MICRO PHOTON DEVICES

PSD

Picosecond Delayer

$PSD-065-A-OEM

Via Stradivari, 4
39100 Bolzano (BZ), Italy
email info@micro-photon-devices.com
Phone +39 0471 051212 • Fax +39 0471 501524

Picosecond Delayer User Manual Version 6.2.2 - July 2018

Table of Contents

Introduction

The MPD Picosecond Delayer, is a solid-state delayer based on programmable-delay chips. Any signal that is fed in the SMA input is reshaped, delayed in time and routed to the SMA outputs. The reshaping generates both a TTL and a NIM pulse independently on the type and form of the input pulse. The minimum delay (Propagation Delay) is on average 15 ns. The user-selectable additional delay (Programmable Delay) can be set from 0 ns to 50 ns within 10 ps steps. The output pulses duration is also user adjustable from 1 ns to 250 ns. The "module" version, i.e. the standalone instrument, of the delayer is shown in Figure 1. It can be controlled directly on the instrument by using a membrane keyboard and a colour LCD or it can be also operated via USB by means of a graphical user interface (GUI) software. This software runs only on PCs running Microsoft Windows® XP® or higher. The provided device drivers install a "Virtual Serial COM" and the delayer is controlled by plain text ASCII commands sent to the correct COM port. Since the commands are here provided (page 17), the Picosecond Delayer can be easily integrated in any custom control software and experimental set-up without the use of the MPD proprietary GUI. Additionally, because the device drivers are provided also for Linux and Mac Os X, the instrument can be effectively operated with no problem by PCs using these operating systems.

The delayer accepts periodic and aperiodic input signals with amplitudes from -2V to +3V and can be triggered either on positive or negative edges, with an adjustable threshold ranging from -2V to +2V. The delayer can act also as a frequency divider by a user selectable positive integer ranging from 1 to 999. The digital outputs have a controllable pulse width ranging from 1 ns to 250 ns for both NIM and TTL outputs. Input and outputs connectors are SMA. In Figure 1 is thus shown the $PSD-065-A-MOD model.

Figure 1. Picosecond Delayer model $PSD-065-A-MOD.

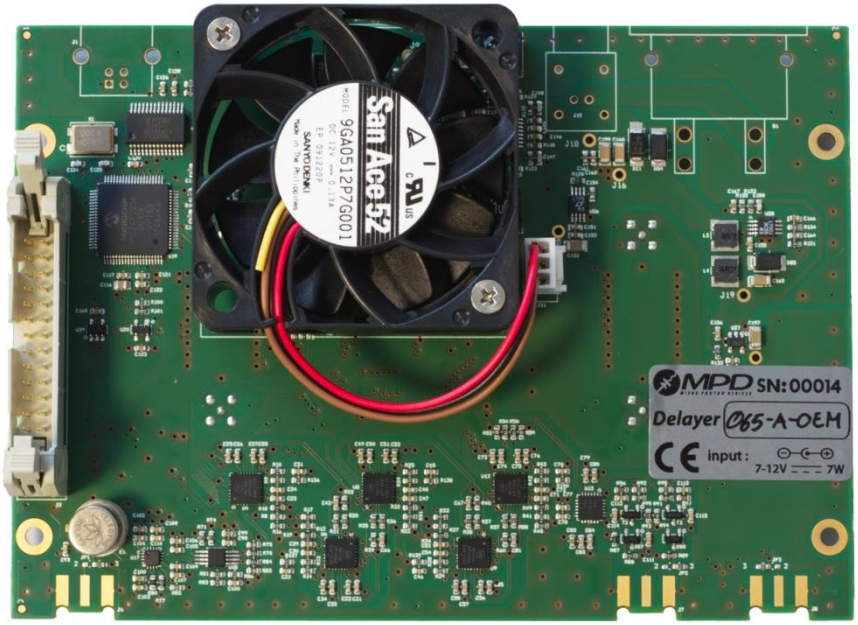


Figure 2. Picosecond Delayer model $PSD-065-A-OEM (TOP view).

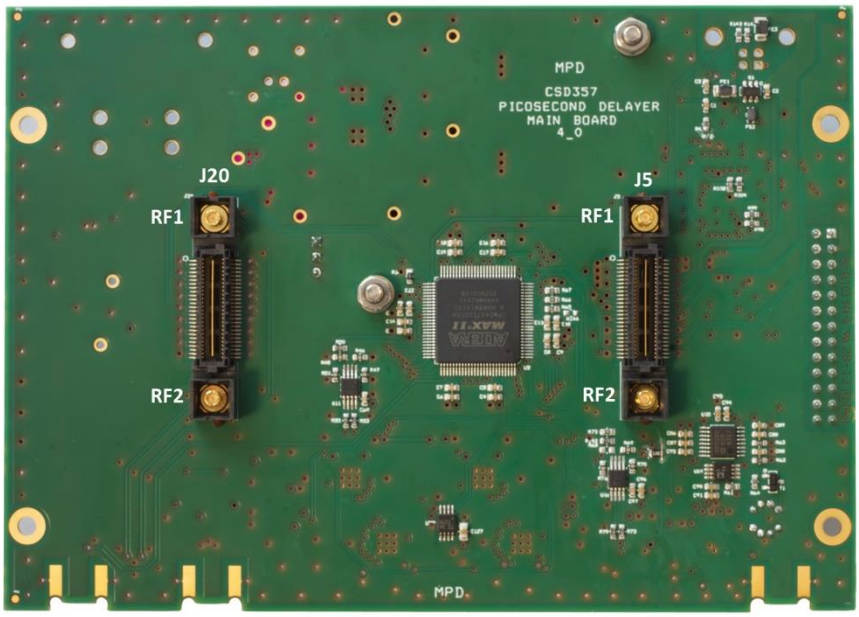


Figure 3. Picosecond Delayer model $PSD-065-A-OEM (BOTTOM view).

For costumers that would like to integrate directly in their instruments the outstanding capabilities of the MPD picosecond delayer it is possible to get the picosecond delayer OEM-version. It is based on the delayer motherboard and it is shown in Figure 2 and in Figure 3. All the signals and power lines are routed and/or fed through two special SAMTEC connectors (J5 and J20), mounted on the bottom of the motherboard and shown in Figure 3.

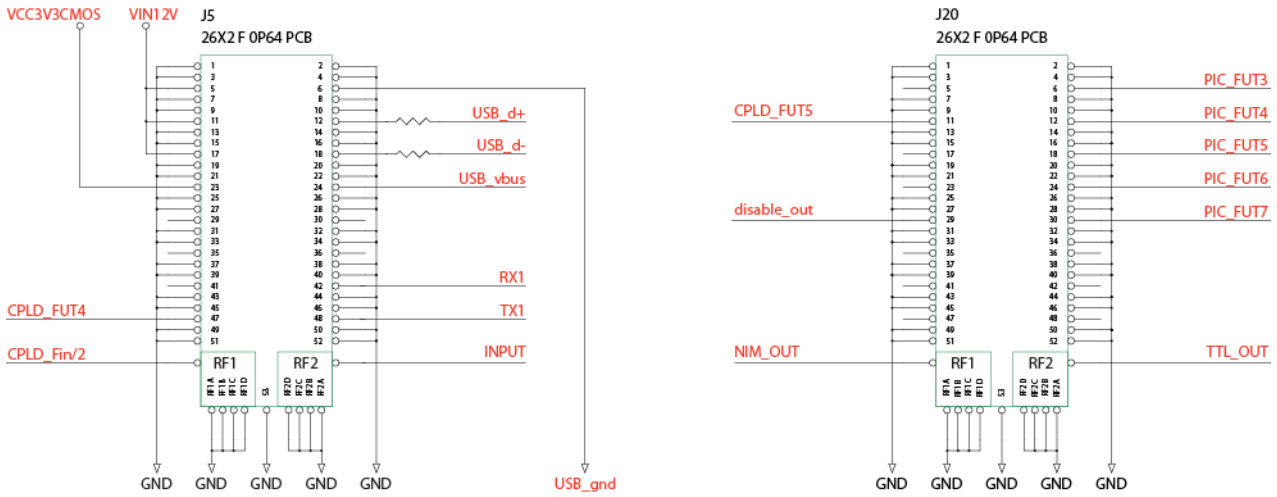


Figure 4. Electrical connection for the two SAMTEC connectors mounted on the bottom of the picosecond delayer motherboard.

