Pulse Width Modulation

# Introduction

This application note provides details on using the Pulse Width Modulation (PWM) peripheral of INP1010/INP1011/INP1012/INP1013 Talaria TWO modules. Two different applications - pwm and pwm\_fade are described in this app note. pwm application generates the PWM frequency of 1MHZ on four different channels simultaneously and pwm\_fade application generates the PWM with varying duty cycles on one of the channels. The output of the PWM is connected to LED D1. The brightness of the LED will change based on varying duty cycle generated by the pwm\_fade application.

# PWM Specifications

The following table contains the PWM specification of Talaria TWO.

|  |  |
| --- | --- |
| **PWM Specification** | **Details** |
| Maximum output frequency | 40MHz |
| Duty Rate Range | 0% to 100% |
| Pulse Alignment | Left Aligned |
| Other | Audio Capable |

Table : PWM specification details

# Block Diagram

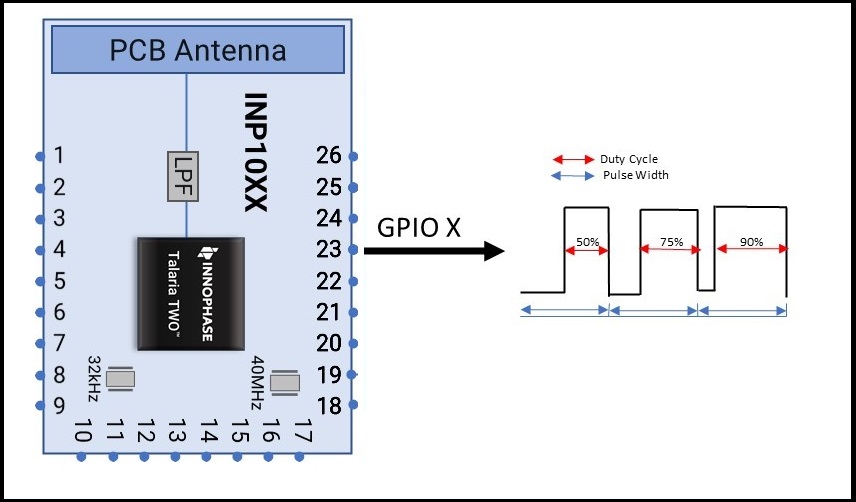


Figure : PWM - Block Diagram

# Building

To build the sample application, execute the following commands:

|  |
| --- |
| cd examples/pwm  make |

The make command should generate the pwm.elf and pwm\_fade.elf in the out directory.

# Sample Code Walkthrough

In this section, source code of pwm application and pwm\_fade applications are described. Following APIs are used in these applications.

## PWM APIs

1. pwm\_enable(): Creates a new PWM output on a given channel with a given period.
2. pwm\_destroy(): Releases a previously created PWM output.
3. pwm\_channel\_cfg\_set(): Configures the PWM Channel.
4. pwm\_output\_cfg\_set(): Configures a specific PWM output.
5. pwm\_output\_cfg\_set\_array(): Configures multiple outputs.
6. os\_gpio\_request(): Allocates specified pins and configures them as GPIO. The pins are initially set up to be input. The direction of the pins can be changed by configuring the specified GPIO pins to be outputs.
7. os\_gpio\_set\_mode(): Sets up GPIO pin mode.
8. os\_gpio\_mux\_sel(): Configures GPIO pin mux.

## PWM Application

pwm application generates a PWM frequency of 1MHZ, 50% duty cycle on all the four channels. In this application, the duty cycle is constant.

To enable PWM output on a specific GPIO, configure the pin mux using os\_gpio\_mux\_sel().

|  |
| --- |
| os\_gpio\_mux\_sel(GPIO\_MUX\_SEL\_PWM\_0P, pwm\_port0);  os\_gpio\_mux\_sel(GPIO\_MUX\_SEL\_PWM\_0N, pwm\_port2);  os\_gpio\_mux\_sel(GPIO\_MUX\_SEL\_PWM\_1P, pwm\_port1);  os\_gpio\_mux\_sel(GPIO\_MUX\_SEL\_PWM\_1N, pwm\_port3); |

os\_gpio\_mux\_sel() function sets the PWM functionality to the GPIO specified in the argument.

