Trigger option for Alpao drive electronics

1 Introduction

The present document describes the trigger option for Alpao Drive Electronics.

This option is useful for the synchronization of the deformable mirror (DM) with an external hardware or vice-versa.

Signal	Usage	Example
Trigger input	Synchronize the DM motion with an external clock signal. The DM is a slave.	Apply the next frame of a predefined command pattern after the end of a camera exposure.
Trigger output	Synchronize external hardware with the DM motion. The DM is the master.	Monitor the exact timing of an adaptive optics loop using an oscilloscope.

Note: This version of document is applicable only if a PEX-292144 PCIe board and its specific cable are used to control the DM.

2 Software

The software examples are given for the Matlab language. The corresponding syntax for other languages can be found in Alpao SDK Programmers Guide.

2.1 Trigger input

To enable the trigger input, the property "TriggerIn" must be set to the value 2.

```
⇨ dm.Set('TriggerIn', 2);
```

Note: The "2" is for falling edge (rising edge "1" is not supported by this PCI card).

The trigger input signal must be disabled when not used:

```
⇨ dm.Set('TriggerIn', 0);
```

When sending value using the Alpao SDK, the first call to "Send" will be non-blocking, but the next calls will block until the end of the previous transfer (allowing the next frame to be computed while the hardware waits for the trigger pulse of the current frame).

More parameters for the trigger input can be found in the ALPAO Mirror SDK Programmers Guide (section 8).

2.2 Trigger output

The trigger output signal is always enable.

3 Typical timing

The following figure shows the timing measured with a standard DM97-15 device and transfer speed @ 2Mhz. (Values may change depending on the mirror type and data transfer speed)

- Data transfer starts when the **trigger-IN falling edge** is detected. In this example, the trigger signal is provided by a general purpose signal generator at 1 kHz. It is not possible to store a pattern and trigger only a part of this pattern with the PEX card. When the trigger In signal is received, the entire command or pattern will be sent to the mirror at the specified transfer speed.
- The **trigger-OUT signal** goes high when data are ready to be transferred and low when the data transfer is finished.
- Deformable mirror typical transfer time:

DM69-15/08	DM97-15/08	DM145-25	DM241-25	DM277-15
66.00 µs	69.50 µs	98.50 µs	130.25 µs	162.50 µs

- The jiter of the transfer time is lower than 1µs.
- Transfer begins with trigger-in and analog output (top trace) changes after it. In this example, the software has been programmed to send a square signal.
- There is virtually no delay (<2µs) between the end of the data transfer and its effect on the analog output.
- All analog outputs (i.e. all channels) are updated simultaneously.
- The mirror starts moving as soon as the analog signal changes. The DM settling time is documented in its test report (typically <0.5ms).

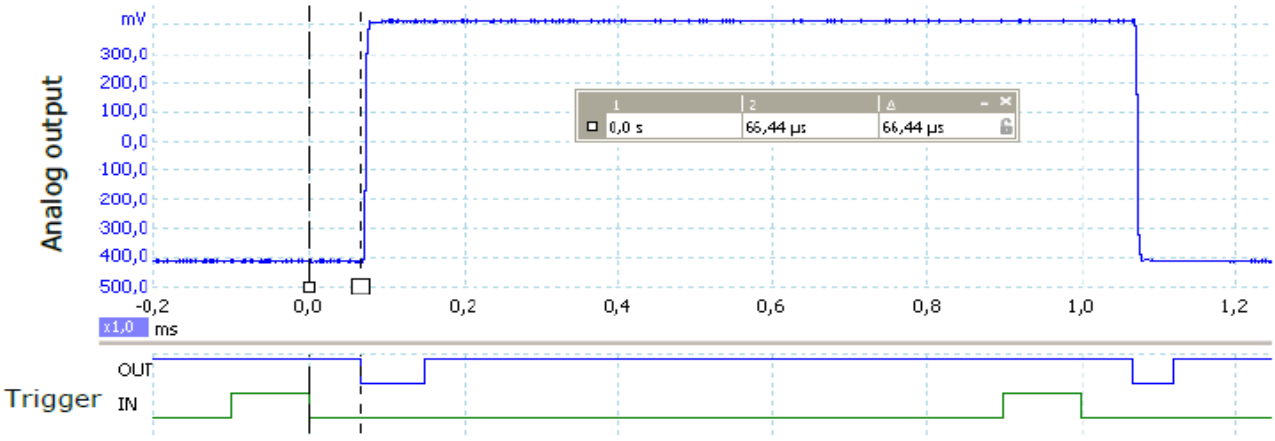

Figure 1 : Trigger In/Out and Analog response of DM

Channel A: Analog signal from electronics channel.
Digital "blue" (OUT): Trigger out of PEX card
Digital "green" (IN): Trigger input of PEX card

4 Hardware signals

External signals are provided by a specific cable converting from Sub-D9 to coaxial connector. 'Trig In' and 'Trig Out' are labelled on the cables.

