Conduct yourself in this world as if you are here to stay forever; prepare yourself for eternity as if you have to die tomorrow

Prophet Muhammad (S.A.W)

University of Alberta

Design and development of a two dimensional scanning molecular tagging velocimetry (MTV) system

by

Farhan Ahmad

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Fall 2011
Edmonton, Alberta

Examining Comittee

Dr. David S. Nobes (Supervisor), Mechanical Engineering

Dr. Roger W. Toogood, Mechanical Engineering

Dr. Anthony Yeung, Chemical and Materials Engineering

To my parents, especially my mother. Thank you for your love, guidance, persuasion and the other countless sacrifices that you have made for me so that I can achieve my goals.

ABSTRACT

The design and development of a MTV approach, based on photobleaching, is discussed. The MTV techniques developed so far for microfluidic application provide either only a one dimensional flow measurement or use a grid and a structured mask for macro-scale and micro-scale flows respectively. The developed system is capable of resolving two dimensional flow velocity information without the limitations associated with other approaches. The salient feature of the presented approach is the use of a laser scanner which allows unrestricted, repeatable and accurate movement of the write laser within the field-of-view. An assessment of the performance of the MTV system will be discussed with a comparison to traditional μ-PIV. The aim of this technique is to perform velocity measurements in a dielectrophoretic flow of a mixture of nano-particles and the caged fluorescent dye, which may show different flow behavior, possibly in two opposite directions due to the charge on them.

ACKNOWLEDGEMENT

It is my pleasure to appreciate and thank the many people who made this thesis possible. I would like to take this opportunity to acknowledge

Dr. David Nobes, Advisor

for his continuous guidance, support and help during my journey as a graduate student. I consider myself lucky to get someone like Dr. Nobes as an advisor. He is always cheerful, calm and cool. Talking to him about my research work was always fun because of his witty remarks in between conversations and his excitement about the project. Despite his busy schedule and a difference between our working hours, the door of his office was always open for me to discuss any issues related to my research work. His advice on proper presentation of the significant data and results helped me grow as a technical writer and presenter.

Dr. Payam Rahimi, Co Advisor

for his help in my transition to a new education system. Although I did not spent much time with him but his guidance in the process of literature review not only helped me during my thesis but will continue to help me during my practical life.

Rick Conrad, Electronic Engineering Technician

Roger Marchand, Technical Services Supervisor

Tuula Hilvo, Technician

Ryan Shoults, Special Projects Technician

for their help during my time spent in the MECE workshop.

Dr. Lorenz Sigurdson, Mechanical Engineering (MEC E 638)

Dr. Morris Flynn, Mechanical Engineering (MEC E 630)

Dr. Subir Bhattacharjee, Mechanical Engineering (MEC E 637)

for their guidance during my courses

Dr. Roger W. Toogood, Mechanical Engineering

Dr. Anthony Yeung, Chemcal and Materials Engineering,

for their input and comments as my committee members and all my colleagues and friends in

MecE 5-27, Optical Diagnostic Lab

MecE 3-31, Thermodynamics and Kinetics Lab

6th floor at the National Institute for Nanotechnology (NINT)

for our inspirational discussions and their support during my graduate studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMECLATURE

Latin symbols

A	Surface area
b	Laser sheet thickness
d_E	Energy density
d_f	Incident beam width
d_{Tot}	Depth of field
E	Laser energy
e	Resolution limit
g	Gravitational acceleration
I_f	Intensity of fluorescence
I_{f0}	Fluorescence intensity at time $t = 0$
I_{Size}	Image size
l	Characteristic length
L	Length

P	Power
pH	Measure of acidity or basicity
P_{size}	Pixel size
Re	Reynold’s number
Re_{Γ}	Reynolds number based on circulation
t	Time or duration
Δt	Time interval between two events
TVL	TV lines
v	Velocity
V	Volume
w	Laser sheet width
X-position	Horizontal position
X-direction	Horizontal direction
Y-position	Vertical position
Y- direction	Vertical direction
2D	Two dimensional
10x	10 times
15x	15 times

Greek symbols

μ	Viscosity
ρ	Density
$\Delta\rho$	Density difference between two medium
σ	Surface tension force
λ	Wavelength
λ_F	Wavelength of fluorescence
λ_{Ucl}	Uncaging laser wavelength
τ	Photobleaching time constant

Abbreviations

AOI	Angle of incidence
APART	Air photolysis and recombination tracking
AVI	Audio video interleave, a multimedia container format and file type
CCD	Charged-coupled device
CW	Continuous wave
EDL	Electric double layer
FFT	Fast fourier transform
FOV	Field of view

FPS	Frames per second
GUI	Graphical User Interface
HTV	Hydroxyl tagging velocimetry
ID	Internal diameter
LED	Light emitting diode
LEI	Laser enhanced ionization
LIPA	Laser induced photochemical anemometry
LOC	Lab on a chip
MAX	Measurement and automation explorer (Software module)
MEMS	Micro-electromechanical systems
M_{Obj}	Magnification of objective
MTV	Molecular Tagging Velocimetry
NA	Numerical apperture
NA_{Obj}	Numerical aperture of objective
Nd:YAG	Neodymium-doped yttrium aluminium garnet
NI	National instruments
OTV	Ozone tagging velocimetry
PAF	Photoactivated fluorophores
PHANTOMM	Photoactivated non intrusive tracking of molecular motion

PIV	Particle image velocimetry
PSL	Polystyrene latex
QDs	Quantum dots
RAM	Random access memory
RELIEF	Raman excitation plus laser induced electronic fluorescence
ROI	Region of interest
SNR	Signal to noise ratio
USB	Universal serial bus
UV	Ultraviolet
μ-PIV	Micro particle image velocimetry

CHAPTER 1

INTRODUCTION

Microfluidics is defined as the science and engineering of devices having the smallest flow design feature on the scale of a micron or larger. As a result of this there is a deviation of fluid behavior from conventional flow theory (Stone & Kim, 2001; Nguyen & Wereley, 2006). The miniaturization of devices is a rapid technology development since Richard Feynman's thought provoking 1959 speech "There's Plenty of Room at the Bottom" (Feynman, 1992; Nguyen & Wereley, 2006). Biological, chemical, industrial and commercial applications supplemented by the development of technology facilitating the fabrication of individual and integrated components for lab-on-a-chip (LOC) devices and micro-electromechancial systems (MEMS) are motivating factors for research in the field of microfluidics (Gad-el-Hak, 1999; Beebe, Mensing, & Walker, 2002; Nguyen & Wereley, 2006). Some of the apparent advantages of microscale devices over conventional equipment include less sample, reagent and power consumption, reduced response time, increased sample throughput, cost effectiveness, increased portability and versatility in design (Hrnčiřík & Náhlík, 2010). The potential of microscale devices for parallel and integerated operation with other microscale devices and their capability to automate complex laboratory processes in a single device can be considered as other prospective applications for microscale devices (Hrnčiřík & Náhlík, 2010).

The application scope of microscale devices incorporating fluidic, optical and electronic components is extensive. Manz et al. (1990) highlighted the significance of microfluidics in the fields of life science and chemistry. The broad range of applications of microfluidic devices involving fluid transport includes tips of scanning electron microscope for imaging single atoms, automobile airbags accelerometers, microheat exchangers, micropumps in inkjet printing, biological cell separation reactors and blood analyzers (Gad-el-Hak, 1999).

Regardless of the vast application of microscale devices, the comprehension of the fluid behavior through these devices plays a significant role in their performance. The following section highlights the importance of the detailed understanding of the flow behavior at the microscale.

1.1 Need for microscale flow measurement techniques

The understanding of the unconventional physics involved in the operation of microscale devices has lagged from the corresponding rapid progress in the fabrication and utilization of these devices (Gad-el-Hak, 1999; Gad-el-Hak, 2006). Extensive knowledge of diverse disciplines is necessary for researchers working in the field of microfluidics. One of these diverse disciplines involved in microfluidics is fluid mechanics. The design and development of microscale devices incorporates atypical geometries and novel fluid transport techniques posing many challenging research and design questions (Stone & Kim, 2001; Stone, Stroock, & Ajdari, 2004). A description of the design, development and operation of individual and integrated systems of micro-actuators, micro-valves, micro-pumps and micro-flow sensors involving fluid transport can be found in Gravesen et al. (1993) and Shoji & Esashi, (1994).

Pressure, electrical, magnetic and surface forces, either alone or as a combination may be the fluid driving force in these microscale devices (Stone, Stroock, & Ajdari, 2004). The unique features of the flow through micro configurations discerning them from their macroscale counterparts are perhaps most intriguing for researchers in basic fluid mechanics (Ho & Tai, 1998). Only some of the interesting and useful characteristics of fluids flowing in microsystems have been exploited (Whitesides, 2006). As the size of the devices decrease the surface-to-volume ratio increases. This leads to the dominance of the surface forces over body forces which deviates the flow field from conventional flow physics as discussed in sections below.

1.1.1 Fluid flow at microscale

The physical phenomena dictating the dynamic behavior of a system are significantly influenced by the physical size of the system (Zhao, 2002). Miniaturization of devices has a significant impact on different physical phenomena giving rise to new behavior with promising applications in microfluidic devices (Abgrall & Nguyen, 2009). The forces governing the physical phenomena in microscale devices are dictated by the length scale (Ho & Tai, 1998). The surface area scales to the square of the characteristic length and the volume scales to the cube of this length. The surface to volume ratio can be expressed as

$$\frac{\text{Surface Area}}{\text{Volume}} = \frac{A}{V} = \frac{L^2}{L^3} = \frac{1}{L} \qquad 1.1$$

Where *A*, *L* and *V* are the surface area, characteristic length and volume respectively. Equation 1.1 shows that the surface-to-volume ratio has an inversely proportional relationship to the characteristic length scale. This ratio, also known as the square-cube law, increases as a result of miniaturization (Abgrall & Nguyen, 2009).

This ratio is significantly larger for microdevices compared to conventional macroscale devices (Ho & Tai, 1998). The typical surface-to-volume ratio for a microfluidic device having a size of 1 µm is 10^6 m^{-1} compared to 1 m^{-1} for a conventional machine with a characteristic length of 1 m (Gad-el-Hak, 2006). The millionfold increase implies that the surface effects become dominant compared to volume effects as a result of miniaturization (Gad-el-Hak, 2006). A significant increase in the radiative and convective heat transfer in microdevices can be attributed to the massive increase in the surface-to-volume ratio of these devices. A consequence of miniaturization is the increased significance of surface forces and the vanishing impact of body forces (Gad-el-Hak, 2006).

1.1.2 Surface forces vs. body forces

Surface forces and body forces are the two distinctive types of all the forces involved in fluid mechanics (Shaughnessy, Katz, & Schaffer, 2005). Body forces also called volume forces are long range forces proportional to the volume of the fluid. Surface forces are short range forces proportion to the surface area of the fluid (Shaughnessy, Katz, & Schaffer, 2005).

Surface forces that are usually ignored in macro scale fluid mechanics leads to new phenomena for fluid flows in microscale devices (Stone & Kim, 2001). In contrast to the body forces influencing the larger scale systems, the smaller scale systems are influenced by the surface forces (Zhao, 2002; Stone & Kim, 2001). The length of demarcation below which surface forces dominate over gravitational forces is around 1 mm based on biological studies (Went, 1968; Gad-el-Hak, 2006).

Non-dimensional numbers eliminate the dimension factor and are commonly used in fluid mechanics analysis. The significance between a pair of forces can be deduced from their ratio which is represented by a number of non-dimensional numbers. Table 1-1 lists non-dimensional numbers which demonstrates an interesting interplay of body forces verses surface forces at the microscale. Typical parameter values encountered at the microscale are used for the calculation of these non-dimensional numbers. The parameters for a small particle in a medium at the microscale are indicated in Table 1-1.

The order of magnitude calculated for the Bond number demonstrates that the gravitational forces become less important at the microscale in comparison to the surface tension forces. Despite the presence of the gravitational force the liquid will rise in the capillary tube due to the high surface force. The capillary number is useful when dealing with flows through porous media as it highlights the dominance of surface tension forces over viscous forces. The order of magnitude calculated for the capillary number clearly indicated that the viscous forces less impact at the microscale as compared to the surface tension forces. The diminishing effect of the inertial force due to the fluid momentum relative to the amplified effect of viscous friction at the microscale is depicted by the Reynolds number. The Weber number signifies the high ratio of impact of the surface tension force to the inertial force. The calculated order of magnitude of the Reynolds number and the Weber number signifies that the reduced influence of the inertial forces in comparison to the viscous forces and the surface tension forces at the microscale respectively.

Table 1-1: Dimensional numbers depicting the magnitudes of characteristic forces at the microscale: g = 10 m/sec^2, l = 1 µm, v = 1 x 10^{-6} m/sec, ρ = 10^3 kg/m^3, $\Delta\rho/\rho$ = 10^{-2}, μ = 10^{-3} Pa sec, σ =0.1 N/m.

Dimensionless Number	**Interpretation**	**Formula**	**Order of Magnitude**
Bond number	$\frac{\text{Gravitational force}}{\text{Surface tension force}}$	$\frac{g\,\Delta\rho\, l^2}{\sigma}$	$\approx 10^{-9}$
Capillary number	$\frac{\text{Viscous force}}{\text{Surface tension force}}$	$\frac{\mu v}{\sigma}$	$\approx 10^{-8}$
Reynolds number	$\frac{\text{Inertia force}}{\text{Viscous force}}$	$\frac{\rho v l}{\mu}$	$\approx 10^{-6}$
Weber number	$\frac{\text{Inertia force}}{\text{Surface tension force}}$	$\frac{\rho v^2 l}{\sigma}$	$\approx 10^{-14}$

It can be concluded from the above discussion that the fluid mechanics at microscale is significantly different from macroscale fluid mechanics. Hence, a comprehensive insight of the hydrodynamic behavior of the fluids used in microscale devices is required which in principle can produce novel, better and more efficient designs serving the specified objectives of the microfluidic devices (Stone & Kim, 2001). Computational fluid dynamics as well as experimental methods are means of investigating the hydrodynamic behavior.

An important fundamental non dimensional number related to the mass transfer and diffusion is the Peclet number. The Peclet number is equal to the ratio of the mass transport by convection and by diffusion and is given by

$$\mathrm{Pe} = \frac{\text{Mass transpot by convection}}{\text{Mass transport by diffusion}} = \frac{vl}{D} \qquad 1.2$$

where *v* is the flow velocity, *l* is the characteristic length and *D* is the diffusivity of the dye tracer in the fluid. The Peclet number gives an idea of the useful lifetime of a dye tracer mixed in the fluid for flow measurements. A high Peclet number means that the rate of diffusion is sufficiently low compared to the rate of convection and tracer motion will be dominated by the flow making it useful for flow measurements. A dye tracer will disappear rapidly due to diffusion if the Peclet number has a low value.

Fluid visualization is a fundamental experimental technique to understand the flow behavior simultaneously providing benchmark data for macro scale measurements. The fluids used extensively for flow measurements are typically transparent and invisible to the naked eye. The basic philosophy of flow visualization is to make the fluid transport detectable by altering the fluid in such a way that the fluid motion remains unchanged (Lindken, Rossi, Grooße, & Westerweel, 2009). This requirement demands totally non intrusive flow visualization techniques which are hardly ever possible. Alternatively, it can be argued that the alterations are either negligible or they can be modeled to extract the undisturbed flow information. A number of flow visualization techniques have been developed for large scale flows as well as flows at the microscale.

1.2 Microscale flow measurement techniques

Flow visualization at the microscale has been pivotal to the development of the field of microfluidics (Sinton D. , 2004). The term visualization, velocimetry and flow profiling are often used interchangeably due to the fact that most of the flow visualization techniques for microscale flows have concentrated on velocimetry or determining the fluid velocity (Sinton D. , 2004). Most of the microscale flow visualization techniques have often evolved through adaptation of their corresponding well developed macroscale without altering the fluid motion. This requirement demands a totally non intrusive technique which is rarely possible. Two non-intrusive microscale flow velocimetry techniques that have emerged from several different attempts to image flows through microchannels are micro-particle image velocimetry and molecular tagging velocimetry (Garbe, Volker, & Jähne, 2008).

1.2.1 Micro-particle image velocimetry (µ-PIV)

Micro-particle image velocimetry (µ-PIV) is a non-intrusive optical flow measurement technique facilitating comprehensive flow measurements in microfluidic devices. µ-PIV is a microscale adaptation of the well developed flow measurement technique particle image velocimetry (PIV) for large scale flows (Lindken, Rossi, Grooße, & Westerweel, 2009). It utilizes the temporal and spatial evolution of microscale particles seeded in the fluid under observation for flow measurements (Wereley & Meinhart, 2010). The basic assumption of this technique is that the particles follow the fluid with negligible influence on the flow field. The quantitative determination of two dimensional velocity data in a two dimensional measurement plane employing µ-PIV utilizes a combination of conventional microscopy and digital imaging (Lindken, Rossi, Grooße, & Westerweel, 2009).

Commercially available pulsed monochromatic light sources specifically designed for PIV applications are commonly used as illumination source (Wereley & Meinhart, 2010). Beam combining optics, two Nd:YAG laser cavities and a frequency doubler crystal allows the generation of two laser pulses of 532 nm (Wereley & Meinhart, 2007). The duration of each pulse ranges between 5-10 nsec whereas the interval between the pulses can range from tens of nanoseconds to a few seconds (Wereley & Meinhart, 2010). Longer pulse length diode pumped laser like Nd:YLF are alternative illumination sources (Lindken, Rossi, Grooße, & Westerweel, 2009). A variety of beam forming optical elements are used to deliver the illumination light to the microfluidic device. The optical elements modify the light to fill the back of the objective lens which results in a broad illumination of the region under observation (Wereley & Meinhart, 2007). Typically large magnification and high numerical aperture (NA) microscope objective are used for µ-PIV measurements (Lindken, Rossi, Grooße, & Westerweel, 2009).

Continuous chromatic light sources have also been used for illumination in μ-PIV measurements (Santiago, Wereley, Meinhart, Beebe, & Adrian, 1998). In order to allow the passage of a narrow wavelength band of light to pass reach the test section an excitation filter is employed (Wereley & Meinhart, 2007). A mechanical or electro-optical shutter is typically used before the image recording device in order to gate the light reaching the recording device (Wereley & Meinhart, 2010).

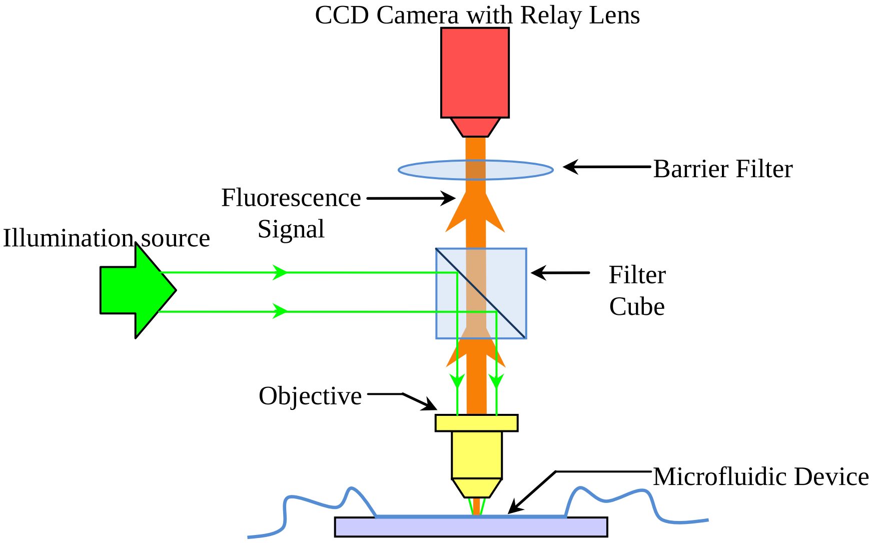


Figure 1.1: A schematic of a typical epifluorescent μ-PIV setup.

A schematic of a typical epifluorescent μ-PIV setup is shown in Figure 1.1 which demonstrates the arrangement of a variety of components utilized in the construction of the setup. A particle seeded working fluid is directed through the test microfluidic device placed in front of the objective lens. The fluorescence signal generated from the particles is relayed back to the imaging device via the same objective lens. The filter cube placed between the barrier filter and the objective lens typically contains a dichroic mirror assembly which reflects the illuminating light whereas permits the fluorescence signal to pass through. The barrier filter shown in front of the camera is typically a high pass filter. This filter is used to filter out the reflected illumination light or noise from the fluorescence signal. In an epifluorescence configuration the fluorescing light and the excitation signal are transmitted through the same objective lens as shown in Figure 1.1. A sensitive large format interline transfer CCD camera is utilized to capture the fluorescence signal of the particles because it increases the spatial dynamic range of the measurements by capturing more particle images. The motion of the fluid is inferred with a good accuracy and spatial resolution by correlating the location of the particles within successive frames taken a suitable interval (*Δt*) apart.

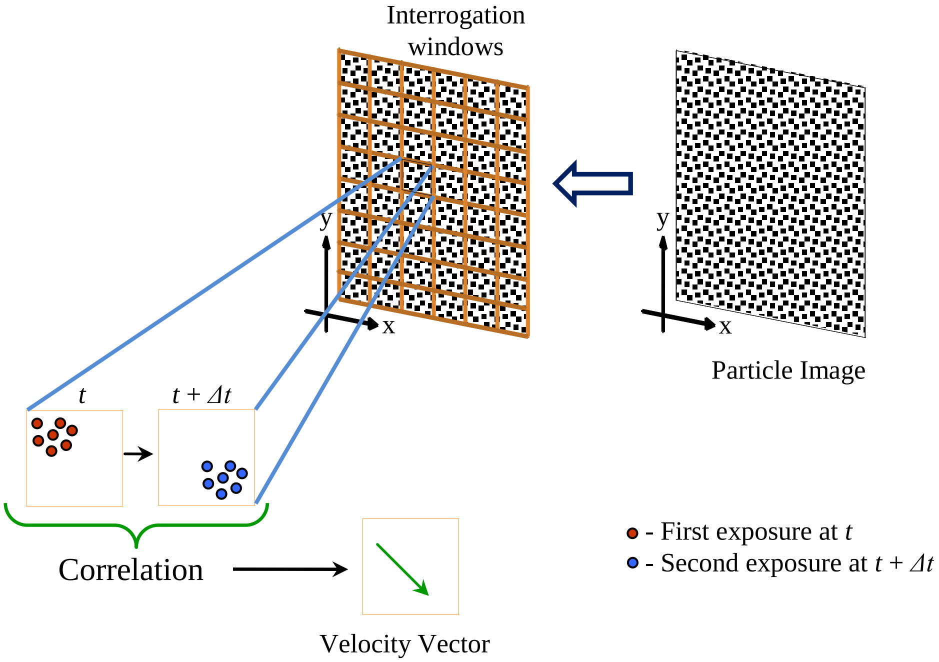


Figure 1.2: Procedure of velocity vector calculation for µ-PIV.

