MICROMIRROR ARRAY (MMA): A PROMISING CANDIDATE FOR MICRO – ELECTROMECHANICAL SYSTEMS (MEMS)

Kittiwat Kamlungsua
DESY Summer Student 2013, Chulalongkorn University, Thailand

Supervisors
Andreas Przystawik and **Sergey Usenko** (Ph.D.)

September 5th, 2013

Abstract

This summer student report contains all important data for controlling the MMA operation cycle, including basic instrumental information about the MEMS Phase Former Kit by Fraunhofer IPMS and programming in LabVIEW. Besides, the front panel and the block diagram are successfully created and their functions still work well for the basic MMA operation. However, further advanced optimization process is required to enhance the program's performance.

ACKNOWLEDGEMENT

First of all, I am extremely grateful to Andreas Przystawik, my supervisor, for helping me in every way even though I am a chemist, having no experience in programming before. Besides, the thankfulness would come to Sergey Usenko, Ph.D., and Elisabeth, a summer student fellow. They are also very kind and friendly and always persuade me to have a conversation more or less although I'm not quite talkative. On top of that, the very encouraging wholehearted comments are from Tim Laarmann, the group leader of the Femtochemistry.

Next I would like to thank Olaf Behnke, Andrea Schrader, and Doris Eckstein for organizing this very impressive DESY summer school. This city is very nice and suitable for studying any disciplines. I will be missing this place where all of summer student spent their time together. Thank you again for everything. It will be in my memory forever.

Of course, many summer student fellows deserve the praise. Sometimes, we have just a little chat. Sometimes, we just meet at the kitchen and also just say 'Hi'. Anyway, they are friendly and amicable, I bet. I hope that we will meet again and can create a huge connection of young scientists from all over the world. Thank for your friendship, fellows.

Finally I would like to thank NSTDA under the patronage of HRH Princess MahaChaki Sirinhorn for this invaluable chance to participate this prefect summer camp and I hope that I will bring the knowledge I have experience to develop the country more or less.

Contents.

The importance of micro – electromechanical system (MEMS)	4
MEMS Phase Former Kit by Fraunhofer IMPS	9
Graphical User Interface (GUI) for Autonomous MMA operation	19
Simulation in LabVIEW	25
Programming and Simulation in LabVIEW	27
Conclusion	46
References	47

CHAPTER 1

The importance of micro – electromechanical system (MEMS)

1. Root of the problem.

Optics and Photonics are branches of Physics dealing with the study of behaviors and properties of light and optically – involving electronic devices. However, the development of so – called optical wavefront controls has increasingly grown in popularity and importance nowadays. This piece of research plays a crucial role in various disciplines, covering scales from enormous as in astronomical telescope to extremely small as in retinal microscopic imaging. An example of the astronomical problem is demonstrated.

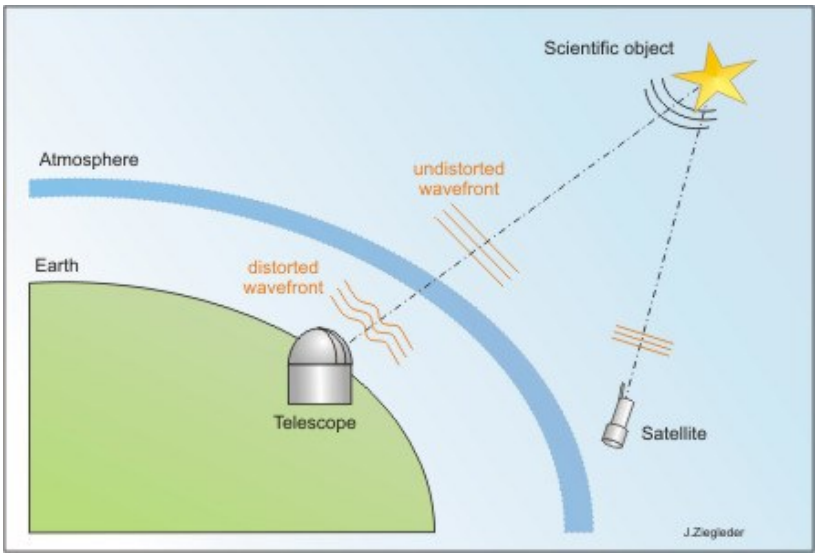


Figure 1.1 The distortion of the wavefront, one major problem in Astronomy.

One major challenge in astronomical observations is called ‘wavefront aberration’ caused by the atmospheric turbulence. This phenomenon renders the wavefront of the incoming light from distant stars distorted when passing through the Earth’s atmosphere. To illustrate, the atmosphere must be perceived as a set of air bubbles having slightly different properties such as temperature and pressure. According to the Snell’s Law, the speed of light varies in different medium of different properties. Therefore, the light travels with different speed after coming into the Earth. In other words, its wavefront is changed by the atmosphere, resulting in the data containing errors.

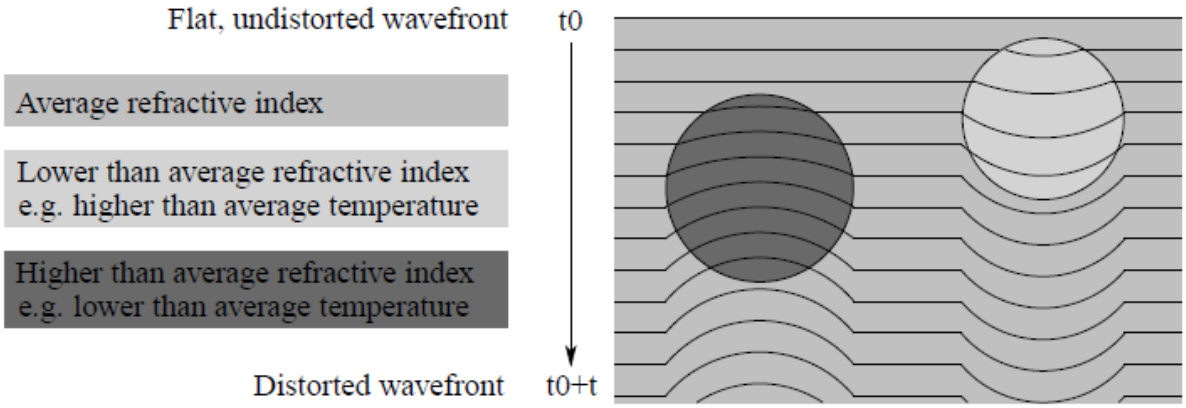


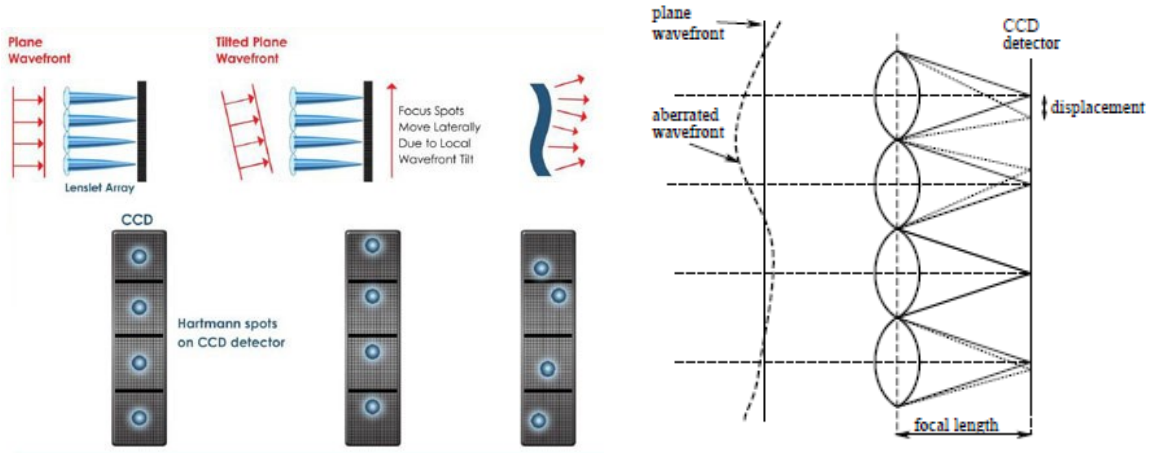
Figure 1.2 The graphical representation of the Earth's atmosphere. The circles with different shades represent the air bubbles with different properties.

2. Adaptive – optics system.

To solve the aforementioned problems, an adaptive – optics system becomes a promising solution. The adaptive - optics system is a state – of – the – art technology used to reduce the effect of wavefront distortions, thus enhancing the performance of the optic system. Most typical adaptive – optics systems consist of five main parts as follows.

1. *Wavefront corrector*. The wavefront corrector performs the physical correction of the distorted wavefront. It comes in many types, sizes and shapes. As named, its purpose is to optically compensate the incoming distorted wavefront to be as correct as possible compared to the original one before perturbed by the wavefront – distorting medium. After the correction, the light is reflected to the beam splitter.
2. *Dichroic beam splitter*. The beam splitter is used to divide the corrected light beam from the wavefront corrector into two parts. One is sampled by the wavefront sensor for the analysis and optimization. The other goes to the detector for demonstration.
3. *Wavefront sensor*. The wavefront sensor is employed to measure the phase aberration and subsequently sends the data to the wavefront control system for iterating computation. Due to its simplicity and manufacturability, Shack Hartmann sensor (SHS) is responsible for the measurement, which is called 'slope method'. The sensor consists of the array of miniature lenslets focusing the wavefront onto a Charge Couple Device (CCD)

camera. The local displacement of each lenslet from the normal wavefront is then measured and integrated for reconstructing the wavefront shape. Other types of the sensors also exist like Curvature Sensor (CS) and Pyramid Sensor (PS).



(a) (b)

Figure 1.3 (a) Shack Hartmann Wavefront Sensor and (**b)** slope measurement method.

4. *Wavefront control system*. The wavefront control system is actually computer software automatically performing iterating calculations. It receives the measurements from the sensor and calculates the corrective movement of the wavefront corrector. Strehl ratio, one necessary value for the calculation, indicates the degree of compensation, which is inversely proportional to the variance of the wavefront aberration. Therefore, new measurements and more corrective movement are received and done repetitively to mathematically minimize this variance, which is the ultimate goal of the system.
5. *Detector and High – resolution camera*.

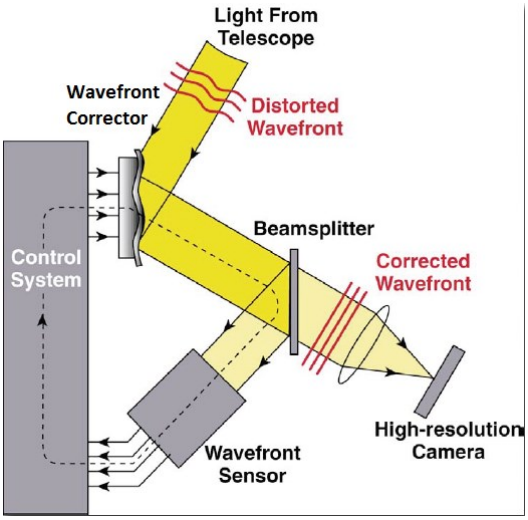


Figure 1.4 Closed – loop feedback control system for typical adaptive optics systems.

3. Dynamic and static adaptive – phase devices.

The most important part of the adaptive – optics system is definitely the wavefront corrector because the quality and degree of phase compensation depends on how well – organized and well – fabricated it is. The wavefront corrector comes in variety but it is, generally speaking, coarsely divided into two main types.

1. *Static adaptive – phase device.* Static devices imprint a fixed phase pattern by either reflective or transmissive means. Their well - known advantages are their high quality of production and ease of use. Such devices always come to users' mind if one is interested in only a specific phase pattern. However, the specificity inherently becomes the major severe problem if more than one pattern is required.

2. *Dynamic adaptive – phase device.* Dynamic devices, or spatial light modulators, provide more pattern flexibility for users. They are mainly classified into two types, liquid crystals (LCs) and deformable mirrors (DMs). Speaking of the liquid crystals, the operational speed is relatively comparable to that of the deformable mirrors and they still have better resolution and lifetime expectation. Yet, a common severe drawback which makes most users reject them is the suffering from the unwanted polarization effect. In case of the deformable mirrors, they are not affected by polarization effect and, even better, able to attain reflectivity close to 100% due to the specular reflection. Those latter reasons make the deformable mirrors prevailing and become the better choice than the liquid crystals.

4. Deformable mirrors and micro – electromechanical system (MEMS).

Deformable mirrors (DMs) are mirrors whose surface can be deformed, mostly electrostatically – driven mechanism. They are designed to achieve wavefront control and the correction of optical aberrations and used in conjunction with the wavefront sensor and the real – time processing wavefront control system. Several types of these deformable mirrors are available which

are different in fabrications and mechanisms of deformation but have the same ultimate goal to compensate for the wavefront distortion. For example, segmented DMs which are light – weight, scalable and have high – quality surface, membrane DMs which are made of conductive reflective metallic membrane, bimorph DMs and ferrofluid DMs. However, all of them cannot still break through the high price threshold of the conventional spatial light modulators.

Recently, micro – electromechanical systems (MEMS) have been introduced as an interesting implementation. They are the state – of – the – art photonic technology which has been of increasingly growing importance among scientists. They comprise a large array of micron – sized mirrors (10^4 – 10^5 mirrors) independently acting as piston/tip – tilt actuators. Also, the micromirrors are usually fabricated on top of the CMOS circuitry with cutting – edge bulk and surface micromachining techniques. How each micromirror works is based on electrostatically – driven piston mechanism by which the micromirror is originally be at the highest position. When the voltage is applied to each address electrode below the mirror, the electrostatic attraction causes the mirror to bend towards the electrode. As a result, the mirror is said to be deflected from the normal position. Typically, MEMS has several advantages over other deformable mirrors. For example, high response rate, high precision, no hysteresis effect, large temporal bandwidth, compactness and cheapness. Because of these good points, MEMS becomes widely used in various disciplines from scientific to industrial application as in medical imaging, laser pulse technology, open – air communication, astronomy, ophthalmology and weapon tracking. In this report, the MEMS provided by Fraunhofer IPMS will be studied in greater details.

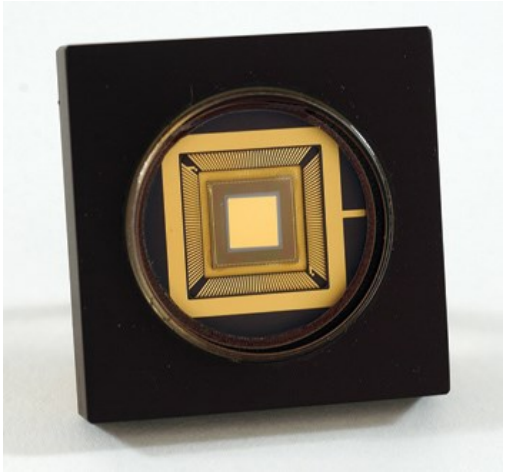


Figure 1.5 Micro – electromechanical system (MEMS).