Next, create a pwm\_device using pwm\_enable().The struct can be used to configure the PWM output ports to operate with a certain duty cycle defined in percentage.

|  |
| --- |
| struct pwm\_output\_cfg cfg[4] = {  { .port = 0, .duty\_cycle = duty\_port\_0}, // Ch0 +  { .port = 2, .duty\_cycle = duty\_port\_2}, // Ch0 -  { .port = 1, .duty\_cycle = duty\_port\_1}, // Ch1 +  { .port = 3, .duty\_cycle = duty\_port\_3}, // Ch1 -  }; |

pwm\_enable() generates PWM output on a given channel with a given period and width of the pulse.

|  |
| --- |
| pwm\_enable(period) |

To set the PWM duty cycle and port, configure pwm\_output\_cfg structure.

|  |
| --- |
| struct pwm\_output\_cfg |

## PWM Fade Application

In the pwm\_fade application, PWM output of 1MHZ is generated on one channel and PWM duty cycle varies continuously. GPIO is connected to LED D1 of the EVB. When the application is loaded on to Talaria TWO, the LED D1 is ON and varies the brightness in proportion to the PWM duty cycle.

**Note:** Ensure that the jumper J3 on Talaria TWO EVB is connected.

Application flow:

1. Set the port to operate with a 50% duty cycle.
2. Create a 1000ns (1Mhz) long PWM signal.
3. The infinite while loop runs the pwm\_fade application.

This section describes configuring the PWM. To create a signal with a period of 1000ns and duty cycle of 50% on channel 0, port 0, configure the PWM via pwm\_enable(), pwm\_channel\_cfg\_set(), and pwm\_output\_cfg\_set().

|  |
| --- |
| #define PWM\_PIN 14  #define PWM\_PERIOD 1000 |

The PWM pin needs to be requested by os\_gpio\_request(). os\_gpio\_set\_mode() sets the operational mode of the pins to the GPIO\_FUNCTION\_MODE as PWM will operate the selected GPIO.

|  |
| --- |
| struct pwm\_output\_cfg cfg ={ .port = 0, .duty\_cycle = 50 };  os\_printf("PWM demo\n");  os\_gpio\_request(PWM\_PIN);  os\_gpio\_set\_mode(PWM\_PIN, GPIO\_FUNCTION\_MODE);  os\_gpio\_mux\_sel(GPIO\_MUX\_SEL\_PWM\_0, PWM\_PIN);  pwm\_enable(PWM\_PERIOD);  if (pwm\_channel\_cfg\_set(0, PWM\_CTRL\_ENABLE)) {  pr\_err("Failed to enable channel 0!\n");  } |

To change PWM width, configure the pwm\_output\_cfg\_set parameter.

|  |
| --- |
| struct pwm\_output\_cfg cfg = { .port = 0, .duty\_cycle = 50 }; |

# Running the Application

## Programming Talaria TWO (pwm.elf)

Program pwm.elf (*sdk\_x.y\examples\pwm\bin*) using the Download tool:

1. Launch the Download tool provided with InnoPhase Talaria TWO SDK.
2. In the GUI window:
   1. Boot Target: Select the appropriate EVK from the drop-down.
   2. ELF Input: Load the pwm.elf by clicking on Select ELF File.
   3. Boot Arguments: Pass the following boot arguments as applicable:

|  |  |  |
| --- | --- | --- |
| **Function** | **Boot Argument** | **Example** |
| Selecting GPIO Pin | pwm.port\_0 ( PWM Channel 0 +) | pwm.port\_0=18 |
| pwm.port\_2 ( PWM Channel 0 -) | pwm.port\_2=20 |
| pwm.port\_1 ( PWM Channel 1 +) | pwm.port\_1=19 |
| pwm.port\_3 (PWM Channel 1 -) | pwm.port\_3=21 |
| Common Duty Cycle Parameter | pwm.duty\_cycle ( For all PWM Pins) | pwm.duty\_cycle=50 |
| Individual Duty Cycle Parameter | pwm.duty0 ( PWM Channel 0 +) | pwm.duty0=25 |
| pwm.duty1 ( PWM Channel 1 +) | pwm.duty1=50 |
| pwm.duty2 ( PWM Channel 0 -) | pwm.duty2=75 |
| pwm.duty3 ( PWM Channel 1 -) | pwm.duty3=90 |
| Setting PWM Period | pwm.period | pwm.period = 1000 |

Table : Boot arguments

* 1. Programming: Prog RAM or Prog Flash as per requirement.

For more details on using the Download tool, refer to the document: UG\_Download\_Tool.pdf (path: *sdk\_x.y\pc\_tools\Download\_Tool\doc*).

**Note**: x and y refer to the SDK release version. For example: *sdk\_2.4\doc*.

## Expected output

Expected output is displayed on the console:

|  |
| --- |
| UART:NWWWWWAEBuild $Id: git-b664be2af $  App=gordon  Flash: Gordon ready!  Y-BOOT 208ef13 2019-07-22 12:26:54 -500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PNWWWAEBuild $Id: git-b664be2af $  PWM demo |

## Programming Talaria TWO (pwm\_fade.elf)

Program pwm\_fade.elf (*sdk\_x.y\examples\pwm\bin*) using the Download tool:

1. Launch the Download tool provided with InnoPhase Talaria TWO SDK.
2. In the GUI window:
   1. Boot Target: Select the appropriate EVK from the drop-down.
   2. ELF Input: Load the pwm\_fade.elf by clicking on Select ELF File.
   3. Programming: Prog RAM or Prog Flash as per requirement.