The steps involved in the processing the particle images are demonstrated in Figure 1.2. Two snapshots (usually referred to as frames) of the flow within a short controlled time interval are used to deduce the velocity information of the flow. The two frames are divided into small regions which are called interrogation windows as shown in the figure. Each interrogation window contains a small number of particles. Each interrogation window in the first frame is correlated to the corresponding window in the second frame to reveal the local displacement vector at the given time interval. All the particles within one interrogation window are assumed to have moved homogeneously between the two frames. Having the local displacement information and the time interval (Δt) the local instantaneous velocity information is revealed. All the interrogation windows of the recording are correlated in the same manner. Hence, the instantaneous velocity field for the fluid flow is determined from the particle images. A more detailed discussion about the processing of particle images can be found in Raffel et al. (2007).

1.2.1.1 Flow tracers

A variety of particles for µ-PIV applications are commercially available. Typically fluorescently labelled polystyrene latex (PSL) is used in the manufacturing of these particles (Lindken, Rossi, Grooße, & Westerweel, 2009; Wereley & Meinhart, 2010). The flow tracing particles used for µ-PIV measurements have typical diameter range from 200 nm to 2 µm (Wereley & Meinhart, 2010). Ideally the particles are required to follow the fluid flow faithfully. This assumption is fulfilled by optimizing the particles in such a way that either their size is sufficiently small compared to the length scale or by matching their density with the fluid under observation (Sharp & Adrian, 2005; Lindken, Rossi, Grooße, & Westerweel, 2009). The size of the particles must also be large enough to significantly dampen Brownian noise, can be imaged easily and do not clog the flow device (Wereley & Meinhart, 2010). Apart from the size of the particles, particle density is another important parameter which needs to be optimized for µ-PIV measurements. A high particle density will result in a high level of background noise from out of focus particles whereas macroscopic correlation methods may become unreliable for low particle densities (Sinton D. , 2004).

One of the drawbacks of using the fluorescently labelled microspheres is that either they fluoresce weakly through the volume or typically fluoresce only on the surface as a result of quenching effects (Singh, Cummings, & Throckmorton, 2001). This limits the signal-to-noise ratio of the particle images using typical µ-PIV configurations (Singh, Cummings, & Throckmorton, 2001). The aggregation of particles, nonspecific adsorption of hydrophobic solutes on the particle surface and adhesion of latex particles to the microchannel surface due to the hydrophobic nature of polystyrene complicates the use of fluorescently labelled PSL microspheres for µ-PIV applications. Fluorescent liposomes were developed by Singh et al. (2001) to be used as alternate flow markers for µ-PIV measurements. These liposomes are hydrophilic, neutrally buoyant in water, approximately 280 nm in diameter and provide a good signal-to-noise ratio because they carry a large number of fluorophoes (up to 80,000 fluorescein equivalent) (Singh, Cummings, & Throckmorton, 2001).

Pouya et al. (2005) introduced a new type of flow tracing particles having several advantages over fluorescent PSL microspheres called quantum dots (QDs). The small size of the QDs of around 18 nm hydrodynamic diameter, inability to photobleaching and the demonstration of a relatively large Stoke's shift are some of the dominating features of the quantum dots over fluorescently labelled PSL microspheres (Wereley & Meinhart, 2010). The QDs are more susceptible to high levels of Brownian motion due to their small size. They also exhibit random blinking behaviour and have weak signal intensity (Lindken, Rossi, Grooße, & Westerweel, 2009). However, these problems were resolved by conjugating around 60 QDs to a single polystyrene nanoparticle which results in a larger particle size (Freudenthal, Pommer, Meinhart, & Piorek, 2007). Apart from the flow tracers described above, there is a larger number of possible choices for the particle types for use in µ-PIV measurements.

1.2.1.2 Conventional PIV and µ-PIV

Although the µ-PIV technique has a lot of similar features to PIV but it is considered as a separate measurement technique due to various considerable optical and mechanical constraints (Wereley & Meinhart, 2010). Wereley et al. (2002) outlined the fundamental aspects differentiating conventional PIV and µ-PIV. The particles utilized for µ-PIV measurements are typically on the order of 100-300 nm so that the particles do not disturb the flow being measured (Prasad, Adrian, Landreth, & Offutt, 1992; Meinhart, Wereley, & Santiago, 1999). The diameter of these particles is less than the illumination wavelength (e.g. $\lambda = 532$ nm). Hence, normal elastic scattering techniques will prove to be difficult for flow measurements in this regime (Meinhart, Wereley, & Santiago, 1999). The utilization of fluorescently labelled particles imaged using an epifluorescent configuration and an optical filter to remove the background noise is a potential solution to this problem (Wereley, Gui, & Meinhart, 2002).

The effect of Brownian motion including individual particle-fluid and particle-particle interactions becomes more significant as the size of the particles is reduced (Sinton D. , 2004). The increased significance of Brownian motion prevents the particles from faithfully following the fluid flow. The error in the velocity measurements arising as a result of Brownian motion is unbiased (Meinhart, Wereley, & Santiago, 1999; Wereley, Gui, & Meinhart, 2002). This unbiased error can be significantly reduced by averaging several particle images either within each single interrogation spot or ensemble averaging over several interrogations (Santiago, Wereley, Meinhart, Beebe, & Adrian, 1998; Meinhart, Wereley, & Santiago, 1999; Wereley, Gui, & Meinhart, 2002).

A laser sheet determines the out of plane depth of the measurement domain in conventional PIV (Sinton D. , 2004). The lack of optical access and the diffraction in light sheet forming optics are two major issues which makes the use of light sheets impractical for µ-PIV (Wereley, Gui, & Meinhart, 2002). Consequently this problem is overcome by utilizing volume illumination in an epifluorescent configuration where a common microscope objective is shared by the fluorescing and excitation light (Wereley, Gui, & Meinhart, 2002; Sinton D. , 2004). The depth of the measurement region is defined by the depth of focus of the imaging optics. The background noise of out of focus particles is removed using digital imaging filter in the case of small depth of focus imaging systems (Sinton D. , 2004). In the case of larger depth of focus imaging systems all the particles appear to be in focus. Advanced processing techniques are utilized to extract the information along the depth of the flow measurements (Wereley, Gui, & Meinhart, 2002; Sinton D. , 2004).

The first application of µ-PIV was demonstrated by Santiago et al., (1998). They presented instantaneous and averaged pressure driven Hele-Shaw flow field around a 30 µm wide obstacle in a 120 µm square field-of-view utilizing 300 nm polystyrene particles. They utilized continuous illumination by means of mercury arc lamp. Meinhart et al., (1999) extended the application of µ-PIV by the use of a 5 nsec pulsed Nd:YAG laser instead of continuous illumination. They presented ensemble averaged velocity measurements in a 30 x 300 µm^2 rectangular cross section microchannel less than 1 µm from the wall for a pressure driven flow using 200 nm flow tracing particles. The µ-PIV technique was first applied to study an electrokinetic flow in planar microchannels by Cummings, (2000). Meinhart & Zhang, (2000) conducted the first application based µ-PIV study. They analyzed the complex pressure driven flow structure inside a microfabricated inkjet printhead using a 532 nm pulsed Nd:YAG laser and 700 nm diameter flow tracing polystyrene particles. Significant amount of work has been done in the development of µ-PIV. Several reviews are available for the advancement in the flow measurements using this technique (Lindken, Rossi, Grooße, & Westerweel, 2009; Lee & Kim, 2009; Wereley & Meinhart, 2010).

1.2.1.3 Limitations of µ-PIV

The implicit assumption for µ-PIV measurements is that the particles follow the flow faithfully with negligible influence on the flow field. Although this assumption is valid for macroscale flow fields but the fact remains that even extremely small particles are incapable of following the flow as efficiently as molecules. In some situations the interaction of the particles with either the flow process or the device may lead to inaccuracies (Koochesfahani & Nocera, 2001). Similarly there are situations where particles fail to track the fluid motion accurately due to their inertia or buoyancy (Koochesfahani & Nocera, 2001). Lim et al., (2003) reported the dynamic pattern formation of colloidal particles within a microchannel flow due to a significant hydrodynamic lift force experienced by the particles.

1.2.2 Molecular tagging velocimetry (MTV)

Molecular tagging velocimetry (MTV) is a laser based, non-intrusive flow measurement technique based on the photoinjection of a specialized molecular scale flow tracer. In contrast to particle based flow visualization techniques such as PIV or PTV, MTV does not requires the use of seeding particles (Koochesfahani & Nocera, 2001; Lempert & Harris, 2000; Maynes & Webb, 2002). Instead, MTV utilizes fluorescent molecular dye tracers premixed in the fluid under observation which are activated upon photoinjection (Beushausen et al., 2009). Due to the molecular dimensions of the flow tracer MTV can be considered as a molecular counterpart of particle image velocimetry particle image velocimetry (PIV) (Bohl, Koochesfahani, & Olson, 2001; Koochesfahani

& Nocera, 2001; Bohl & Koochesfahani, 2004). A detailed description of MTV is provided in Chapter 2.

Molecular tagging velocimetry offers several advantages over µ-PIV. The particle induced disturbances in the flow are eliminated as a result of the use of fluorescent molecules as flow tracers. The tracer molecules can accurately track the fluid as compared to particles. A uniform distribution of molecular tracing is more easily achieved because they do not settle out like particles (Hsu, Srinivasan, Bowersox, & North, 2009). The density mismatch of the particles with the fluid under observations may lead to complications and errors in the velocity measurement using tracer particles (Hu & Koochesfahani, 2006). However, the molecular tracers used for MTV are not prone to density mismatch because they are dissolved in the fluid used for the measurements.

The use of particles for flow measurements through dense internal geometries such as packed columns is prohibitive because the particles will essentially be filtered out of the fluid (Maynes & Webb, 2002; Thompson, Maynes, & Webb, 2002; Garbe, Volker, & Jähne, 2008). The use of molecular tracers allow the flexibility of choosing the time and site of tracer introduction by means of photoinjection. When using fluorescent tracer particles for µ-PIV measurements all the particles exposed to the excitation light source will fluoresce simultaneously. Therefore, it can be argued that MTV complements PIV where the use of particles leads to complications and does not provide reliable flow information (Koochesfahani & Nocera, 2001; Bohl, Koochesfahani, & Olson, 2001).

1.2.2.1 Current limitations of MTV

The measurements performed using MTV at the microscale have been mostly one dimensional (Paul, Garguilo, & Rakestraw, 1998; Maynes & Webb, 2002; Mosier, Molho, & Santiago, 2002; Sinton, Escobedo-Canseco, Ren, & Li, 2002; Thompson, Maynes, & Webb, 2002; Shelby & Chiu, 2003; Sinton & Li, 2003; Thompson, Maynes, & Webb, 2005). Typically, the flow information is inferred by tracking the evolution of a line tagged perpendicular to the channel axis. This reveals the flow profile and one dimensional velocity information in the cross stream.

Laser grid tagging has been utilized for two dimensional flow measurements using MTV at the macro scale (Gendrich, Koochesfahani, & Nocera, 1997). Due to the lack of sufficient optical access the grid method cannot be utilized for macroscale flows.

A mask based approach is recently developed for two dimensional flow measurements based on MTV for microscale flows (Garbe, Volker, & Jähne, 2008; Beushausen, Roetmann, Schmunk, Wellhausen, Garbe, & Jähne, 2009). However, the use of the mask restricts the write laser energy which may lead to improper uncaging or photobleaching. The requirement of separate individual masks in order to accommodate different tagging patterns also complicates this method. A detailed description of the two dimensional methods for MTV measurements is given in section 2.6. In order to avoid these limitations a scanning two dimensional velocity measurement system has been designed and developed.

1.3 Summary and thesis outline

The thesis discusses the design and development of a two dimensional microscale flow velocity measurement system on the principle of MTV. This design and development includes the selection and assembly of specific hardware components based on performed design calculations. The sequential operation of these hardware components is crucial for the successful operation of the developed measurement system. It is a complex system that requires the precise temporal operation of several individual components utilized in the design of the system. The quality of the acquired data is also crucial for the accurate processing of the data. The design of the system is hence considered incomplete without a program capable of organizing the hardware operation and controlling the data properties. As part of the design process, an image acquisition graphical user interface is developed which not only controls the timely sequential operation of the hardware components but also enables the user to adjust the quality of the acquired data in real time.

A review of the molecular tagging velocimetry technique will be discussed in chapter 2 of this thesis. The different mechanisms of MTV based on the type of flow tracers will be described in this chapter. This description will include the type of the flow tracers used, the mechanism utilized for flow measurement measurements, typical hardware operating parameters and the advantages and disadvantages of each mechanism. The description of application of MTV to microscale flow concludes the chapter with emphasis on the two dimensional microscale measurements

The hardware design of the two dimensional microscale scanning molecular tagging velocimetry system will be presented in chapter 3. The motivation behind the development of the system will be followed by the basic concept behind the design of the system. The design calculations important for the selection of the actual hardware components are performed. The selection procedure of the major components imperative for building the designed measurement system is described. The chapter finishes off with the discussion of possible hardware configurations highlighting the potential of the utilizing the same system for various flow measurement mechanism.

Chapter 4 is dedicated to the design of the program for controlling the operation of the designed system and the acquisition and the processing of the data. The need for the requirement of a dedicated program harbouring the necessary controls is highlighted. A description of the various controls of the designed program is provided. A sequence of operation for collecting flow data using the developed software for MTV and µPIV is discussed. The acquired data is processed to reveal flow velocity information using commercially available software. The procedure involved in extracting the velocity information for each of the flow measurement techniques is elucidated.

The velocity results obtained using µ-PIV and MTV are examined in chapter 5. The properties of the designed system are reviewed. Raw data with corresponding processed velocity each for µ-PIV and MTV are presented and discussed. A comparison of the data and the velocity fields obtained using the two flow measurement technique is performed. The advantages and disadvantages of MTV mechanism in comparison with µ-PIV will conclude this chapter. An error analysis of the results obtained for the photobleaching MTV is presented at the end of this chapter.

Despite of all the efforts to develop a comprehensive system there is always room for improvement. Chapter 6 will venture the possibilities of the performance enhancement of the overall designed system. The hardware modifications promising efficiency enhancement of the system with regard to potential future applications are discussed.

A set of appendices are given following chapter 6 for completeness. Appendix A contains the diagrams of all fabricated parts for use in the assembly of the system. The specification of the hardware components used for the assembly of the system is listed in Appendix B. The parameters used in the processing of the images in DaVis 7.4 are mentioned in Appendix C. The code operating at the back of the graphical user interface for the operation of the system is given in Appendix D.

CHAPTER 2

MOLECULAR TAGGING VELOCIMETRY

2.1 Introduction to MTV

Molecular tagging velocimetry (MTV) is a laser based, non-intrusive, time resolved, whole field technique capable of providing complete velocity profile information of fluid flows from a single measurement (Bohl, Koochesfahani, & Olson, 2001; Maynes & Webb, 2002; Sinton & Li, 2003). This technique is based on molecular compounds, used as flow tracers that exhibit long lifetime emission, compared to the flow convection time scale, upon exposure to photons of appropriate wavelength (Gendrich & Koochesfahani, 1996; Bohl, Koochesfahani, & Olson, 2001; Koochesfahani & Nocera, 2001)The molecular flow tracers are useless until exposed to a light source. This process known as photoinjection is used to activate the molecular tracers for conducting flow measurements.

A discussion of molecular tagging velocimetry technique is presented in detail in this chapter. An overview of the different mechanisms that are grouped under the common heading of MTV is discussed. This discussion includes the physics involved in each mechanism, the type of tracers used and the use of the technique in terms of advantages and disadvantages. The application of MTV to microscale flows is also discussed in the later half of this chapter.

2.2 Basic principle

Molecular tagging velocimetry includes various mechanisms based on contrasting photophysics of the utilized molecular compounds. A generalized principle of MTV is represented in Figure 2.1. The two important parameters to be considered when using MTV are the choice of a molecular tracer and the practical implementation of tagging methods, detection and processing (Koochesfahani M. M., 1999). Specialized molecular tracers have been developed for various applications such as velocimetry and thermometry (Hu & Koochesfahani, 2006). The selection of the tracers for MTV is based on the medium of fluid and the speed range of the measured flow field keeping in view the compatibility issues of the flow facility (Koochesfahani M. M., 1999). These specialized molecular tracers are engineered to be invisible until exposed to a certain wavelength of light. The type and the number of required photon sources are dictated by the photophysics of the selected molecular tracer whereas the number of required velocity components dictates the method of tagging and detection (Koochesfahani M. M., 1999). The selection of the data processing approach is influenced by the selected tagging method (Koochesfahani M. M., 1999). Regardless of the selected flow tracer, method of tagging, detection and the data processing scheme, a flow chart representing the general steps involved in MTV are shown in Figure 2.1.

There are three basic steps involved in molecular tagging velocimetry, ‘Writing’, ‘Reading’ and ‘Interrogation’. Application specific flow tracers are premixed homogeneously in the fluid under observation. This mixture is made to flow through the desired flow geometry. This flow is exposed to an intense beam of light of a specific wavelength. This photo-injection activates the molecular tracer only in the region exposed to the light. This process is usually termed as “Writing” or “Tagging”. Typically a pulsed laser is utilized to execute this step, tagging the flow multiple times during each measurement. The laser used for “tagging” may be termed as the “Write Laser" or the “Tagging Laser”. The flow tracers are activated only in the region exposed to this ‘Write Laser’.

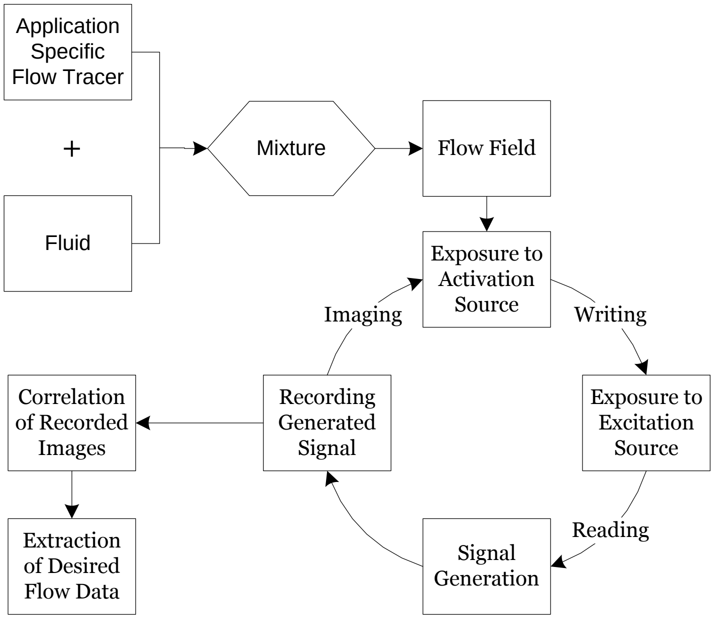

Application Specific Flow Tracer
+
Fluid
Mixture
Flow Field
Exposure to Activation Source
Writing
Exposure to Excitation Source
Reading
Signal Generation
Recording Generated Signal
Imaging
Correlation of Recorded Images
Extraction of Desired Flow Data

Figure 2.1: Flow chart representing the different steps involved in a typical molecular tagging velocimetry system.

Even after activation the flow tracers are invisible and do not show any significant physical change that can be utilized for flow measurements. A second light source (excitation source) is required to excite the activated tracer molecules. This exposure to the second light source, typically a continuous wave laser, may be termed as “Reading”. The laser used to perform this process is called the “Read Laser”. Since the position of the activated tracer molecules is ambiguous, the read laser covers the entire field-of-view (FOV) either illuminating a volume or a sheet of certain thickness depending on the region under investigation. This results in the generation of a signal from the activated tracer molecules that can be utilized for flow measurements.

This signal is then captured onto an image recording device either as a time series or a set of time resolved single frame or double frame images that can be processed using the appropriate correlation algorithm. The processing of the recorded images reveals the desired flow property information.

2.3 Mechanisms of MTV

Molecular tagging velocimetry is a broader common heading encompassing diverse flow measurements involving molecular flow tracers regardless of the details of the photophysics associated with each molecular design (Koochesfahani M. M., 1999). Keeping in view the various inter-related parameters described above, it is necessary to familiarize with the different mechanisms of MTV in order to select the suitable one for the desired application. MTV can be divided into four basic mechanisms, based on the photophysics of the molecular tracer used in the design. The four mechanisms that have been reported in literature till date are as follows

1. Laser induced photochemical anemometry (LIPA)

2. Raman excitation plus laser induced electronic fluorescence (RELIEF)

3. Photoactivated non intrusive tracking of molecular motion (PHANTOMM)

4. MTV by direct phosphorescence.

The mechanisms of MTV are based on the unique chemical properties exhibited by the molecular tracers in their metastable electronically and/or vibrationally excited states (McKeon, et al., 2007). The highly reactive electronically excited state is usually achieved upon photon absorption. These excited molecules tend to return either to their original ground state or to a new ground state with (intermolecular decay) or without (intramolecular decay) a reacting partner (McKeon, et al., 2007). The decay of the molecule from the excited state is an amalgamation of photon and thermal emission. The photon emission is characterized by the radiative rate constant whereas the thermal emission is depicted by the non-radiative rate constant (McKeon, et al., 2007).

The change in molecular absorbance or luminescence of the molecule upon photon excitation is utilized in the design of different MTV mechanisms. The two types of luminescence utilized by the MTV mechanisms are fluorescence and phosphorescence. Fluorescence is a short live radiative process which supplements the transition of the molecule from the quantum mechanically permitted singlet excited state to a singlet ground state (McKeon, et al., 2007). The radiative rate constant for fluorescence is significantly higher than the non-radiative rate constant which in turn leads to an intense short duration emission typically on the order of 1 to 100 nsec (mostly 2 - 10 nsec) (McKeon, et al., 2007). Phosphorescence however, is a long lived radiative process which accompanies a quantum mechanically forbidden transition between the excited state and the ground state, having different spin (McKeon, et al., 2007). In this case the radiative and the non radiative rate constant are comparable with the radiative time scale ranging from micro to milliseconds (McKeon, et al., 2007). This longer lifetime makes the phosphorescence signal prone to quenching as a result of reaction with the water vapours or oxygen present in the environment (McKeon, et al., 2007). Molecular tagging velocimetry based on different mechanisms is described in detail below.