Thanks to these connectors it is possible not only to access to the delayer's inputs and outputs but also to talk directly to the internal microcontroller both via a USB bus (the same one employed in the $PSD-065-A-MOD) and an additional UART interface. On these connectors are also present other useful signals or unused links to both the internal CPLD and Microcontroller. These unused links will allow the introduction of new features without HW changes. Finally both the CPLD and the Microcontroller can be re-programmed through the USB interface and a custom PC software that is provided.

Picosecond Delayer hardware characteristics

The connection with the delayer, $PSD-065-A-OEM version, is thus obtained through two Samtec (www.samtec.com) connectors. On the board are mounted two QMS-026-5.75-LD-RF1 thus customers should use model FS-026-04.25-L-D-RF1 for their motherboard. Figure 4 illustrates the electrical connections to the picosecond delayer through the described connectors. The function of each used pin is illustrated in Table 1, while the electrical specifications of all the inputs and outputs are reported in Table 2.

Table 1. Connection to the Picosecond Dealyer OEM version: description of all the used inputs and outputs.

NIM OUT	RF output, requires 50 Ohm DC terminated transmission lines. The output is a NIM pulse, which means that the low logic level is 0 V and the high logic level is -800 mV. The falling edge of the pulse marks, with very low jitter, the delayed signal; the output pulse width is user adjustable.

TLL OUT	RF output, requires 50 Ohm DC terminated transmission lines. Electric pulses of 3.3V LVTTL are generated at this output. The TTL output pulse width is also user adjustable and it is typically 1 ns longer than NIM OUT one.
INPUT	RF input, 50 Ohm with fixed DC impedance. The trigger signal can be positive or negative between -2 V and +3 V and the input internal threshold is fully programmable. The input signal must have a duration of at least 100 ps, 100 mV overdrive and a minimum slew rate of 100 V/µs in order to be detected correctly by the internal comparator.
CPLD_FUT*X*	CPLD_FUT*X* are connected to pins of the internal CPLD. They are currently not used and are reserved for future developments. They should be drawn to ground with a 10kΩ pull-down resistor or routed to customer's microprocessor.
PIC_FUT*X*	PIC_FUT*X* are connected to pins of the internal Microprocessor. They are currently not used and are reserved for future developments. They should be drawn to ground with a 10kΩ pull-down resistor or routed to customer's microprocessor.
USB_*xxx*	USB_*xxx*, namely USB_vbus, USB_d+, USB_d- and USB_gnd, are the pins needed to use in order to communicate via USB to the internal Microprocessor. Names follow the international convention on the USB bus.
CPLD_Fin/2	This pin is an output and clocks out the INPUT's frequency divided by a factor of 2. Used to monitor if any input is detected.
RX1, TX1	These are the pins used to communicate with the internal microcontroller through a 3.3V UART, i.e. the high level ('1') is 3.3 V and the low level ('0') is 0 V.
disable_out	input pin, LVTTL levels (0 – 3.3 V), 50 Ohm terminated with fixed DC impedance. When the input is high the output is disabled.
VIN12V, VCC3V3CMOS, GND	These are the power lines of the board. VIN12V is used to power the board while GND is the ground connection. VCC3V3 is a power output given for user convenience. It is generated by an internal regulator and it is used also to power the internal logic ICs.

Picosecond Delayer's Electrical Characteristics

The logic values and the timing properties of all the inputs and outputs, particularly of INPUT, NIM OUT and TTL OUT, are described in Table 2. Figure 5 illustrates graphically all the terms and definitions given in Table 2. In Table 2, it is also indicated the programming time which is defined as the time from the end of the serial communication to the time the new configuration is fully applied, when changing a single parameter.

Table 2. Electrical Characteristics of the Picosecond Delayer input and outputs.

Parameter	Symbol	Description	Min.	Typ.	Max.	Unit
Input high voltage	V_{IN_High}				3	V
Input low voltage	V_{IN_low}		-2			V
Input differential range	$V_{D,MAX}$		-2		2	V
Input termination	R_{IN}			50		Ω
Input voltage overdrive	V_{OV}		100			mV
Input pulse width	t_{IN}		100			ps
Input Slew Rate	SR		100			V/µs
Input overdrive dispersion		100 mV < V_{OV} < 1 V		10		ps
Input slew rate dispersion		2 V/ns < SR < 10 V/ns		15		ps
Input Edge				Neg/pos		
Input Threshold	V_{TH}		-2		2	V
Input threshold resolution	ΔV_{TH}		10	18	30	mV
NIM Output low logic level	V_{NIM_High}	50 Ω termination required		0		V
NIM Output high logic level	V_{NIM_Low}	50 Ω termination required		-800		mV
NIM Output Bandwidth	BW_{NIM}		300	380		MHz
TTL Output Bandwidth	BW_{TTL}		100	120		MHz
TTL Output low logic level	V_{TTL_Low}	50 Ω termination required		0		V
TTL Output high logic level	V_{TTL_High}	50 Ω termination required	2.4			V
Propagation delay INPUT - NIM OUT	t_{PD_NIM}		12	15	18	ns
Propagation delay INPUT - TTL OUT	t_{PD_TTL}		16	19	22	ns
Delay programmable range	t_{RANGE}	Equivalent to t_{DELAY} (max)	45	50	55	ns
Delay programmable range Temperature variation	Δt_{RANGE} (T)			75		ps/°C
Random timing jitter (RMS)	Rt_{jitter}	t_{DELAY} = 0 ns		2	5	ps
		$t_{DELAY} = t_{range}$		5	12	ps
Delay step	Δt_{DELAY}			10		ps
Delay integral non linearity	INL	Full scale range	-100		50	ps
Input Frequency divider factor			1		999	