## Expected Output

Expected output is displayed on the console:

|  |
| --- |
| UART:NWWWWWAEBuild $Id: git-b664be2af $  app=gordon  flash: Gordon ready!  [10.874, 479] partitions mounted  UART:NWWWWWAEBuild $Id: git-b664be2af $  app=gordon  Flash: Gordon ready!  Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PNWWWAEBuild $Id: git-b664be2af $  PWM FADE demo |

# PWM Test Setup

Figure 2 represents the PWM test setup, where Talaria TWO evaluation board is powered through USB and the pwm.elf is flashed onto the module using the Download Tool. The GPIO which is configured to work as PWM is connected to the Oscilloscope along with ground. Once the module is released from Reset, the waveforms can be observed on the oscilloscope.

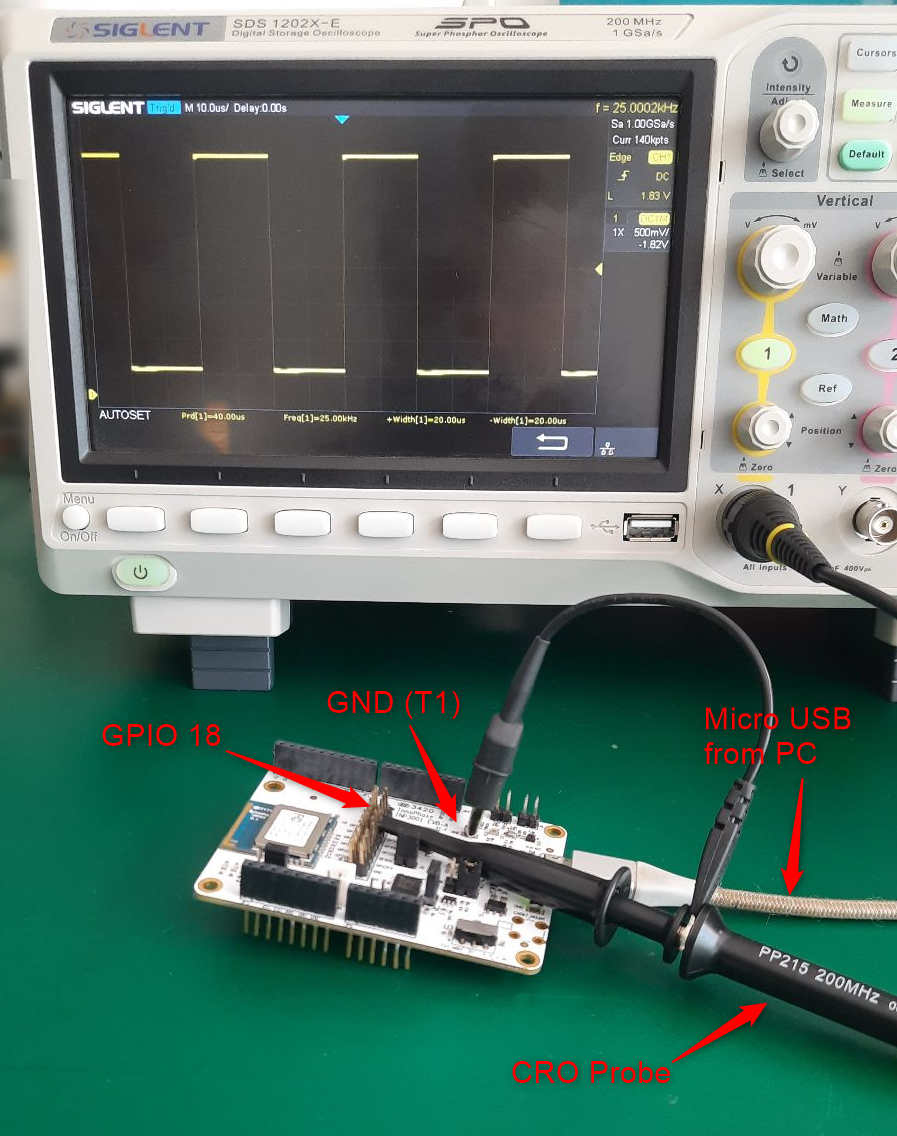


Figure : PWM Test Setup

Waveforms captured for 1MHz frequency with less than 5% duty cycle is as shown in Figure 3.