2.3.1 Laser induced photochemical anemometry (LIPA)

Laser induced photochemical anemometry (LIPA), a mechanism of MTV, is a non intrusive optical technique capable of revealing qualitative as well as quantitative information about flow phenomenon in liquid and gaseous flows (Chu & Liao, 1992; Chu, Wang, & Hsieh, 1993). Instantaneous velocities on a planar area at multiple locations in unsteady flows can be measured simultaneously using LIPA (Chu & Liao, 1992; Chu, Wang, & Hsieh, 1993). It is the only MTV mechanism which utilizes absorbance instead of luminescence for flow measurements (McKeon, et al., 2007). Miller, (1967) was the first to use this mechanism for fluid mechanics measurements using a UV flash tube to excite the photochromic solution in his undergraduate work at the University of Toronto (Sinton D. , 2004). However Popovich and Hummel may be considered as the pioneers in using LIPA for flow visualization. They studied the turbulent flow characteristics near a channel wall using a photochromic pyridine dye (Popovich & Hummel, 1967). The properties of the flow tracers and the mechanism are described below.

2.3.1.1 Photochromic dye tracers for LIPA

A chemical species exhibiting a reversible color change between two states with significantly different absorption spectra stimulated in at least one direction by exposure to electromagnetic radiation may be referred to as a photochromic dye tracer (Kurada, Rankin, & Sridhar, 1994). The photochromic dye is mixed in the fluid under observation to give a colorless solution (Kurada, Rankin, & Sridhar, 1994). Upon exposure to a certain characteristic wavelength of ultraviolet light the forward reaction (transparent to color) is induced within a few microseconds however the rate of reverse reaction depends on the temperature and the type of the solvent used. The two most widely used photochromic dyes is the benzyl pyridine and the spiropyran dye which are listed in Table 2-1. These photochromic tracers are insoluble in water and are reusable as they convert back to the original form after a non-radiative (heat) release of energy. Most studies using photochromic dyes have used organic solvents (e.g. kerosene) for experimentation as can been seen in the table.

2.3.1.2 LIPA measurement technique

Fluid flow measurements using LIPA utilize the properties of a photochromic molecular tracer which are normally colorless when dissolved in a non polar liquid (Kurada, Rankin, & Sridhar, 1994). In this technique a nanometer duration pulse from a write laser, usually a UV laser (e.g. 351 nm) converts the photochromic tracer to a tautomer which is more absorbing than the initial photochromic molecule (McKeon, et al., 2007). The absorption spectrum of the tautomer is different from that of the original tracer and usually lies within the visible spectrum (Frantisak, De, Smith, & Hummel, 1969). This exposure to the write laser generates a colored region along the path of excitation. This flow field under investigation is then exposed to another light source, which is usually a white light source, to read the motion of the excited region along with the flow. The photochromic process is a reversible process and the tracer lifetime is about 3 or 4 msec (Frantisak, De, Smith, & Hummel, 1969). The chemical reaction after exposure to the tagging laser is instantaneous however the duration of the color change may be on the order of seconds to minutes (Sinton D. , 2004).

Molecular tagging velocimetry using photochromic tracers offers several advantages. The photochromic tracer converts rapidly to the more absorbing tautomer form within the duration of a nanosecond laser pulse generating a signal promptly (McKeon, et al., 2007). This tautomer form can persist for several seconds to minutes before it converts back to its original form which allow flow measurements in slow speed flows. The photochromic dye tracer can be used multiple times since it returns to its original form after each excitation. One of the drawbacks of this technique is the insolubility of the photochromic dye tracers in water. It is evident from Table 2-1 that most of the studies using photochromic dyes have incorporated organic solvents. However, water soluble compounds are becoming available (Sinton D. , 2004). The overall applicability of photochromic dyes to molecular tagging velocimetry is limited due to the complications in data collection and analysis The captured images record the change of absorbance which requires the measurement of the difference between the incident and transmitted light (McKeon, et al., 2007). Measuring emitted light against a dark background is favorable and accurate compared to measuring the change of absorbance. Therefore MTV based on mechanisms utilizing luminescence has seen more advancement as compared to MTV based on LIPA (McKeon, et al., 2007).

Table 2-1: A review of the various flow tracers and solvents used for laser induced photochemical anemometry (LIPA).

Researcher	Dye	Solvent	Flow visualization or velocimetry
Park et al., (1999)	1,3,3,-trimethyl-6-nitro-indoline-2-spiro-2-2-benzopyrane	Deodorized kerosene (Shell-Sol 715)	Axial and radial velocity fields of a steady two dimensional flow field in the plane of symmetry of a 90º curved tube at *Re*=250.
Couch et al., (1996)	1,3,3-trimethylindoline-6-nitro-benzospiropyran	Deodorized kerosene (Shell-Sol 715)	Full field velocity field measurement in a human carotid bifurcation model at *Re*=1200 using a photochromic grid based technique.
Chu et al., (1995	1,3-trimethyl-8-nitrospiro[2-H-l -benzopyran-2,2'-indoline	Deodorized kerosene (FC-180)	The effect of orifice dimensions, Reynolds number and initial vorticity distribution of two head on colliding vortices.
Zalzal et al., (1994)	l ,3 ,3 -trimethyl-6-nitroindolene-2-spiro-2-benzopyran	Deodorized kerosene	Photochromic dye used for visualizing transition pipe flow.
Chu & Liao, 1992	1,3-trimethyl-8-nitrospiro[2-H-l -benzopyran-2,2'-indoline	Deodorized kerosene (FC-180)	Langrangian flow velocity measurement behind an impulsively started cylinder between *Re* = 500 and *Re* = 3000.

Chu et al., (1993)	1,3-trimethyl-8-nitrospiro[2-H-1 -benzopyran-2,2’-indoline	Deodorized kerosene (FC-180)	Effect on laminar vortices approaching a solid planer surface and a free surface over a range of Re_{Γ} from 900 to 2350.
Yurechko & Ryazantsev, (1991)	1,3,3-trimethylindolino-6-nitrobenzopyrylospiran (TMINBPS)	Water/Alcohol	Evaluation of LIPA based on the fluid motion in a flow-through and non-flow-through sytem.
Ojha et al., (1989)	1, 3, 3 - trimethylindoline-6-nitrobenzospiropyran (TNSB)	Deoderized kerosene (Shell-Sol 715)	One dimensional pulsatile flow measurements in constricted tubes for different constrictions.
Frantisak et al., (1969)	2-(2, 4 – dinitrobenzyl) pyridine	95% Ethyl alcohol	Measurement of mean velocity profile and mean wall shear stress in turbulent flow for various Reynolds number.
Popovich & Hummel, (1967)	2-(2, 4 – dinitrobenzyl) pyridine	95% Ethyl alcohol	Pioneers of LIPA. Study of viscous sub layers in a turbulent pipe flow.

2.3.2 Raman excitation plus laser induced electronic fluorescence (RELIEF)

Raman excitation plus laser induced electronic fluorescence or RELIEF, a mechanism of MTV, is a non intrusive optical flow velocity measurement technique utilizing naturally present species in air flows. No seeding is required since naturally occurring oxygen molecules are used as flow tracers in air. This technique was developed by Miles, et al., (1989) for instantaneous velocity measurement in air flows. RELIEF consists of two separate processes, stimulated Raman excitation (RE) of oxygen molecules and laser induced electronic fluorescence (LIEF) of these excited molecules from the Schumann-Runge band (Noullez, Wallace, Lempert, Miles, & Frisch, 1997). MTV based on RELIEF is the most spectroscopically complicated MTV technique (McKeon, et al., 2007).

2.3.2.1 Flow tracer for RELIEF

Vibrationally excited oxygen molecules are used as flow tracers for flow measurements based on RELIEF. Naturally occurring oxygen in air flows is vibrationally excited by stimulated Raman scattering. Homonuclear diatomic oxygen molecules have a relatively long lifetime in this vibrationally excited state (Miles & Lempert, 1997). Deactivation because of dipole radiation is prevented due to the molecular symmetry of oxygen whereas the collision interactions are curtailed as a result of the absence of a dipole moment and a mismatch between the vibrational energy and the translational or rotational energy of the molecule (Miles & Lempert, 1997). This long lifetime makes it appropriate for use as flow tracer. Nitrogen being more abundant in air is not suitable as a flow tracer for RELIEF because of the unavailability of commercial ultraviolet lasers for interrogation (Miles & Lempert, 1997). Compared to oxygen, the high energy ultraviolet absorption spectrum of nitrogen masked by oxygen absorption makes it impractical to be interrogated using laser induced fluorescence (Miles & Lempert, 1997).

2.3.2.2 RELIEF measurement technique

A short pulsed, two color, high power laser beam is used to vibrationally excite the naturally occurring oxygen molecules in the air flow converting them into flow tracers in a non linear process involving two photon transition (Miles, Grinstead, Kohl, & Diskin, 2000). The selection of the colors of the excitation laser is based on the energy difference between the vibrationally excited and ground states of the oxygen molecule (Miles, Grinstead, Kohl, & Diskin, 2000). A

532 nm second harmonic of a Nd:YAG laser combined with either 580 nm of a dye laser or a Raman cell frequency shifter driven by the 532 nm line of the same Nd:YAG is used for generating the two color laser beams (McKeon, et al., 2007). The excitation pulse length is approximately 10 nsec and typical pulse energy is approximately 50 mJ (Miles & Lempert, 1997). The lifetime of vibrationally excited state of oxygen is of the order of 30 ms at 1 atm (Miles, Grinstead, Kohl, & Diskin, 2000). The oxygen molecules are excited equivalent to the beat frequency of the two laser beams and the probability of excitation is proportional to the product of their intensities (Miles & Lempert, 1997). This excited state of oxygen is highly prone to dissociation into oxygen atoms due to interaction with water vapors.

The excited molecules are advected with the flow and are interrogated by electronic fluorescence using another laser, typically an ultraviolet laser pulse at 193 nm produced by an Argon Fluoride laser (Noullez, Wallace, Lempert, Miles, & Frisch, 1997). Exposure to the ultraviolet interrogation laser excites the vibrationally excited oxygen molecules to the Schumann-Runge band. These excited molecules return to the ground state emitting a fluorescence signal in the near ultraviolet (200 - 400 nm) range (Noullez, Wallace, Lempert, Miles, & Frisch, 1997). Most of the molecules dissociate into oxygen atoms and do not participate in fluorescence due to the high predissociation rate of the Schumann-Runge band (Miles & Lempert, 1997). The electronically excited state of oxygen has a lifetime of the order of 2.5 psec whereas the fluorescence lifetime is around 30 – 50 nsec (Miles, Grinstead, Kohl, & Diskin, 2000). This results in the fluorescence of 1 molecule out of every 10^4 excited oxygen molecules. Typical pulse length for the interrogation laser is also on the order of 10 nsec and pulse energy range between 50 – 100 mJ. Both the tagging and interrogation processes are instantaneous. The temporal and the spatial resolution of RELIEF depend on the tagging and interrogation time resolution and the optical system spatial resolution respectively (Noullez, Wallace, Lempert, Miles, & Frisch, 1997). For practical MTV applications the time delay between tagging and interrogation has been of order 10 µsec or less however the usable lifetime of the tracer is in the 100 ms range due to deactivation of the vibrationally excited oxygen with water and carbon di oxide (McKeon, et al., 2007).

A process known as vibrational up-pumping between vibrational states is observed in vibrationally excited oxygen. This process is a characteristic of diatomic molecular collision processes and is a result of the anharmonicity among the vibrational states (Miles, Grinstead, Kohl, & Diskin, 2000). This vibrational up-pumping drives the vibrationally excited molecules, in timescales of microseconds, to a state with higher fluorescence yields whose laser induced fluorescence can be separately identified (Miles & Lempert, 1997; Miles, Grinstead, Kohl, & Diskin, 2000). This improves the signal-to-noise ratio (SNR) of the flow tagging measurements.

RELIEF offers several advantages and has some disadvantages. This technique can be used where the use of seed particles is undesirable or will lead to inaccurate measurements. Examples include large scale facilities where uniform seeding is highly difficult since the particle seeding density is restricted by secondary scattering, sampling ambiguities and coherent scattering effects, highly sensitive equipment such as aircraft engines where the use of particles may damage the components, reactive processes where particles might react with the environment, and high speed flow measurements prone to inaccuracies resulting from particle lag. Simple observation of RELIEF images provides instantaneous appreciation of the velocity profile and the turbulence intensity. Typically for MTV applications two laser frequencies are required, one for tagging and one for interrogation. However, for MTV based on RELIEF three laser frequencies are required, two for tagging and one for interrogation. MTV based on RELIEF is restricted to flow measurements in gaseous phase oxygenated flows. In addition the selection of the flow tracers is limited to oxygen due to the unavailability of commercial lasers appropriate for use with other potential tracers e.g. nitrogen. The use of ultraviolet compatible experimental equipment poses another limitation for MTV based on RELIEF.

2.3.3 Photoactivated non intrusive tracking of molecular motion (PHANTOMM)

Photo activated non intrusive tracking of molecular motion (PHANTOMM) is a molecular tagging velocimetry mechanism based on photo activated fluorophores (PAF) or caged compounds (Biage & Campos, 2003). This mechanism is analogous to laser induced photochemical anemometry (LIPA) and may be considered as its luminescent counterpart (McKeon, et al., 2007). It is based on the selective photolysis and fluorescence of the otherwise non fluorescent caged compound premixed in the test fluid. The first application of caged compounds for flow visualization termed as PHANTOMM was reported by (Lempert, Ronney, Magee, Gee, & Haugland, 1995). The work involved the study of a fully developed laminar flow in a 1.4 cm internal diameter pipe as well as wing tip vortices in a large scale water channel facility with small seeded portions of the flow with the caged fluorescent dye. They also conducted a detailed investigation of the properties of the photo activated fluorophores critical for flow velocimetry applications. PHANTOMM based on different flow tracers can be found under different acronyms in literature. These acronyms include HTV (hydroxyl tagging velocimetry), OTV (ozone tagging velocimetry), APART (air photolysis and recombination tracking), LEI (laser enhanced ionisation) and MTV (PHANTOMM) based on photobleaching. Details of PHANTOMM under the mentioned acronyms can be found in McKeon, et al., (2007). In view of the work described in the forthcoming chapters, following two approaches of PHANTOMM will be discussed here.

1. PHANTOMM based on caged fluorescence.

2. PHANTOMM based on photobleaching.

2.3.3.1 PHANTOMM based on caged fluorescence

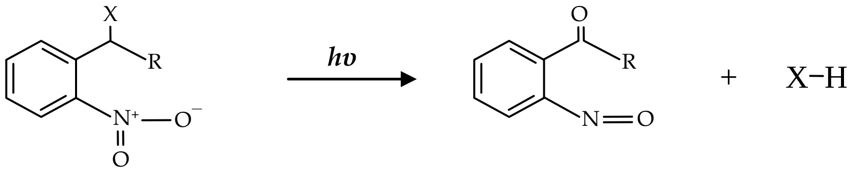
Photo activated fluorophores or caged fluorescent compounds are used as flow tracers for PHANTOMM based on caged fluorescence. A description of the photo chemistry of the photo activated fluorophores or the caged fluorescent compounds and the experimental technique is as follows.

2.3.3.1.1 Photo activated fluorophores (PAF) for caged fluorescence based PHANTOMM

Photo activated fluorophores are molecules that are formed by chemically coupling a photo-removable protecting group, also known as a caging group, to a fluorescent molecule (Pelliccioli & Wirz, 2002). This caging group cleaves upon irradiation, typically to an ultraviolet light, revealing the fluorescing molecule with emission and absorption spectrum in the visible range (Pelliccioli & Wirz, 2002). These caged compounds were originally designed for time resolved studies of cellular functions (McCray & Trentham, 1989; Lempert & Harris, 2000). Lempert et al., (1995) demonstrated the potential of these caged compounds for flow velocity measurements. A variety of photo activated fluorophores have been engineered. These include caged fluorescein, caged rhodamine, caged rosrufin and caged 8-hydroxypyrene-1, 3, 6-trisulfonic acid (caged HPTS) (Pelliccioli & Wirz, 2002). Both caged fluorescein and caged rosrufin are readily subject to photobleaching post activation whereas rhodamines have a resistance for photobleaching (Pelliccioli & Wirz, 2002; Mitchison, Sawin, Theriot, Gee, & Mallavarapu, 1998). The lowest molecular weight caged rosrufin is the easiest to synthesize (Mitchison, Sawin, Theriot, Gee, & Mallavarapu, 1998).

Typically variants of the o-nitrobenzyl group are employed to quench the fluorescence of caged compounds (Mitchison, Sawin, Theriot, Gee, & Mallavarapu, 1998). Fluorescein exists naturally in two tautomeric forms, a fluorescent carboxylic acid and a non-fluorescent lactone (Pelliccioli & Wirz, 2002).The molecule is constrained in its lactone form by covalently attaching two nitrobenzyl groups at the phenolic oxygens (Pelliccioli & Wirz, 2002). The nitrobenzyl groups undergo an internal redox reaction upon photolysis in the UV, with one of the nitro groups transferring oxygen atoms to the benzylic carbon atoms (Guilkey, Gee, McMurtry, & Klewicki, 1996). The caged molecule dissociates into two photoproducts, the “uncaged” leaving group and a nitrobenzoyl derivative, as a result of the disintegration of the original substituent form the benzylic carbon atoms as shown in the Figure 2.2 (Guilkey, Gee, McMurtry, & Klewicki, 1996).

The nitrobenzyl group has the simplest chemistry and the corresponding compounds are easy to synthesize (Mitchison, Sawin, Theriot, Gee, & Mallavarapu, 1998). However, the caged fluorescent compounds formed upon addition of o-nitrobenzyl groups have poor water solubility (Mitchison, Sawin, Theriot, Gee, & Mallavarapu, 1998). This discrepancy led to the replacement of the o-nitrobenzyl group with the 5-carboxymehoxy group. The 5-carboxymethoxy group has good water solubility. Other advantages of the 5-carboxymethoxy group compared to the nitrobenzyl groups include faster uncaging and relatively less toxic by-products (Mitchison, Sawin, Theriot, Gee, & Mallavarapu, 1998). Table 2-2 below describes the different flow tracers, concentration and solvents used for flow measurements.



Caged Molecule

o-nitrobenzyl derivative and the uncaged leaving group

Figure 2.2: Formation of the "uncaged" leaving group and an o-nitrobenzoyl derivative upon exposure of an o-nitrobenzyl derivative to UV irradiation (Adapted from Guilkey et al., 1996).

2.3.3.1.2 Caged fluorescence based PHANTOMM measurement technique

Photo activated fluorophores or the caged fluorescent dye is premixed homogeneously in the fluid under investigation. The compositions of the mixture in terms of mixture concentration and the selection of solvents are listed in Table 2-2. The mixture is made to flow through the experimental flow geometry. Typically the mixture is colorless. The flow field is exposed to an intense beam of an ultraviolet laser tagging the desired region in the flow. The caging group cleaves off permanently due to exposure to the ultraviolet laser, in turn activating or regenerating the fluorescent molecule. This process may be termed as tagging, writing or photoinjection. A user defined fluorescent pattern can be photochemically produced via the tagging laser beam (Lempert & Harris, 2000). Typically, nsec duration pulses of a powerful light source within the range of 300 – 380 nm are used for photolysis (process of removal of the caging group) (Bernardinelli, Haeberli, & Chatton, 2005). The removal of the caging group is not instantaneous and is defined by a time constant which is on the order of few milliseconds (Koochesfahani M. M., 1999). Each caged molecule can be utilized only once since the uncaging process is irreversible (Guilkey, Gee, McMurtry, & Klewicki, 1996). However, once the dye is uncaged, it can be excited multiple time to get the fluorescence signal. The fastest flow that can be measured using PHANTOMM is dictated by the time duration between the tagging and the interrogation (Koochesfahani M. M., 1999).

The entire fluid region under observation is exposed to a second light source in order to get a fluorescence signal from the uncaged dye molecules. The flow field is flood illuminated because the uncaged molecules are also colorless and their exact location within the region under observation is unknown. This fluorescence signal reveals the velocity information of the tagged region. This process may be termed as reading. Typically a continuous wave laser within the visible spectrum is used for reading. The absorption spectrum of uncaged PAFs is strong, fairly broad (~30 nm) and centered at 480 nm (Lempert & Harris, 2000). They exhibit an intense fluorescence signal centered at 550 nm with a high quantum yield (~0.90) and a fast radiative lifetime which results in low concentration requirements (Lempert, Ronney, Magee, Gee, & Haugland, 1995; Lempert & Harris, 2000). This fluorescence signal is then captured onto an imaging device. The captured images are processed using appropriate image correlation algorithms to reveal the desired flow information. This step may be termed as interrogation.

As compared to LIPA the use of caged compounds offer a high quantum yield fluorescence signal which is favorable for imaging applications yielding simultaneous high contrast and high intensity images (Lempert, Ronney, Magee, Gee, & Haugland, 1995). The fluorescence signal has a fast radiative lifetime which prevents quenching due to water or oxygen in air. Selective photoactivation of the molecules allows the interrogation of the desired region in the flow geometry under observation. There are several disadvantages associated with utilizing caged compounds as flow tracers. One of the major disadvantages is the speed limit placed on the flow that can be measured using MTV based on PHANTOMM. This speed limit is dictated by the finite kinetic rate of the permanent photochemical removal of the caging group. The minimum reported time delay for caged dye PAF experiments is 200 μsec (Lempert, Ronney, Magee, Gee, & Haugland, 1995). The caged compound can only be utilized once since the uncaging is permanent. After each experiment the fluid should be mixed in order to homogeneously mix the uncaged molecules. However after several experiments utilizing the same caged dye solution, a time will come when the contrast will be lost because most of the molecules will be uncaged. In that case the mixture will not be usable and will require replacement. The caged compounds are also subject to significant uncaging upon exposure to normal room lighting if proper precautions are not taken (Lempert, Ronney, Magee, Gee, & Haugland, 1995). All these disadvantages are compounded by the high cost of the caged dyes (Lempert & Harris, 2000). A major constraint and a high cost requirement of PHANTOMM based on caged fluorescence is the requirement of ultraviolet permissible optical infrastructure.

Table 2-2: A review of the various flow tracers and solvents used for caged fluorescence based PHANTOMM.