CHAPTER 2

MEMS Phase Former Kit by Fraunhofer IMPS

The MEMS Phase Former Kit is a piston – type micromirror array provided by Fraunhofer IPMS. It is also a complete spatial light modulator, used for high – resolution, high – precision and high – speed wavefront control and designed for the convenient integration into the user's own application. Apart from the micromirror itself, it still consists of the complete electronic devices with easy – to – use software interface for any operating systems. All detailed description related to its components and how it works has been introduced in this chapter.

1. Kit components.

1.1 Micromirror Array Board (MMA Board)

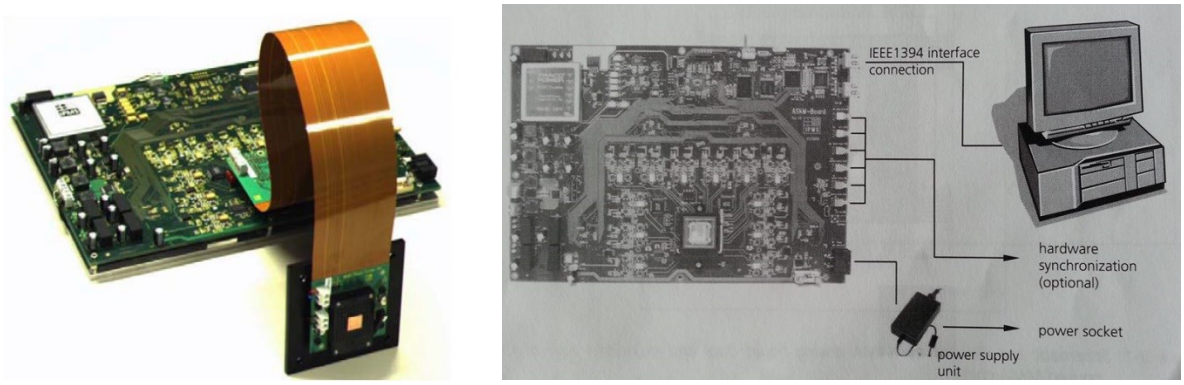


Figure 2.1 (left) MMA board with ZIF socket shown and **(right)** MMA board correct wiring.

MMA board is a central printed MMA driving board containing all necessary electronic components. It also acts as a headquarter to control other parts of the whole MEMS and to transfer data from internal computational system to external sources and vice versa. Provided by Fraunhofer, it has many interesting special features and provisions different from other integrated circuit boards as follow.

- Easily – replaceable ZIF socket.
- On – board power supply for logic electronics, Data Acquisitions (DACs), amplifiers and the MMA device.
- Digital – to – analog data conversion and the amplification of MMA data.
- High – speed data communication provided by the IEEE 1394

Firewire interface.

- Surveillance of the MMA operational conditions.
- Software and hardware synchronization with external devices.

In order to achieve the smooth normal operation, the PC and the power supply are essentially required to connect to the board via IEEE 1394 Firewire interface. The two available on – board IEEE connectors are equivalent and inter - exchangeable. Furthermore, a rigid Al back plate is closely attached to the rear for the purposes of better heat dissipation, mechanical stability mounting of the optical instrumentation.

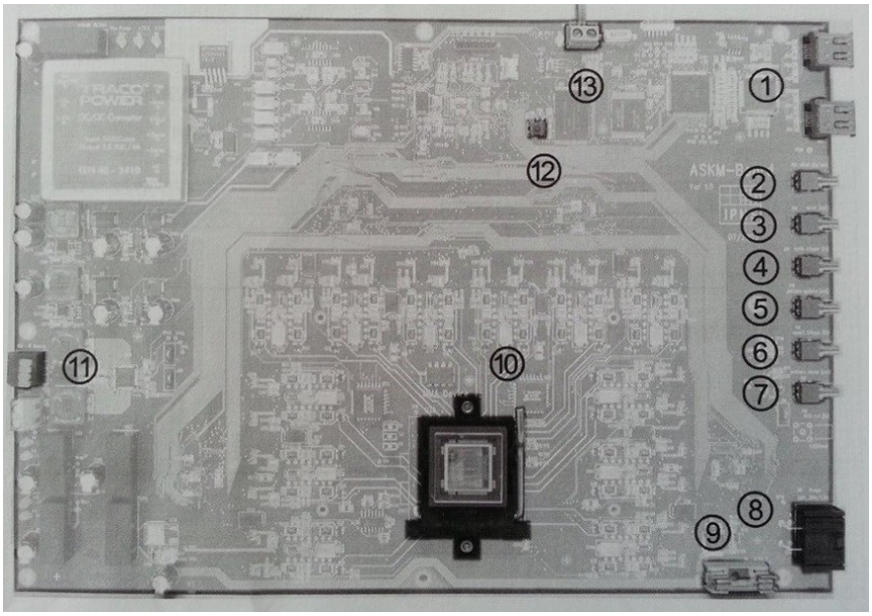


Figure 2.2 MMA board connectors and buttons. (1) IEEE 1394 Firewire connectors, (2) – (7) Ports for external hardware control, (8) Power supply connector 20 … 26 V/2.5A, (9) Fuse 2.5 A, time – lag, (10) ZIF socket for the MMA device, (11) Connector for temperature control, (12) Reset button and (13) Board temperature connector.

1.2 Micromirror Array Chip (MMA Chip)

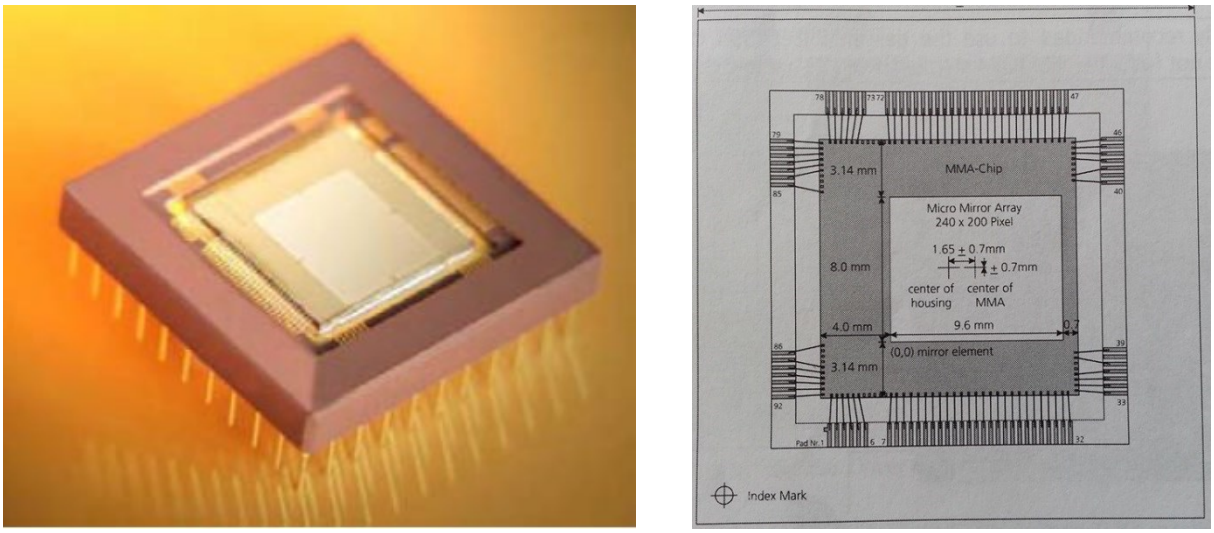


Figure 2.3 (left) MMA chip and **(right)** detailed description of housed MMA chip.

Micromirror array chip is an electronic device containing an array of functional micromirrors. The chip provided by Fraunhofer IPMS has several outstanding characteristics as shown below.

- Integrated 200 X 240 micromirror arrays.
- Chip programming > 5 kHz.
- Overall data transfer up to 100 fps.

1.3 Micromirror Array

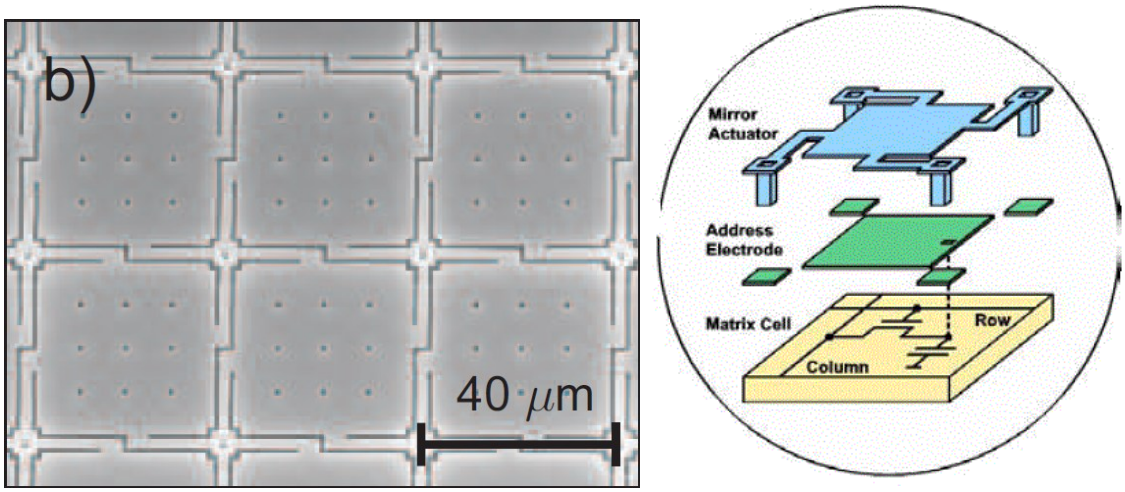


Figure 2.4 (left) Magnified image of micromirror array and **(right)** the diagram of each mirror element.

Micromirror array is a functional building block of the MEMS Phase Former Kit. Unlike other types of deformable mirrors, many properties have been provided by Fraunhofer IPMS as followings.

- Single – element size of 40 X 40 μm^2.
- Mirrors suspended from their supporting posts by flexible hinges.
- Substrate fabricated from a monolithic aluminum – metal – alloy integrated on top of a CMOS circuitry.
- Independently – addressing electrodes and 8 bit – resolution deflection.
- Possible range of deflection up to 400 nm.
- 2¶ - phase modulation suitability for the visible spectral region.
- Long recovery time after deflection, preventing from the permanent deformation of the microscopic hinges.
- Operation cycle with 1:17 on/off rate.

The mechanism of deflection is based on electrostatically – driven piston mechanism. Originally, the hinge is positioned at the highest level or, in other words, at the same level as the post. When the voltage is applied to a particular address electrode, the positive charge is generated on the electrode whereas the negative charge on the mirror. Consequently, the electrostatic attraction causes the mirror to bend towards the electrode or deflect from the original position. The degree of deflection varies with the amount of applied voltage.

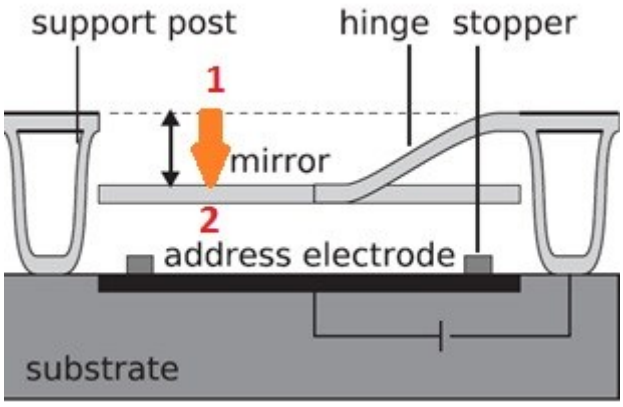


Figure 2.5 Mechanism of how each element can be deflected. The mirror moves from the position 1 to 2 after the voltage is applied, resulting in the ‘deflection’.

2. Dataflow in the closed – loop MMA operation.

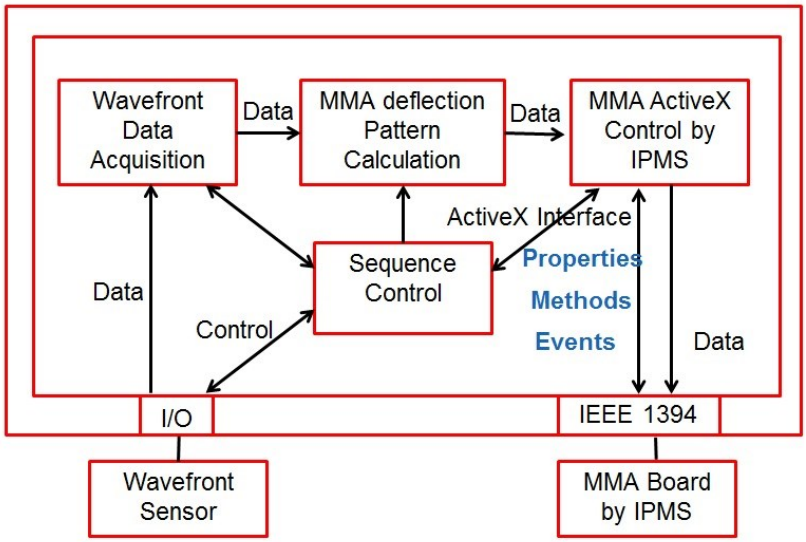


Figure 2.6 Software integration block diagram.

The ActiveX control provides the user with the incorporation of the MMA data transfer and the control of functionalities into his own software environment, thus enabling the automation of the data transfer from data sources to the MMA for closed – loop operation. The interaction between the user's controlling software and the ActiveX control is illustrated in figure 2.6 Three items called properties, methods and events describe the interface of an ActiveX control. Properties typically define the behavior of the control. In this case, they are primarily used to set various configuration values of the MMA board and the chip. Moreover, some certain assigned functions make them readable and editable during the run – time simulation. Speaking of methods, they are meant to initiate actions of the ActiveX control and able to call parameters and return values. Most of the methods here are used for the data transfer and the execution of the commands on the MMA board. Finally, events can be understood as a kind of software trigger.

3. General data transfer.

3.1 Mode of data transfer from PC to MMA board.

- Single Pattern Mode

Figure 2.7 Only a single pattern is transferred in Single Pattern Mode.

In Graphical User Interface, only a single pattern currently selected from the file list in the GUI mainframe is transferred to the MMA board only if the 'load' button (in LabVIEW, UPLOAD) is activated.