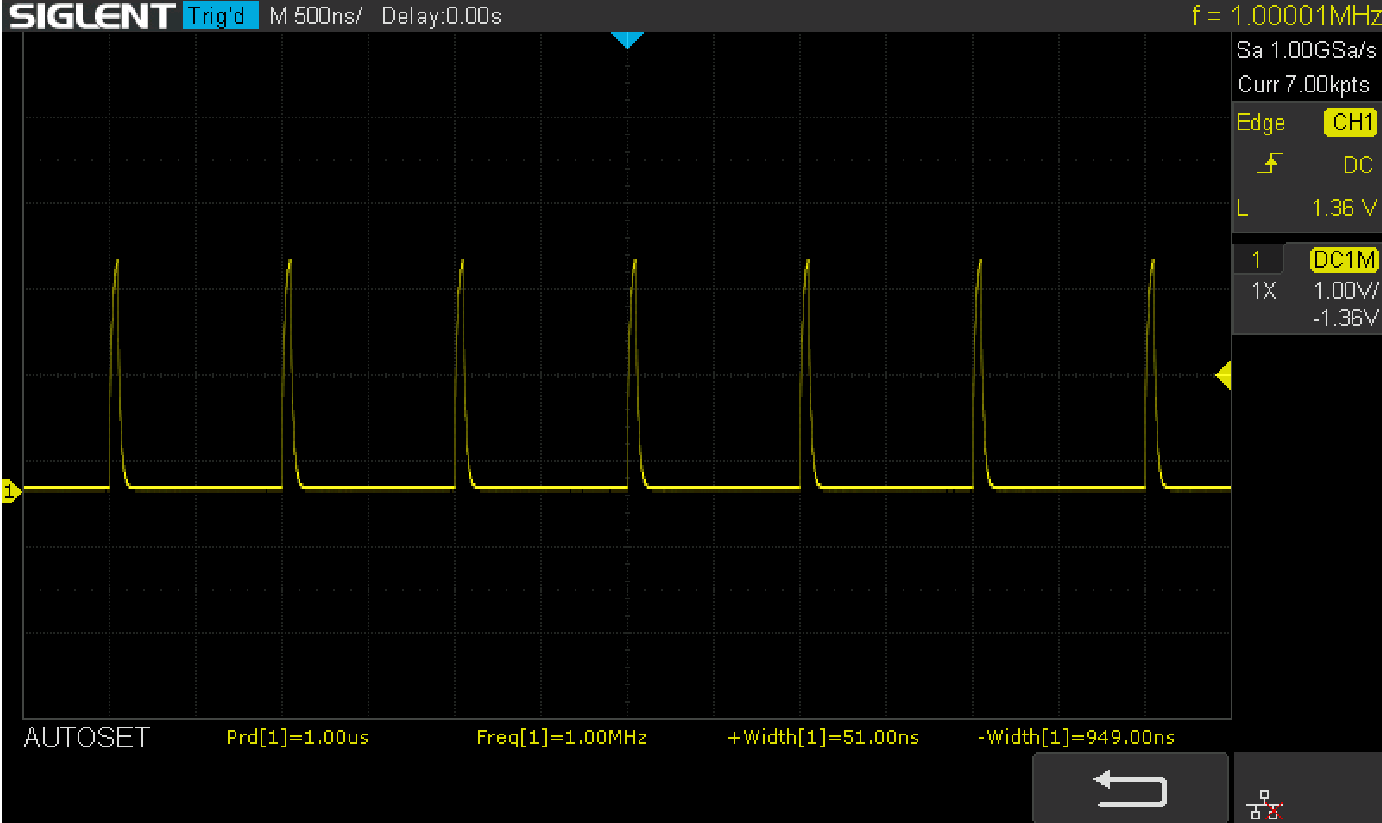


Figure : Waveforms for 1MHz frequency with less than 5% duty cycle

Waveforms captured for 1MHz frequency with 50% duty cycle is as shown in Figure 4.