Researcher	Dye	Dye concentration in Solvent	Flow visualization or velocimetry
Beushausen et al., (2009)	Caged carboxy – Q- Rhodamine	500 mg/litre mixed in demineralized water	Pressure driven flow in a square and rectangular cross section millimeter dimension channel.
Garbe et al., (2008)			
Roetmann et al., (2008)			
Sinton D. , (2004)	5-Carboxymethoxy-2-nitrobenzyl (CMNB) caged fluorescein	1.5 mM mixed in 25 mM sodium (bi-) carbonate buffer (*pH* = 9.0)	Pressure driven and Electroosmotic flow in a circular cross section capillaries.
Shelby & Chiu, (2003)	5-Carboxymethoxy-2-nitrobenzyl (CMNB) caged fluorescein	0.3 mg/litre and 0.1 mg/litre mixed in 100 mM sodium borate (*pH* = 9.3)	Pressure driven flow in a simple rectangular channel and a diamond shaped chamber attached to a straight channel.
Sinton & Li, (2003)	5-Carboxymethoxy-2-nitrobenzyl (CMNB) caged fluorescein	1.5 mM in TAE buffer (*pH* = 9.0)	Electroosmotic flow in square and circular cross section microchannels.
Sinton et al, (2002)	5-Carboxymethoxy-2-nitrobenzyl (CMNB) caged fluorescein	1.5 mM in Tris-acetate (*pH* = 8.0) and Tris-borate (*pH* = 8.3) with EDTA	Electroosmotic flow in circular cross section microchannels.

Lempert & Harris, (2000)	DMNB caged fluorescein	1.6 mg/litre in water	Flow below a rotating disc in a quartz cylinder.
Paul et al., (1998)	4,5-Dimethoxy-2-Nitrobenzyl (DMNB) caged fluorescein dextran	5 mg mixed in 3 ml 0.1 M sodium carbonate buffer, (*pH* = 9.0)	Pressure driven and electroosmotic flow in circular cross section microchannels.
	((5-Carboxymethoxy-2-nitrobenzyl)oxy) carbonyl (CMNCBZ) – caged 5-(-and -6)-carboxy- Q- Rhodamine	1 mg mixed in 0.5 ml 70% acetonitrile and 30% 5mM Tris buffer (*pH*= 8.0)	
Guilkey et al., 1996)	5-Carboxymethoxy-2-nitrobenzyl (CMNB) caged fluorescein	10^{-6} M in aqueous solution of sodium hydroxide (*pH* = 10)	Turbulent passive scalar mixing in a channel consisting of eight circular quartz pipes.
Lempert et al., (1995)	Dextran carboxy fluorescein PAF	5 mg/ litre in water	Pressure driven poiseulle flow in a circular cross section quartz tubing and the test section of a water channel facility.
	8 hydroxypyrene - 1,3,6 trisulfonic acid	2 mg/litre in water	

2.3.3.2 PHANTOMM based on photobleaching

Photobleaching may be defined as a process in which a fluorophore is chemically altered in a way such that it no longer participates in the absorption/emission process, due to photodecomposition, collisional quenching etc., as a result of exposure to a high intensity light source (Wang & Fiedler, 2000; Mosier, Molho, & Santiago, 2002; Sinton D. , 2004). The ability of the fluorescent organic dye molecules to absorb photons and emit a fluorescence signal is reduced over time due to extended exposures to a luminous source which reduces the effective active dye molecule concentration in turn reducing the fluorescence signal (McKeon, et al., 2007). This photodecomposition of the fluorescent dye in aqueous solutions is a permanent and irreversible process (Mosier, Molho, & Santiago, 2002; Sinton D. , 2004). However, depending on the solute, solvent and analyte interactions reversible photobleaching is also possible (Mosier, Molho, & Santiago, 2002). The effect of photobleaching on laser induced fluorescence experiments has been reported by Wang & Fiedler, (2000). However, photobleaching may be used as a flow marking technique. A flow measurement technique based on temporal tracking of a photobleached region in a fluorescein mixed fluid is the basis of PHANTOMM based on photobleaching. The detail of the flow tracers and the measurement technique is described below.

2.3.3.2.1 Flow tracers for photobleaching PHANTOMM

The flow tracers utilized commonly for photobleaching based flow velocity measurements is Uranin, commonly known as fluorescein sodium salt or simply fluorescein (Rička, 1987; Mosier, Molho, & Santiago, 2002; Wang G. R., 2005; Hosokawa, Fukunaga, & Tomiyama, 2009). Other commercially available photobleachable tracers include erythrosine and eosine (Rička, 1987). The fluorescent dye required for photobleaching flow velocity measurements are required to be photochemically unstable with a strong luminous absorption. This requirement makes it difficult to find flow tracers for photobleaching flow measurements because most of the fluorescent dyes are designed to have photochemical stability (Rička, 1987). The excitation source photon flux and the properties of the fluorescent dye, such as the absorption cross section and the photobleaching quantum efficiency, are in direct proportion to the rate of effective photobleaching (McKeon, et al., 2007). Hence for a low intensity excitation source longer exposure times are required, whereas for high intensity excitation sources a shorter exposure is sufficient to acquire effective photobleaching (Sinton D. , 2004).

The fluorescence intensity is a function of dye concentration. However, the fluorescence intensity depends on flow velocity as well as dye concentration for sufficiently low flow velocities at the microscale (Wang G. R., 2005; Kuang & Wang, 2010). A simplified model by Wang (2005) demonstrates the relationship between the flow velocity (v) and the intensity of fluorescence (I_f) in equation 2.1.

$$I_f = I_{f0} \times \exp\left(-\frac{d_f}{v\tau}\right) \tag{2.1}$$

Where I_f is the intensity of fluorescence, I_{f0} is the fluorescence intensity at time $t = 0$, d_f is the incident beam width, v is the fluid velocity and τ is the photobleaching time constant.

It can be observed from equation 2.1 that the fluorescence intensity I_f is directly related to the flow velocity (v). The intensity of fluorescence increases with increasing velocity. This principle can also be used to measure fluid velocity if the fluorescence intensity is known (Wang G. R., 2005). At higher velocities the sensitivity is relatively lower because of the exponential nature of the relationship.

2.3.3.2.2 Photobleaching PHANTOMM measurement Technique

PHANTOMM based on photobleaching may be considered as a reverse approach to PHANTOMM based on caged fluorescence. Instead of the generation of luminescent tracer from a dark background, a dark signal is generated on a fluorescent background. This may be considered as a ‘negative’ image in contrast to the caged fluorescence based approach (McKeon, et al., 2007). Typically two laser sources are required for PHANTOMM based on photobleaching. A relatively low intensity laser is required to generate the fluorescence signal whereas another high intensity laser produces the photobleached region which is eventually tracked temporally to reveal the flow measurements. Typically the fluorescent dye used for photobleaching measurements are used in dilute concentrations on the order of 0.1 – 1 mM (Rička, 1987; Mosier, Molho, & Santiago, 2002; Wang G. R., 2005). The high intense laser beam employed for bleaching the fluorescence is on the order of 400 -500 mW, typically from a 488 nm Argon Ion laser (Sinton D. , 2004; Hosokawa, Fukunaga, & Tomiyama, 2009). Wang (2005) has reported a stronger photobleaching signal with a high sensitivity using an ultraviolet laser ranging between 337.5 – 356.40 nm. For the fluorescence signal various different sources have been used. Mosier et al. (2002) utilized a broad spectrum high pressure mercury arc lamp whereas Hosokawa et al. (2009) employed a low intensity laser light sheet from a 514 nm laser to generate the fluorescence signal.

Flow measurements based on photobleaching have several advantages over some other velocimetry techniques. Only lasers within the visible spectrum are used for flow measurements based on photobleaching, which allows the use of both polymer and glass flow geometries to be visualized (Mosier, Molho, & Santiago, 2002). This is a major advantage of PHANTOMM based on photobleaching in terms of wider applicability and cost reduction as compared to its caged fluorescence based counterpart (Mosier, Molho, & Santiago, 2002). However, the SNR for photobleaching fluorescence measurements is as much as one order of magnitude lower as compared to caged fluorescence based measurements as a result of out of plane velocity gradients (Mosier, Molho, & Santiago, 2002; Sinton D. , 2004). The fact that measuring emitted light against a dark background is favorable and accurate compared to measuring the change of absorbance limits the overall dynamic range of the photobleaching approach.

2.3.4 MTV by direct phosphorescence

MTV by direct Phosphorescence is a non intrusive whole field optical flow velocimetry technique which relies on tracking of molecules that can be transformed into long duration flow tracers upon exposure to photons of appropriate wavelength (Koochesfahani, Cohn, & MacKinnon, 2000). This mechanism of MTV is different from the other mechanisms in that it utilizes phosphorescence instead of fluorescence for flow measurements and require only one laser source. Aside from velocity measurements MTV by direct phosphorescence has been used extensively for thermometry applications. The phosphorescent flow tracers used by this mechanism of MTV are discussed in detail below.

2.3.4.1 Flow tracers for MTV by direct phosphorescence

A chemical compound with a long luminescence lifetime is suitable for MTV based on direct phosphorescence. This long lifetime is actually the excited state of the molecular complex (Koochesfahani M. , Cohn, Gendrich, & Nocera, 1996). As described earlier, the long lifetime of the phosphorescence signal makes it relatively more prone to oxygen and water vapor quenching. Therefore, different molecules have been tested for MTV based on direct phosphorescence to get desired results.

The primary candidate that was used for MTV measurements based on this mechanism was 2,3-butanedione (CH_3-$(CO)_2$-CH_3) commonly known as Biacetyl. It is a non-toxic (used as a food additive), highly flammable (flashpoint 70 ºF) molecule with a sweet odor and a high vapor pressure (5.3 kPa at room temperature) (Hiller, Booman, Hassa, & Hanson, 1984). Formerly it was used for quantitative density and concentration measurement experiments using laser induced fluorescence diagnostics. The potential for the application of biacetyl for velocity based measurements was first demonstrated by Epstein (Hiller, Booman, Hassa, & Hanson, 1984). The absorption and emission spectrum of biacetyl can be observed in Figure 1 of Hiller et al. (1984). It has a broad absorption spectrum peaking at around 270 nm and 420 nm permitting the use of commercially available lasers. It can be observed from the figure (Figure 1 in Hiller, Booman, Hassa, & Hanson, (1984)) that the excitation of biacetyl leads both to fluorescence and phosphorescence. The fluorescent lifetime is too short (of the order of 10 nsec) to be applied for velocity measurement in most applications. However, it proposes a long phosphorescent lifetime on the order of 10^{-2} sec with a high quantum yield (Hiller, Booman, Hassa, & Hanson, 1984). Typical reported phosphorescent lifetimes are of the order of 1.5 msec (Stier & Koochesfahani, 1999). Biacetyl has been used for flow measurements by Hiller et al. (1984), Liu et al., (1988) and Thurlow & Klewicki, (2000). However, annihilation reaction occurring between two electronically excited molecules in the triplet state, collisional de-excitation and oxygen quenching are the factors leading to a shorter phosphorescent lifetime of biacetyl (Hiller, Booman, Hassa, & Hanson, 1984). Quenching due to oxygen requires the experiments to be performed in an oxygen-free environment; hence nitrogen is used as a flowing medium (Stier & Koochesfahani, 1999).

Lempert et al., (2003) performed phosphorescence based experiments for high-speed microjet flows using acetone because it has similar photophysics to biacetyl molecule in various aspects. However, the excitation spectrum of acetone ranges from 230 nm to 340 nm. CH radicals are formed upon photodissociation of acetone readily under certain conditions which can be used advantageously.

The limitation of the previously described phosphorescent flow tracers led to the development of water soluble phosphorescent molecular complex with longer phosphorescent lifetime. 1-BrNp.Gβ-CD.ROH or 1-BrNp.Mβ-CD.ROH is the ternary complex engineered for phosphorescence based measurements exempt of the limitations posed by biacetyl or acetone. These molecules are also termed as superamolecules. 1-Bromonaphthalene (1-BrNp) is a lumophore with a 9 nsec bright fluorescence emission (McKeon, et al., 2007). A cup shaped molecule constructed from sugars connected in a head-to-tail arrangement is a cyclodextrin (CD) (McKeon, et al., 2007). Their size is dictated by the number of sugars in the structure. The phosphorescent lifetime of a lumophore (1-BrNp) can be increased by mixing an aqueous solution of a cyclodextrin (CD) containing the lumophore with a mixture of certain alcohols (collectively represented as ROH). The alcohol acts as a lid for the CD cup containing the lumophore which prevents quenching due to water or oxygen as successfully shown by (Ponce, Wong, Way, & Nocera, (1993) (Gendrich, Koochesfahani, & Nocera, 1997). This can be better understood from the top portion of Figure 5.156 in McKeon, et al., (2007) which shows the lumophore in a cyclodextrin cup with and without the addition of the alcohol.

The cyclodextrins used for phosphorescence based measurements are glucosyl-β-cyclodextrin (Gβ-CD) and maltosyl-β-cyclodextrin (Mβ-CD) (Hu & Koochesfahani, 2006). The triplex based on either of the cyclodextrans show similar measured properties and both of them can be used interchangeably (Hu & Koochesfahani, 2006). The type of alcohol however determines the phosphorescence intensity and lifetime of the complex. The dependence of alcohol concentration of three different alcohols (*tert*-butanol (*t*-buOH), cyclohexanol (cycOH) and neopentanol (neopOH)) on the total phosphorescence emission was demonstrated by Gendrich et al. (1997). The selection of appropriate alcohol may lead to significant phosphorescence enhancements (10^4-10^5) (McKeon, et al., 2007). 1-BrNp exhibits a bright green phosphorescence signal with a natural lifetime of 10 msec in the presence of an appropriate alcohol whereas a short lived blue fluorescence signal is observed in the absence of the alcohol as can be observed in the lower portion of Figure 5.156 in McKeon, et al., (2007)

2.3.4.2 MTV by direct phosphorescence measurement technique

Two photon sources, one for tagging and the other for interrogation, are required by all the formerly discussed MTV mechanism. Unlike them, MTV by direct phosphorescence requires only one laser for generating the luminescence signal required for performing desired flow measurements, which makes it the easiest MTV mechanism in terms of implementation. The procedure of this technique is analogous to laser induced fluorescence with the exception of the long lifetime of the flow tracer for MTV by direct phosphorescence. The phosphorescent flow tracers are premixed in the fluid under observation. The fluid flow is exposed to a pulse of an ultraviolet laser which results in the generation of a phosphorescence signal in the visible spectrum, usually in the green region. Typically 20 nsec duration pulses of a 308 nm excimer laser are used for excitation. This phosphorescence signal is captured at two successive intervals within the emission lifetime of the molecule which reveals the flow velocity information. The first image is acquired immediately after the pulse laser fires, in order to capture the undeformed tagged region. A second image is recorded after a short interval which captures the displacement of the tagged region. The time interval depends upon the speed of the flow.

The major difficulty encountered during phosphorescence based measurements is quenching due to the presence of oxygen or water in the experimental environment (Koochesfahani M. M., 1999). For the case of superamolecules, the quenching is overcome by the addition of appropriate alcohols. The alcohol concentrations may be of the order of 100 mMol or higher which often limits the application of this technique for electrokinetic flows. The lower quantum efficiency of the phosphorescent flow tracers (φ_e = 0.035) relative to caged fluorescent dye tracers (φ_e = 0.90) is another disadvantage of this technique (Gendrich, Koochesfahani, & Nocera, 1997). In addition to these disadvantages, MTV by direct phosphorescence offer several advantages. The electromagnetic excitation and emission process of the phosphorescent tracers is reversible, hence the flow tracers are reusable and can be utilized multiple times (Koochesfahani M. M., 1999). The requirement of a single laser compared to two or more laser sources required for other mechanisms highly reduce the cost, which may be approximately 5 to 100 times less relative to caged fluorescein based experiments based on the choice of the laser (Gendrich, Koochesfahani, & Nocera, 1997; Koochesfahani M. M., 1999). This also simplifies the execution of the MTV mechanism. The time when the emission has decayed to 37% (e^{-1}) of the initial signal, may be referred to as the luminescent lifetime of a flow tracer (Gendrich, Koochesfahani, & Nocera, 1997). The usable delay between tagging and interrogation for a phosphorescent complex is significantly longer relative to its luminescent lifetime (Koochesfahani M. M., 1999). Hence, slow speed flows that require a long duration between tagging and interrogation can be studied using MTV by direct phosphorescence.

2.4 Macroscale MTV

All the MTV mechanisms described above were initially developed to perform flow velocity measurements at the macroscale. With the development of technology and the transfer of attention to microscale flows, these mechanisms were modified to perform microscale flow velocity measurements.

2.5 Microscale MTV

Similar to other microscale flow measurement techniques microscale molecular tagging velocimetry has been adapted from its macroscale counterpart. Paul et al. (1998) are considered as the pioneers in application of MTV to microscale flow. Microscale MTV, so far, has been limited to one dimensional flow measurements only.

Typically, the velocity measurement is estimated by calculating the displacement of a tagged line perpendicular to the axial flow direction. The micro laser sheet of the write laser is used to tag lines in the field-of-view. This method of tagging can be referred to as the 'line tagging method'. This leads to the measurement of only one component of velocity, typically, normal to the tagged line. The measured flow velocity is subject to an inherent error if the flow has a non zero perpendicular velocity component (*v*) as inferred from the analysis of Hill & Klewicki, (1996). The flow profile can also be observed along with the one dimensional velocity information using the line tagging method.

The first application of molecular tagging velocimetry at the microscale utilizing caged fluorescent dye was reported by Paul et al. (1998). They developed a method to extract velocity profile data from time series image data and successfully applied it to pressure driven flows. They analyzed the velocity profile data both for pressure driven and electro kinetically driven flow through open fused silica capillaries. Mosier et al. (2002) used photobleaching MTV for visualizing microscale flows and obtaining molecular diffusion and advection information. A detailed analysis of microscale MTV measurements in a 705 μm ID tube, based on the line tagging method utilizing MTV by phosphorescence, was presented by Maynes & Webb, (2002).

2.6 Two dimensional MTV

As discussed in the previous section most of the flow measurements reported using MTV till date provides only one dimensional velocity measurements. However, two dimensional flow velocity measurements have been successfully performed employing utilizing molecular tagging velocimetry both at the macro and micro scale. Spatial gradients in the luminous intensity of the tagged region in at least two directions, preferably orthogonal directions, are compulsory to obtain two components of velocity explicitly (Gendrich, Koochesfahani, & Nocera, 1997). Hence, all the two dimensional MTV velocity measurement approaches are based different tagging methods to generate the desired spatial gradient necessary to obtain the two components of velocity.

For two dimensional flow velocity measurements at the macroscale a grid approach is utilized. A similar approach is employed for microscale two dimensional molecular tagging velocimetry which makes use of a mask instead of a grid. The basic principle underlying both the grid and the mask approach is the same. A novel two dimensional MTV approach for microscale flows utilizing a laser scanning system, manoeuvring the write laser across the entire field-of-view is utilized in this work. The details of these two dimensional MTV approaches is as follows.

2.6.1 Grid approach:

The grid tagging approach is utilized only for two dimensional flow velocity measurements based on MTV for macroscale flow. This approach was first proposed by D’Arco et al., 1982) and was later refined by Falco & Chu, (1987) (Gendrich, Koochesfahani, & Nocera, 1997). A grid of intersecting laser lines formed using a pair of crossing laser beams enables multipoint velocity measurements. The deformation of the grid in two spatial directions provides the necessary spatial gradient to perform two dimensional flow velocity analyses. Depending on the variety of available optics, the laser tagging grid can be generated in a number of different ways.

The first approach to generate a mesh of intersecting laser lines from a single laser beam utilized a beam divider (Chu & Liao, 1992). It is basically an array of small mirrors and in terms of appearance this beam divider resembles an oversized diffraction grating but provides a specular reflection (Chu & Liao, 1992; Gendrich, Koochesfahani, & Nocera, 1997). A sketch of a typical beam divider is shown in Figure 2.3 below. This method of generating the grid has the advantage that no laser light is lost because no beam blockers are used (Chu & Liao, 1992). The facet design controls the line width and the spacing between the lines (Chu & Liao, 1992). Typical laser grids have a line width of 150 μm and a mesh size of 2 mm square (Chu & Liao, 1992; Chu, Wang, & Hsieh, 1993).

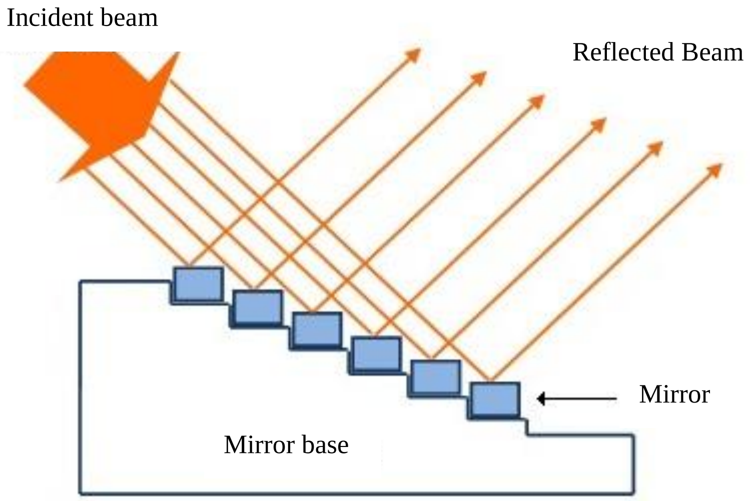


Figure 2.3: A sketch of a typical beam divider (Adapted from Chu & Liao, (1992)).

The grid of intersecting laser lines can also be generated in an alternative way as proposed by Gendrich et al. (1997) (Hu & Koochesfahani, 2006). In this method a single laser beam is split into at least two beams using a 50:50 beam splitter. These beams form an intersecting laser grid after passing through a series of mirrors and beam blockers. The beam blocker is typically a thin metal plate with alternating slots and portions that blocks the incident beams (Gendrich, Koochesfahani, & Nocera, 1997). This generates a grid of intersecting laser lines and the number of lines depends on the number of slots of the beam blocker. A simplified schematic of the setup for generating a grid using a pair of beam blockers is shown in Figure 2.4.

An actual setup will require more optical components to generate the laser grid. The use of beam blockers allows only a fraction of the laser beam energy to be utilized for tagging applications (Gendrich, Koochesfahani, & Nocera, 1997). Gendrich et al. (1997) reported high quality data with beam blockers which transmits less than 30% of the incident beam energy. However, the manufacturing and the design of the beam blockers is a quick and a low cost process, which facilitates the production of the beam blocker with any arbitrary slot pattern (Gendrich, Koochesfahani, & Nocera, 1997). This proves helpful for flow measurements requiring an optimization of the line width and the number of laser lines or where a grid of non uniform beam thickness is required (Gendrich, Koochesfahani, & Nocera, 1997).