OUTPUT pulse duration	t_{OUTPUT}		1		250	ns
OUTPUT pulse step	Δt_{OUT}	Pulse width incremental step		3.3		ns
OUTPUT off time	t_{NIM_OFF}, t_{TTL_OFF}	t_{NIM_OFF} and t_{TTL_OFF} are not guaranteed to be the equal on a module.	1	3		ns
OUTPUT pulse width jitter	$t_{OUT,jitter}$	% of pulse duration			1	%
OUTPUT pulse width INL	$t_{OUT,INL}$	Referred to Applied value, NIM OUT		max (±5%, 1 ns)		
OUTPUT pulse width difference between TTL and NIM (TTL-NIM)	$\Delta t_{TTL-NIM}$			1		ns
OUTPUT pulse width INL	$t_{OUT,INL}$			max (±5%, 1 ns)		
Programming time	t_{PROG}	Not valid for input edge change		1	2	ms
12V Power Supply (INPUT)	VCC12V	12 V board's power supply	11	12	13	V
3.3V Power Supply (OUTPUT)	VCC3V3CMOS	3.3V internal logic power supply	3.2	3.3	3.4	V
12V Power supply -current- SINK	VCC12V	Maximum DC current absorbed by the 12V power supply			1	A
3.3V Power supply -current- SOURCE	VCC3V3CMOS	Maximum current that can be sourced by the 3.3V power supply			500	mA
GND	GND	Board ground		N/A		
$V_{ILs\_I/O}$ (I/O INPUT)	RX1 CPLD_FUT_X PIC_FUT_X Disable_output		0		0.4	V
$V_{IH\_I/O}$ (I/O INPUT)	RX1 CPLD_FUT_X PIC_FUT_X Disable_output		2.8		3.3	V
$V_{OL\_I/O}$ (I/O OUTPUT)	TX1 CPLD_FUT_X PIC_FUT_X CPLD_Fin/2	High Z termination	0		0.4	V
		50 Ohm termination	0		0.2	V
V_{OH_IO} (I/O OUTPUT)	TX1 CPLD_FUT_X PIC_FUT_X CPLD_Fin/2	High Z termination	2.8		3.3	V
		50 Ohm termination	1.6		2.4	V
CPLD_Fin/2 - frequency			100	120		MHz
USB power supply	USB_vbus	Refer to standard USB specifications	4.5		5.5	V
USB data signal plus	UBS_d+	Refer to standard USB specifications		N/A		
USB data signal minus	USB_d-	Refer to standard USB specifications		N/A		
USB ground	USB_gnd	Refer to standard USB specifications		N/A		
Temperature	The delayer board has been designed with a thermal control loop that, by continuously adjusting the fan rotation speed, sets the board temperature to 55°C wit ± 0.2°C. For this reason, the board should be inserted in a carefully designed enclosure, in order to optimize the instrument dissipation and to avoid thermal gradients on the MPD board. Please contact MPD for further details and support.					

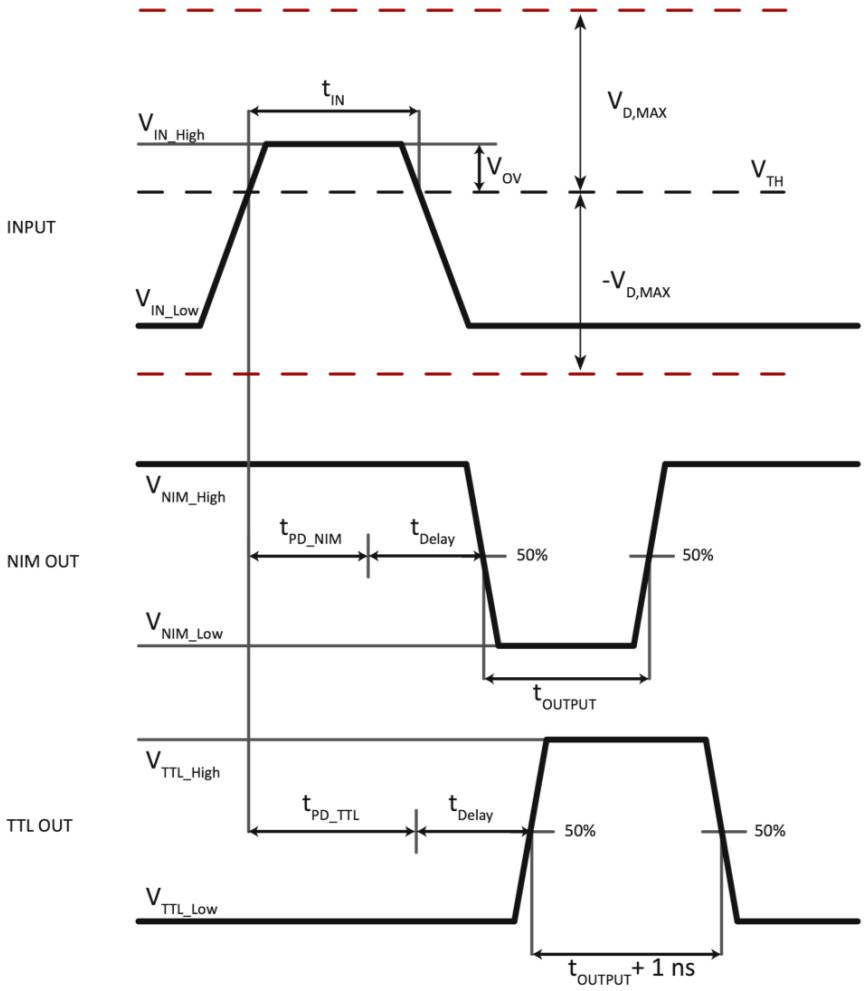


Figure 5. Timing NIM OUT and TTL OUT whit respect to the INPUT signal.

Calculation of the maximum repetition rate

The picosecond delayer input is internally connected to a fast comparator whose output is fed to a pulse shaper. The input pulse duration (t_{IN}) is thus totally un-related to the internal reshaped-pulse duration. As a consequence, the maximum repetition rate at the input side can be calculated as follows:

$$f_{max-input} = \frac{1}{t_{in} + 100\ ps}$$

where 100 ps is the minimum, both negative and positive, input pulse duration (t_{IN}) according to Table 2. Considering the NIM output pulse length equal to t_{output}, the maximum repetition rate, at the output side, depends on the maximum frequency of the selected output (BW_{NIM} for NIM OUT and BW_{TTL} for TTL OUT), on the duration of the output pulse (t_{output}) on the minimum output off time (t_{NIM_OFF} or t_{TTL_OFF}) and on the input frequency divider factor *n*. It follows that the maximum repetition rate, at the output side is:

$$f_{max-output,NIM} = \frac{1}{n} \times min\left(\frac{1}{t_{output} + t_{NIM-OFF}}, \mathrm{BW}_{NIM}\right)$$

$$f_{max-output,TTL} = \frac{1}{n} \times \min\left(\frac{1}{(t_{output} + 1\,ns) + t_{TTL-OFF}}, BW_{TTL}\right)$$

The picosecond delayer maximum repetition rate is than the minimum between $f_{max\text{-}input}$, $f_{max\text{-}output}$.

Picosecond Delayer Operation

Normally, as shown in Figure 6, the *Picosecond Delayer* generates an output digital pulse, with an user-adjustable delay, for every input trigger event. As can be seen in Figure 6, the delayer actually generates always two digital pulses one negative, based on the NIM standard at the NIM OUT output, and one positive, based on the LVTTL standard at the TTL OUT output. Of course the input trigger period can be shorter than set delay: the module simply delays in time the input pulses while keeping their repetition rate. The minimum delay between an input trigger event and an output pulse is given by intrinsic propagation delay (t_{PD_NIM} for NIM OUT and t_{PD_TTL} for TTL OUT). An additional programmable delay can be set by the user from 0 ns to an achievable maximum value (t_{RANGE}), in order to have the desired total delay. Users can also set the output pulses width. Particularly the NIM OUT pulse width (t_{OUTPUT}) is adjustable from 1 ns to 250 ns in fixed steps; TTL-OUT's pulse width is typically 1 ns longer than NIM OUT's one. The delayer can also act as a frequency divider by a user selectable positive integer ranging from 1 to 999; of course, in case of aperiodic input signals, the divider acts as a pulse skipper where, given *n* the dividing factor, (*n-1*) input pulses are skipped after each valid input.