Signal	Voltage level	Max current
Trigger IN	LVTTL 3.3V to 5V	0.8mA
Trigger OUT	LVTTL 3.3V	-10mA to +20mA

5 Loop triggering

The triggering can be used in a classic way. But another use can be to prepare a set of patterns to be send to the DM and send of pattern at each trigg IN signal.

For example, for a Z-scan along a sample using the defocus of the DM.

This is feasible using a thread as in annex next page (coded in c but can be done in every version of the SDKs – Matlab, Labview and Python).

If any questions remains after reading this technical note, please feel free to contact us for assistance: support@alpao.com.

example_wrapper_trigger:

```
#include "asdkWrapper.h"
#include <stdio.h>
#include <stdlib.h>
#include <Windows.h>

/* Number of predefined patterns */
#define N_PATTERNS 10

/* Wait for human action */
void pause()
{
    printf( "Press ENTER to exit... \n" );
    getchar();
}

/* Main program */
int main( int argc, char ** argv )
{
                UInt nbAct, idx, pattern;;
    COMPL_STAT ret;
    asdkDM * dm = NULL;
    Scalar tmp;

                UInt       nbPatterns = N_PATTERNS;
                Scalar *  datas[N_PATTERNS];

                char   serialName[128] = "";

                /* Get serial number */
                printf("Please enter the S/N within the following format:
BXXYYY (see DM backside)\n");
                scanf_s("%s", serialName, sizeof(serialName));

                /* Load configuration file */
                dm = asdkInit(serialName);

    if ( dm == NULL )
    {
        /* Print last error */
        asdkPrintLastError();
        pause();
        return -1;
    }

    /* Get the number of actuators */
    ret = asdkGet( dm, "NbOfActuator", &tmp );
    nbAct = (UInt) tmp;

    /* Check errors */
    if ( ret != SUCCESS )
    {
        /* Print last error */
        asdkPrintLastError();
        pause();
        return -1;
```

```
    }

    printf( "Number of actuators: %d\n", nbAct );

                if (nbPatterns < 2) {
                        printf("Number of patterns should be strictly higher
than 1\n");
                        return -1;
                }

                /* Initialize patterns, for example, with offset from -0.1
                to 0.1 */
                for (pattern = 0; pattern < nbPatterns; pattern++) {
                        datas[pattern] = (double*)calloc(nbAct,
sizeof(double));
                        for (idx = 0; idx < nbAct; idx++) {
                                datas[pattern][idx] = -0.1 + 0.20*pattern /
((double)nbPatterns-1);
                        }
                }

                /* Enable trigger mode */
                ret = asdkSet(dm, "TriggerIn", 2);


                /* Set Timeout of, e.g., 5 seconds */
                ret = asdkSet(dm, "Timeout", 5);

                /* Check errors */
                if (ret != SUCCESS) {
                        asdkPrintLastError();
                        pause();
                        return -1;
                }

                /* Begin sending patterns to the DM */
                for (pattern = 0; pattern < nbPatterns; pattern++) {

                        /*
                        - The first call to this function will prepare
                        the pattern to be sent to the DM.
                        - The second call will block until a trigger signal
                        is sent.
                        - Once the trigger signal received, the first
                        pattern will be
                        sent and the second pattern will be ready to be sent
                        and
                        so on until the end of the pattern list
                        */

                        ret = asdkSend(dm, datas[pattern]);

                        /* Check errors */
                        if (ret != SUCCESS) {
                                asdkPrintLastError();
                                pause();
                                return -1;
```

```
                        }
                }

                /* Release memory */
                for (pattern = 0; pattern < nbPatterns; pattern++) {
                        free(datas[pattern]);
                }

        /* Reset mirror values */
        asdkReset( dm );

        /* Release */
        asdkRelease( dm );
        dm = NULL;

        /* Print last error if any */
        asdkPrintLastError();

        pause();
        return 0;
}
```