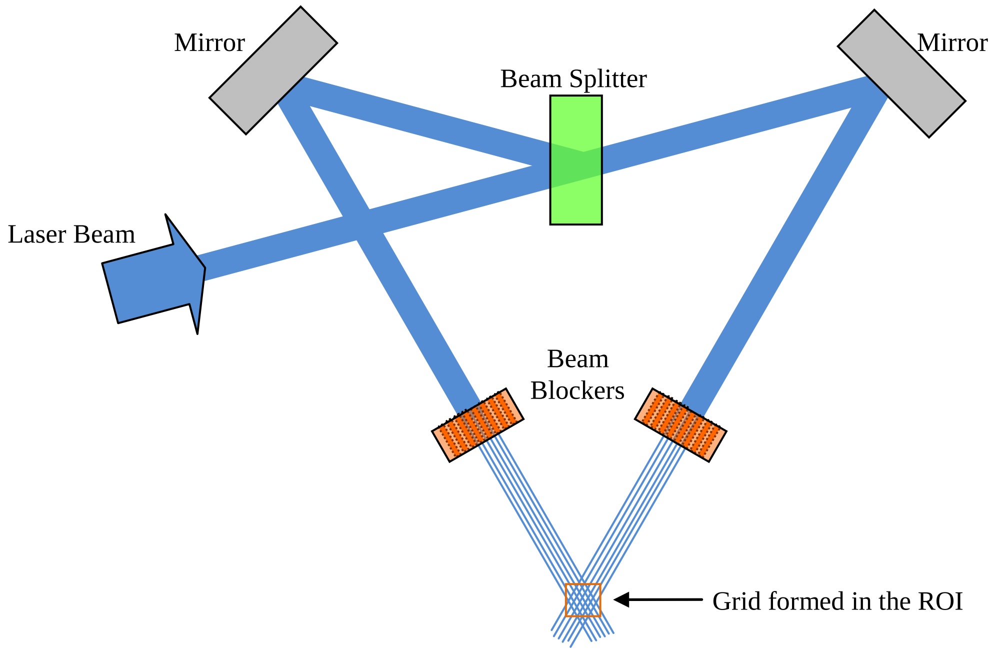


Figure 2.4: A simplified schematic of the grid formation employing the beam blockers (Adapted from Gendrich, Koochesfahani, & Nocera, (1997)).

2.6.2 *Mask approach.*

Molecular tagging velocimetry measurements at the microscale have been mostly one dimensional as discussed before. However, recently two dimensional velocimetry based on MTV have been reported by Garbe et al. (2006). The basic concept utilized by Garbe et al. (2006) is similar to the grid based approach. In this approach instead of generating a grid of intersecting laser lines, the write laser is spatially structured using a mask thereby generating a well defined pattern in the dye mixed fluid. The mask is placed in the path of the write laser which is then demagnified and imaged onto the field-of-view (Roetmann, Schmunk, Garbe, & Beushausen, 2008). The mask is basically a beam blocker with the desired pattern cut out of it. Garbe et al. (2006), Garbe et al. (2008), Roetmann et al. (2008) and Beushausen et al. (2009) reported a complete illumination of a 40 mm x 40 mm structured mask with a well defined transmission pattern. They generated a regular dot pattern to mark the fluid with a dot size of 160 μm and a period of three times the diameter after the imaging process. Any imaginable pattern can be generated using the structured mask (Roetmann, Schmunk, Garbe, & Beushausen, 2008; Beushausen, Roetmann, Schmunk, Wellhausen, Garbe, & Jähne, 2009).

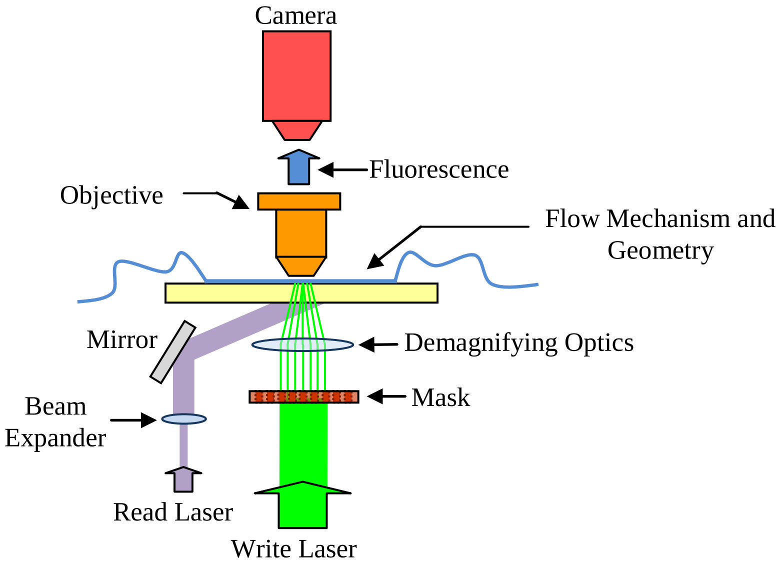


Figure 2.5: A schematic of the typical mask based approach.

Typically an epi-fluorescent configuration is utilized for microscale flow measurements as shown in the Figure 2.5. However, in the cases where a structured mask is utilized to generate a pattern, the epi-fluorescent configuration cannot be used because of the lack of optical access due to the presence of the mask. Instead of an epi-fluorescent configuration, dark field illumination is utilized. The mask is also generated and projected in a back illumination configuration. A typical experimental setup utilizing a structured mask is shown in the figure above. Similar to the beam blocker, the use of a structured mask only transmit some percentage of the write laser beam energy. If more than one pattern needs to be imaged for the same flow geometry, the structured mask need to be designed, manufactured and replaced with the previous mask.

2.7 Conclusion

Molecular tagging velocimetry is molecular tracer based whole field instantaneous flow velocity measurement technique. There are four major mechanisms of MTV that are based on the type of the molecular tracer used for the measurement. The selection of the molecular tracer determines the mechanism of MTV and the hardware configuration required for the measurements. One dimensional and two dimensional flow measurements have been successfully carried out using the different mechanisms of MTV at the macroscale. However, the flow measurements at the microscale are mostly one dimensional. Two dimensional flow measurements at the microscale using MTV have been carried out but there are some limitations to the attempted techniques. There is a strong need for the development of multi-dimensional flow measurement techniques ath the microscale based on molecular tagging velocimetry.

CHAPTER 3

EXPERIMENTAL SETUP DESIGN

3.1 Introduction

A molecular tagging velocimetry system capable of performing two dimensional microscale measurements is designed, assembled and tested. The salient feature of the designed system is a laser scanner which is used to move the write laser beam to any desired location within the area under observation. This method can be called the laser scanner approach because of the use of the laser scanner. The laser scanner approach for two dimensional MTV system is devoid of the limitations of the approaches discussed in the previous chapter. The various steps involved in the design process include design calculations, component selection, system assembly and software development. The concept, design, component selection and the assembly of the scanning two dimensional microscale molecular tagging velocimetry system are discussed in this chapter.

3.2 Laser scanner approach:

The design and operation of the laser scanner based MTV system for microscale flows is the topic of the thesis. As the name indicates, the designed system utilizes a scanner to control the movement of a laser beam. The laser scanner maneuvers the write laser to any point in the field-of-view (FOV) quickly compared to the flow, accurately and with high repeatability. The motion of the write laser in the FOV provides the spatial gradient of fluid motion in at least two directions which is required for two dimensional microscale flow measurements based on MTV. Flow measurements can be performed at any arbitrary point within the field-of-view using the laser based scanning measurement system.

3.2.1 Concept

Figure 3.1 shows a schematic of the proposed laser scanner based microscale molecular tagging velocimetry system. The schematic shown in this figure is basically for a PHANTOMM based MTV system. However, it can be used to explain the implementation of the other mechanisms of MTV. The laser scanner based microscale MTV system is described in detail below.

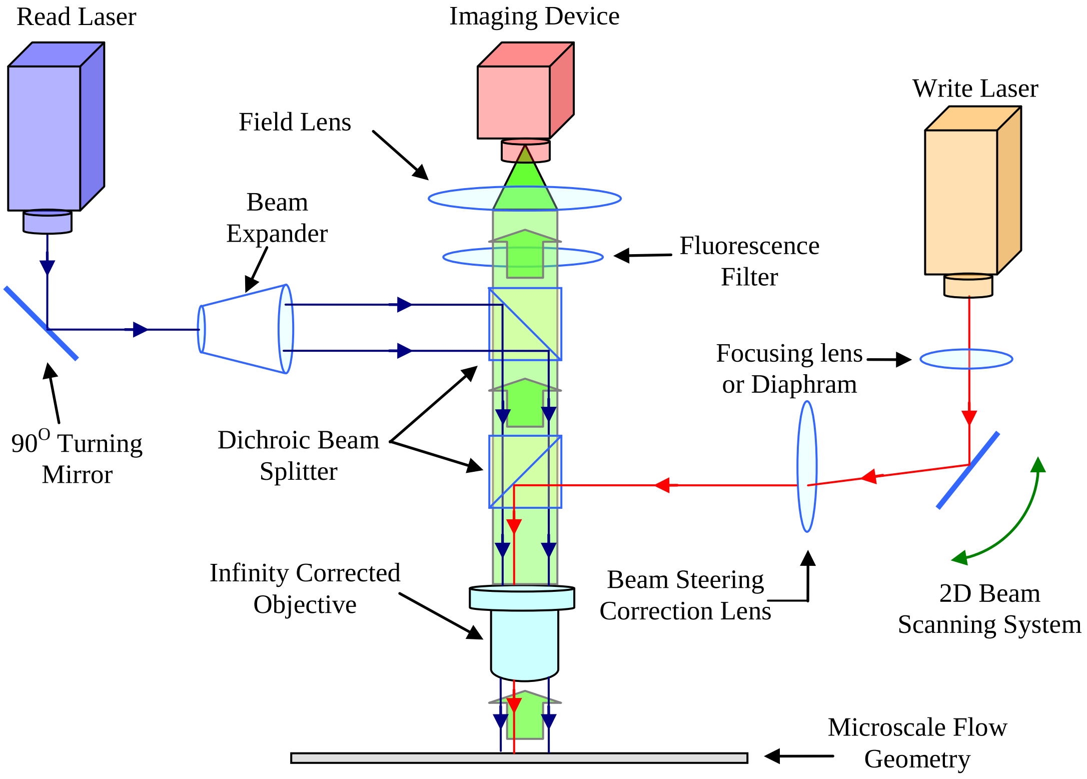


Figure 3.1: Schematic of a laser scanner based microscale two dimensional MTV system.

As discussed in section 2.3.3, MTV based on PHANTOMM utilizes either uncaging of a caged fluorescent dye tracer or photobleaching of fluorescein. The system is capable of operation with both types of tracers. The description of the system presented here is generic keeping both the flow tracers in consideration. It can be observed from Figure 3.1 that three significant light paths are involved in the measurement. The first path is of the write laser, the second path is of the read laser and the third path is the signal that is captured by the camera.

The main components involved in the path of the write laser are the write laser, the 2D beam scanning system, the focusing lens or diaphram, the beam steering correction lens and the dichroic beam splitter. The 2D laser beam scanner is used to maneuver the write laser beam to any desired location in the FOV. A focusing lens is a high focal length focusing lens which is used in combination with the objective to focus the beam sharply onto a point. The focal length of the focusing lens should be equal to the sum of the distance from the lens to the image plane. Alternatively a diaphram can also be used to control the diameter of the beam with the consequence that only some percentage of the write laser beam energy will be transmitted for either uncaging or photobleaching. The beam steering correction lens is a simple concave lens which is included to make the beam parallel to the optical axis so that it can be transmitted to the specified spot at the focal plane of the objective lens. The dichroic mirror used in the path of the write laser is selected so that it reflects only the write laser whereas allows the read laser and the fluorescence signal to pass through. Depending on the type of tracer used, the transmitted write laser will either uncage the caged fluorescent dye or photobleach the fluorescing dye. The region exposed to the write laser will appear as a small dot.

The main components utilized in the path of the read laser are the read laser, a turning mirror, a beam expander and a dichroic mirror. The bulk illumination is achieved by transmitting the read laser through the infinity corrected microscope objective. This type of configuration is typically known as an epifluorescent configuration. The laser beam is expanded using the beam expander in order to fill the back of the infinity corrected objective lens. The objective focuses the expanded beam onto the focal plane bulk illuminating the entire FOV. The 90° turning mirror is used in the path of the read laser to provide the ease of alignment. If a caged fluorescent dye is used, the tagged region will appear as a bright spot against a dark background. A dark spot against a bright fluorescing background indicates the tagged region if fluorescein is used as a flow tracer. The selection of the dichroic mirror used in the path of the read laser allows the read laser to be reflected while allowing the fluorescence signal to pass through.

The main components within the path of the signal captured by the camera are the infinity corrected objective, two dichroic beam splitters, a fluorescence filter, a field lens and an imaging device. An objective lens is used to transmit the signal from the measurement volume to the imaging device. An infinity corrected objective lens is required because it allows the use of other optical components in the path of the transmitted signal. A fluorescence filter is basically a high pass filter which is used to filter out the noise arising from the reflected read laser. The imaging device is used to capture the signal transmitted through the objective lens. A field lens is used to focus the signal transmitted from the infinity corrected objective to form an image at the imaging sensor. At least two images at a suitable time interval apart are required to reveal the flow velocity information from the recorded images.

3.2.2 Implementation

A flow chart of the operation of the laser scanner based microscale molecular tagging velocimetry system is shown in Figure 3.2. The flow chart describes the general steps involved in the implementation of the concept of the laser scanner approach. These steps include scanning, reading, writing, imaging and interrogation.

The molecular tracers are mixed homogeneously in the fluid under observation. The flow of fluid-tracer solution is initiated using the desired mechanism. The software and the hardware setup utilized for the experiments are initialized. The laser scanner is moved to the predefined location in the FOV. The write laser is pulsed during each writing step for a short duration. However, the read laser is continuously switched on during the entire experiments. An image is captured and transferred to memory. The laser scanner is moved to a new location and the process is repeated until a sufficient number of temporally and spatially resolved images have been recorded. The recorded sequence of images is processed using an algorithm which reveals the two-dimensional flow information.

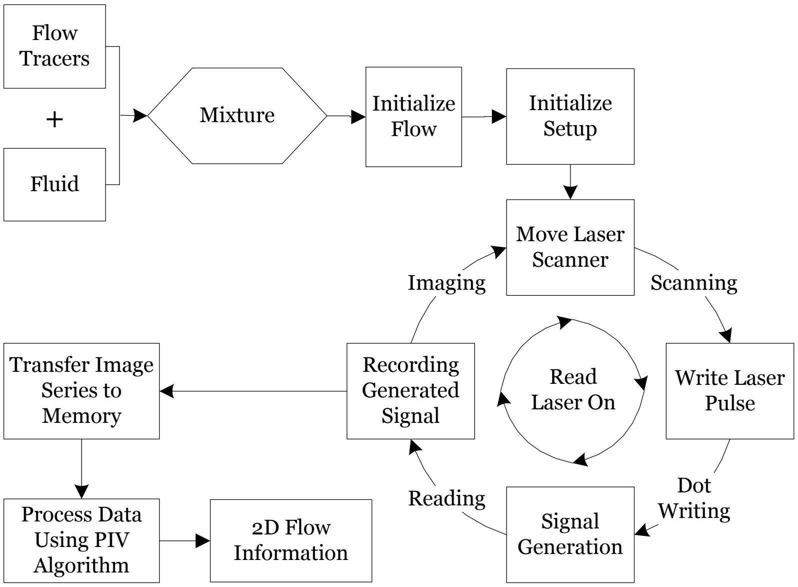


Figure 3.2: A flow chart of the laser scanner based microscale two dimensional MTV system.

3.3 Design Calculations

Design calculations can be considered as the first step in the development of any system. These calculations provide the criteria for the selection of various components used in the construction of a system. Here, design calculations are performed in order to determine the technical specification of the individual components required for the construction of the experimental setup.

Paul et al.,(1998) is considered as a pioneer in the application of molecular tagging velocimetry to microscale flows. In the design of the laser scanning MTV system, the parameters employed by Paul et al. (1998) are taken as a reference. The beam energy density of the laser sets the criteria for the laser energy required to remove the caging group from the caged molecules. The beam energy density and the power calculations are performed below.

Uncaging laser energy at capillary	=	E	=	100	μJ
Laser sheet thickness	=	b	=	20	μm
Laser sheet width	=	w	=	500	μm
Beam area at capillary	=	A	=	$b \times w$	
			=	1 x 10^{-10}	m^2
Energy density of beam	=	d_E	=	$\frac{E}{A}$	
				1 x 10^4	J/m^2
Uncaging laser pulse duration	=	t	=	5	nsec
Power per pulse	=	P	=	$\frac{E}{t}$	
			=	20	kW

The calculations based on the pulse energy and the pulse duration from Paul et al., (1998) leads to a laser power of around 20 kW. A 337 nm laser capable of emitting 170 μJ per pulse is selected as the write laser for the laser scanning MTV system using caged fluorescent dye tracers. The details of the lasers are discussed in section 3.7.1. The beam energy density calculations for the 337 nm uncaging laser are performed below in order to calculate the power of the selected laser.

Wavelength of the uncaging laser	=	λ_{Ucl}	=	337	nm
Energy per pulse	=	E	=	170	μJ
Rectangular beam length	=	b	=	7	mm
Rectangular beam width	=	w	=	3	mm
Area of the uncaging laser beam	=	A	=	2.1 x 10^{-5}	m^2
Energy density of beam	=	d_E	=	8.1	J/m^2
Minimum pulse duration of laser	=	t	=	3.5	nsec
Power per pulse of uncaging laser	=	P	=	48.6	kW

The beam energy density calculated for the selected laser is significantly lower than the reference calculations. The beam energy density calculated from Paul et al (1998) is based on the area calculated from the beam dimensions at the capillary (measurement region). However, the calculations for the selected laser are performed based on the area calculated from the unprocessed beam dimensions. The calculated power of the selected uncaging laser is well above the 20 kW calculated from the reference specifications. The selected laser has almost more than double the laser power used by Paul et al, (1998) to ensure that sufficient energy density of the laser beam is available at the measurement region.

The designed digital imaging system is basically a digital imaging device attached to microscope. The two main components of an infinity corrected microscope that defines the quality of the imaging system are the objective lens and the tube lens. Infinity corrected microscope objectives are commonly used to accommodate various optical components in the path of the light in order to process the transmitted signal. The objective used by Paul et al. (1998) was a 10x microscope objective. A higher magnification objective (15x) capable of operation in the ultraviolet range is selected for the system (details can be found in section 3.7.5). The resolution of the microscope objective defines the smallest resolvable distance between two independent point sources. The resolution of the objective depends on its numerical aperture (NA) and the wavelength of the emitted fluorescence signal.

Numerical aperture of objective = NA_{Obj} = 0.32

Wavelength of fluorescence signal = λ_F = 520 nm

Diffraction limited resolution of objective = R_{Obj} = $\dfrac{0.61\,\lambda_F}{NA_{Obj}}$

~ 1 µm

The resolution of the digital imaging device attached to the microscope defines the spatial resolution of the optical system. The size of the digital sensor, the number of pixels and the pixel size matching the resolution and the magnification of the selected objective lens needs to be calculated for the selection of the digital imaging device. The calculations for the number of required pixels and the pixel size are performed according to reference (CCD resolution calculation) and reference (Matching CCD resolution) respectively.

Magnification of objective lens $= M_{Obj} =$ 15x

Image size projected onto the CCD $= I_{Size} = R_{Obj} \times M_{Obj}$

$=$ 15 µm

Minimum number of pixels required for imaging (Nyquist limit) $=$ 2

Pixel size matching objective resolution $= P_{size} = \dfrac{I_{Size}}{\text{Nyquist limit}}$

$\sim$ 8 µm

Line pairs per mm projected onto CCD $= \dfrac{lp}{mm} = \dfrac{1}{I_{Size} \text{ in mm}}$

$\sim$ 67

Sensor dimension (assumed) $= H \times V$

$= 10 \times 10$ mm^2

Sensor aspect ratio $=$ 1

Equivalent TV lines obtained from sensor $= TVL = \dfrac{lp}{mm} \times \mathrm{V} \times 2$

$=$ 1340

Maximum sensor resolution $= TVL \times$ Sensor Aspect ratio

= 1340

Minimum number of pixels required = $TVL \times$ Maximum Sensor Resolution

= 1.8 million

Since it is not possible to manufacture the sensor of the digital imaging device according to the calculated pixel dimension, a compromise has to be made between the size of the sensor, number of required pixels and the size of each pixel. This will have an effect on the overall image resolution and magnification of the system. A digital imaging device with a sensor size of 10.9×10.9 mm and an individual pixel size of 10.6 μm is selected for the designed system. The details of the imaging device can be found in section 3.7.4.

The thickness of the plane where the subject being imaged is in focus is defined by the depth of field of the optical digital imaging system. The depth of focus depends on a number of factors including the magnification and numerical aperture of the objective, the refractive index of the medium and the smallest resolvable dimension of the imaging sensor.

Size of one pixel = 10.6×10.6 μm^2

Resolution limit of imaging sensor = e = Pixel size × Nyquist limit

~ 21 μm

Index of refraction (air) = n = 1

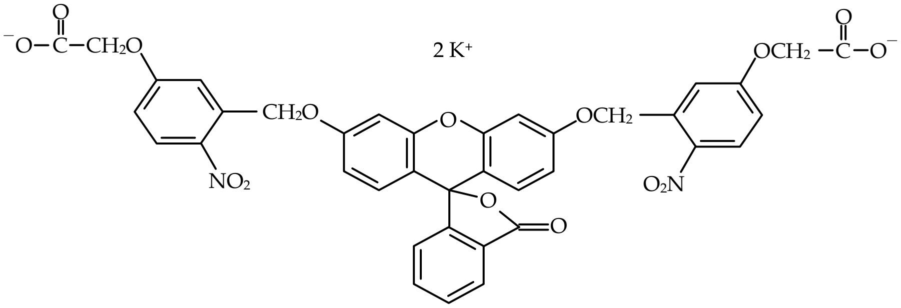
Depth of field = d_{Tot} = $\frac{\lambda_F \,.\, n}{{NA_{Obj}}^2} + \frac{n}{M_{Obj} \,.\, NA_{Obj}} e$

~ 9.5 μm

The design calculations are concluded with the calculation of the depth of field of the designed optical digital imaging system which is ~10 μm. The selection of other necessary components is discussed in the later sections.