- Sequence Mode

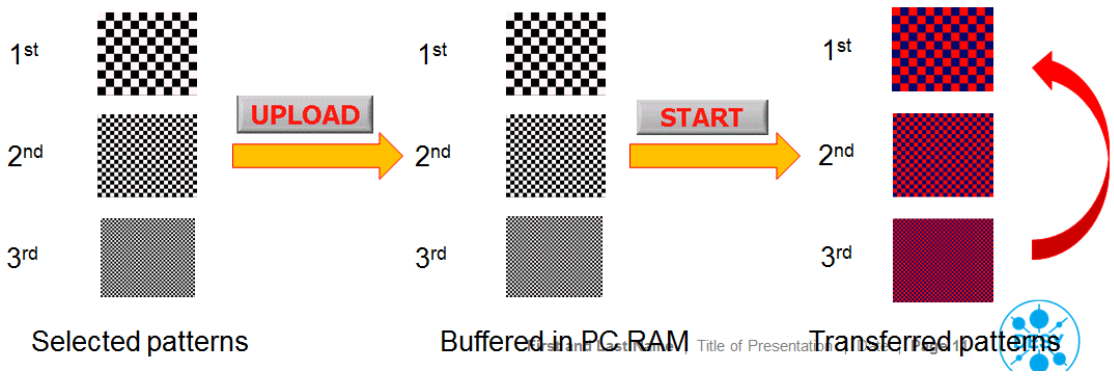


Figure 2.8 Several data patterns are sequentially transferred in Sequence Mode.

In Graphical User Interface, the user is allowed to select several data patterns at a time. When the 'load' button (in LabVIEW, UPLOAD) is pressed, all selected data are first buffered in the PC RAM waiting for another transfer. Then the buffered data patterns are sequentially transferred to the board after the activation of the 'start' button (in LabVIEW, START) on the GUI mainframe, with regard to the order they have been chosen. Anyway, the procedure automatically keeps going repetitively unless the 'stop' button is pressed.

3.2 MMA operation cycles.

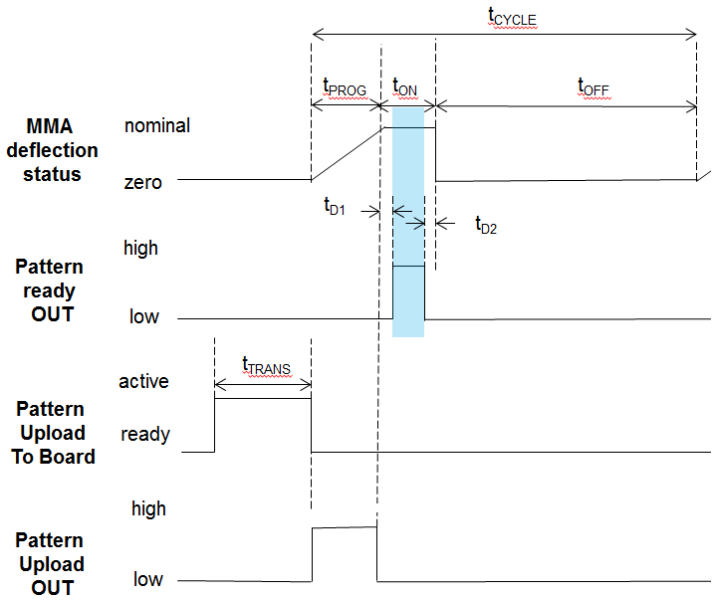


Figure 2.9 Typical MMA operation cycle.

Typical MMA operation cycles are performed based on a timely discrete basis. Firstly, the user is required to select data patterns and then press the button 'load' on the GUI mainframe. After that, 'Pattern Upload To Board's status changes from ready to active, waiting for the time t_{TRANS} to expire. Following the expiration of the time t_{TRANS} is the change in the status of 'Pattern Upload OUT' from low to high, indicating that the data patterns are being transferred. At the same time, the MMA deflection status gradually increases from zero to a nominal value until it levels off and the data transfer is also completely finished ,which is illustrated by the falling edge of the 'Pattern Upload OUT's status. This period of time is called t_{PROG}. However, the 'MMA deflection status' remains at the nominal value for a period of time t_{ON}. Within this duration, the user is allowed to adjust the time margins t_{D1} and t_{D2} for fine – tuning the 'Pattern Ready Signal' t_{PR}. The purpose of fine – tuning is for special trigger modes. When the time t_{ON} is no longer valid, the deflection status returns to zero for a long period of time t_{OFF}. Notice that the time t_{OFF} has been made user – accessible to facilitate the synchronization with the external hardware. Anyway, one operation cycle includes three periods of time, t_{PROG}, t_{ON} and $t_{OFF,}$ which leads to the definition of the term t_{CYCLE}.

In term of programming, a new MMA operation cycle is always done after the complete upload of the previous data patterns onto the on – board RAM. Yet, the new data patterns available in the RAM buffer cannot be written

into the MMA unless the previous operation cycle is truly finished. Therefore, the maximum possible re –programming rate depends on the MMA cycle rate as shown in the expression below.

$$f_{CYCLE} = \frac{D}{t_{ON}} \; and \; D = \frac{t_{ON}}{t_{CYCLE}} \; \ldots equation\ 1$$

where f_{CYCLE} represents the rate of operation cycle and D is the duty factor. More importantly, the frame rate or the overall data transfer must be properly adjusted and should not be equal to or greater than the rate of operation cycle. If not, the new MMA pattern may not be fed into the MMA in time.

3.3 Modes of Cycle Trigger

- *Auto*

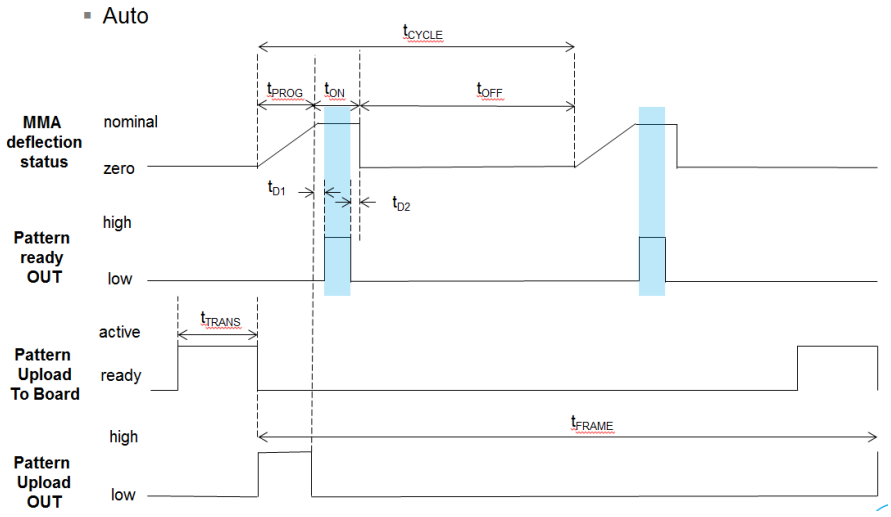


Figure 2.10 Auto trigger mode.

In case of 'Auto', each MMA operation cycle automatically starts after the previous cycle has finished, taking either already – existing pattern or the new one into the operation. Repeated programming with the same pattern is basically suitable for this triggering mode.

- *New Pattern*

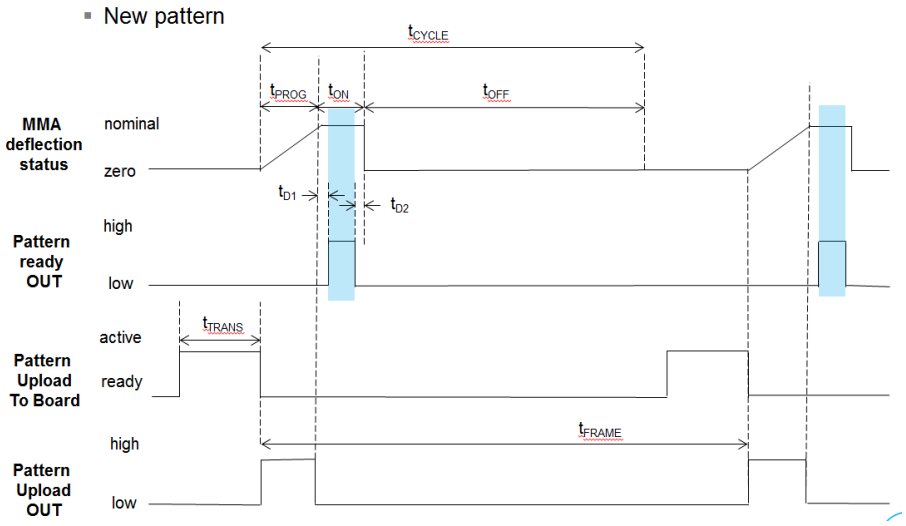


Figure 2.11 New Pattern trigger mode.

For 'New Pattern', a new MMA operation cycle is started only upon a switchover of the memory buffers after a complete upload of a new pattern to the on - board RAM. As a result, this triggering mode provides the synchronization of the pattern transfer to the board and the MMA programming. Anyway, proper and careful time adaptation is required in order to avoid any data overwriting.

- *External*

It is possible that a new MMA operation cycle can be started by an external Cycle Trigger IN signal applied via an external port on MMA board.

4. Timing conditions

The following equations and the table are the summaries of the timing conditions mentioned in the previous chapters.

$$D = \frac{t_{ON}}{t_{CYCLE}} \leq 5\% \ldots equation\ 2$$

$$t_{CYCLE} = t_{PROG} + t_{ON} + t_{OFF} \ldots equation\ 3$$

$$t_{ON} = 0.2 \ldots 1000\ ms = t_{D1} + t_{D2} + t_{PR} \ldots equation\ 4$$

$$t_{OFF} \geq 17 * t_{ON} \ldots equation\ 5$$

$$f_{FRAME} \leq f_{CYCLE} \ldots equation\ 6$$

SYMBOL	MIN	TYP	MAX	UNIT	COMMENT
t_{PROG}					$t_{PROG} = 1100 / f_{CLK}$
t_{ON}	0.2		1000	ms	
t_{OFF}	17* t_{ON}				
f_{CLK}		1000		kHz	
t_{D1}	50			µs	$t_{ON} > t_{D1} + t_{D2}$
t_{D2}	50			µs	$t_{ON} > t_{D1} + t_{D2}$
t_{TRANS}		15		ms	Depends on PC hardware
t_{FRAME}	t_{CYCLE}				

Table 2.1 Summary of important timing conditions and relationships.

CHAPTER 3

Graphical User Interface (GUI) for Autonomous MMA operation

The Graphical User Interface (GUI) provides the user with a convenient and uncomplicated access to MMA programming with individual deflection patterns and to the hardware control features for the MMA autonomous operation. The data source of the GUI is based on pre – defined data patterns previously stored by the user in the Windows file systems using a common file format (Bitmap or ASCII). Also, the selection of a single pattern or complete pattern sequences for transferring to the MMA board is allowed in the GUI. In this chapter, some fundamental requirements for the operation and basic instructional commands in GUI are introduced here.

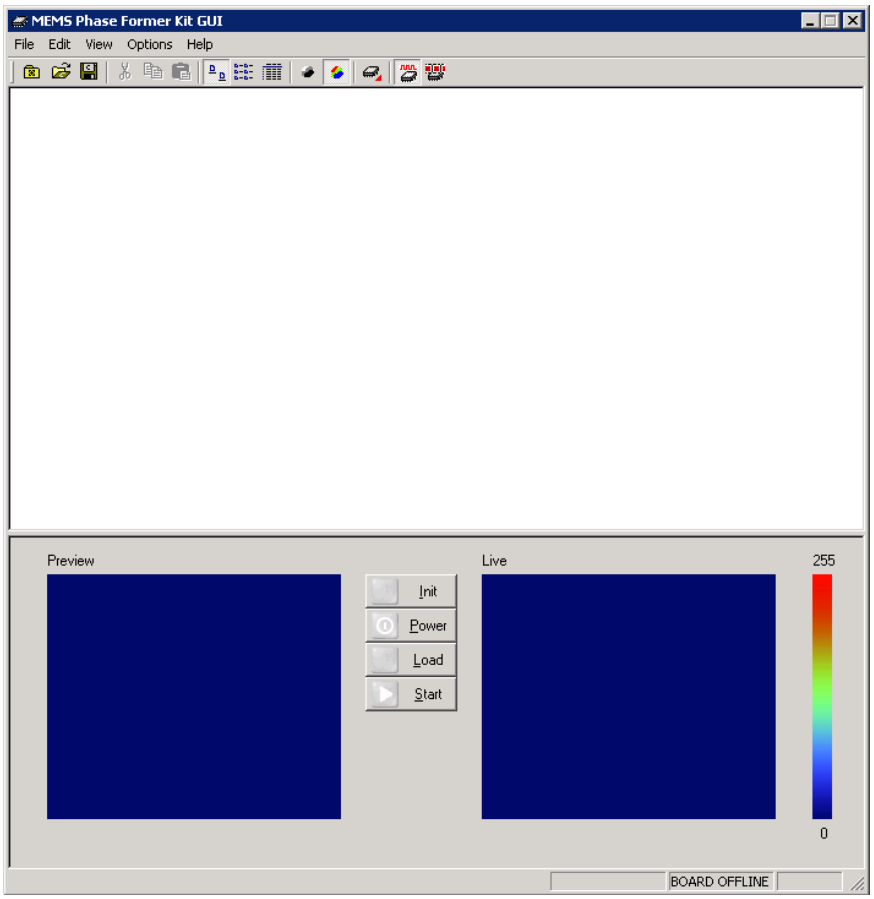


Figure 3.1 GUI mainframe.

1. Loading patterns.

In order to load data patterns into the MMA, a directory containing the Bitmap or ASCII data files must be selected by pressing  which is embedded onto the toolbar. The selected patterns will be displayed in the lower left preview frame, getting ready for the transfer to the MMA board. Multiple selections can be done by pressing the Shift or Ctrl key while selecting them.

Speaking of the input files for GUI, each file must contain 240 X 200 elements of the mirror deflection arranged in the same orientation as appearing on the MMA chip. The (0,0) – mirror element is located at the lower left of the chip. Thereby, the degree of deflection has to be linearly encoded by an 8 – bit value and the maximum deflection (tilt or stroke) is represented by the number 255. Anyway, two acceptable file types are valid by GUI.

1. *Window Bitmap Format (extension BMP)*. It must be 240 X 200 pixel monochrome ranging from 4 to 32 bit color resolution. Additionally, both grayscale and colored images are allowed but only 8 – bit values are solely for colored images.

2. *ASCII Format (extension ASC or CSV)*. It must also be 240 X 200 pixel with 8 – bit values as well, ranging from 0 to 255. Each column is separated by a comma, semicolon or tabulator and each line with CR/LF.

If a particular pattern is larger than 240 X 200 pixels, other portions in the upper right exceeding this boundary will be truncated. In contrast, if the pattern is smaller than 240 X 200 pixels, all missing pixels in the upper right part are supplemented with an entry zero.

2. Basic board operations.

There are four main controlling buttons in the GUI mainframe that you should carefully consider and think twice before using them, tabulated as follow.