As already specified, the minimum INPUT pulse width for positive and negative pulses (t_{in}) is 100 ps. After every output pulse, NIM and TTL outputs are off for a period of time, called minimum off time (t_{NIM_OFF} and t_{TTL_OFF} respectively). This off-time can vary from 1 ns to 3 ns depending on the actual sample and might be different for the two outputs. As said, normally for each input pulse, NIM and TTL output pulses are generated. Anyway, as shown in Figure 7, this might not always be the case. As a matter of fact, given an input pulse, considering its significant edge (rising or falling), all the next ones within $t_{output} + t_{NIM\_OFF}$, in case of the NIM output, or $t_{output} + \Delta t_{TTL-NIM} + t_{TTL\_OFF}$, in case of the TTL output, are masked. This happens because the masked pulses would have to be generated before the end of the previous output pulse. In case of a periodic input signal, the condition to be satisfied in order not to have the masking effect, is the following:

$$^1/_{fin} > t_{output} + t_{NIM\_OFF} \quad \text{in case of the NIM output}$$

$$^1/_{fin} > t_{output} + \Delta t_{TTL-NIM} + t_{TTL\_OFF} \quad \text{in case of the TTL output}$$

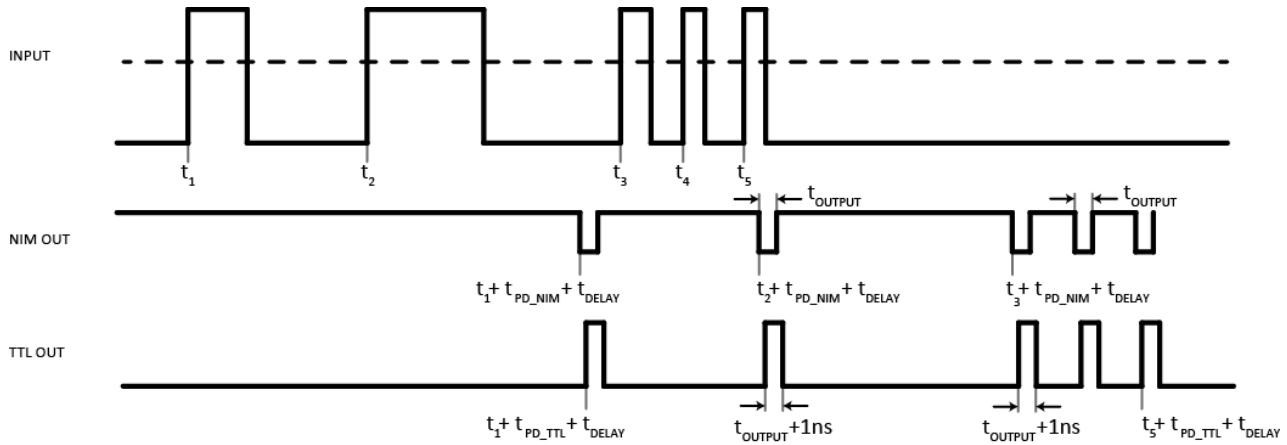


Figure 6. Picosecond Delayer timing signals. Rising edge selected as significant edge.

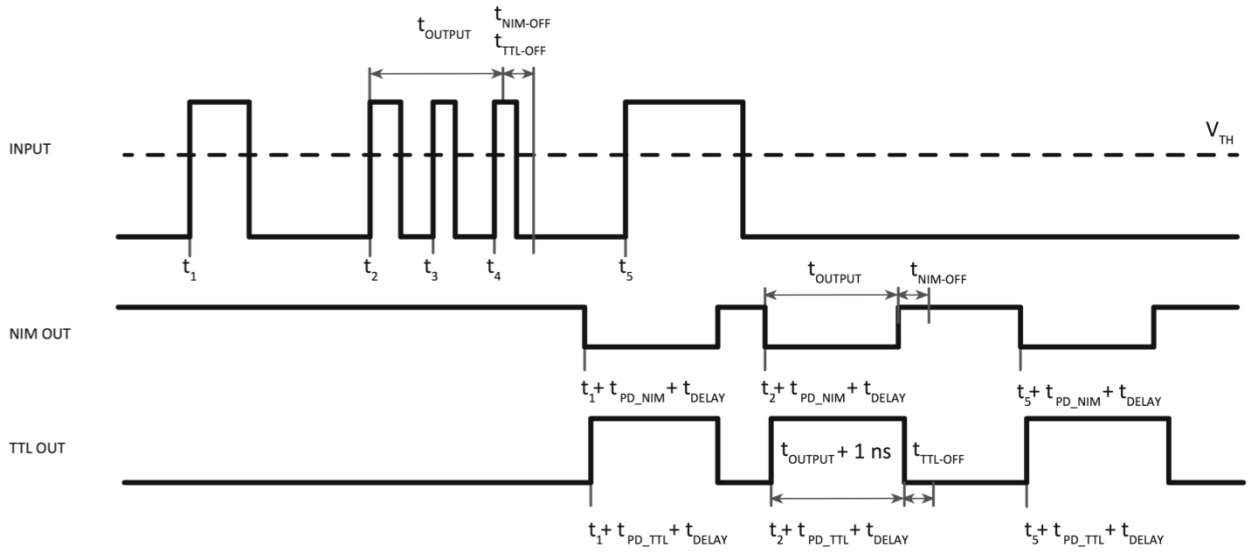


Figure 7. Picosecond Delayer timing signals. The third and fourth pulses are fed within $t_{OUTPUT} + t_{NIM\_OFF}$ from the rising edge of the second pulse (t2) and thus the NIM OUT hides their corresponding output pulses. The same happens for the TTL OUT. Rising edge selected as significant edge.

Picosecond Delayer precision and temperature dependence

The Picosecond Delayer has a nominal step delay of 10 ps. In order to reduce as much as possible the integral non-linearity and the differential non-linearity each instrument is fully characterized and calibrated. As a result the curve, *measured-delay* versus *set-delay,* is almost ideal and the integral non-linearity error, defined as the *measured-delay* minus the *set-delay*, as a function of the set-delay, is always smaller than 100ps in absolute values. Actually, the excellent linearity of the curve *measured-delay* versus *set-delay* is proved by the very good integral non-linearity error of about (peak to peak) +50ps/-100ps over the full range of the programmable delays, as shown in Figure 8.

As any electronic device, the Picosecond Delayer experiences propagation delay changes with temperature variations. In order to mitigate the effects of the latter, the temperature is stabilized by means of variable speed fan inserted in a thermal control loop. After the initial warm-up, the operating temperature of the delayer board stabilizes at 55 °C. This value is kept then constant by the thermal control loop with an accuracy of ±0.2 °C. Additionally, whenever the set temperature cannot be kept at 55 °C, the delayer compensates the delay as a function of temperature to reduce delay variations in order to keep the same state of the art performances as shown in example in Figure 9 at three different temperatures.

Even if temperature compensations maintain extraordinary INLs, the delay programmable range can't be compensated and it varies at a rate of 75 ps/°C.

Figure 8. Typical Picosecond Delayer integral non linearity error (INL) as a function of the set delay after calibration.

