz	0,05%
10,05 MHz	-3313	10,053 MHz	0,03%

Edition 1, October 2024

DS1150-90002

www.keysight.com