3.4 *Caged fluorescent dye*

The caged fluorescent dye is the most important component in the design of a MTV system. The optical and chemical properties of the caged fluorescent dye are directly responsible for defining the capabilities of the technique and the developed system. The theory of caging and uncaging has been described in detail previously in section 2.3.3.1. The caged fluorescent dye used for experimentation is the fluorescein bis-(5-carboxymethoxy-2-nitrobenzyl) ether, dipotassium salt (CMNB caged fluorescein) (Molecular Probes Inc.). The molecular structure of the caged dye is shown in the figure 3.3.



$^{-}O-\overset{O}{\overset{\|}{C}}-CH_2O$
2 K+
$OCH_2-\overset{O}{\overset{\|}{C}}-O^{-}$
CH_2O
O
OCH_2
NO_2
O_2N
O
O

Figure 3.3: Molecular structure of fluorescein bis-(5-carboxymethoxy-2-nitrobenzyl) ether, dipotassium salt (CMNB caged fluorescein) (Adapted from F7103 Product Structure).

It is a highly water soluble caged fluorescent molecule having a molecular weight of 826.81. It can be observed from Figure 3.3 that the caging group is attached on both sides of the fluorescent molecule which renders it non-fluorescent. A ~310 nm laser is required to remove the caging group from the caged fluorescent molecule. Figure 3.4 shows the uncaged fluorescent molecule and the removed caging group after exposure to an ultraviolet laser.

The spectral properties of this dye in a pH 9.0 buffer are shown in figure 3.5. It can be observed that the absorption and emission spectrum of the caged fluorescent dye overlap each other. The absorption spectrum ranges approximately from 420 nm to 520 nm peaking at ~ 488 nm. The emission spectrum ranges from 470 nm to 650 nm reaching a

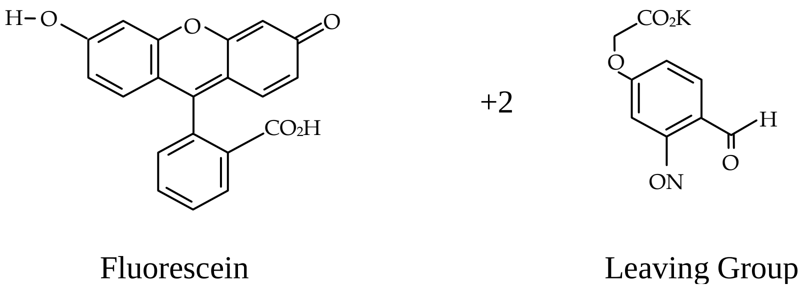


Figure 3.4: Uncaged CMNB caged fluorescein and the removed caging group after exposure to ultraviolet laser (Adapted from Guilkey, Gee, McMurtry, & Klewicki, (1996)).

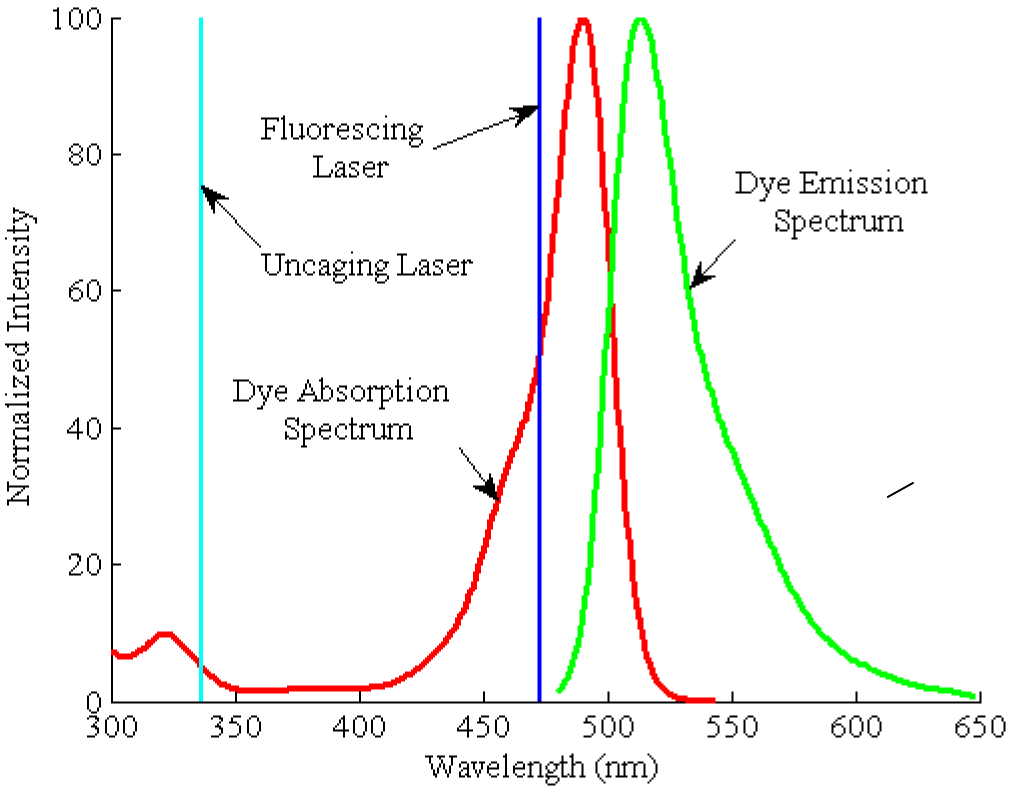


Figure 3.5: Absorption and emission spectrum of uncaged fluorescein bis-(5-carboxymethoxy-2-nitrobenzyl) ether, dipotassium salt (CMNB caged fluorescein) in a pH 9.0 buffer (Adapted from F7103 Spectrum data).

maximum at ~ 520 nm. The selected write laser and the read laser (described later) are also shown on the absorption and emission diagram. These spectral properties of the caged fluorescent dye dictate the selection of the major components used in the assembly of the molecular tagging velocimetry system. The spectral position of the selected uncaging and fluorescing laser with respect to the absorption and emission spectrum of the caged fluorescent dye are also shown in Figure 3.5.

The major properties of the CMNB caged fluorescent dye are described in Table B-1. Fluorescein bis-(5-carboxymethoxy-2-nitrobenzyl) ether, dipotassium salt (CMNB caged fluorescein) has been applied to flow visualization as well as biochemical applications (Johnson, Ross, Gaitan, & Locascio, 2001; Pellliccioli & Wirz, 2002; Ross & Locascio, 2003; Bernardinelli, Haeberli, & Chatton, 2005; Korkotian & Segal, 2007).

3.5 Photobleaching dye

The fluorescent molecular compound used for photobleaching experiments is Uranine (Fisher Scientific, NJ). It is an orange color compound with a molecular weight of 376.27. Uranine has a molecular formula $C_{20}H_{10}O_5.2Na$ and is commonly known as fluorescein sodium salt or simply fluorescein. It is available in powder form and has high solubility in water. The detailed specifications of Uranine are mentioned in Table B-2.

The absorption and emission spectrum of Uranine in a 0.1 N NaOH solution is shown in Figure 3.6. The absorption and emission spectrum of Uranine overlap each other. The absorption spectrum ranges approximately from 420 nm to 530 nm peaking at ~ 490 nm. The emission spectrum ranges from 470 nm to 650 nm reaching a maximum at ~ 510 nm. The spectral position of the photobleaching laser is also indicated on the absorption and emission spectrum of Uranine. The wavelength of the photobleaching laser is within the absorption spectrum of Uranine.

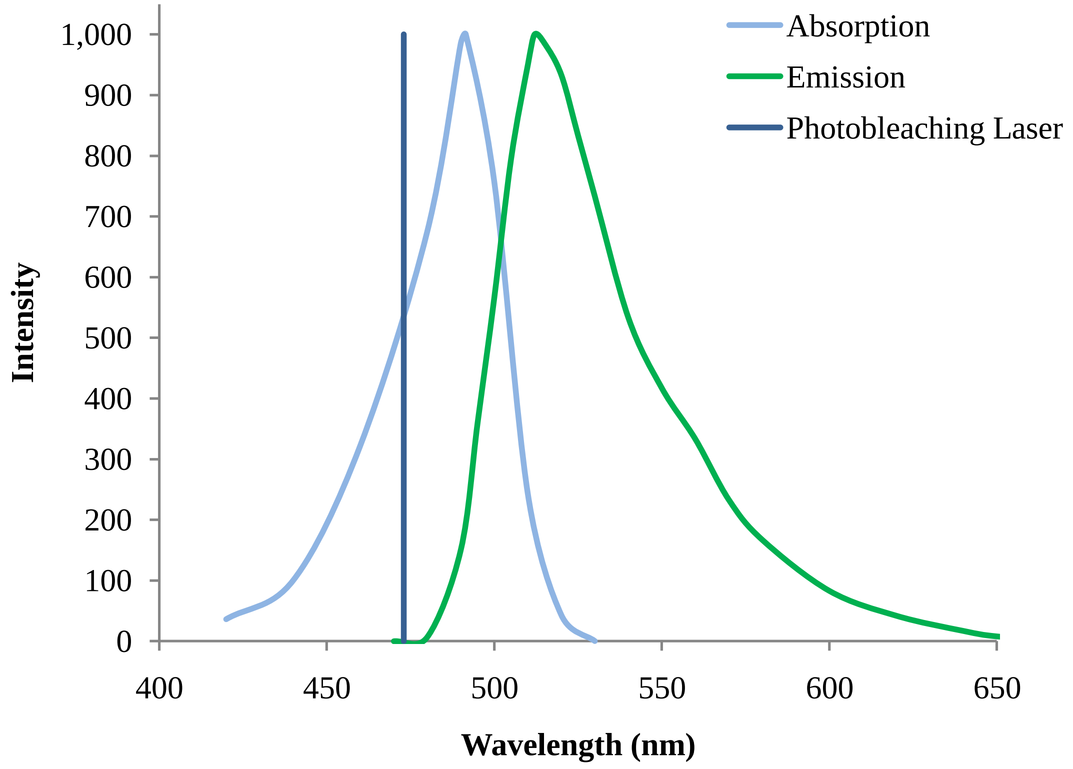


Figure 3.6: Absorption and emission spectrum of Uranine in 0.1 N NaOH solution (Adapted from Heller, Henry, McLaughlin, & Bliss, (1974)).

3.6 *μ-PIV particles*

The particles used for μ-PIV are 1.9 μm fluoro-max dyed green aqueous fluorescent particles (Duke scientific) capable of producing bright, high contrast colors. These particles are made up of polystyrene and are internally dyed with the green fluorescent dye. The fluorescent dye is incorporated within the polymer matrix rather than simply coating the surface of the particles. These particles have an excitation maxima ranging in the blue (468 nm) whereas the emission maxima lies within the green (508) range. An epifluorescence setup or a confocal microscope can be used to detect these particles using standard lasers. These particles are available in the form of aqueous suspensions as 1% solids. The specifications of the fluorescent polystyrene particles are mentioned in Table B-3.

3.7 Components selection

The spectral properties of the caged fluorescent dye determine the wavelength of the two lasers utilized in the construction of the molecular tagging velocimetry system. The selection of the compatible components is crucial for the successful construction and operation of the scanning two dimensional microscale MTV system. A brief description of the selected components used in the assembly of the developed MTV system is given below.

3.7.1 Ultraviolet laser

An ultraviolet laser is required for the experiments performed using the CMNB caged fluorescent dye. A 337 nm, nitrogen laser (NL100, Stanford Research Systems) is used for removing the caging group from the caged fluorescent molecules premixed homogeneously in the fluid. This laser is capable of being triggered both internally and externally up to 20 Hz with pulse energy of 170 µJ. The minimum pulse duration that can be achieved using this laser is around 3.5 nsec. The compact size of the laser makes the laser easier to mount in order to form a single assembly of the measurements system. This laser cannot be operated in continuous mode and is capable of operation only in the pulsed mode. The trigger and the pulse length can be controlled either internally or by means of an external trigger signal. This laser is not used for photobleaching or PIV experiments. The laser will be triggered externally during the caged fluorescence based experiments. The detailed specifications of the NL100 nitrogen laser are described in Table B-4

3.7.2 Visible blue laser

In order to determine the location of the uncaged molecules and imaging the flow, for CMNB caged fluorescent dye based experiments, a second laser is required to read the uncaged molecules. A 473nm diode pumped solid state (DPSS) laser (LRS-0473-PFM-0200-05, Laserglow Technologies) is used to flood illuminate the entire field of view. The uncaged molecules fluoresce upon exposure to this read laser. The laser has a relatively high power output compared to its compact size. The compact size allows the easier mounting of the laser in order to form a single assembly. It is capable of operation in either a continuous mode or a pulsed mode based on an external trigger signal. For CMNB caged fluorescent dye applications it is operated in a continuous mode, in order to flood illuminate the region of interest continuously.

For photobleaching applications the 473 nm laser is used for bleaching the flow because of the high energy density of the laser beam. This laser acts as a write laser for photobleaching application instead of a read laser, as in the case of the caged fluorescent dye experiments. The laser is operated in the pulsed mode for the photobleaching experiments. The instance and the duration of the pulse are controlled by means of an external trigger signal.

This laser is operated in continuous mode providing flood illumination for µPIV measurements. The laser beam is expanded using a 10X beam expander (details provided in section 3.7.6) in order to fill the back of the objective lens, in turn illuminating the entire FOV. The specifications of the 473 nm DPSS laser are mentioned in Table B-5.

3.7.3 Laser scanner

The salient feature of the designed molecular tagging system is the capability of writing any desired pattern in the field of view. This requires the convenient maneuvering of the write laser beam. The write laser is moved to the desired location utilizing a laser scanner. This laser scanner consists of two motorised mirrors. The position of the write laser beam is controlled via the motion of these two mirrors as shown in Figure 3.7. The angular motion of the *X*-position mirror and the *Y*-position mirror is responsible for the maneuvering of the write laser within the FOV. This type of laser scanner is typically known as a galvomirror.

The laser scanner has an optical angular excursion of 40° corresponding to a voltage range of +5 volts to -5 volts. An important feature of the laser scanner is the high response time of 160 µsec for steps of 0.1° with a maximum of 800 µsec for steps of 20°. The high repeatability of 8 µrad facilitates multiple accurate measurements at the same location. The specifications of the laser scanner are described in Table B-6. The high response time of the mirrors enables the writing of the entire pattern without distortion.

The position of the mirrors is controlled through an external voltage signal (+5 V to -5 V) through a data acquisition module (National Instruments NI-9263). One pattern can be written several times without any distortion due to the high repeatability of the laser scanner.

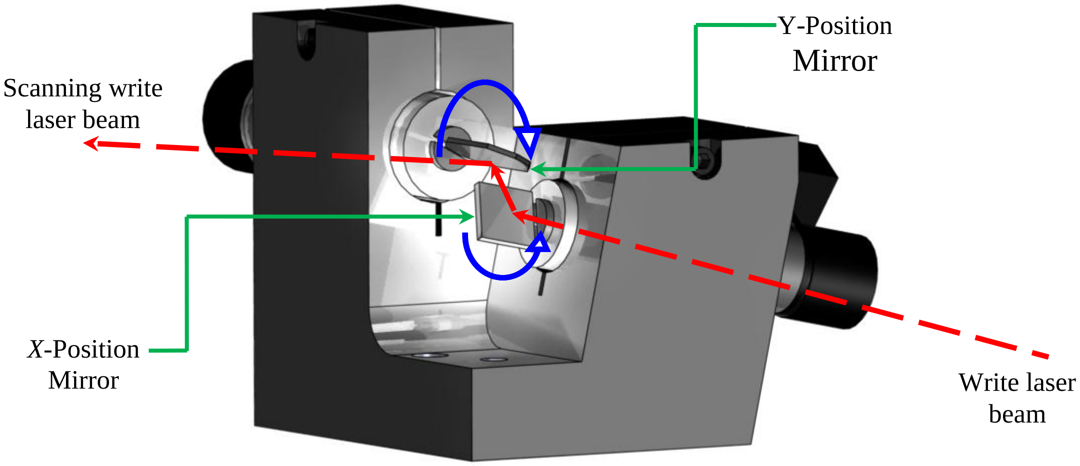


Figure 3.7: Schematic of the operation of the laser scanner for the write laser.

3.7.4 Camera

An imaging device is required to capture either the fluorescence or the photobleached signal. A 1024 x 1024 pixel monochrome CMOS camera (MV-D1024E-160-CL-12, PhotonFocus) with a digital CameraLink interface is used for this purpose. The camera is capable of imaging at about 150 frames per second at full resolution. It is capable of operation both in external and internal trigger modes.

During experimentation the camera is operated in external trigger mode in order to control the instance of image acquisition and the duration of exposure. The image acquisition can be actuated either at the rising edge or the falling edge of the trigger pulse. The pulse width of an external trigger signal determines the duration of the exposure time. The camera supports three grey scale resolutions which can be chosen depending upon the image quality requirement of the processing algorithm. The brightness of the image can be adjusted using the black level offset or the digital gain of the camera. The digital gain can be increased upto 4 times that of a regular image. The camera is capable of applying correction for offset, gain and hot pixel for the captured images instantly before they are recorded. The camera is capable of operation at user selectable region-of-interest depending on the requirements of the experiment.

The frame rate of the camera increases as the ROI decreases. In addition to the capability of supporting different region-of-interest, it can handle 16 multiple ROI simultaneously. Instead of cropping the images after recording, this feature facilitates the capture of the desired region with the amenity of a higher frame rate. These multiple ROI is transformed into a single image before it is transferred to the frame grabber. The detailed specifications of the camera are given in Table B-7.

3.7.5 Microscope objective

A microscope objective serves dual purpose in an epifluorescent configuration. It is used to transmit both the laser beams focusing them within the desired FOV and to transmit the fluorescence or the bleaching signal onto the imaging device. The selected objective is a 15x microspot focusing objective (LMU-15X-NUV, ThorLabs). It is an achromatic objective which means that it is corrected for spherical aberrations in the green (546 nm) and color corrected for two wavelengths, red (656 nm) and blue (486 nm). This microscope objective has a broadband anti-reflecting coating ranging from 325 nm to 500 nm. This broadband AR coating makes it suitable for use in multi-wavelength applications such as UV fluorescence spectroscopy or confocal microscopy. The high damage threshold of the objective lens allows applications involving lasers with a high beam energy density. The minimum resolution obtained using this objective is calculated to be around 1 μm. The specifications of the microscope objective are mentioned in Table B-8.

3.7.6 Laser beam expander

Caged fluorescence based MTV and μPIV measurements require flood illumination. A 10x laser beam expander (X10DR-H (Hercules), Laserglow) is used in the path of the fluorescing laser for both the techniques in order to fill the back end of the objective lens. The maximum input beam width for this beam expander is 2 mm which makes it suitable for use with the 473 nm laser which has a similar beam output diameter (Table B-5). It is easily accommodated in the assembly because of its compact size. The beam expander has an adjustable focus. The beam expander consists of two parts. The distance between the two parts controls the output beam diameter. Hence, in order to utilize this feature the beam expander is disassembled and the parts are used separately with the maximum distance between them. The detailed specifications of the beam expander are mentioned in Table B-9.

3.7.7 Analog output module

The sequential operation of the lasers, the scanning system and the camera is very important for the successful operation of any measurement system. A 4 channel analog output module (NI USB-9263, National Instruments) is used to control the sequential operation of the laser scanner, camera trigger and the laser pulse. It is a 10 V analog output device with an update rate of 100 kS/s. The analog output module connects via a USB interface. The module is programmed (LabWindows CVI) to output a voltage signal which triggers the operation of the different components. The first two channels of the DAQ control the operation of the laser scanner in the *X* and *Y* directions. The third channel controls the instant and duration of the camera trigger. The camera captures the images on the rising edge of the signal and the exposure time depends on the length of the trigger pulse. The fourth channel controls the triggering of the laser. Similar to the camera, the laser is triggered on the rising edge of the signal and the duration of the laser pulse depends on the length of the trigger signal. The details of the operation of the different channels will be explained in the next chapter. The detailed specifications of the analog output module are presented in Table B-10.

3.7.8 LED assembly

In the case of photobleaching experiments the 473 nm laser is used for bleaching the fluorescence instead of illumination. A back illumination configuration is employed for photobleaching experiments. This is achieved by means of a broad band high power LED (M470L2, ThorLabs) assembly. The major parts include the M470L2 LED, two aspheric condenser lenses, a lens tube and an excitation filter.

The nominal wavelength emitted by the LED is 470 nm. The output power of the LED ranges from 830 mW to 1560 mW. The power decay of the LED is reduced to a minimum due to a unique thermal design. The broad spectrum of the M470L2 is shown in Figure 3.8. The detailed specifications of M470L2 LED are mentioned in Table B-11. The total power of the LED is not utilized because a 1200 mA T-cube LED driver (LEDD1B, ThorLabs) is used whereas the LED can accommodate up to 1600 mA.

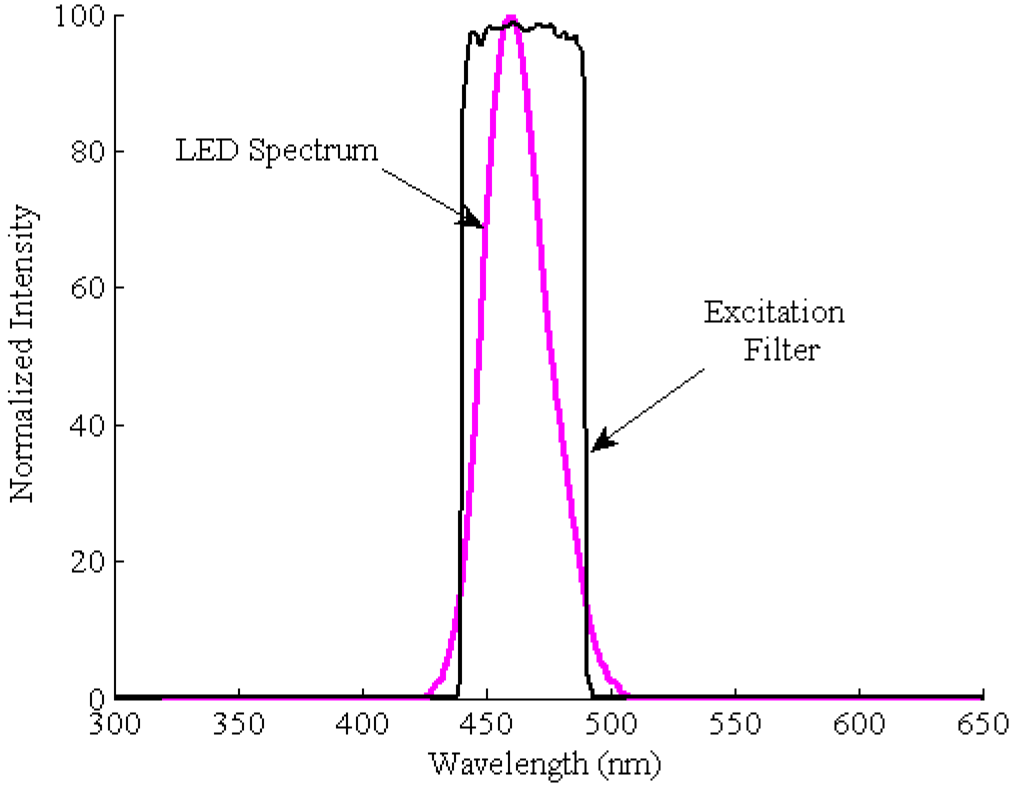


Figure 3.8: Broad spectrum of the M470L2 LED and the effect of ET465/50m emission filter .