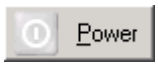
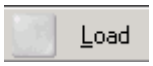
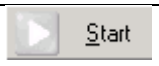
Operation	Description
	Four main actions will be proceeded if this button is activated, the loading of the driver for IEEE 1394, the initialization of the interface, the resetting of the MMA driving board and the transfer of all initial parameters required for the initialization. Moreover, a selection dialogue box appears if more than one MMA board is connected. Anyway, the button becomes enabled when the initialization succeeds.
	When the 'Power' button is activated, it causes the board supply voltage to be applied to the MMA board and to energize the amplifiers for the analog MMA data channels as well. In case of power – off state, all pins of the ZIF socket are idle and the MMA device can be exchanged without any risk.
	The pre – selected files in the list of the GUI mainframe will be transferred to the board when this button is pressed. Two modes of data transfer are involved, single pattern mode and sequence mode as mentioned in the previous chapter.
	Using the Start button can really start and terminate the MMA operation cycle

Table 3.1 Description of all necessary buttons in the GUI mainframe.

3. Configurations.

3.1 General parameters.

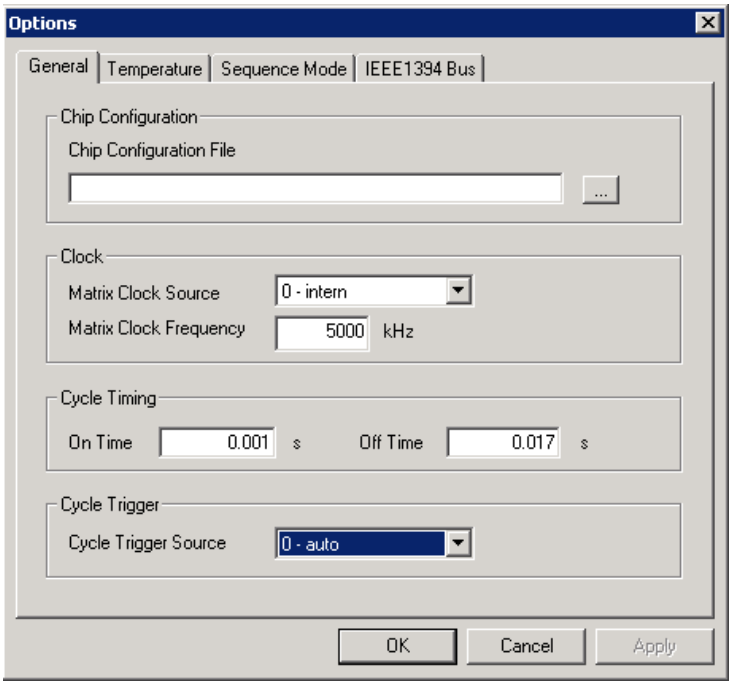


Figure 3.2 Options Dialog – Page General.

- *Chip Configuration.* The chip configuration file contains all important chip – related characteristic data needed for the proper and smooth MMA operation. It is obligatory that the user select this file. Otherwise, the initialization of the MMA driving board fails with an error message shown. Be careful, wrong configuration files with potentially – problematic data patterns may result in incorrect deflection values. Even worse, the MMA board and other electronic components can be severely damaged.

- *Cycle Timing.* 'On Time" denotes the time of the nominal deflection, i.e. the time during which each deflection pattern is statically applied to the MMA. In term of the 'Off Time", the values which are greater than 17 times of the 'On Time' are acceptable.

- *Cycle Trigger.* The description of how each mode works is already described in Chapter 2. Keep in mind that this triggering mode is set to be read – only during the MMA operation cycle.

3.2 Temperature control.

A peltier element mounted on a cooper block is the MMA temperature controller, which is in close contact with the back of the MMA housing. Its nominal value can be set as well. Moreover, two sensors are used to monitor the temperature. One is the peltier sensor inside the cooper block (Peltier sensor) and the other is on the backside of the cooling plate (Board sensor).

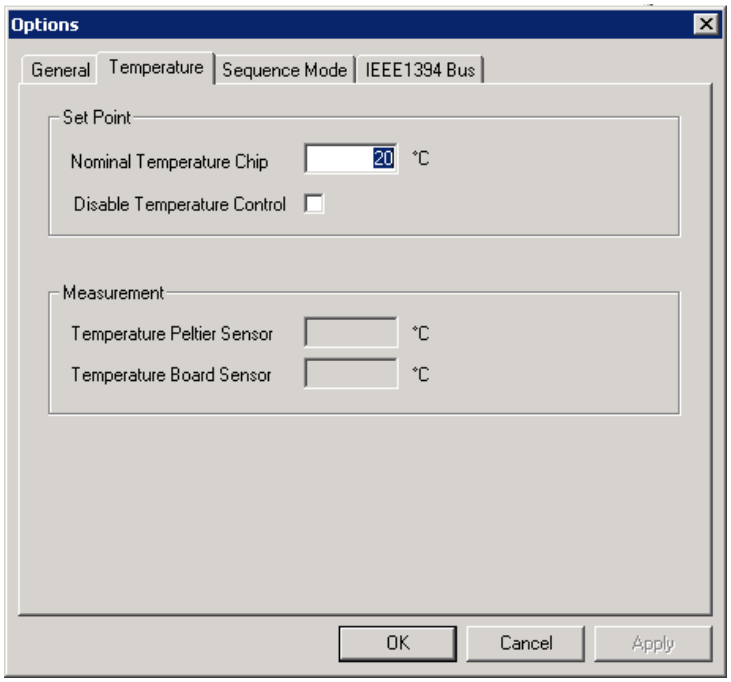


Figure 3.3 Options Dialog – Pate Temperature.

3.3 Sequence mode parameters.

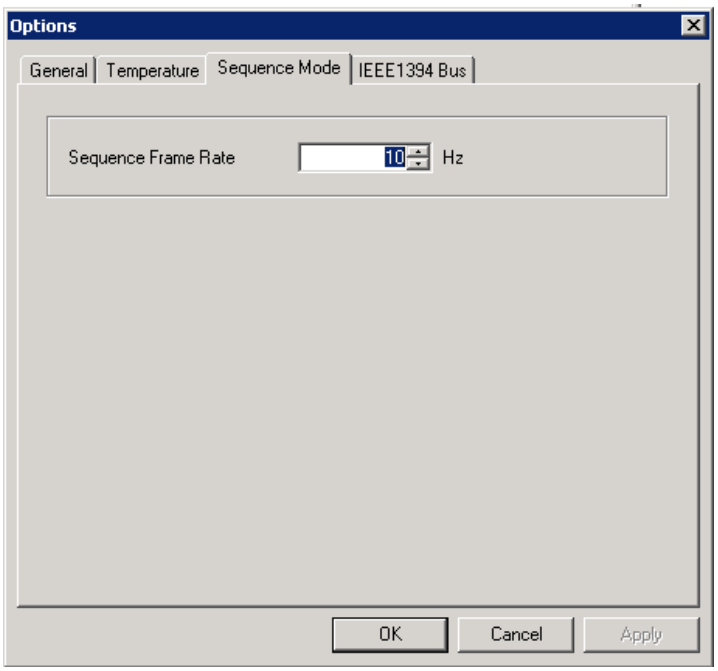


Figure 3.4 Options Dialog – Page Sequence Mode.

The 'Sequence Frame Rate' represents the number of transferred MMA data patterns per second from the PC to the board. Furthermore, it is limited by the PC hardware's performance. If it is set to be greater than the value offered by the hardware, the data transfer will proceed only at the maximum rate without any data loss.

3.4 IEEE 1394 bus parameter.

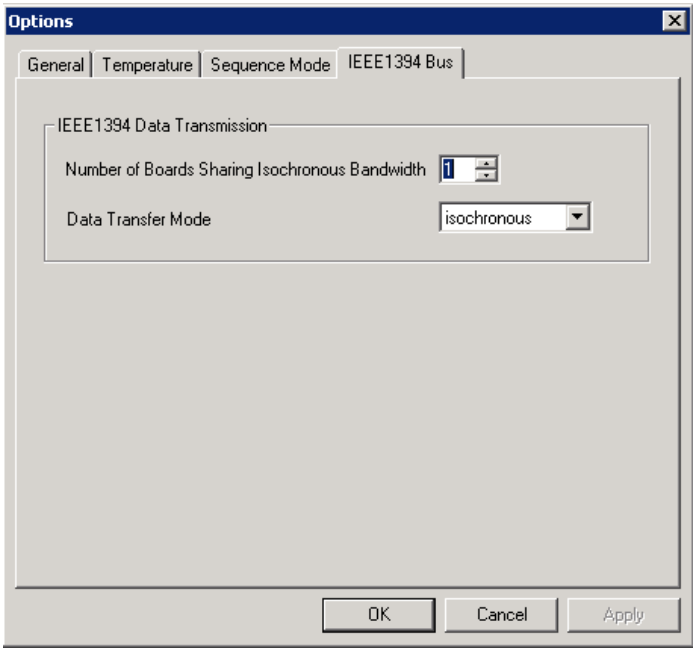


Figure 3.5 Options Dialog – Page IEEE 1394 Bus.

- *Data Transfer Mode.* Three methods of the data transfer are available, i.e. asynchronous, isochronous and isochronous with external IEEE 1394 sources. For isochronous mode, the data transfer is allocated within certain timing constraints via a broadcast data delivery at regular time intervals. Normally, the control commands for the interaction of the PC with the MMA board abide by this method. However, the method can be changed and it is highly recommended to use the isochronous mode.

- *Number of Boards Sharing Isochronous Bandwidth.* If more than one board is connected to the IEEE 1394 bus, each allocated bandwidth must be reduced in order to equally share the bandwidth to other boards. Therefore, the maximum frame rate for each further additional board linked to the bus is decreased.

CHAPTER 4

Simulation in LabVIEW

1. LabVIEW's advantages

LabVIEW stands for Laboratory Virtual Instrument Engineering Workbench. It is a programming environment in which the program is created by graphical notations. However, it is not only a programming language but also an interactive program development and execution system widely used by scientists and engineers of several disciplines. Compared to other types of programming, LabVIEW provides users with stunning interesting advantages to be given as examples below.

1. LabVIEW can greatly reduce the amount of time the user needs to create one particular program. Because of the special design for taking measurements, analyzing the data, presenting the results and versatile graphical user interface, it increases the productivity of the user's work several orders of magnitude.

2. LabVIEW offers more flexibility than standard laboratory instruments. Due to being software – based, the program exactly created can be modified in moments. Moreover, a fraction of the cost of traditional instruments is able to achieve with virtual instrument.

3. LabVIEW has the extensive libraries of functions and subroutines helping the user solve most programming tasks. Furthermore, the userdo not have to face difficulties of the fuss of the pointers, memory allocation, and arcane programming problems.

4. LabVIEW eliminates a lot of syntactical details associated with text – based languages. Thus, even an amateur programmer or a person who has no experience in programming can create such a powerful virtual instrument.

5. LabVIEW performs parallel and multiple executions at the same time. Based on the flow of the data, the program is executed only if the dataflow reaches that object. Moreover, the wiring of multiple nodes is allowed so that data flows into many applications simultaneously.

Figure 5.1 LabVIEW starting window.

2. LabVIEW's components.

All programs created by LabVIEW consist mainly of three main parts.

2.1 Front panel.

The front panel is the interactive user interface of a program. As named, it simulates the front panel of a physical instrument. It can contain knobs, push buttons, graphs and many other controls and indicators.

2.2 Block diagram.

The block diagram is the program's source code, constructed in LabVIEW's programming language. It is also the actual executable program, comprising lower – level programs, built – in functions, constants and program execution control structure. Noticeably, the front panel objects have corresponding terminals on the block diagram so that data can pass from the user to the program and back to the user.

2.3 Icon.

The icon is a program's pictorial representation and is used as an object in the block diagram of another program.

CHAPTER 5

Programming and Simulation in LabVIEW

1. Front panel.

Front panel is simply the interactive window for a user to communicate with a program. While running the program, the front panel must be open to allow the user to input data to the executing program. It could be very simple, consisting of a few controls and indicators, or sophisticated so that several comments should be implanted on it. In this project, the front panel controlling the MMA operation has been created and its description will be discussed here.

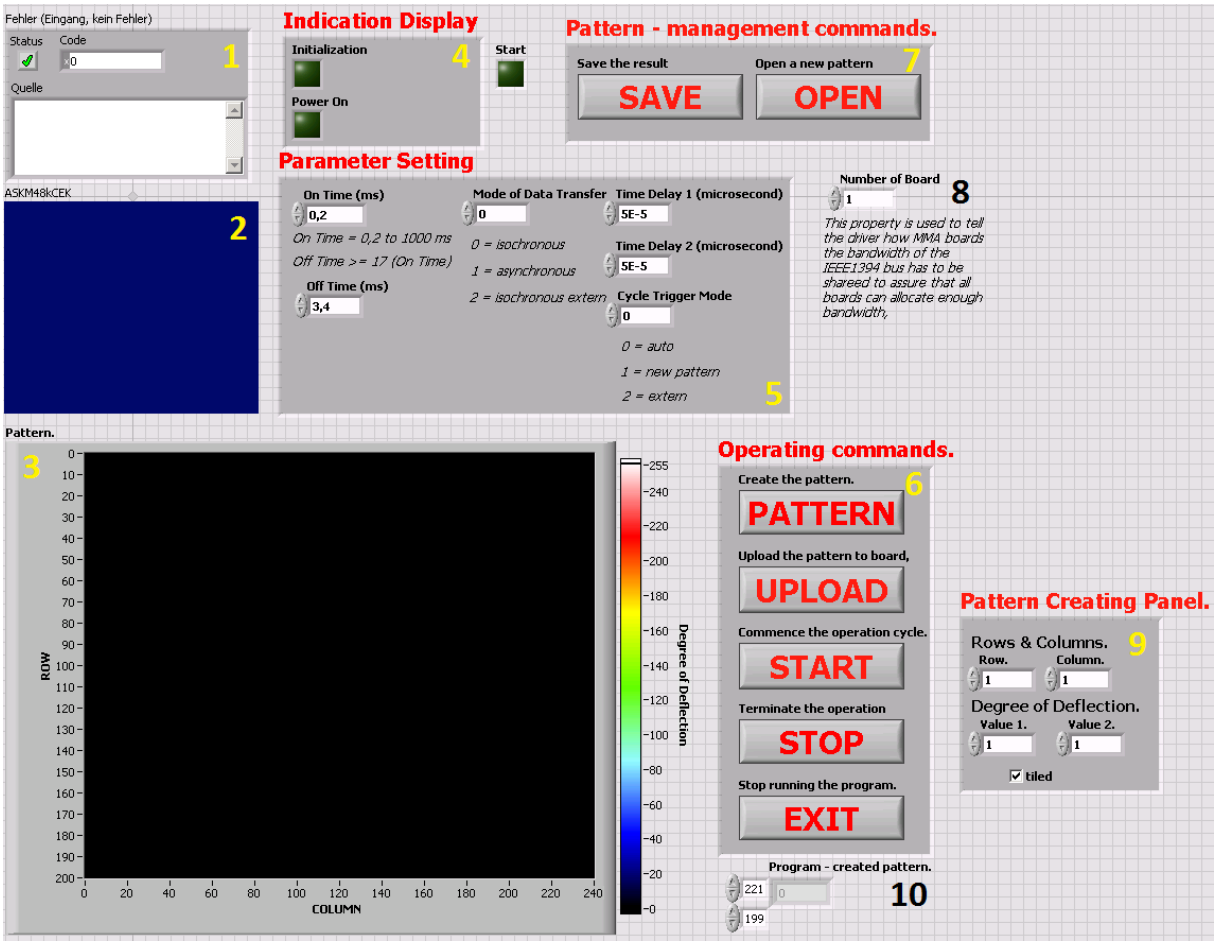


Figure 5.1 Front panel for controlling the MMA operation cycle.