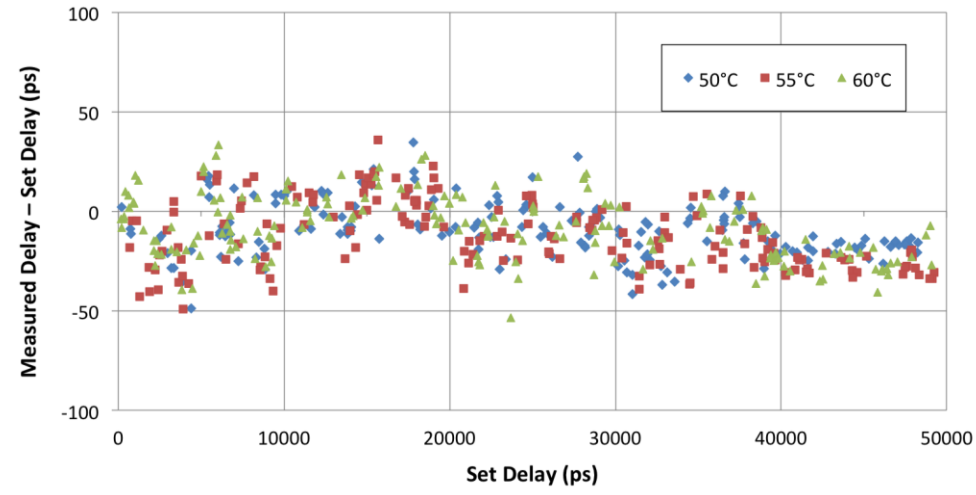


Figure 9. Typical Picosecond Delayer integral non linearity error (INL) as a function of the set delay after calibration at different temperatures.

$PSD-065-A-MOD Picosecond Delayer's mechanical dimensions

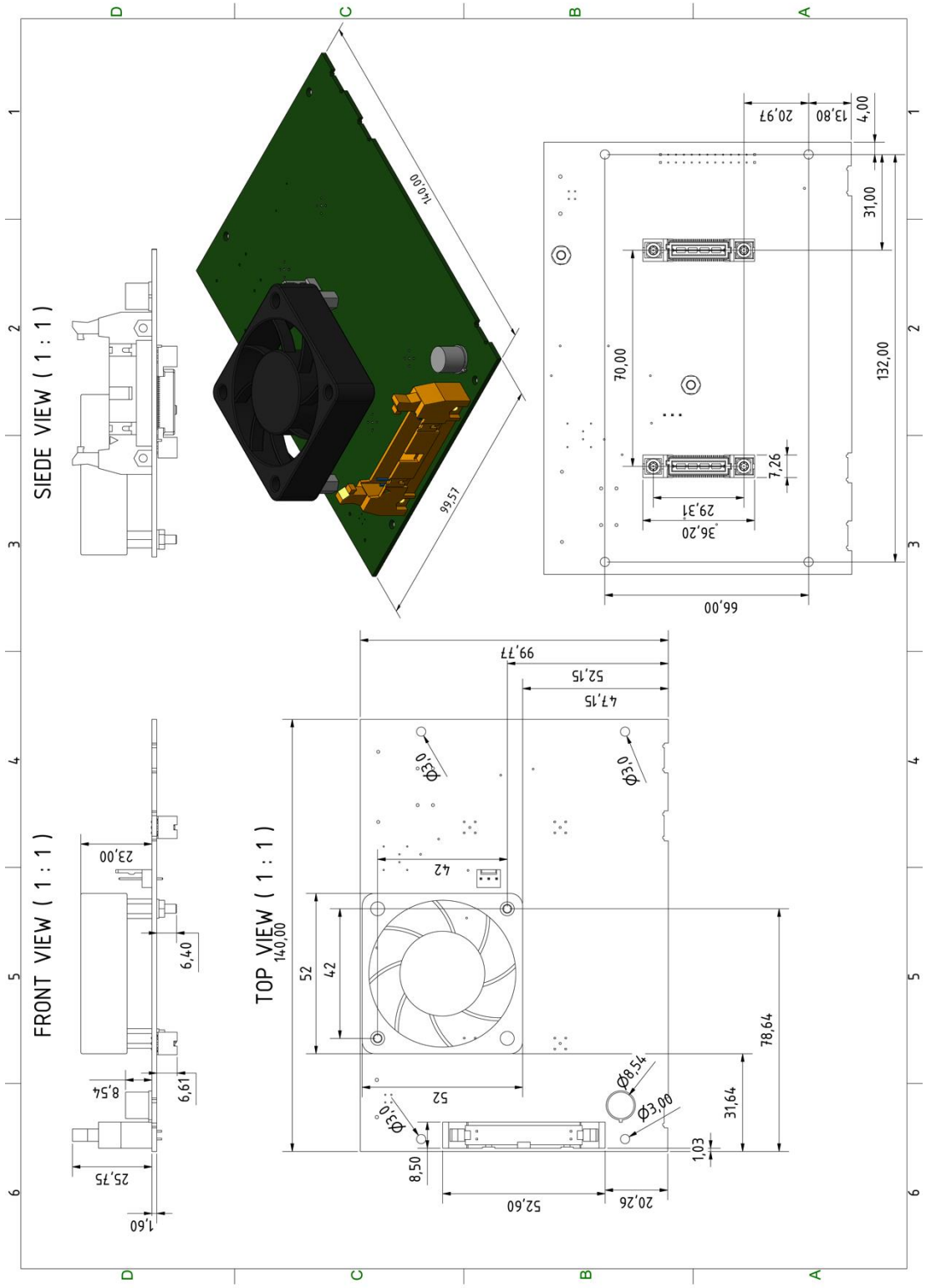


Figure 10. Picosecond Delayer $PSD-065-A-OEM mechanical dimensions. All dimensions in mm.

Picosecond Delayer Software interface (Windows only)

The Picosecond Delayer is provided with an acquisition and control software, which is running on Microsoft Windows operating systems. This software works by communicating through the USB connection of the board. The USB connection is actually an FTDI® chip that creates, through the available drivers, a virtual serial port on the host PC.

Installation

Complete the following steps in order to install the controlling software for Windows®:

- Log in as system administrator or as a user with administration privileges.
- Disable any automatic virus detection programs before you install. Some virus detection programs could interfere with installation.
- Insert the USB key and copy the zip file that contains the installer in a temporary location (for example your Desktop)
- Unzip the file and double click on the file named setup.exe located inside the folder you just unzipped. Then follow the instructions on the screen until the installer ends.
- Windows® will automatically recognize the device and install the correct drivers. If a "found new hardware" window appears, select the option to automatically download recommended drivers from internet. For this reason, the computer on which the drivers will be installed, MUST BE CONNECTED to the internet. The drivers will create a virtual serial COM associated with the Picosecond Delayer.
- Once completed, you can start the controlling software from Windows *Start Menu → Programs → MPD →Picosecond Delayer*.
- Remove the temporary install folder.

Software Interface

The software interface allows the control of the delayer settings with a simple and user friendly window, which is shown in Figure 11 and in Figure 12. The window shows always the current status of the delayer. In order to understand in detail the interface, please refer to Table 3.