The light from the LED is collimated by means of two aspheric condenser lenses (ACL5040-A, ThorLabs). These lenses are suitable for high efficiency illumination applications and facilitate diffraction-limited performance at their design wavelength. The lenses have an antireflection coating ranging from 350 nm to 700 nm. The lenses are mounted in a lens tube. An excitation filter (ET465/50m, Chroma Technology

Corporation) is used in front of the LED before the condenser lenses in order to limit the broad spectrum of the LED. It is an image quality excitation filter with a centre wavelength of 465 nm. Figure 3.8 demonstrates the effect of the excitation filter on the emission spectrum of the LED. The filter restricts the spectrum of the LED between 440 nm and 480 nm in order to utilize the exact same system without changing the optical components. The detailed specifications of the condenser lenses and the excitation filter are mentioned in Table B-12 and Table B-13 respectively.

3.7.9 Dichroic mirrors

A dichroic mirror is a type of mirror which reflects one wavelength (lower) of light and allows another wavelength (higher) to pass through. Dichroic mirrors are essential parts of an epifluorescent configuration. Two dichroic mirrors (DMLP425 and DMLP505, ThorLabs) are utilized in the development of the MTV setup employing the caged fluorescent dye. Only one dichroic mirror (DMLP505) is utilized in the case of photobleaching and µPIV experiments. These dichroic mirrors posses a high transmission and reflectivity with a flat response and are resistant to damage from UV light and chemicals.

Figure 3.9 and Figure 3.10 elaborate the spectral range for the transmissivity and reflectivity of the DMLP425 and DMLP505 dichroic mirrors respectively. The DMLP425 reflects the 337 nm laser and allows the 473 nm laser and the fluorescence signal to pass through. The DMLP505 reflects both the 337 nm and the 473 nm laser and allows the fluorescence signal to pass through. The dichroic mirrors are designed for use at an angle of incidence of 45º and reflect/transmit 50% of the incident light at the design wavelength. However, at wavelengths below the design wavelength they reflect more than 90% of the incident light and at wavelengths above the design wavelength they transmit more than 90% of the incident light. The design wavelength for the DMLP425 dichroic mirror is 425 nm whereas the design wavelength of the DMLP505 dichroic mirror is 505 nm. The detailed specifications of the dichroic mirrors are mentioned in Table B-14 and Table B-15.

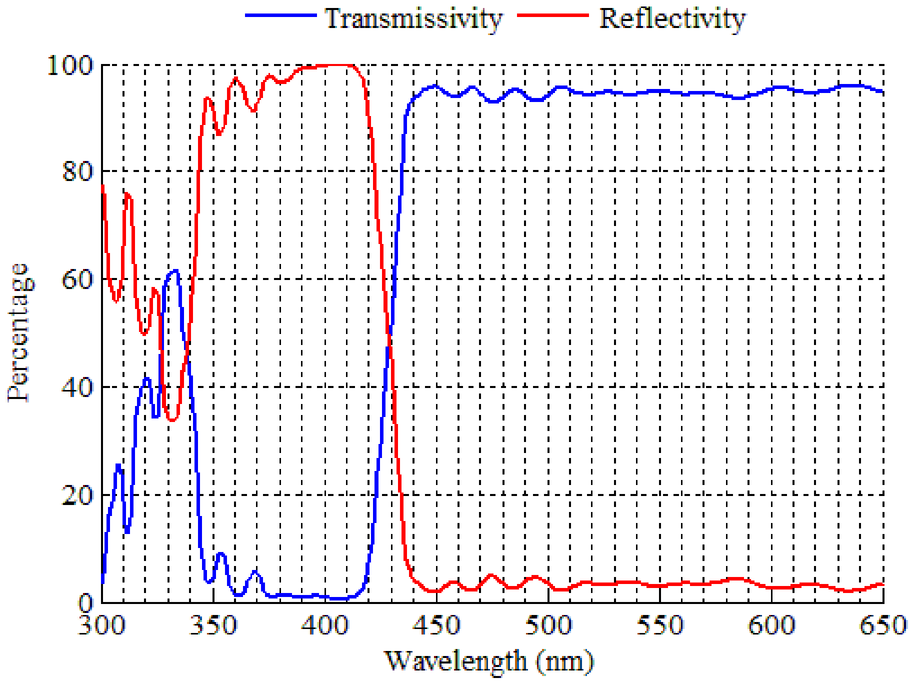


Figure 3.9: Spectral range of transmittivity and Reflectivity of the DMLP425 dichroic mirror (Adapted from Thorlabs DMLP425 plot data).

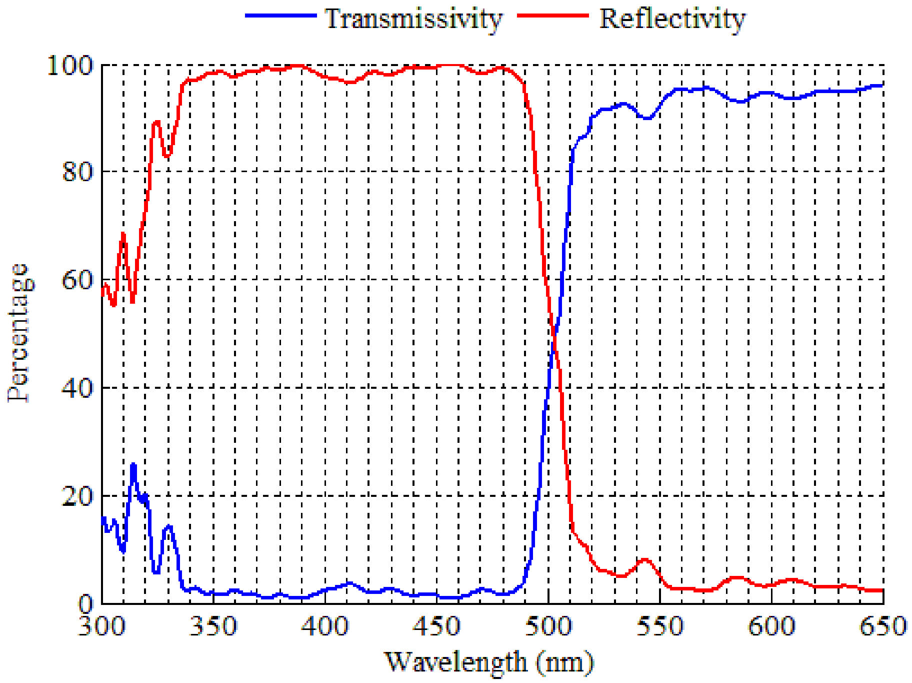


Figure 3.10: Spectral range of transmissivity and reflectivity of the DMLP505 dichroic mirror (Adapted from ThorLabs DMLP505 plot data).

3.7.10 Auxiliary Optics

A long pass filter (E505LP, Chroma Technology Corporation) is used in the imaging path. In the case of photobleaching a back illumination configuration is used which results in a lot of noise coming into the camera along with the fluorescence signal. This filter is used to remove this unwanted noise from the fluorescence signal. In the case of caged fluorescence and μPIV experiments utilizing an epifluorescent configuration, reflections from the surface of the microchannel results in additional noise to the fluorescence signal. This additional noise is removed using the long pass filter. The filter is placed right before the field lens. Figure 3.11 demonstrates the capability of the filter with respect to fluorescence signal and the illuminating wavelengths. The detailed specifications of the long pass filter are mentioned in Table B-16.

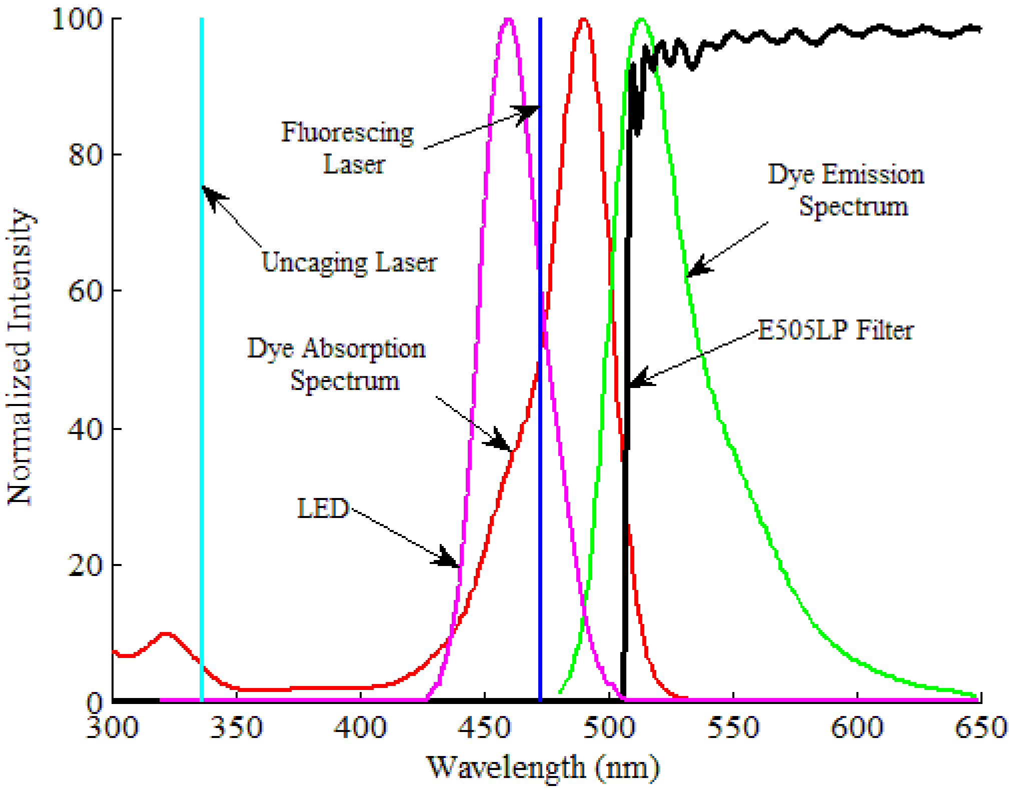


Figure 3.11: The effect of the long pass filter on the incoming signal.

A focusing lens is required to focus the image because an infinity corrected objective is used. A field lens (LB1676-A, ThorLabs) is used to focus the image onto the camera sensor. It is a 100 mm focal length convex lens with an antireflection coating ranging from 350 nm to 700 nm. The detailed specifications of the field lens are mentioned in Table B-17.

The laser scanner changes the orientation of the beam at an angle from the optical axis. In order to make it parallel to the optical axis a beam steering correction lens is used. A 75 mm focal length bi-convex lens (LB4330-UV, ThorLabs) is used for beam steering correction. The detailed specifications of the beam steering correction lens are mentioned in Table B-18.

3.8 Flow Cell

A number of flow cells have been designed and fabricated for use with the two dimensional scanning MTV system. Three major designs of flow cells have been used to conduct experiments. The detailed description of each flow cell design is provided below.

An exploded view of the first flow cell designed for μ-PIV experiments is shown in Figure 3.12. The flow cell consists of four major parts; the channel holder, the sealing o-ring, a quartz window and the top plate. The flow channel is embedded in the channel holder which is made of acrylic based photopolymer plastic (Fullcure 720). The channel holder is built from a solid model design that was made using a rapid prototyping technique (Object, Eden 350) in 16 µm layers. The 1 mm x 1 mm channel is straight for the first 30 mm and then transforms into a serpentine shape for the rest of the length of the cell. Each turn of the serpentine channel decreases by 1 mm which provides sharp two-dimensional flow. A custom made o-ring is used to seal any leakage of the fluid from the sides the cell. A 76.2 mm x 25.4 mm x 1 mm quartz window (Ted Pella Inc., 26011 GE-124 quartz window) is used to seal the channel providing sufficient optical access for flow measurements. The aluminum top plate holds the entire flow cell together with the help of six #4-40 screws. This top plate also helps in mounting the flow cell onto the precision three axis stage for experimentation. Commercial

(Swagelok, 1/16” NPT) fittings on both side of the flow cell allows the flow into and out of the channel.

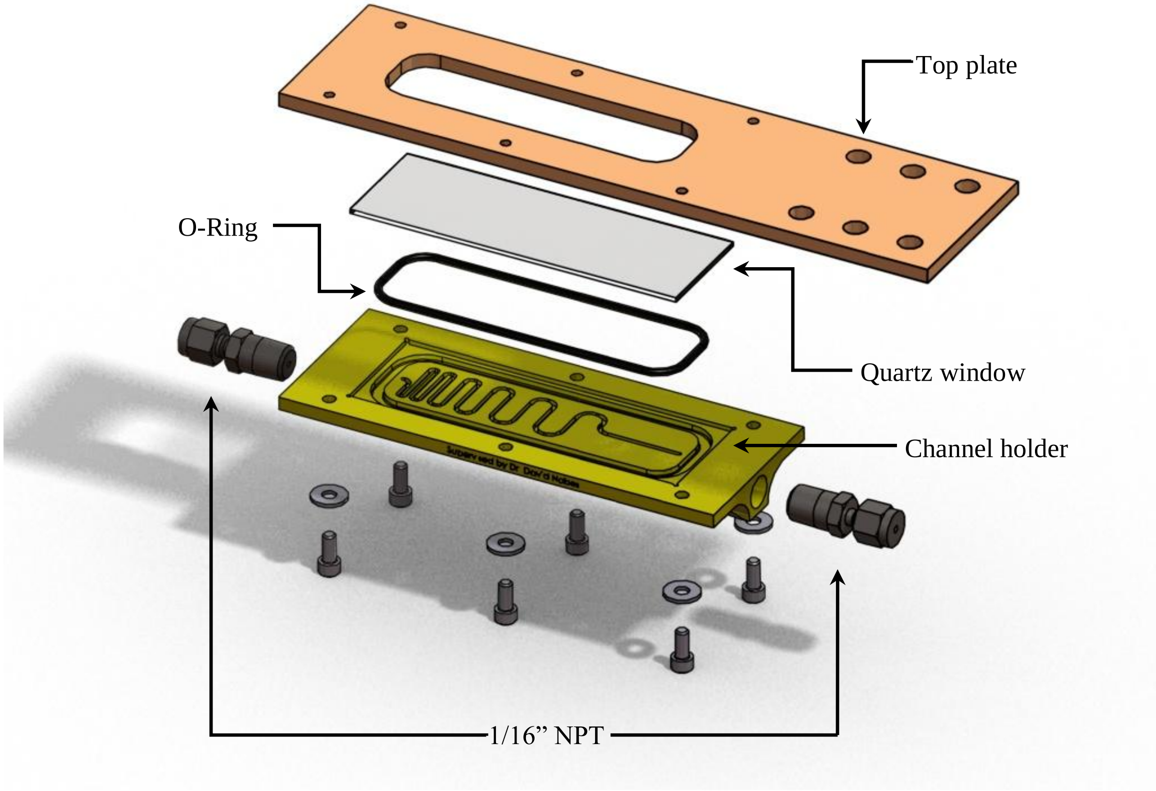


Figure 3.12: An exploded view of the first flow cell design.

An exploded view of the second flow cell design used for scanning MTV experiments is shown in Figure 3.13. The channel geometry is cut out from a 0.3 mm thick hard plastic sheet using a water jet. This channel geometry is sandwiched between two standard glass slides of size 3 in x 1 in x 1 mm (L x W x T). Two holes are drilled in the top glass slides in order to allow the fluid to flow through the channel. The two glass slides and the channel geometry are glued together using transparent silicon. In order to allow the flow of fluid the glass slide assembly is held in between a top cover and a bottom plate. The top cover is made up of an acrylic based polymer plastic (Fulcure 720) using a rapid prototyping technique (Object, Eden 350). However, the bottom plate is made up of aluminum to provide support and stability to the channel. The top cover has a provision for 1/16” NPT fittings. Two standard O-Rings are used between the top cover and the top glass slide to prevent any leakages. The entire assembly is held together using nuts and bolts with washers on each side.

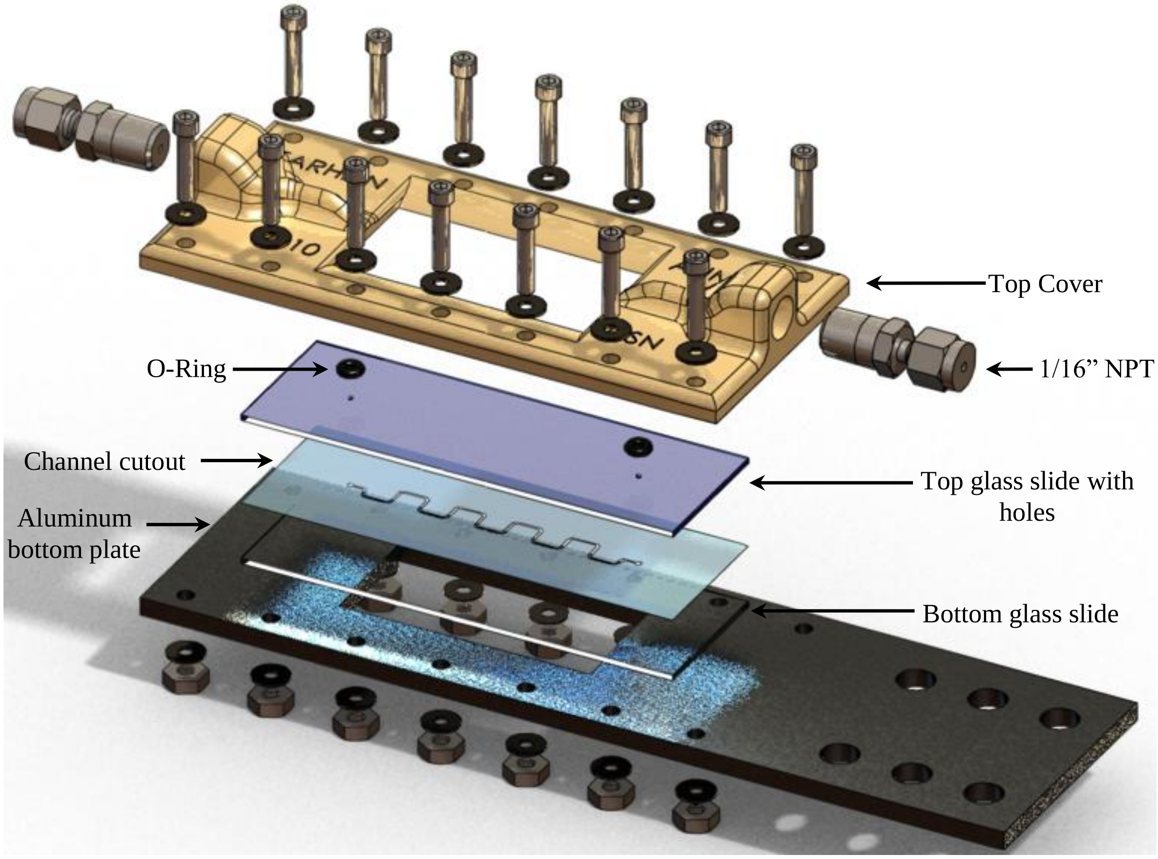


Figure 3.13: An exploded view of the second flow cell design.

3.9 Flow mechanism

The flow through the cells described in section 3.8 is a pressure driven flow which is generated using a syringe and a syringe pump. The syringe pump (PHD 2000, Harvard Apparatus) can accommodate various syringe sizes ranging from 0.5 μl to 140 ml. It employs a microcontroller along with a micro stepping technique to control small step angles to eliminate flow pulsations. It incorporates an internal syringe lookup table containing cross sectional area of selected syringe and automatically calculates the corresponding flow rate and volume accumulation. It has a high reproducibility which makes it favourable to produce the same flow rate for multiple measurements. The syringe pump is capable of operation in both infuse and withdraw direction which facilitates the flow in both directions without changing the connections. The specifications of the syringe pump are mentioned in Table B-19.

3.10 Assembly

A solid model is shown in the Figure 3.14 in order to demonstrate the experimental arrangement and highlight the location and the physical size of the components with reference to the entire assembly. The experimental setup is assembled in a typical epifluorescent configuration for caged fluorescence based measurements.

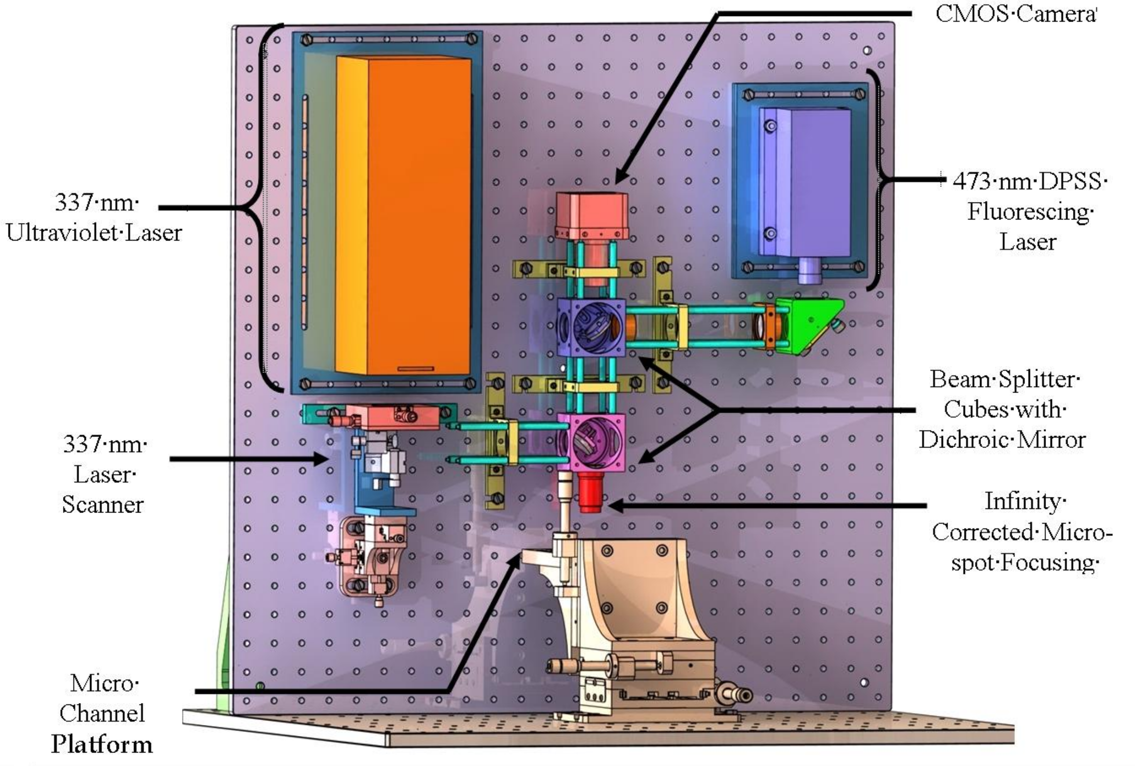

CMOS Camera
337 nm Ultraviolet Laser
473 nm DPSS Fluorescing Laser
Beam Splitter Cubes with Dichroic Mirror
337 nm Laser Scanner
Infinity Corrected Micro-spot Focusing
Micro Channel Platform

Figure 3.14: Solid model of the setup assembly for the caged fluorescence based experiments.