1. *Error display*. This message board displays any errors occurring during the flow of data. Additionally, software error codes are usually shown in order to help the user solve the problem or debug the program.

2. *ASKM48kCEK ActiveX container*. This container is a displaying window showing the deflection pattern of a particular micromirror array during the MMA operation cycle.

3. *Intensity graph 'Pattern'*. This graph illustrates the deflection pattern of the only currently – existing pattern after the user creates it.

4. *Indication display*. The indication display contains two BOOLEAN indicators, square LEDs, indicating the status of the initialization and the power. If these processes are successfully triggered, the LEDs automatically light up.

5. *Parameter setting panel*. The panel provides the user with the real – time manipulation of the modifiable parameters for controlling the MMA operation cycle.

6. *Operating command panel*. All basic commands to control the MMA operation cycle are included here.

7. *Pattern – management command panel*. This panel deals with file management, either saving or opening the patterns.

8. *Number of board control*. This control receives the number of MMA boards for sharing the bandwidth. However, after the initialization process, it is set to be read – only.

9. *Pattern creating panel*. The panel is responsible for designing the pattern either a random pattern or a tiled pattern, by alternatively checking the 'tick box'.

10. *Program – created pattern array indicator*. The array indicator demonstrates the value of degree of deflection for each element in the pattern.

2. Block diagram.

These following block diagrams are created to help control the instrumental system for the pump – probe experiment. Many important functional parts are included in the diagram and most of them correspond to the objects on the front panel. Keep in mind that the diagram works by the left – to right dataflow, by which the order of action is indicated by the number in the block diagram, and multiple wiring is allowed. Therefore, the subsequent explanations are mainly described part by part for short.

2.1 Pre – setting step.

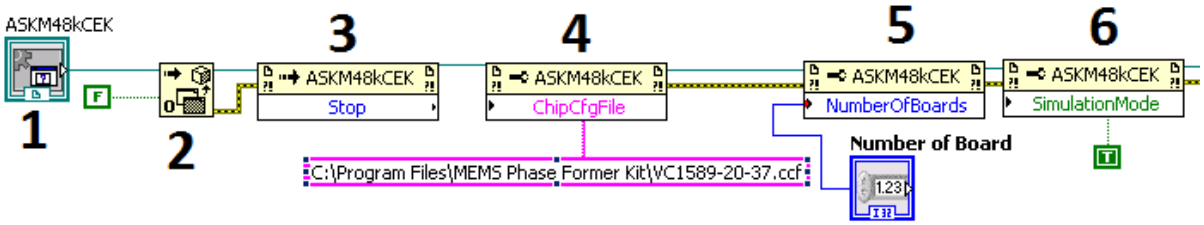


Figure 5.1 Pre - setting part of the block diagram in LabVIEW.

1. *ActiveX container ASKM48kCEK*. The container provides the program with information and ActiveX objects needed for the MMA operation cycle and the simulation.

2. *Automation open.* The purpose of this function is to return the 'automation refnum', pointing to the specific ActiveX object. Furthermore, the BOOLEAN value 'F' wired to function indicates that LabVIEW tries to reach the 'instance' of the refnum which is already open. If it fails, LabVIEW will create a new instance. Noticeably, the error information is first produced.

3. *Invoke node 'Stop'*. This node is placed at the very first part of the block diagram to stop the running MMA operation cycle if the user forgets to stop it before exiting the program.

4. *Property node 'ChipCfgFile'*. The node reads all necessary characteristic data of an individual MMA chip contained in the chip configuration file for the preparation of the new operation. The file path is wired to the node as shown.

5. *Property node 'NumberOfBoards'*. It is required to tell the driver how many MMA boards of which the bandwidth has to be shared is currently used by receiving the number from the 'Number of board control'. However, the property is set to be read – only after the initialization process.

6. *Property node 'SimulationMode'*. The ActiveX control, this node, is set to allow testing of the container software without board connection. Anyway, it is performed only if the BOOLEAN value 'T' is wired to the node and, similarly, set to be read – only after the initialization process.

2.2 Initialization step and MMA board preparation.

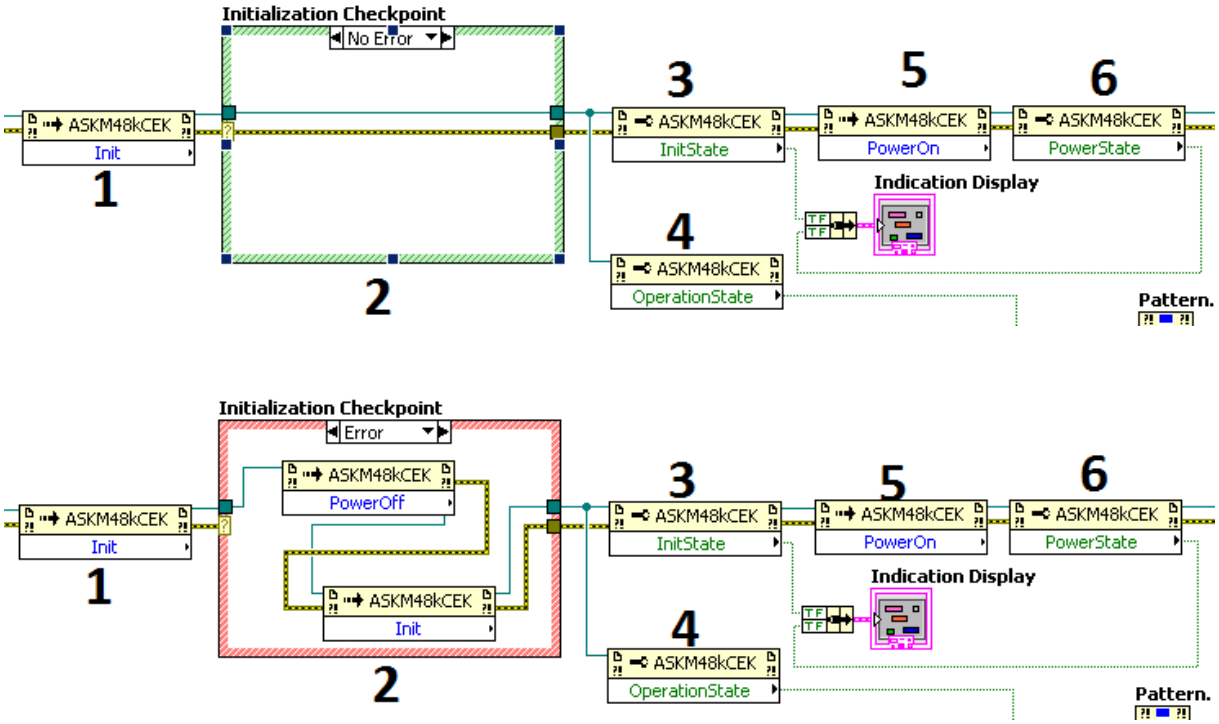


Figure 5.2 (top) Initialization checkpoint in case of **(top)** no error and **(bottom)** any errors**.**

1. *Invoke node 'Init'.* The node initializes the MMA board, the IEEE1394 bus driver and opens the interface. However, this step is highly crucial for the MMA operation and requires the checkpoint. Therefore, the subsequent 'Case structure' is created to examine whether the initialization process succeeds.

2. *Case structure 'Initialization checkpoint'.* This is meant to fix the problem when the initialization fails. The structure is also connected to the previous 'Property node Init' via the error – containing wire. If the initialization process is successful, no errors are found and the structure does nothing here whereas if it fails, the node transfers the error information to the structure, causing the power applied to the board to be shut and the board will be re – initialized.

3. *Property node 'InitState'.* This node returns the BOOLEAN value 'TRUE' to the LED indicator 'Initialization' when the initialization succeeds, causing the LED to light up. In contrast,

it returns “FALSE’ to the indicator, provided that the initialization fails, and the LED does not light up.

4. *Property node ‘OperationState’.* The node returns the BOOLEAN value ‘TRUE’ to the LED indicator ‘Start’ if the MMA operation is still running, causing it to light up. However, it returns ‘FALSE’ to the indicator provided the MMA operation is stopped. As a result, the LED does not glow up.

5. *Invoke node ‘PowerOn’.* The node is responsible for the application of the voltage to the board.

6. *Property node ‘PowerState’.* The node returns the BOOLEAN value ‘TRUE’ to the LED indicator ‘Power On’ when the initialization succeeds, causing the LED to light up. In contrast, it returns “FALSE’ to the indicator, in case of the failure, and the LED does not light up.

Figure 5.3 The glowing of ‘Initialization’ and ‘Power On’ indicators when they obtain ‘TRUE’ BOOLEAN value.

2.3 MMA operation cycle.

- *Error.*

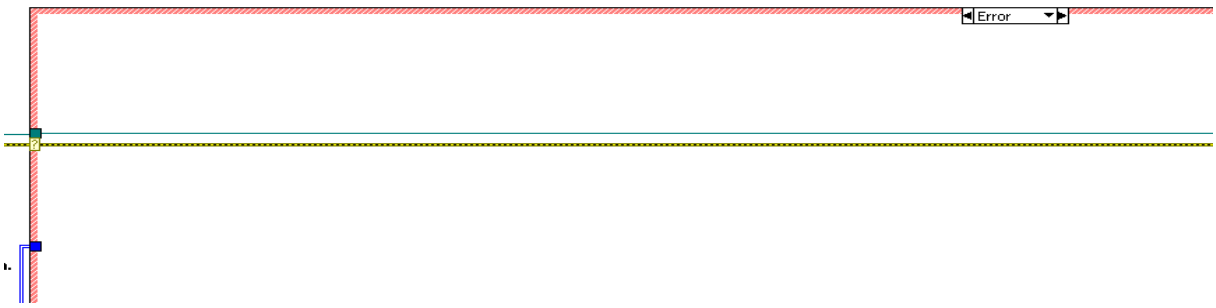


Figure 5.4 Error condition of the Case Structure. No executions are performed.

The next step after the Initialization Checkpoint is the MMA operation cycle. However, in case of the emergence of the errors no matter where they come from, the operation cannot be executed to avoid any possible errors and eventual damage to the board. Consequently, the ‘Case structure’ linked to the error – containing wire does nothing here. Eventually, the MMA

operation is terminated and the board is powered off. The mechanism of how to terminate the MMA operation and stop the application will be discussed later.

- *No error.*

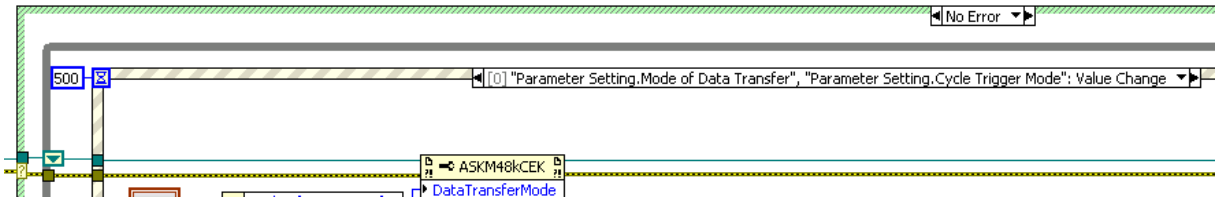


Figure 5.5 While loop and Event Structure for controlling MMA operation are inside the 'No Error' case.

In contrast, the execution can be further performed when no errors are discovered. Under this condition, the 'While loop' containing the 'Event structure' for controlling the MMA operation is engaged. It means that the user can arbitrarily manipulate the parameters to be input in the MMA operation unless the 'While loop' is terminated. Moreover, the special feature of the combination between the 'While loop' and the 'Event structure' is the limitless parameter manipulation until the event which is meant to stop the while loop is triggered. The followings are the detailed description of each event.

2.4 Data manipulating events.

1. Mode of data Transfer and cycle trigger mode.

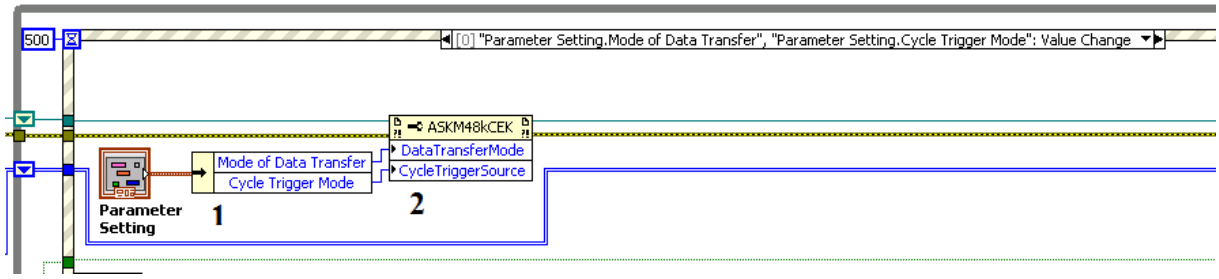


Figure 5.6 An event for the change in values of Mode of Data Transfer and Cycle Trigger Mode.

1.1 *Cluster 'Parameter setting' and 'Unbundle by name'.* The ' Unbundle by name' unbundles the cluster 'Parameter setting' and receives two input parameters 'Mode of Data transfer' and 'Cycle Trigger Mode' from the 'Parameter setting panel'.
These parameters are then transferred to the double Property node 'DataTransferMode' and 'CycleTriggerSource'.

1.2 *Double property node 'DataTransferMode' and 'CycleTriggerSource'.* The node 'DataTransferMode' and 'CycleTriggerSource' obtains the input parameters 'Mode of data transfer' and 'Cycle Trigger Mode' respectively. Then, it implements these values to the board for the MMA operation.