In order to change the delay, the threshold level, the output pulse duration or the input frequency divider, the desired value(s) can be inserted in the correct input box(es) and, after pressing the enter key, the delayer is immediately updated. In order to modify the input signal valid edge, the "CHANGE EDGE" button has to be pressed. The "TOGGLE OUTPUT STATUS" enables or disable the output generation. A green LED marks the enabled output. The "SAVE STAUS" button saves the entire delayer configuration: this is the delayer

configuration that will be loaded when the delayer will be switched on again after having being switched off. In order to save different configurations on a PC and recalling them afterwards, a drop-down menu is provided (see Figure 12 on the left). The provided help menu (Figure 12 on the right) is also very useful because it allows to open the instrument user manual and the possibility to check updates for the user manual and the software to the latest version through the internet. The '*COM Port'* dropdown-list box allows the user to select the correct serial COM for communicating with the delayer. A correct COM is mandatory for the communication with the delayer. A frame surrounds the dropdown-list box: its actual colour shows the serial link status. A red colour indicates a broken link, a green colour (as shown in Figure 11) indicates an established link. The "CLOSE COM & EXIT", as well as the "exit" item in the File menu and the usual "x" button at the top right corner, closes the serial link and terminates the program.

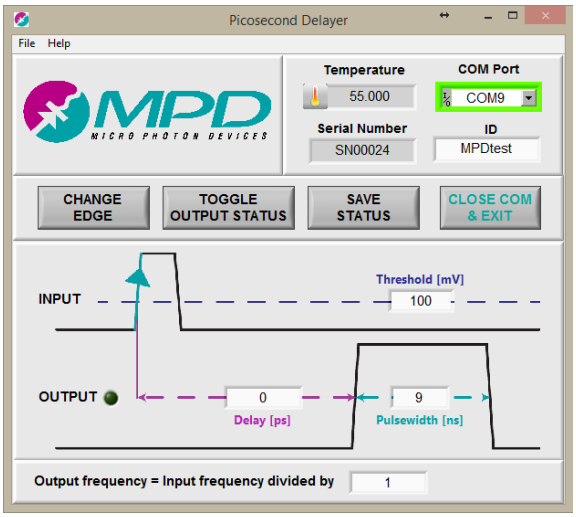


Figure 11. Screenshot of the graphic user interface.

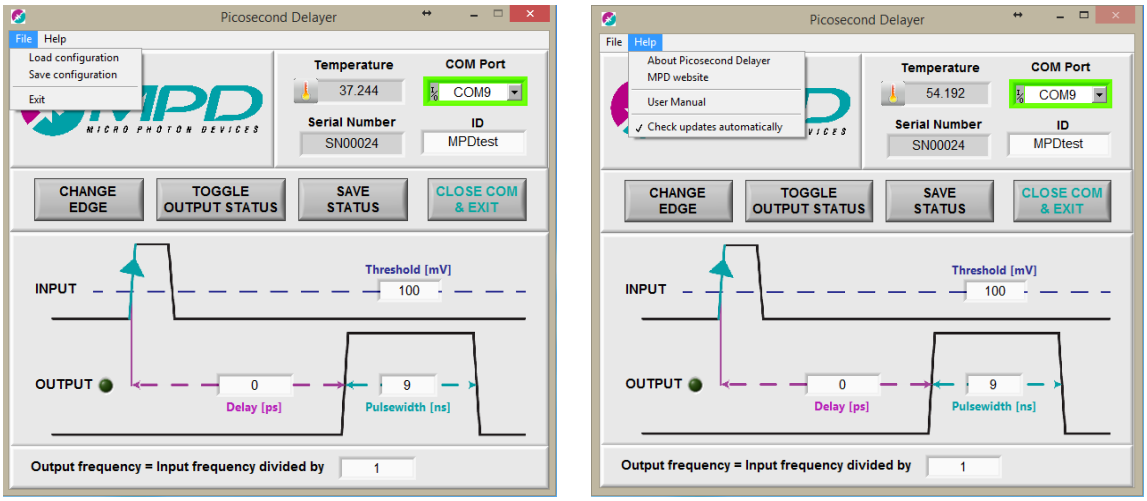
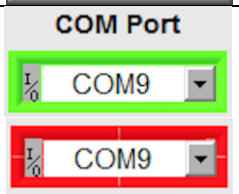
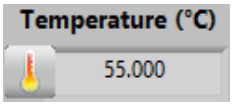
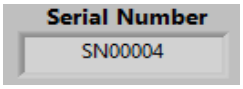
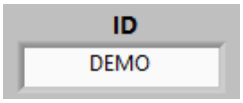
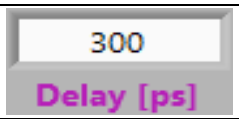


Figure 12. Screenshot of the graphic user interface (menus).

Table 3. Software interface explained.

	OUTPUT STATUS MODIFIER. By pressing this button, the output status changes from disabled to enabled or vice-versa. A virtual LED is switched on whenever the output is enabled.
	EDGE SELECTION. By pressing this button, the valid edge selection changes from low→high to high→low or vice-versa.
	SAVE STATUS. When this button is pressed, the actual configuration (input threshold level, input significant edge, delay, output pulse width and input frequency divider) overwrites the default configuration inside the module. This configuration is loaded at module power on. Please not that the default configuration does not include the output status which at power on is always disabled.
	CLOSE COM & EXIT. Stops the communication and exit the program.
	SERIAL LINK STATUS. A red frame around the COM port indicates a broken link, a green frame indicates an established link. The combo box shows ALL the connected COM ports. Thus the correct one must be chosen.
	BOARD TEMPERATURE. The box shows the board temperature. The temperature is controlled by a closed loop and should be stable at 55°C.
	SERIAL NUMBER. It shows the MPD serial number. The number matches the on the Picosecond delayer label on the back panel.
	ID. This is a custom name which the user can change to identify the instrument. The name inserted in the ID box is updated on the Picosecond Delayer and will be recalled on the next connection. The ID max length is 15 chars and cannot contain "#" or ";".
	DELAY. Sets t_{DELAY} on the picosecond delayer. The current value can be changed by simply modifying it and then pressing the enter key or by using the up and down directional keys. The delay is automatically updated when changed.
	THRESHOLD LEVEL. Sets and shows the threshold level (ThLev) on the input discriminator. The current value can be changed by simply modifying it and then pressing the enter key or by using the up and down directional keys. The threshold is automatically updated when changed.
	OUTPUT PULSE WIDTH. Sets and shows the NIM_OUT pulse width (t_{OUTPUT}). The current value can be changed by simply modifying it and then pressing the enter key or by using the up and down directional keys. The pulse width is automatically updated when changed.
	FREQUENCY DIVIDER. Sets/shows the input frequency divider factor. The current value can be changed by simply modifying it and then pressing the enter key or by using the up and down directional keys. The factor is automatically updated when changed.

Picosecond Delayer Serial Commands

Picosecond Delayer (PSD) also accepts remote commands through a Serial "Virtual COM". Commands are simply ASCII strings sent over the virtual COM by any program communicating with a RS-232. The delayer will process every transmitted command string only after having received the string terminator '#'. The PSD will also respond to every sent command with a string containing either the answer to a request or the actual applied new setting. When in Echo Mode (EM), the delayer will also reply back the string received with the serial interface. Echo Mode is enabled by default at power up and should never be turned off, when using the PC software, since it might interfere with its communication with the delayer. Table 4 shows the serial port configuration parameters.