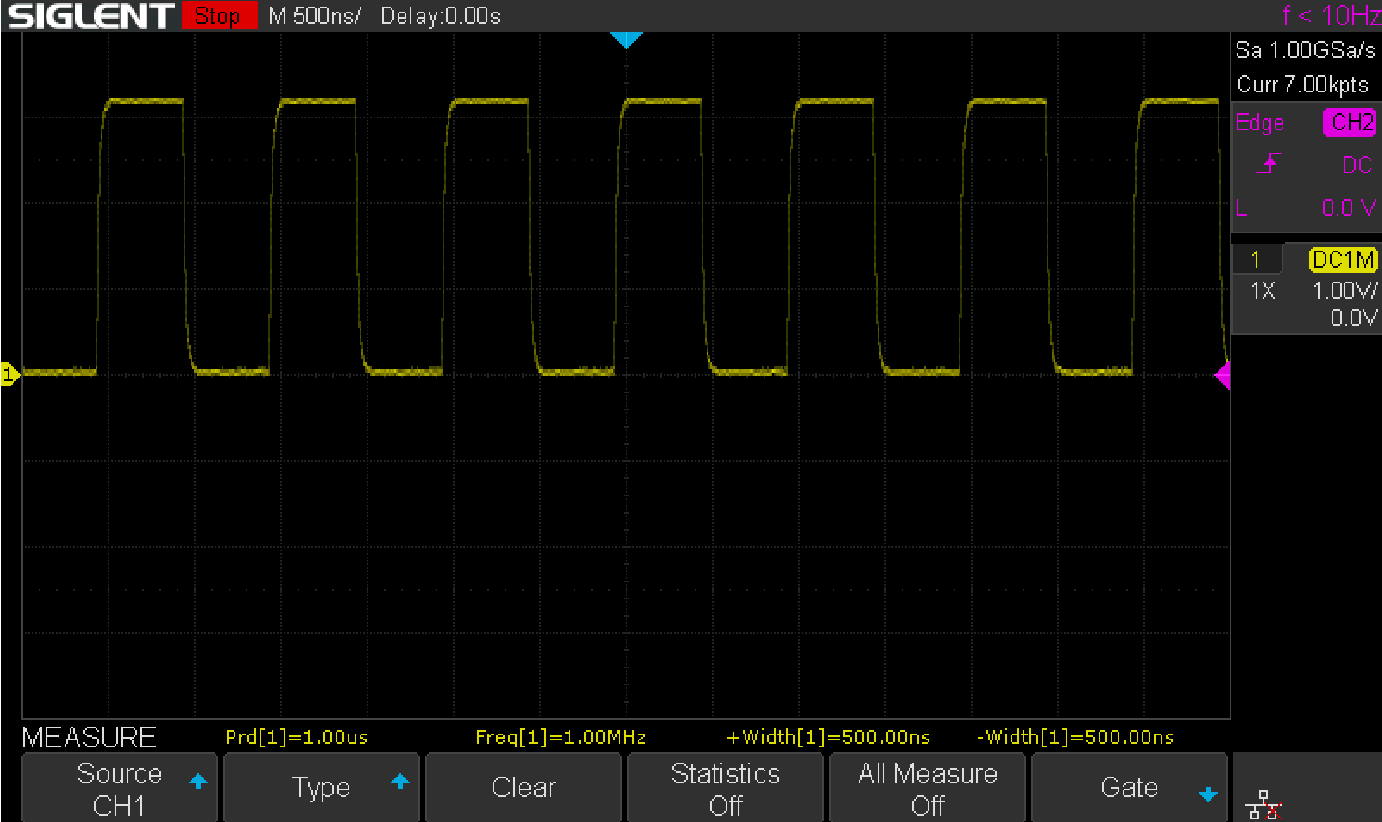


Figure : Waveforms for 1MHz frequency with 50% duty cycle

Waveforms captured for 2MHz frequency with 50% duty cycle is as shown in Figure 5.