2. On – Time, Off – Time, Pattern Ready Delay1 and Pattern Ready Delay2.

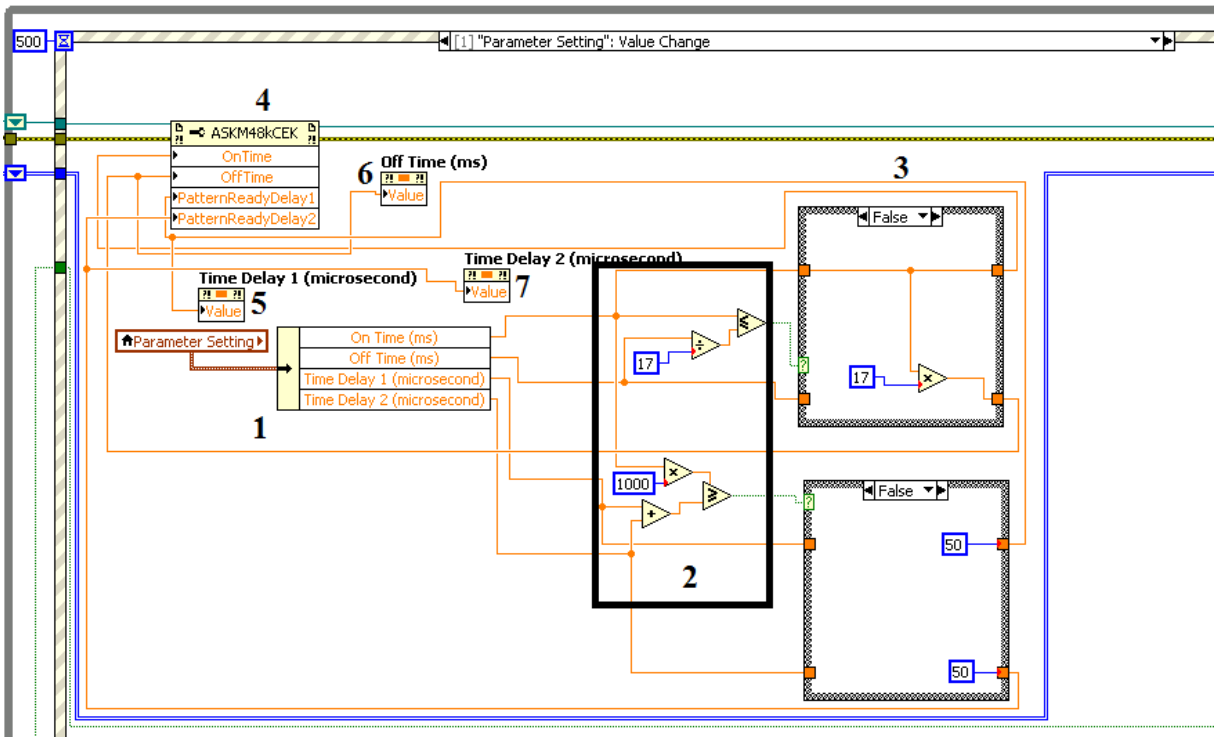


Figure 5.7 An event for the changes in values of On – Time, Off – Time, Pattern Ready Delay 1 and 2.

2.1 *Local variable 'Parameter setting' and 'Unbundle by name'.* The local variable' Parameter setting' is meant to represent the corresponding cluster because the cluster has been used in another event. Yet, its function is the same as the original one. The 'Unbundle by name' unbundles the cluster as shown and receives the related input parameters from the 'Parameter setting' front panel. The parameters are then transferred to the next relationship– checking functions.

2.2 *Relationship – checking functions.* There are two sets of functions examining the validity of the input parameters. The first upper group confirms if 'On – Time' is equal to or less than

‘Off – Time’ divided by a number 17. Another lower group tests whether the sum of ‘Time Delay 1’ and ‘Time Delay 2’ is less than or equal to ‘On – Time’ multiplied by 1000. Thereby, both comparison functions returns the BOOLEAN value ‘TRUE’ if the relationship is conserved.

2.3 *Condition – executing ‘Case structure’.* Two case structures are involved in the execution. The upper structure generates a new ‘Off – Time’ by replacing the old value with ‘On – Time’ multiplied by 17 whereas nothing happens to the ‘On – Time’, in case that the relationship between On – Time and Off – Time is not true. For the lower structure, it uses the value ‘50 microseconds’ in place of both original ‘Time Delay 1’ and ‘Time Delay 2’ if these values do not conform to the relationship. Anyway, both structures do nothing and transfer these parameters to the corresponding property node.

2.4 *Multiple property node ‘OnTime’, ‘OffTime’, ‘PatternReadyDelay1’, ‘PatternReadyDelay2’.* This node receives the unproblematic values which correspond to the node’s function. After that, the parameters are implemented to the MMA operation cycle.

2.5 – 2.7 *Implicit property nodes ‘Time Delay 1’, ‘Off Time’ and ‘Time Delay 2’.* These nodes also receive the corrected input parameters ‘Time Delay 1’, ‘Off – Time’ and ‘Time Delay 2’ respectively in order to display the adjusted suitable values to the user.

3. Operation state (Timeout event).

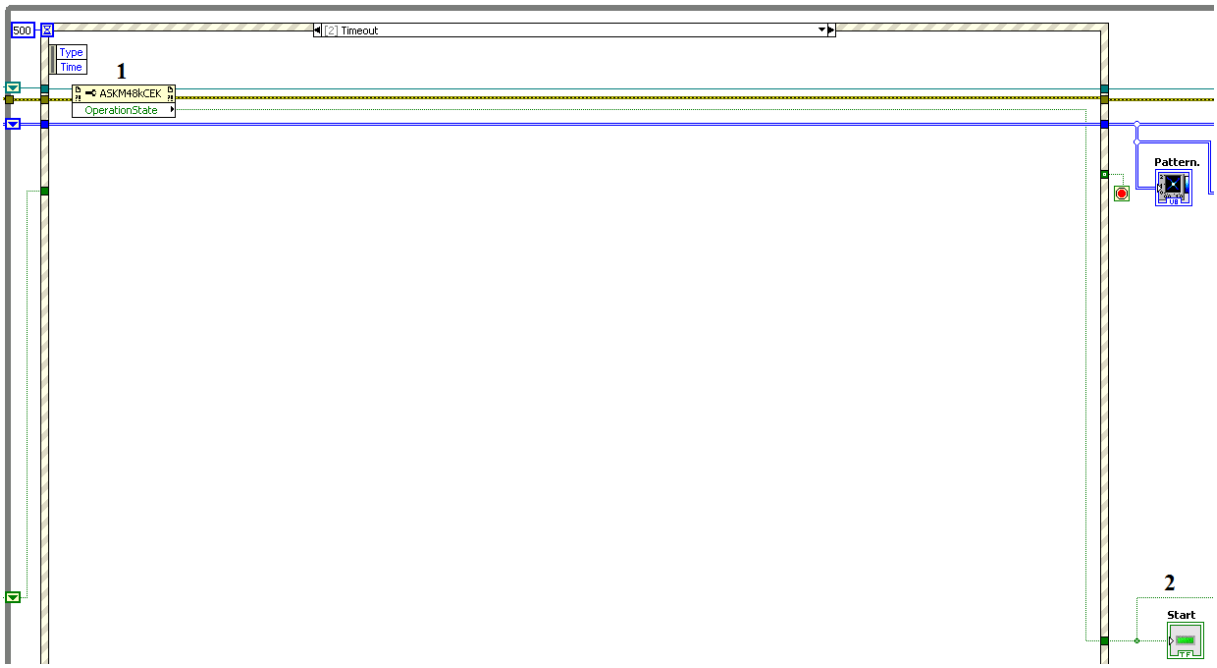


Figure 5.8 The 'Timeout' event that indicate the state of the MMA operation.

3.1 *Property node 'OperationState'.* The node returns the BOOLEAN value 'TRUE' to the LED indicator 'Start' if the MMA operation is still running but 'FALSE' if the operation stops.

3.2 *Square LED indicator 'Start'.* The purpose of this indicator is to inform the state of the MMA operation cycle to the user. If the operation is running, the LED lights up where it does not glow up if the operation stops.

4. Create the pattern.

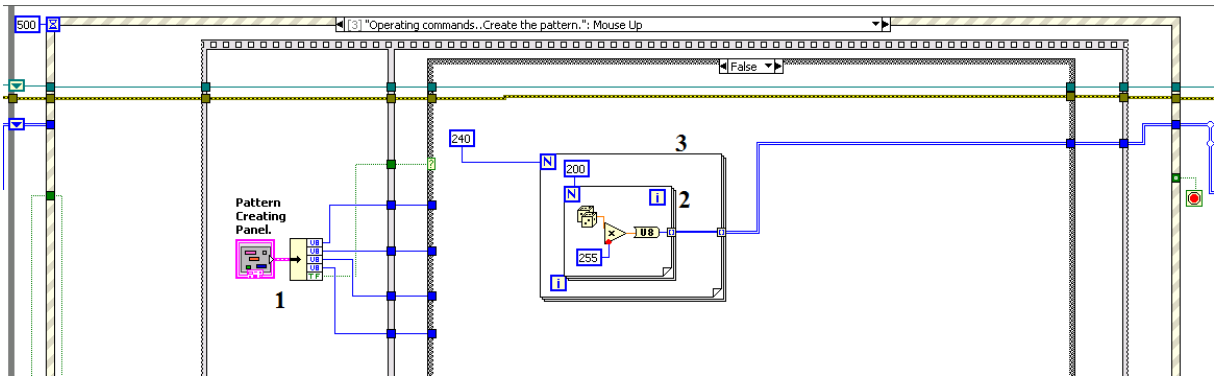


Figure 5.9 An event generating a random pattern.

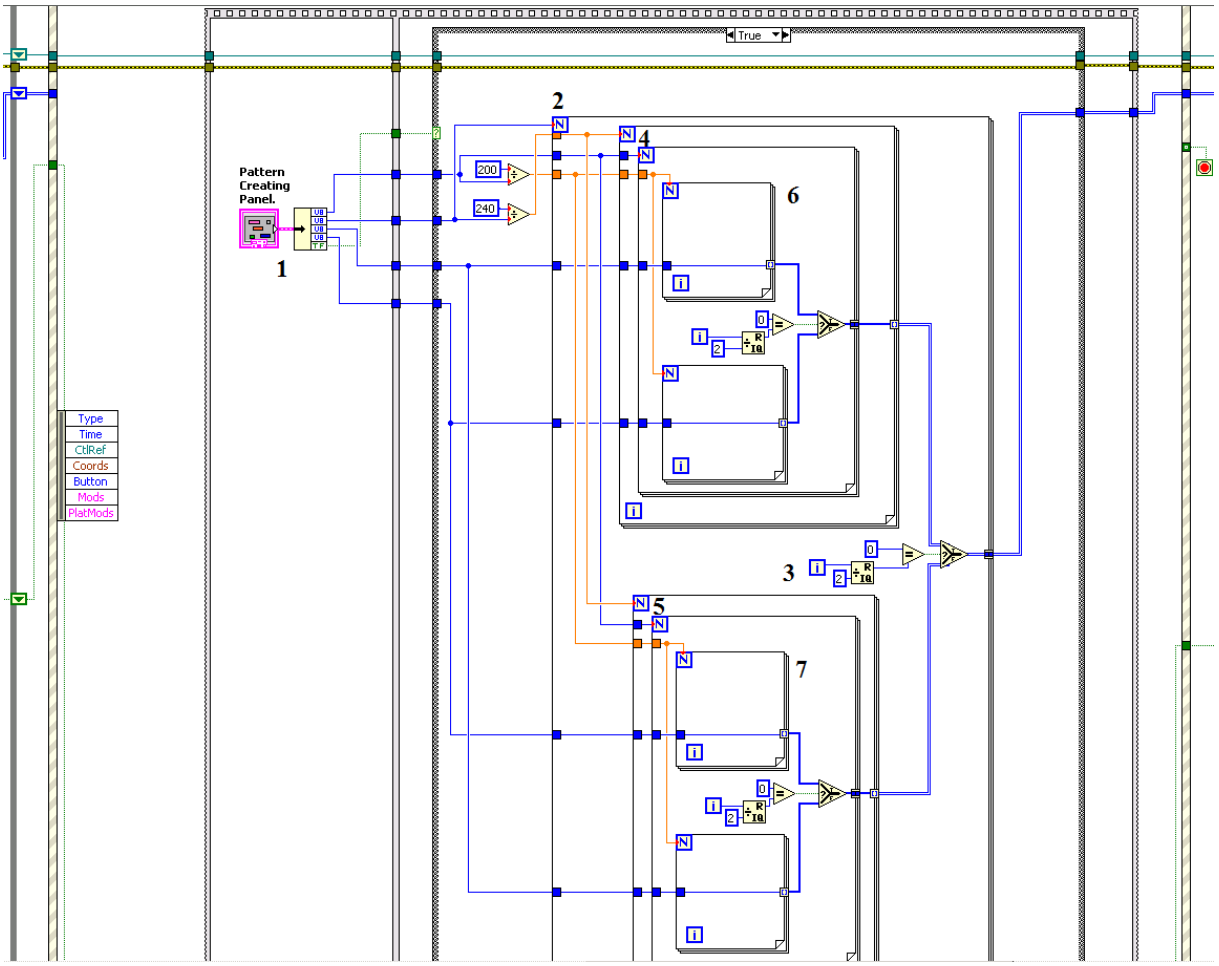


Figure 5.10 An event generating a tiled pattern.

4.1 *Cluster 'Pattern creating panel' and 'Unbundle by name'.* The 'Unbundle by name' receives the information for creating the pattern from the cluster 'Pattern creating panel' in the front panel, including the number of rows and columns, two values of different degrees of deflection and the BOOLEAN value from the tick box . 'TRUE' is for creating a tiled pattern and 'FALSE' for a random one. After that, these parameters are transferred to the next case structure.

4.2 *Case structure for creating the pattern.* The structure executes the creation of a random pattern when receiving the 'TRUE' value but a tiled pattern when getting the 'FALSE' value.

4.3 *Mechanisms of creating the patterns.*

- Random pattern.

The first 1 – dimensional array is initiated with the 'For loop' #3, starting from the iteration index of '0'. Then, the numeric function 'Random number' in the 'For loop' #2 starts producing a random number subsequently multiplied by 255 and converted to 8 – bit unsigned value. After the loop#2 completes the first 1 X 200 – dimensional array, the loop #2 ends and the loop #3 begins the second iteration. Another 1 X 200 - dimensional array is created and later concatenated to the previous array, resulting in the 2 X 200 – dimensional array. Repeatedly, the iteration keeps on until the 240 X 200 – dimensional array finally results. The graphical illustration is shown in figure 5.11.