Command strings are composed by the actual command followed by a hash (#):

command#
*(**#** is the string terminator)*

For understanding in detail all the possible commands, please refer to Table 5. It is worth noting also that in all the commands described in Table 5, the number of characters reserved for the parameters is not fixed. That's why there is an ASCII character (string terminator) that ends every command sent. For example, in case of setting a delay of 12ns the command string is *SD12000#*, but if only 1ns has to be set, then the command is *SD1000#*. Also all the strings sent from the PSD are always terminated with the '#'. Thus, following our example above, in case of sending SD1000#, the answer from the delayer would be 1000# with the EM disabled and SD1000#1000# with EM enabled; the 'echo' is always transmitted as first.

It is also possible to send more commands at the same time to the delayer: in these case the commands are simply "daisy-chained" one after the other separated by a ';'. Do not use any space to separate them. The termination hash is used only once at the end of the string with all the commands as shown here:

*command0**;**command1**;…;**commandx**#*
*(**;** is the string divider) (**#** is the string terminator)*

The answers from the PSD will never be daisy chained, even in case the commands were, but the answer's sequence will respect the transmitted sequence. Thus in case of sending SD100;SE1# the answer from the PSD would be 100#1# with the EM disabled and SD100;SE1#100#1# with the echo mode enabled (Echoes are also processed only after the receiving of the string terminator '#').

Table 4. Serial Port configuration parameters.

Bit per second	115200
Data Bits	8
Parity	None
Stop bits	1
Flow control	None

Table 5. Picosecond Delayer Serial Commands.

COMMAND	***DESCRIPTION***	***EXAMPLE***
SDxxxxx	***Set Delay*** Sets the preferred delay in ps. The delay can be set in 10 ps steps, from 0 to MAX-DELAY. In case of setting a delay value, whose units digit is different from 0, the delay is rounded to the nearest tens digit. The string sent back is the actual set delay.	Command = SD12346 Delay = 12.35 ns String sent from Board = 12350
SPxxxxx	***Set Pulse width*** Sets the preferred pulse width duration in ns. The pulse width duration has non linear steps from 1ns to 250ns. The duration is rounded to the nearest possible value. The string sent back is the actual set pulse-width duration.	Command = SP22 Pulse width duration = 21 ns (rounded from 22 ns to 21 ns) String sent from Board = 21
SHxxxxx	***Set tHreshold*** Sets the preferred threshold voltage in mV. The threshold can be set in 10 mV steps from -2V to +2V. In case of setting a value, whose units digit is different from 0, the threshold is rounded to the nearest tens digit. The string sent back is the actual set threshold.	Command = SH1505 threshold voltage = 1.5 V String sent from Board = 1500
SVxxx	***Set frequency diVider*** Sets the preferred frequency divider factor (integer) from 1 to 999. The string sent back is the actual set divider value.	Command = SV82 Frequency divider = 82 String sent from Board = 82
SEx	***Set Edge*** Sets the significant edge for the trigger in input. The command accepts only one digit (0 or 1) after the 'SE' string and sends back the same digit as reply. 0: high -> low; 1: low -> high.	Command = SE1 significant edge = low->high String sent from Board = 1
EOx	***Enable Output*** Enables or disables the output signal. The command accepts only one digit (0 or 1) after the 'EO' string and sends back the same digit as reply. 0: disable, 1: enable	Command = EO1 Output enabled String sent from Board = 1

EMx	***Echo Mode*** Enables/disables the echo mode. Echo Mode is enabled by default at power up. The command accepts only one digit (0 or 1) after the 'EM' string and sends back the same digit as reply. 0: disable; 1 enable	Command = EM1 echo mode = enabled String sent from Board = 1
SS	***Save Status*** Saves the current configuration, by overwriting the "PC saved" configuration, and makes it default. The string sent back is composed by the parameters saved: "Dxxxxx;" delay in ps "Pxxxxx;" pulsewidth in ns "Txxxxx;" threshold in mV "ESx;" significant edge low->high (1) or high->low (0) "Vxxx" frequency divider factor	Current threshold voltage = 1210 mV Current Delay = 12.3 ns Current significant edge = low->high Current pulse width duration = 21 ns Current divider = 100 Command = SS String sent from Board = D12300;P21;T1210;ES1;V100
RT	***Request Temperature*** Requests the current temperature of the board.	Current temperature = 52.15°C Command = RT String sent from Board= 52.150
RD	***Request Delay*** Requests the current delay of the board.	Current delay = 13230 ps Command = RD String sent from Board= 13230
RP	***Request Pulse width*** Requests the current pulse width duration of the board.	Current pulse width duration = 32 ns Command = RP String sent from Board= 32
RH	***Request tHreshold*** Requests the current threshold voltage of the board.	Current threshold voltage = 1210 mV Command = RH String sent from Board= 1210
RE	***Request Edge*** Requests the current significant edge for trigger in of the board.	Current significant edge = low->high Command = RE String sent from Board= 1
RO	***Request Output*** Requests the current output of the board.	Current output status = disabled Command = RO String sent from Board= 0
RV	***Request diVider*** Requests the current input frequency divider factor.	Current divider = 823 Command = RV String sent from Board= 823
RA	***Request All*** Requests the current parameters of the board. The string sent is composed by the following part: "Dxxxxx;" delay in ps "Pxxxxx;" pulsewidth in ns "Txxxxx;" threshold in mV "EOx;" enable output enable (1) or disable (0) "ESx;" significant edge low->high (1) or high->low (0) "Vxxx" frequency divider factor	Current output status = disabled Current threshold voltage = 1210 mV Current Delay = 12.3 ns Current significant edge = low->high Current pulse width duration = 21 ns Current divider = 100 Command = RA String sent from Board = D12300;P21;T1210;EO0;ES1;V100

RMD	***Request Maximum Delay*** Requests the maximum delay of the board.	Current Maximum delay = 51230 ps Command = RMD String sent from Board= 51230
RSN	***Request Serial Number*** Requests the MPD serial number of the board.	SN = SN00001 Command = RSN String sent from Board= SN00001
RID	***Request ID*** Requests the custom ID of the board. In case the current board's ID is null, the string sent back is a space ' '.	Current board's ID = 'test board' Command = RID String sent from Board = 'test board'
MID	***Memorize ID*** Saves the Custom ID of the board. Max length is 15 char. The name doesn't accept string separator (;) and string terminator (#) chars.	Command = MIDname ID = name String sent from Board = 'name'
RIPD	***Request Intrinsic Propagation Delay*** The returned value is expressed in ps.	Intrinsic propagation delay = 14.25 ns Command = RIPD String sent from Board = 14250
FV	***Request Firmware Version*** With this command, the delayer sends back the firmware version.	Current Firmware version = 5.1.2 Command = FV String sent from Board = 5.1.2
RHW	***Request Hardware Version*** Send back the hardware version.	Current Hardware version is 5.1 Command = RHW String sent from Board = 5.1
HSx	***High Speed mode*** Enables or disables the high speed mode. This command doesn't have any effect on OEM versions (as it disables the display refresh in order to achieve the maximum possible set-delay update rate). The command accepts only one digit (0 or 1) after the 'HS' string and sends back the same digit as reply. 0: disable, 1: enable	Command = HS1 High speed mode enabled String sent from Board = 1

Picosecond Delayer Serial Commands Error

In case of an unrecognized command, a request for a setting outside the PSD's limits or the impossibility to set a value, the command is not executed and an error is sent back instead of the expected reply. The number of characters reserved for the error string is fixed and equal to 5 chars: 'ERRxx' plus the string terminator '#'. The 'xx' value identifies the error type as described in Table 6. Of course, in case of command daisy chaining, each transmitted 'ERRxx' string will positioned, inside the PSD's replay sequence, as the command that

generated it: thus, following our previous example with the EM enabled, in case of sending SD100;SE2;SH3500#, the PSD reply would be SD100;SE2;SH3500#100#ERR01#ERR05#.