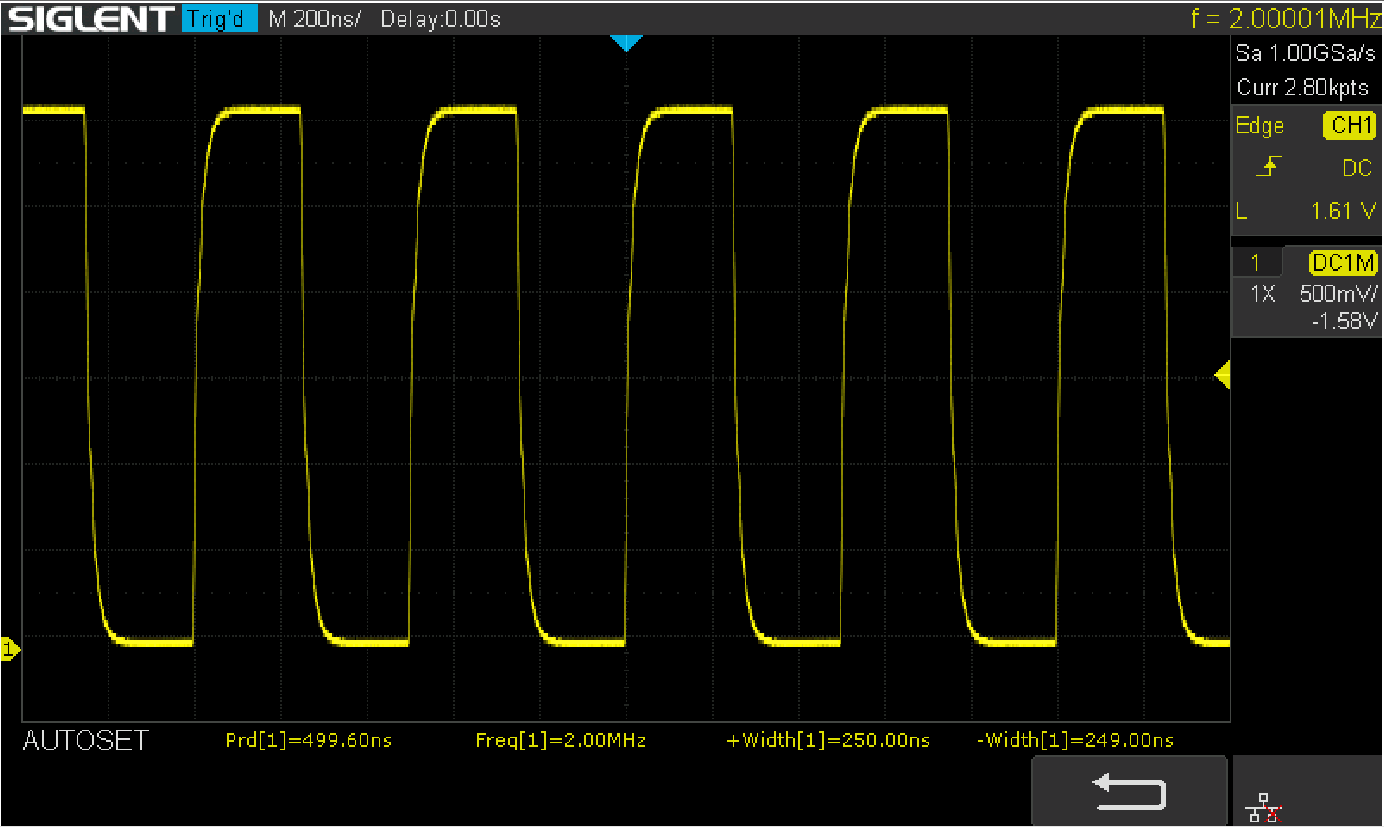


Figure : Waveforms for 2MHz frequency with 50% duty cycle

Waveforms captured for 4MHz frequency with 50% duty cycle is as shown in Figure 6.