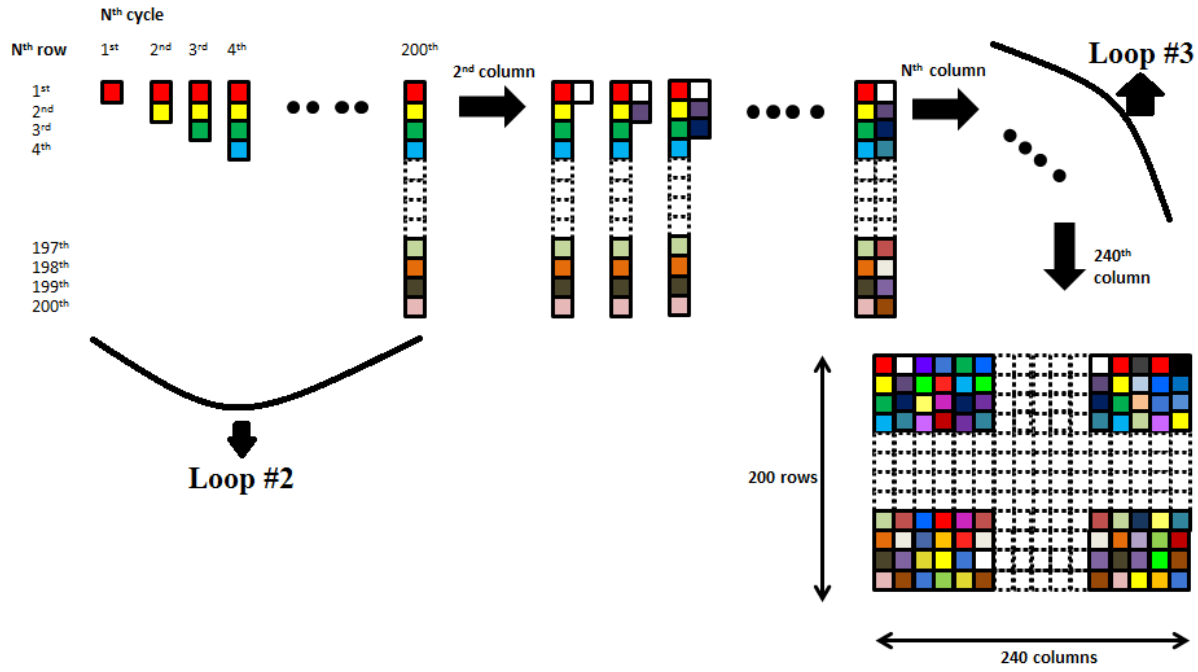


Figure 5.11 Conceptual idea of creating a random pattern.

- Tiled pattern.

The idea of constructing the pattern is based on creating the tiles with alternatively – arranged fashion. Firstly, the loop #2 controls the generation of two different patterns, one beginning with a patch of higher degree of deflection and another starting with a lower degree of deflection. The condition (#3) of which a particular manner is chosen is also based on the iteration number of the loop #2. If the iteration number is even number, the former pattern is executed while the latter is created provided that the number is

odd. Anyway, the loop keeps on running until it completes the number of columns.

Secondly, after one pattern – creating style is reached, the loop #4 (or #5) starts running. Inside the loop #4 (or #5) are two 'For loops', upper and lower loops #6, producing different one – dimensional array with alternate degrees of deflection. The criterion for choosing the loops #6 is also based on whether the iteration number is even or odd. If the iteration number is even, the upper loop #6 (or #7) creates the array. Otherwise, the lower loop #6 (or #7) creates the array instead. No matter which loop creates the array, each array is continually concatenated so that the 1 X 200 – dimensional array is produced when the loop #4 (or #5) ends. However, the loop controlling the loop #4 (or #5) keeps on iterating, creating many more 1 X 200 – dimensional arrays concatenated to one another. Eventually, the array with (240/#column) X 200 dimension results.

Thirdly, the arrays produced from either loop #4 or #5 are also concatenated to one another until it completes the whole pattern.

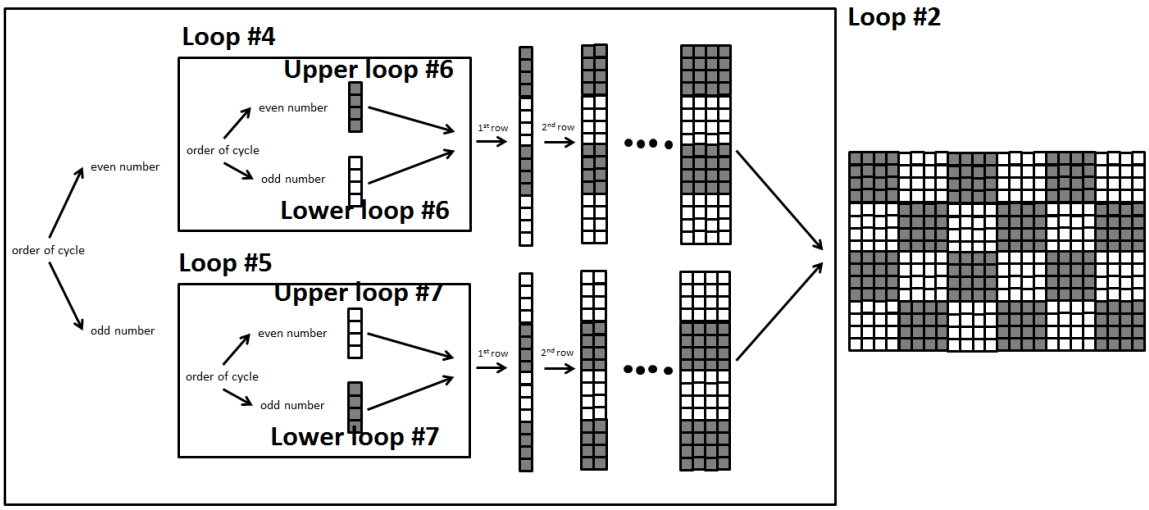


Figure 5.12 The concept of creating the tiled pattern. The 20 X 24 – dimensional array with four rows and six columns is demonstrated. Each square represents each element of the array.

5. Stop running the program.

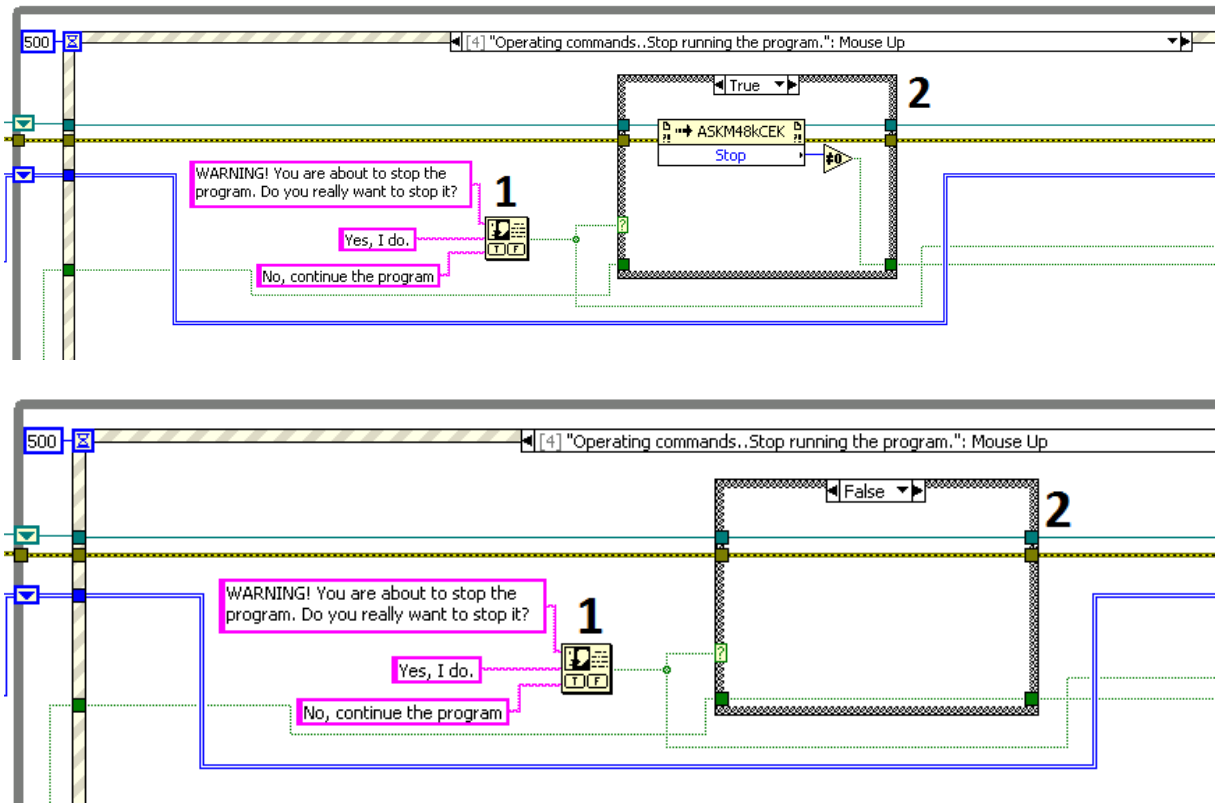


Figure 5.13 An event to exit the application. The top shows the 'true' case where as the bottom shows the 'false' case.

5.1 *Two button dialog.* The dialog box appears ensuring the user that he really wants to exit the application. If the user selects 'Yes, I do.', the dialogue box returns 'TRUE' value whereas it returns 'FALSE' to the adjacent case structure if the user selects 'No, continue the program.

5.2 *Case structure.* The case structure does nothing if it receives the 'FALSE' value and the program is still running. On the other hand, the case structure performs the execution to stop the MMA operation cycle when the 'TRUE' value is obtained. Moreover, the Invoke node stop inside returns '0' to the comparison function 'Not equal to zero', causing the LED indicator 'Start' to light down and the dialogue also returns the 'TRUE' value to the conditional terminal. So the program stops.

6. Stop the operation cycle.

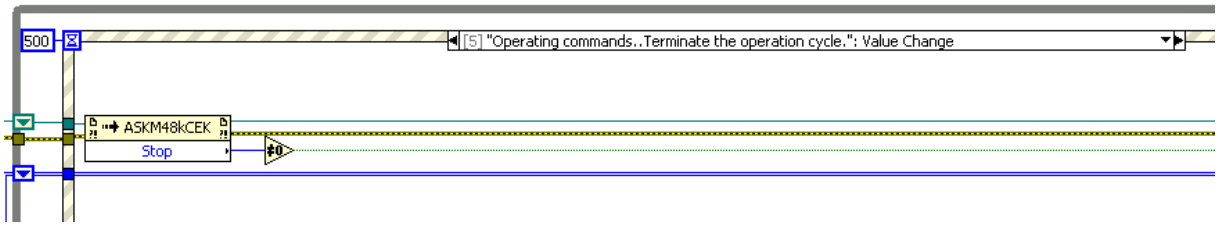


Figure 5.14 An event to terminate the MMA operation cycle.

6.1 *Invoke node 'Stop'.* The node terminates the MMA operation cycle and returns '0' to the comparison function 'Not equal to zero' if it succeeds.

6.2 *Comparison function 'Not equal to zero'.* It returns the 'FALSE' value if it receives 'o', resulting in the lighting down of the LED indicator 'Start'. Otherwise, the BOOLEAN value 'TURE' is produced.

7. Open a new pattern.

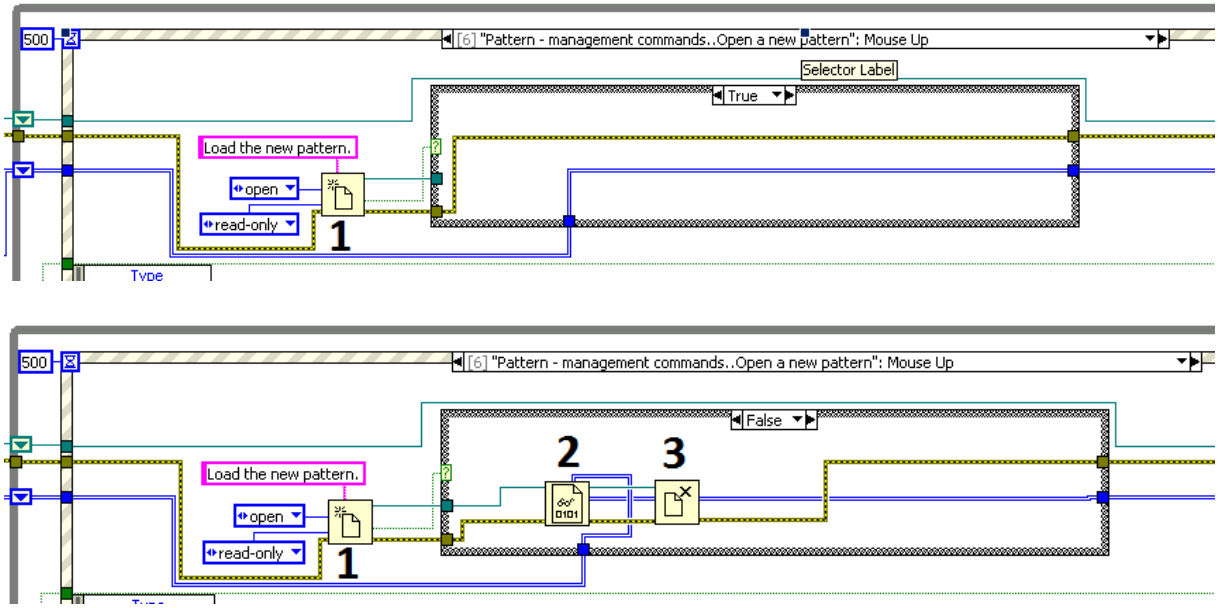


Figure 5.15 An event to open a new pattern from a folder. **(top)** Nothing occurs if the user denies opening the file and **(bottom)** the mechanism of opening the file is shown.

7.1 *Open/Create/ReplacefFile.* The function pops up the dialogue box for selecting a data pattern file to be opened. It returns the 'TRUE' if the user decides not to open the file but the 'FALSE' provided the users selects the file and chooses 'Open'. Both BOOLEAN values go to the case structure.

7.2 *Read from binary file.* It reads the binary data from the file path converts it into data. The data type is determined by the data wired to this function.

7.3 *Close file.* The function closes the file specified by the refnum.

8. Upload the pattern to the board.

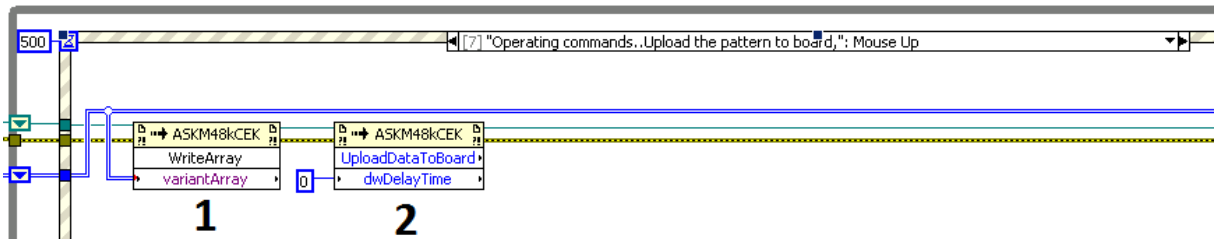
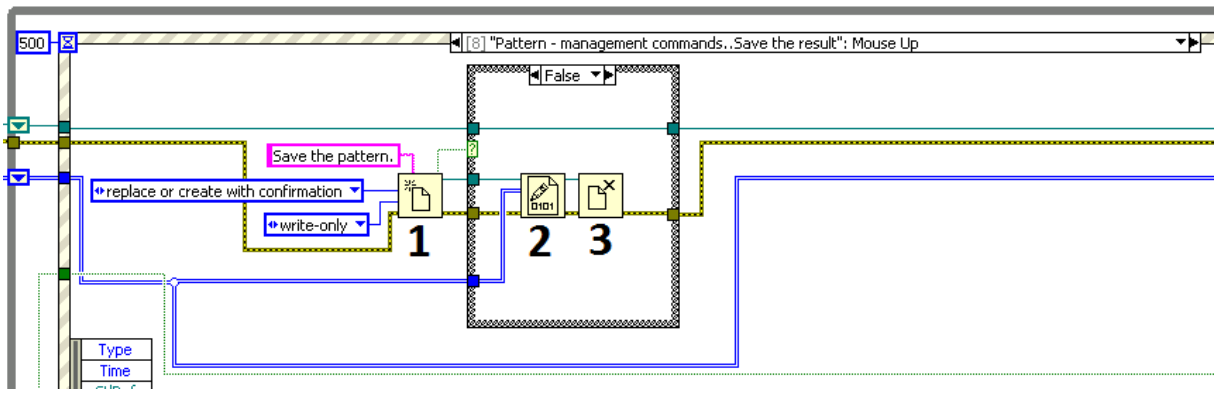


Figure 5.16 An event for uploading the pattern to the board.