Table 6. Picosecond Delayer Serial Error Commands.

COMMAND	***DESCRIPTION***	
ERR01	***String Error***	The string is not recognized
ERR02	***Local Mode***	The Picosecond Delayer is in local mode and the parameters can't be set.
ERR03	***Max Divider***	The divider value is higher than 999. The parameter is not set.
ERR04	***Min Divider***	The divider value is lower than 1. The parameter is not set.
ERR05	***Max Threshold***	The Threshold value is higher than 2 V. The parameter is not set.
ERR06	***Min Threshold***	The Threshold value is lower than -2 V. The parameter is not set.
ERR07	***Max Delay***	The Delay value is higher than the Maximum Delay. The parameter is not set.
ERR08	***Min Delay***	The Delay value is lower than 0 ps. The parameter is not set.
ERR09	***Max Pulse Width***	The Pulse Width value is higher than 250 ns. The parameter is not set.
ERR10	***Min Pulse Width***	The Pulse Width value is lower than 1 ns. The parameter is not set.

Picosecond Delayer Firmware updates

In order to introduce new features or correct found bugs it is possible to update the Picosecond Delayer microcontroller's firmware and the CPLD firmware. This can be achieved only through the use of the USB Virtual COM port and a PC with Microsoft Windows® installed. Other solutions may exist but in order to explore this possibility the signing of a nondisclosure agreement and a subsequent conference call with the engineers at MPD will be needed. The following two paragraphs will describe the procedures and the software used to re-program the Picosecond Delayer's microcontroller and CPLD.

Reprogramming the delayer Microcontroller

The MPD picosecond delayer internally employs a microcontroller whose firmware can be reprogrammed in order to solve accidentally discovered bugs or for introducing new features and capabilities. The reprogramming of the firmware is done through the delayer virtual COM and by using the delayer firmware programmer which is shown in Figure 13. The programmer works only on Microsoft Windows® XP or higher

and is installed simply by double clicking the provided installer and following the on-screen instructions. When installed, by pressing the Load button the hex file is loaded by the software in the PC. By pressing the Read button it is possible to get the current version of the firmware in the microcontroller. By pressing the Program button the PC-loaded firmware is programmed in the microcontroller.

Reprogramming the delayer CPLD

The delayer unit employs also a CPLD which can be reprogrammed by software through the delayer virtual COM. The CPLD programmer is shown in Figure 14. The programmer works only on Microsoft Windows® XP or higher and is installed simply by double clicking the provided installer and following the on-screen instructions. After a successful installation, the program can be started by using the Start menu link. By pressing the program button the software asks for the jam file to be loaded in the CPLD, programs the CPLD and verifies that everything has ben done correctly.

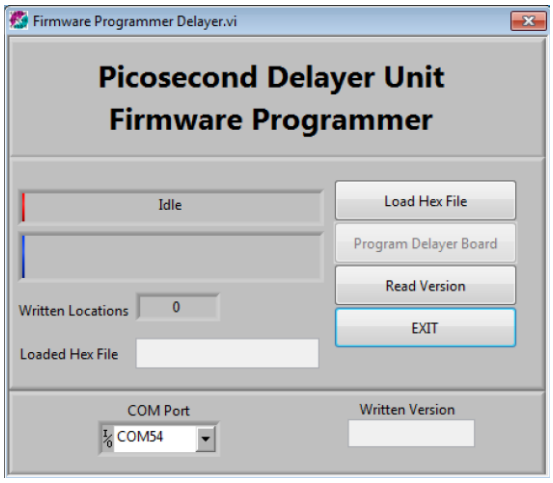


Figure 13. Delayer Unit Firmware programmer PC interface.

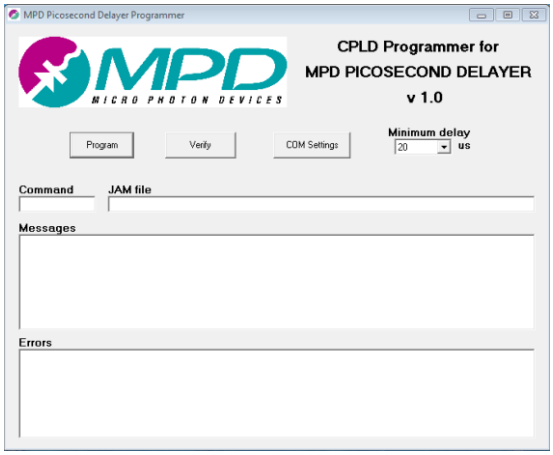


Figure 14. CPLD programmer PC interface.

The verify button verifies a jam file against the loaded CPLD code version. COM settings button allows to choose the serial COM speed and settings. Leave everything with the default values except for the correct COM number. The “minimum delay” dropdown list is for an internal parameter that should changed only in case of problems. Shorter times allows for slower CPLD reprogramming. With the set value the programming (loading + verifying) lasts ~12 minutes. If a too long value is set the CPLD might not be programmed correctly and will fail the verify.

Picosecond Delayer (Linux and Mac Os X)

The Picosecond Delayer can be effectively and successfully controlled even by computers operated by Linux or Mac Os X operating systems. This is possible because the MPD delayer modules employ internally an FTDI integrated circuits (IC) (www.ftdi.com). This IC allows an easy implementation of the bridging between the USB port to an UART interface. As a consequence, the only thing, needed for communicating with the MPD instrument, is creating on the host computer a Virtual Serial COM. For this purpose, then, the user is required only to correctly download, install and configure the FTDI appropriate drivers, by visiting the following web pages:

http://www.ftdichip.com/Support/Documents/InstallGuides.htm.

http://www.ftdichip.com/Drivers/VCP.htm

Please note also that in case of Linux, starting from Ubuntu 11.10, kernel 3.0.0-19, all FTDI devices are directly supported. Once the Virtual serial COM has been created, the Picosecond Delayer is controlled through the provided ASCII commands listed and explained in the previous paragraph. Expert users can also build their own visual interface using Xcode®, in case of Mac Os X, or their preferred tools normally employed in case of Linux.

Differences between HW v5 (or higher) and HW v4

With Picosecond delayer HW v5 we added the possibility to divide the input frequency by a positive integer, all the other characteristics remain unchanged. PC software version 6.x.x works seamlessly with both hardware v4 and v5. In case your delayer is HW version 4 the following differences, compared to what is described in this manual, will apply:

- The input frequency divider control box will be disabled (greyed out) and set to 1
- All the serial commands related to the divider will not be recognised; RA command will send back 1 fewer parameter (the divider)

System requirements

- USB 1.1 and 2.0 interface
- Host computer (minimum requirements)
  - 300 MHz processor and 256 MB of RAM
- Supported operating systems
  - Picosecond Delayer software
    - Microsoft Windows XP, Vista, 7, 8, 32 or 64 bit versions
  - Virtual COM
    - Microsoft Windows XP, Vista, 7, 8, 32 or 64 bit versions
    - Linux Ubuntu 12.04 LTS, Fedora Core 15 or compatible distributions, 32 or 64 bit versions. Different distributions should work, but were not tested.
    - Mac OS X 10.7.5 and above

Copyright and disclaimer