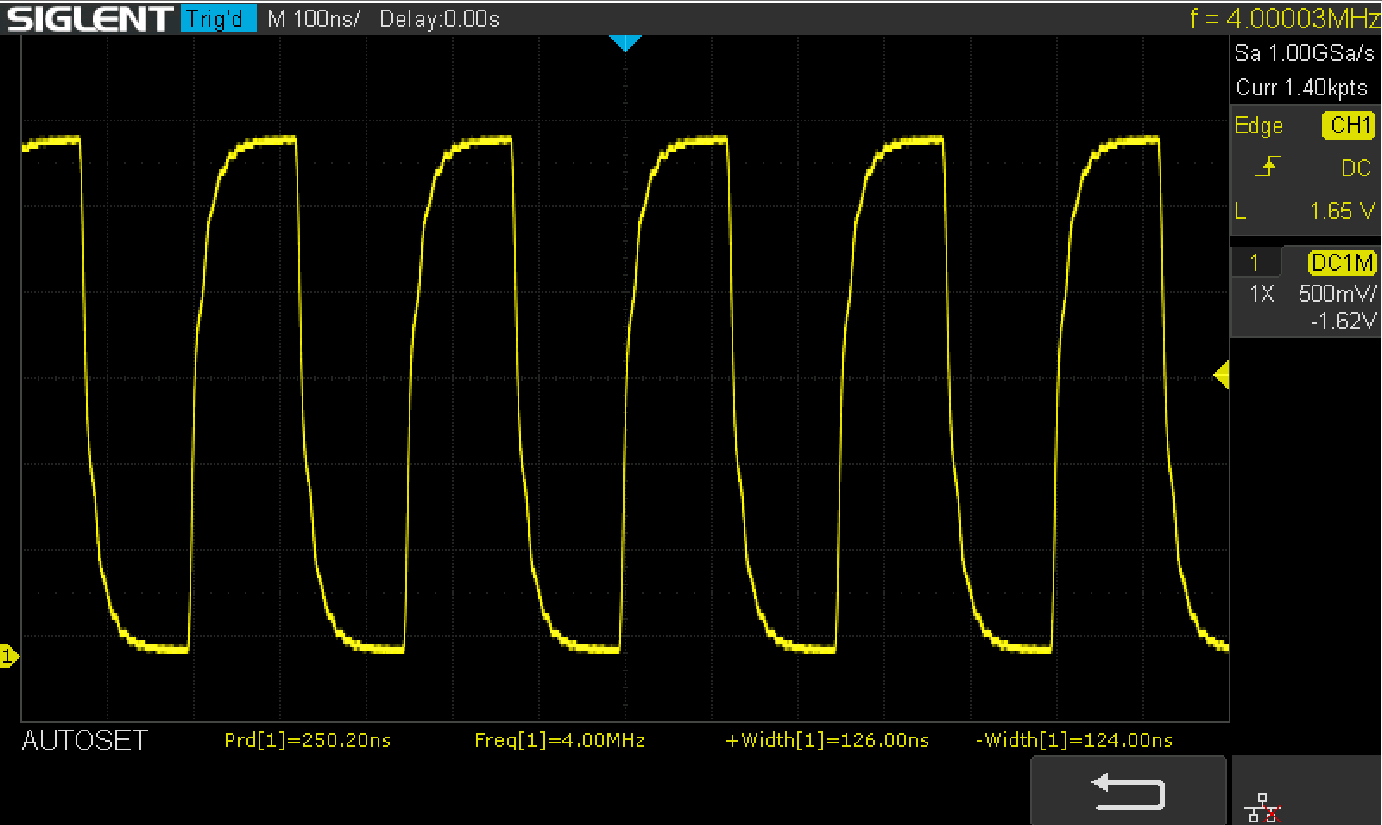


Figure : Waveforms for 4MHz frequency with 50% duty cycle

Waveforms captured for 10MHz frequency with 50% duty cycle is as shown in Figure 7.