8.1 *Invoke node 'WriteArray'.* It writes an array by receiving the array already available and converts it into a variant array compatible with ActiveX Control.

8.2 *Invoke node 'UploadDataToBoard'.* This node transfers the latched array to the MMA board with 'dwDelayTime' after the previous upload. It means that if the time does not expires, the control waits for the remaining time and then uploads the pattern. In case, '0' is wired to it, indicating that the pattern will be uploaded immediately.

9. Save the pattern.



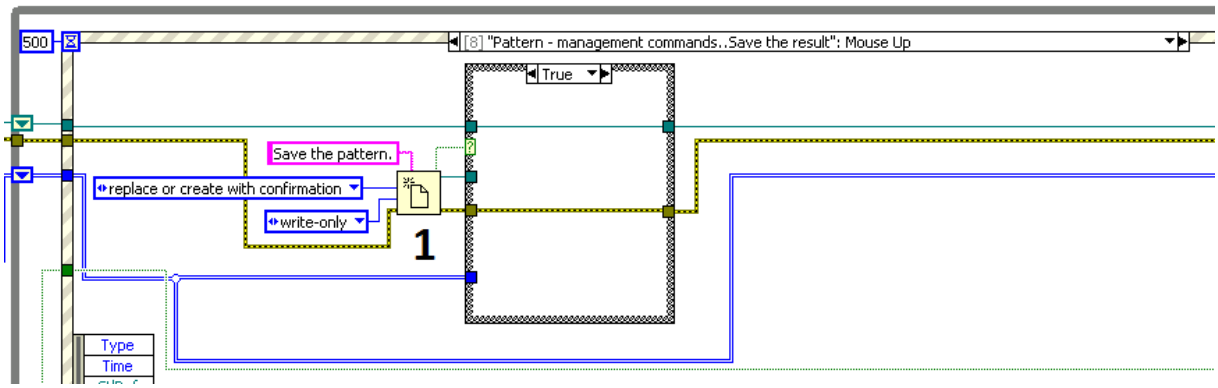


Figure 5.17 An event for saving the existing pattern. **(top)** The mechanism of how to save the pattern and **(bottom)** nothing happens if the user denies saving the pattern.

9.1 *Open/Create/ReplacefFile.* The function pops up the dialogue box for selecting a file directory to save the pattern. It returns the 'TRUE' if the user decides not to write the data file but the 'FALSE' provided the users really wants to save the file and then chooses 'Save'. Both BOOLEAN values go to the case structure.

9.2 *Write to binary file.* It writes the binary data to the new file, appends the data to an already – existing file or replaces a content of a file.

9.3 *Close file.* The function closes the file specified by the refnum.

10. Start the operation cycle.

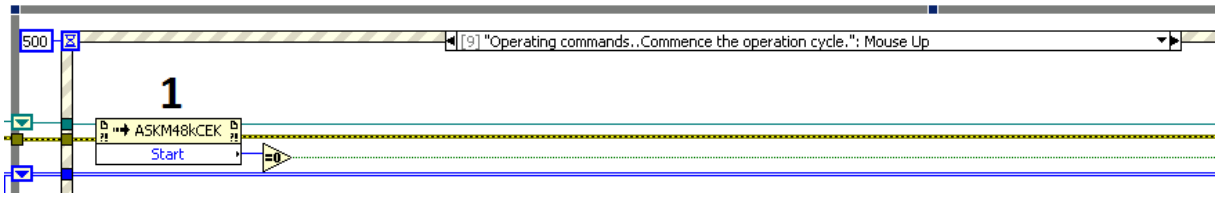


Figure 5.18 An event to start the MMA operation cycle.

10.1 *Invoke node 'Start'.* The node starts the MMA operation cycle and returns '0' to the comparison function 'Equal to zero' if it succeeds.

10.2 *Comparison function 'Equal to zero'.* It returns the 'TRUE' value if it receives 'o', causing the LED indicator 'Start' to light up. Otherwise, the BOOLEAN value 'FALSE' is produced.

2.4 Data display and system shut – down.

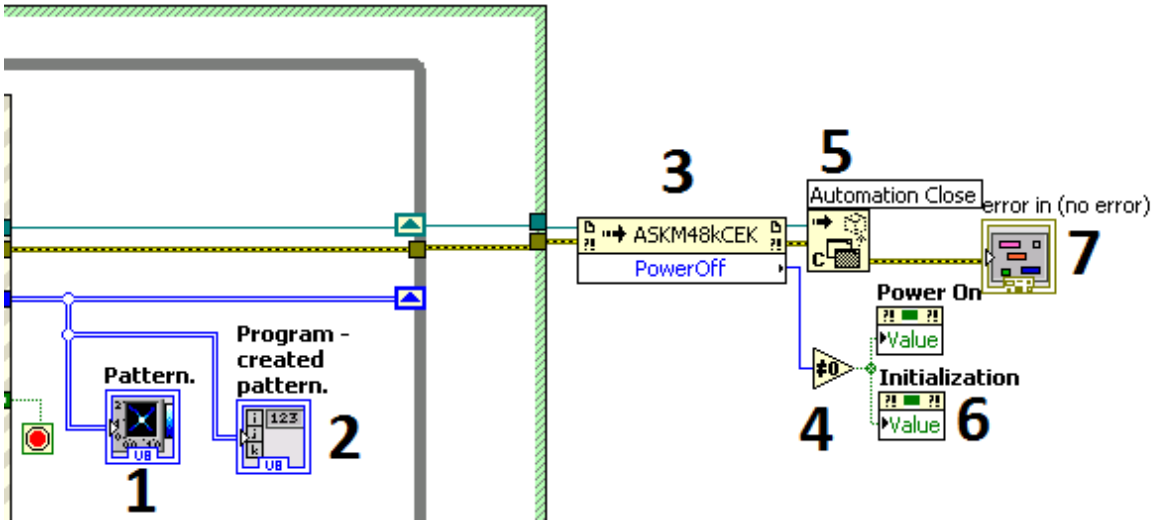


Figure 5.19 Last part of the block diagram.

1. *Intensity graph 'Pattern'.* The graph shows the deflection pattern of the recently – created pattern. Each degree of deflection is represented by spectral colors, ranging from the lowest values with black to the highest value with white.

2. *Array indicator 'Pattern created program.* The indicator displays the real numeric value of degree of deflection for each element.

3. *Invoke node 'PowerOff'.* The node powers off the MMA board before the program really stops.

4. *Comparison function 'Not equal to zero'.* The function returns 'FALSE' to the nexyt properties if it obtains '0'. Otherwise, it returns 'TRUE'.

5. *'Automation close'.* All ActiveX objects are closed and no longer used in LabVIEW.

6. *Properties 'Power On' and 'Initialization'.* The properties represent the real LED indicators 'Power On' and 'Initialization'. These LEDs light down when they receive the BOOLEAN value 'FALSE'.

7. *Error in.* The function receives the error information and displays the error and its code on the message board in the front panel.

2.5 Event – triggering conditions.

Event	Buttons or panel required for triggering	Action
Mode of data transfer and cycle trigger mode.	Parameter setting panel	Change in the corresponding values.
On – Time, Off – Time, Pattern Ready Delay 1 & 2.		
Operation state. (Timeout event)	-	-
Create the pattern.	PATTERN button.	Pressing the button.
Stop running the program.	EXIT button.	
Stop the operation cycle.	STOP button.	
Open a new pattern.	OPEN button.	
Upload the pattern to the board.	UPLOAD button.	
Save the pattern.	SAVE button.	
Start the operation cycle.	START button.	

Table 5.1 Summary of conditions required for triggering the data – manipulating events.

3. Simulation.

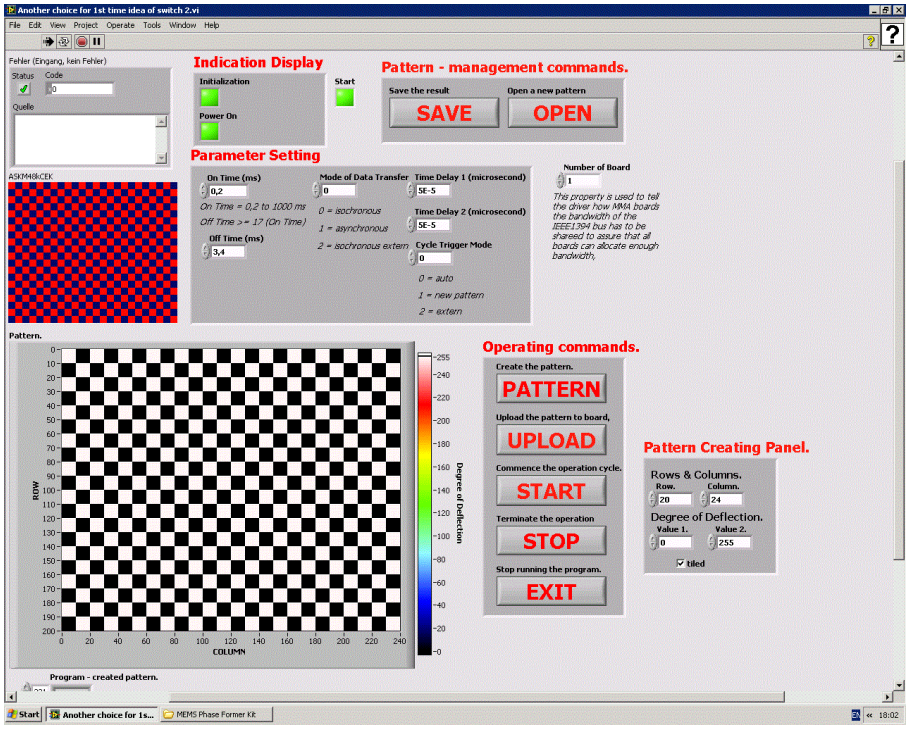


Figure 5.20 The simulation in LabVIEW for a tiled pattern.

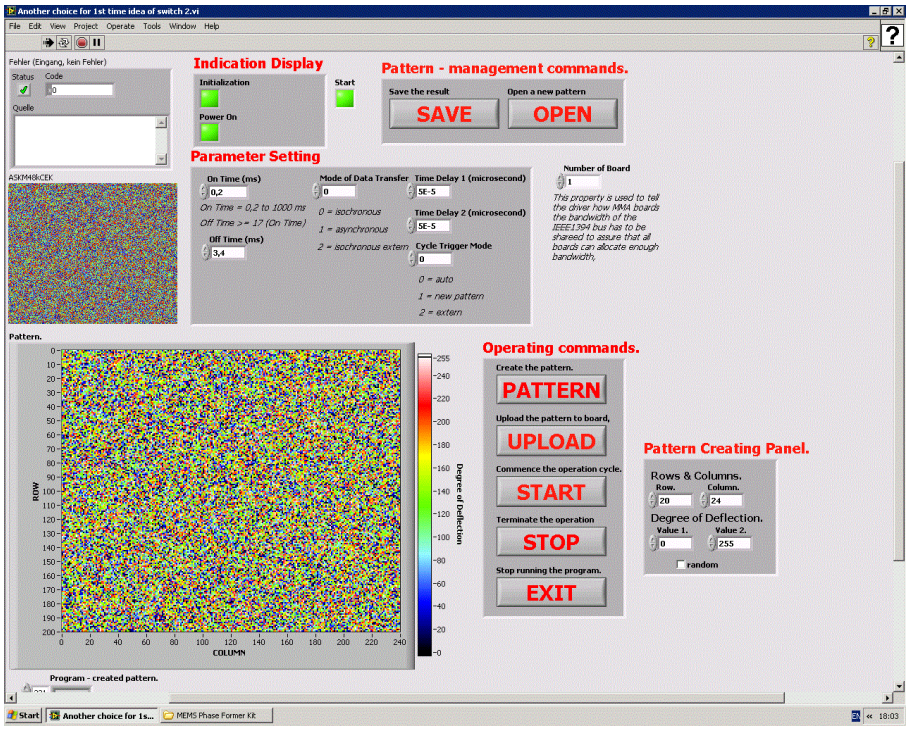


Figure 5.21 The simulation in LabVIEW for a random pattern.

CHAPTER 6

Conclusion

The LabVIEW program for controlling the MMA operation cycle in the pump – probe experiment has been created. The program can be mainly divided into two parts, the front panel and the block diagram. The front panel simulates the interactive interface to manipulate the physical instruments. All important panels required to control the instruments are also created. For example, saving the pattern, loading the pattern, manipulating the parameters, starting and terminating the MMA operation cycle. Moreover, the indication display contains the LED indicators, with the purpose of informing the status of the MMA operation cycle to the user.

Speaking of the block diagram, necessary error – handling application loop is also created to deal with some common problems. Besides, the relationships for suitable input parameters are programmed to check whether the parameters are valid or not.

However, the LabVIEW program can still perform just fundamental controls. Further optimization process of the MMA deflection pattern is required.

REFERENCES.

1. Fabrication of binary phase plates for the management of orbital-angular-momentum superposition states.
2. A. Preumont, R. Bastaits, M. Horodinca, G. Rodrigues, I. Romanescu and I. Surdej. Segmented Deformable Mirror for Adaptive Optics.
3. Hamelinck, Roger. Adaptive Deformable Mirror Based on Electromagnetic Actuators. Eindhoven: Technische Universiteit Eindhoven, 2010.
4. Weber S. M., Waldis S., Noell W. Linear micromirror array for broadband femtosecond pulse shaping in phase and amplitude. *Proc. of SPIE,* 7208: 720805 – 1 -720805 – 6.
5. Madec P.Y. Overview of Deformable Mirror Technologies for Adaptive Optics and Astronomy.
6. Bronson, R. J. (2007). Modeling and Control of MEMS Micromirror Arrays with Nonlinearities and Parametric Uncertainties (Doctoral Dissertation). University of Florida.