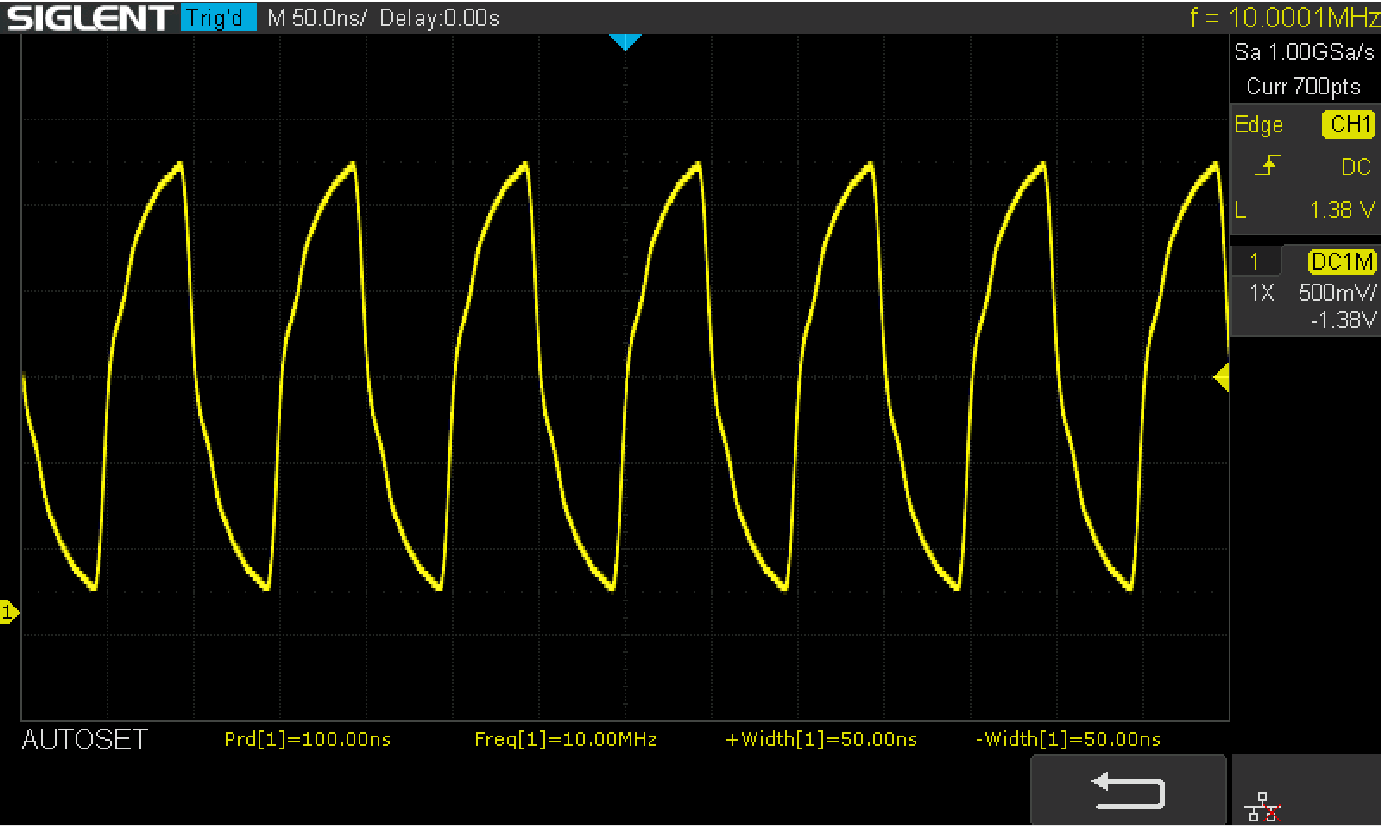


Figure : Waveforms for 10MHz frequency with 50% duty cycle

Waveforms captured for 25KHz frequency with more than 50% duty cycle is as shown in Figure 8.

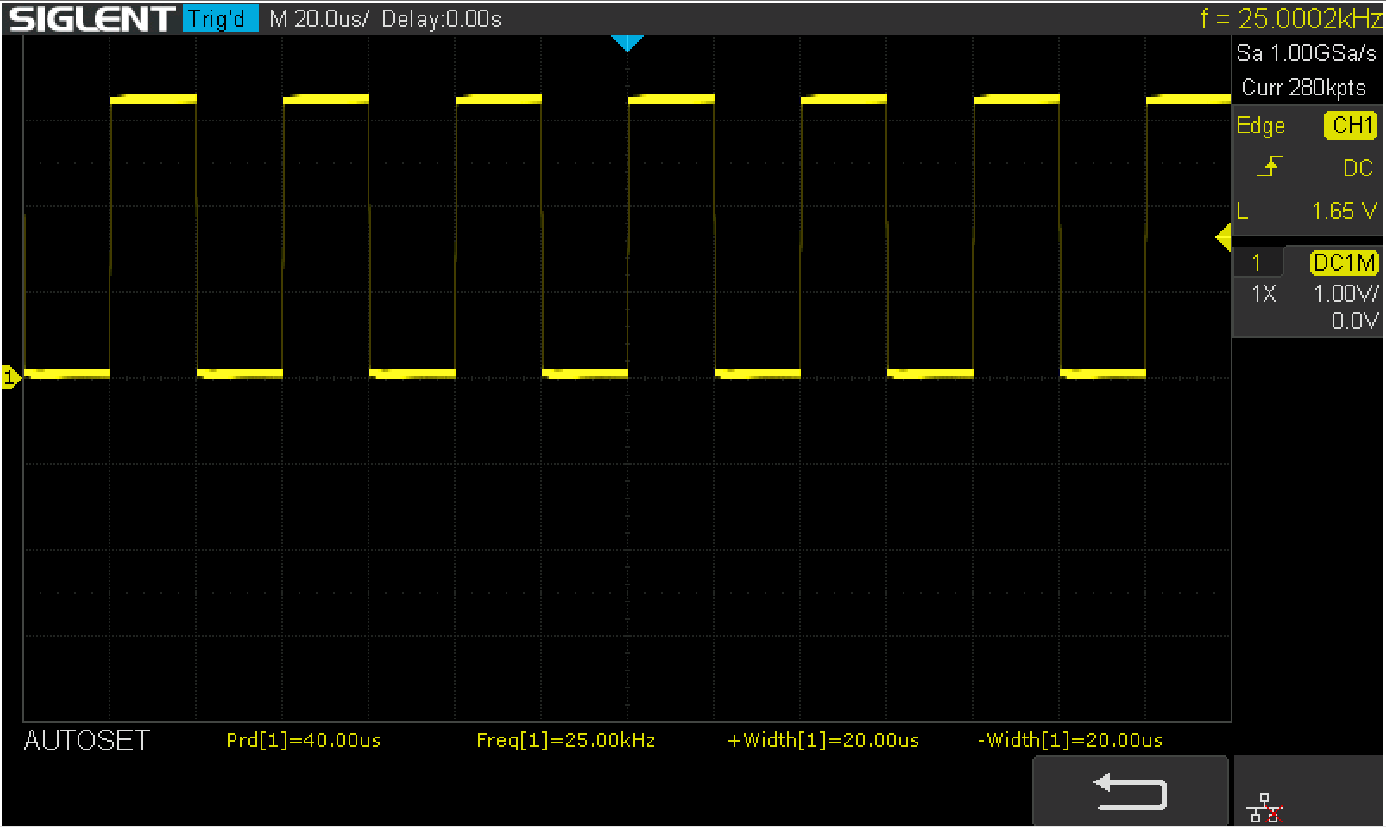


Figure : Waveforms for 25KHz frequency with more than 50% duty cycle