The 337 nm and 473 nm lasers, beam expander, dichroic mirrors, microscope objective and camera are all mounted as a single assembly on an optical breadboard for the stability of the optical system. The laser scanner and the microchannel each is mounted on a precision three axis stage for the ease of alignment. A manual three-dimensional traverse shown below the objective is used to hold and maneuver the micro-channel test piece relative to the measurement.

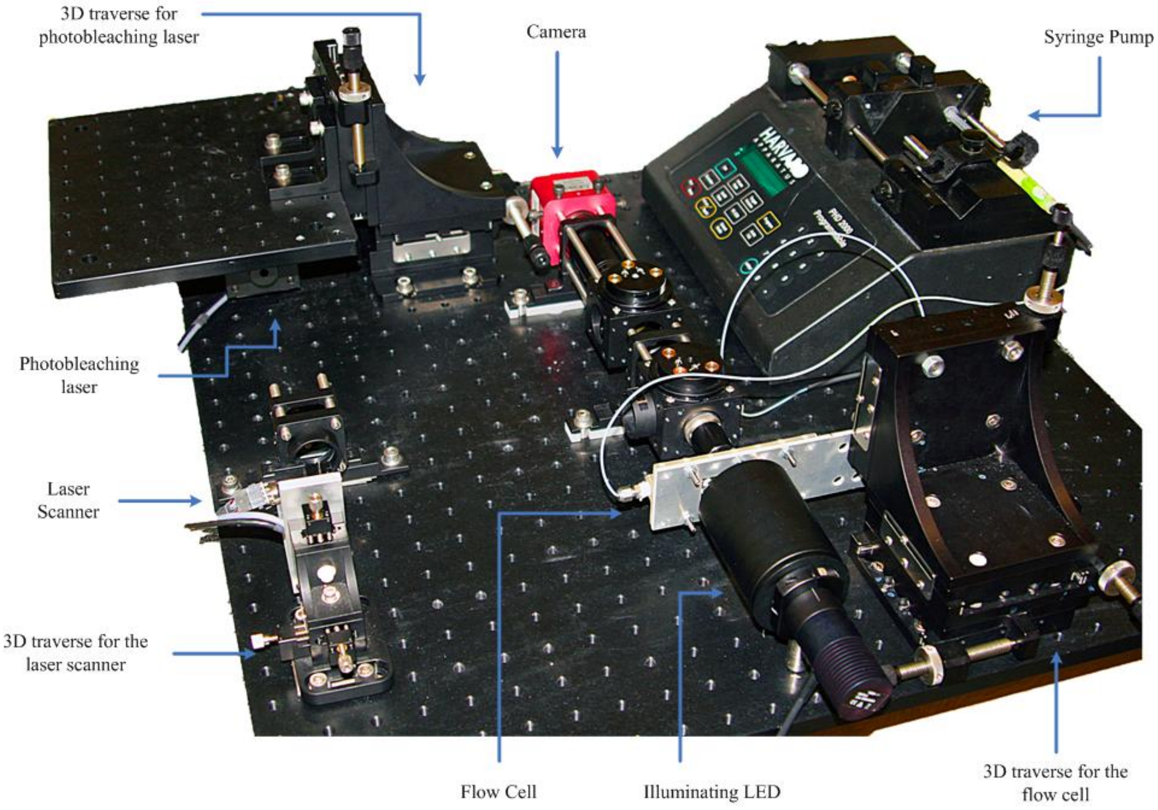

3D traverse for photobleaching laser
Camera
Syringe Pump
Photobleaching laser
Laser Scanner
3D traverse for the laser scanner
Flow Cell
Illuminating LED
3D traverse for the flow cell

Figure 3.15: Experimental setup arrangement for the 2D scanning photobleaching MTV measurements.

The location in the assembly of the two-dimensional scanning photobleaching MTV system is shown in Figure 3.15. The measurement system is assembled in a back illumination configuration. Similar to the caged fluorescence setup the camera, the dichroic mirror and the microscope objective are mounted as a single assembly on an optical breadboard to provide support to the optical system. The write laser, the laser scanner and the flow cell each is mounted on a precision three axis stage to provide easy alignment. The bulk illumination is done by means of the LED mounted in an opposite direction to the camera. The syringe pump used to generate the flow is shown in the top right of Figure 3.15.

3.11 Conclusion

A new two dimensional molecular tagging velocimetry system has been developed. The concept of the two dimensional scanning MTV approach is based on the use of a laser scanner to maneuver the write laser beam. The tagged region can be tracked using algorithms to reveal two dimensional velocity information. The main components required for the construction of the setup include write laser, read laser, microscope objective, an imaging device and necessary optical components. The selected components can be used to construct setup capable of performing MTV measurements based on photobleaching as well as MTV based on caged fluorescence. The selected components are assembled together forming a single assembly.

CHAPTER 4

IMAGE ACQUISITION & PROCESSING

4.1 Introduction

It can be inferred from Figure 3.2 that the successful operation of the scanning molecular tagging velocimetry system requires a number of integrated sequential operations of the write laser, camera and the laser scanner horizontal and vertical position. The laser scanner position should be readily moveable in order to define the coordinates of the regions to be tagged. The positioning of the laser scanner should be accurately followed by the firing of the write laser pulse. The duration of the write laser pulse should be enough to impart sufficient energy for the bleaching or the uncaging process. The initiation of the camera exposure should never overlap with the pulse of the write laser. The exposure time should be enough to capture the image with a temporal resolution necessary to reveal the velocity vector keeping in view the integration time of the camera. The laser scanner should be moved to the new location before the end of this process. A single graphical user interface (GUI) capable of controlling the integrated sequential operation of all the components is required for the accurate and prompt operation of the scanning MTV assembly. This graphical user interface is an integral part of the design of the scanning MTV system.

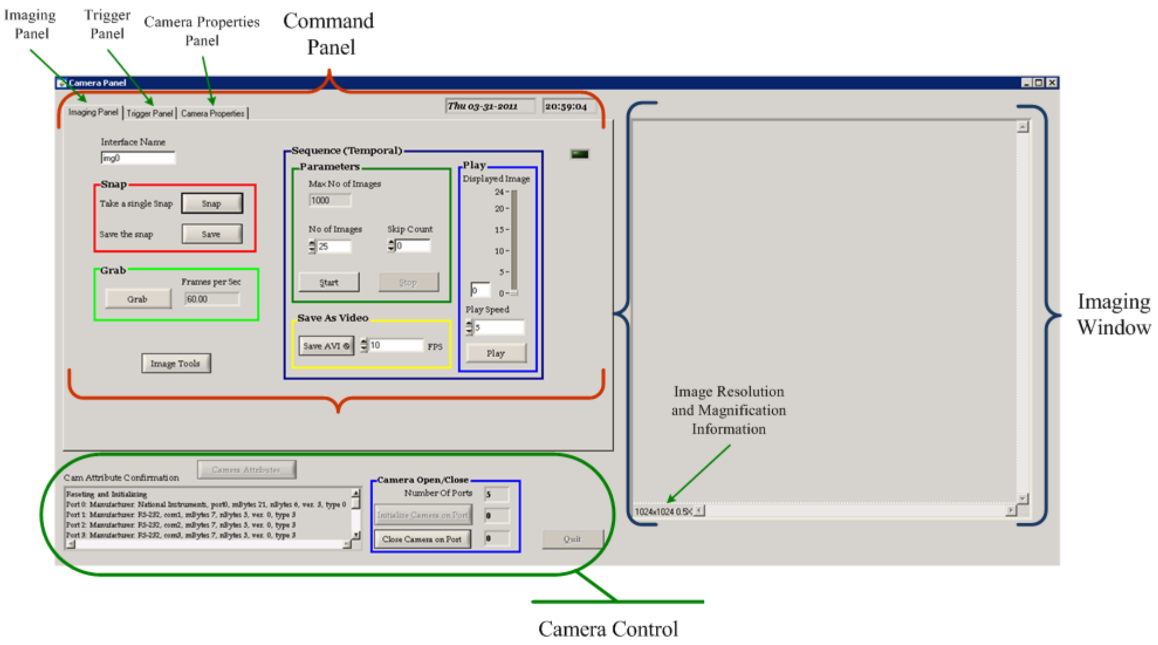
The flow measurements techniques discussed previously depend on spatially and temporally resolved images of the tracer mixed flow field. These images are acquired using the in house developed acquisition system. The acquired images are then processed using a typical set of parameters to reveal the desired flow information. Commercially available software (DaVis 7.4) is used to process the acquired images of the flow field.

The graphical user interface developed for the integrated sequential operation of the hardware components for the scanning MTV system is presented in detail in this chapter. The sequence of operation involved in the acquisition of images and typical parameters used for their processing to reveal required flow information for the photobleaching MTV technique and the µPIV technique are also discussed in the second half of the chapter.

4.2 Graphical User Interface

A graphical user interface (GUI) is developed to control the sequential integrated operation of the different components utilized in the image acquisition process for the scanning MTV system. This graphical user interface is developed using commercially available software (National Instruments, LabWindows CVI 9.0.1). At the back end of this GUI a C language based program is responsible for carrying out the specified operations of the in house developed acquisition software. A detailed description of the operation of the software is necessary in order to understand the image acquisition procedure. A detailed description of the operation of different controls of this GUI is described in the following sections. The C language based program operating behind the controls of the GUI is provided in Appendix D.

The GUI can be divided into three major areas. The first area is classified as the imaging window, the second area is labelled as camera control and the third area is termed as the command panel. The command panel is further divided into three panels namely the imaging panel, the trigger panel and the camera properties panel. The different areas of the graphical user interface are shown in Figure 4.1. Any control which is greyed out or dimmed indicates that it is not available for that current configuration. Any greyed out or dimmed control will be referred to as a deactivated control hereafter.



```
Reseting and Initializing
Port 0: Manufacturer: National Instruments, port0, mBytes 21, nBytes 6, ver. 3, type 0
Port 1: Manufacturer: RS-232, com1, mBytes 7, nBytes 3, ver. 0, type 3
Port 2: Manufacturer: RS-232, com2, mBytes 7, nBytes 3, ver. 0, type 3
Port 3: Manufacturer: RS-232, com3, mBytes 7, nBytes 3, ver. 0, type 3
```

Figure 4.1: Indication of the major areas of the image acquisition control graphical user interface.

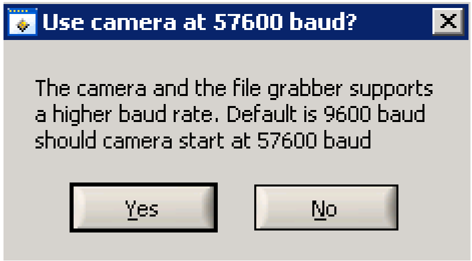
4.2.1 Imaging Window

As the name indicates, the imaging window is the area where the acquired images are displayed. This imaging window is located on the right side of the GUI which is indicated within the blue braces Figure 4.1.

The acquired images are displayed at 0.5x zoom in the imaging window by default. The zoom can be changed using the zoom control in the ‘Image Tools’ described in section 4.2.3.1. It can also be changed from within the program manually. The resolution of the image and the image magnification are displayed in the lower left side of this imaging window. Horizontal and vertical scroll bars appear automatically when the displayed image is larger than the size of the imaging window.

4.2.2 Camera Control

The camera control area controls the opening and closing of the camera on a port defined within the program. The number of available ports is also displayed in the camera control area. Five camera ports, ranging from 0 to 4, are available for the camera currently in use (PhotonFocus, MV-D1024E-160-CL). The first port is for the cameralink interface and the other four are RS-232 interface ports. The camera is connected to the computer via a cameralink interface hence Port 0 is used for the operation of the camera. The ‘initialize camera on port’ button initializes the camera on the defined port (Port 0). The program checks for the highest baud rate available for the camera and the frame grabber before initializing the camera. The current camera and frame grabber supports a baud rate of upto 57,600 baud whereas the default is 9,600 baud. If a higher baud rate is supported a pop up appears prompting the user to select the baud rate. The user can choose between the supported higher and the default baud rate. The baud rate selection popup is shown in Figure 4.2.

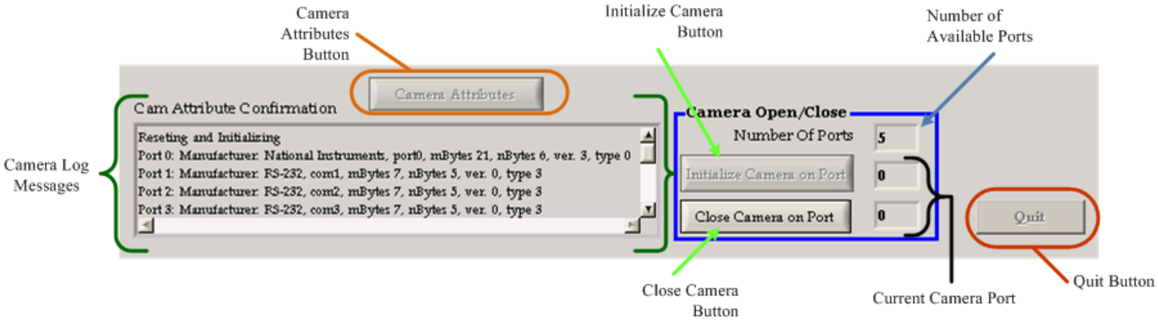


Use camera at 57600 baud?

The camera and the file grabber supports
a higher baud rate. Default is 9600 baud
should camera start at 57600 baud

Figure 4.2: The baud rate selection popup for the camera.

The ‘Camera Attributes’ button is used to acquire and display the current camera attributes in the ‘Camera Properties Panel’ which is a sub panel of the command panel. The ‘Cam Attributes Confirmation’ textbox display the log messages only related to the camera. Once the camera port is closed and reopened the log in this textbox are also reset. After the camera is initialized and the properties are acquired the two buttons, used for opening the camera and acquisition of camera attributes, are deactivated whereas the ‘Close Camera on Port’ button is activated. In addition to the ‘Initialize Camera on Port’ and ‘Camera Attributes’ button, the ‘Quit’ button is also deactivated until the camera port is closed. The camera control area is shown in Figure 4.3 with the camera initialized on port 0 and acquired camera attributes.



```
Reseting and Initializing
Port 0: Manufacturer: National Instruments, port0, mBytes 21, nBytes 6, ver. 3, type 0
Port 1: Manufacturer: RS-232, com1, mBytes 7, nBytes 5, ver. 0, type 3
Port 2: Manufacturer: RS-232, com2, mBytes 7, nBytes 5, ver. 0, type 3
Port 3: Manufacturer: RS-232, com3, mBytes 7, nBytes 5, ver. 0, type 3
```

Figure 4.3: Camera control area with the camera open on Port 0 and acquired camera attributes.

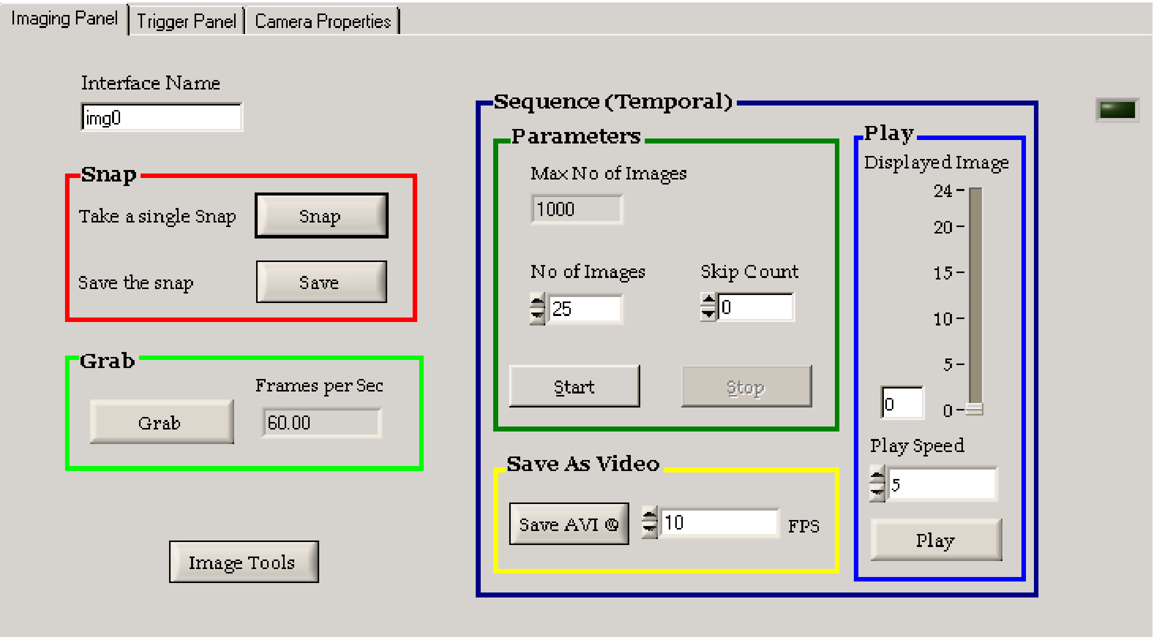
4.2.3 Command panel

The command panel is the area of the GUI which hosts all the controls of the acquisition software. It is located on the left side of the GUI as shown in Figure 4.1. These controls are divided into three major sub panels based on their application. These panels include the imaging panel, the trigger panel and the camera properties panel. The description of the controls included in each of these panels is provided in the sections below.

4.2.3.1 Imaging panel

The first sub panel of the command panel is the ‘Imaging Panel’. This panel contains all the controls related to the acquisition and display of the images. The ‘Imaging Panel’ can further be divided into three major areas. These areas are named as Snap, Grab and Sequence. The controls related to each area are divided using a different colored border.

The Snap area contains two buttons, ‘Snap’ and ‘Save’ within the red border. The ‘Snap’ button is used to acquire and display a single image of the field-of-view. The ‘Save’ button is deactivated until an acquired image is available in a buffer for saving to memory. This is useful for acquiring a reference image for the photobleaching MTV or a calibration image used for scaling the images.



Interface Name: img0

Frames per Sec: 60.00

Max No of Images: 1000

No of Images: 25

Skip Count: 0

10 FPS

Displayed Image: 0

Play Speed: 5

Figure 4.4: The imaging panel, a sub panel of the command panel, for the acquisition of images.

The Grab area, confined in the green border, contains a ‘Grab’ button and a display for the frame rate of the camera. The ‘Grab’ button is used for the continuous temporal acquisition of the field-of-view into a single buffer. These continuously acquired images are simultaneously displayed in the ‘Imaging Window’. The number of frames acquired and displayed per second is displayed in the ‘frames per second’ display in front of the grab button. This frame rate can be changed using corresponding controls in the ‘Camera Properties Panel’ discussed in section 4.2.3.3. The ‘Grab’ button changes to ‘Stop’ when the continuous acquisition is initiated. This ‘Stop’ button is used to stop the acquisition. It converts back to grab once the acquisition is stopped.

The controls in the Sequence area, isolated within the dark blue border, are used to acquire, display and save a temporal sequence of images. This approach forces the allocation of memory and allows the collection of images at full frame rate without missing any frames. The maximum number of images that can be acquired in a sequence is displayed in the maximum number of images display. The required number of images can be specified in the ‘No of Images’ input dialogue. The start button is used to initiate the acquisition of sequence. This sequence operation continues until the required number of images has been acquired. The ‘Stop’ button is activated during the acquisition of the sequence. The sequence acquisition can be discontinued using the ‘Stop’ button.

The images are not displayed until all the images specified in the ‘No of Images’ have been acquired. The acquired sequence of images is displayed like a video using the ‘Play’ button. Similar to the ‘Grab’ button the ‘Play’ button also converts to a ‘Stop’ button during the playback of images. When this ‘Stop’ button is pressed it converts back to the ‘Play’ button. The speed at which the images are displayed can be adjusted using the ‘Play Speed’ dialogue. The number of image currently displayed is updated on the ‘Displayed Image’ slide. The actual number of the displayed image is updated on the ‘Displayed Image’ slide as well as numerically in the dialogue located on the lower left side of the slide. Individual images can be displayed by using the numeric slide or entering the number of the desired image. Finally the sequence can be stored as a video file using the ‘Save AVI’ button at a frame rates specified in the FPS dialogue in front of the button. The ‘Play’ and the ‘Save AVI’ buttons are deactivated until a sequence of image is available for displaying in the imaging window and saving to memory respectively.

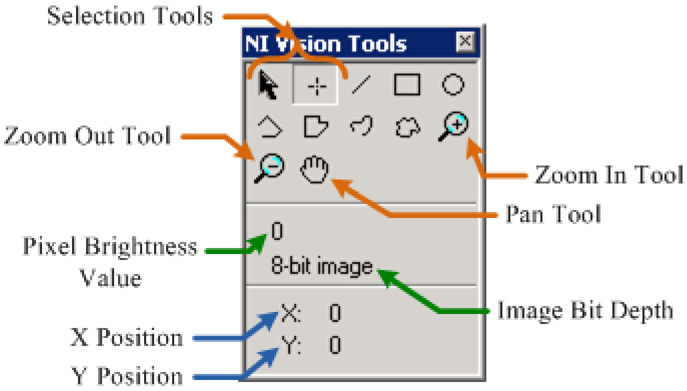


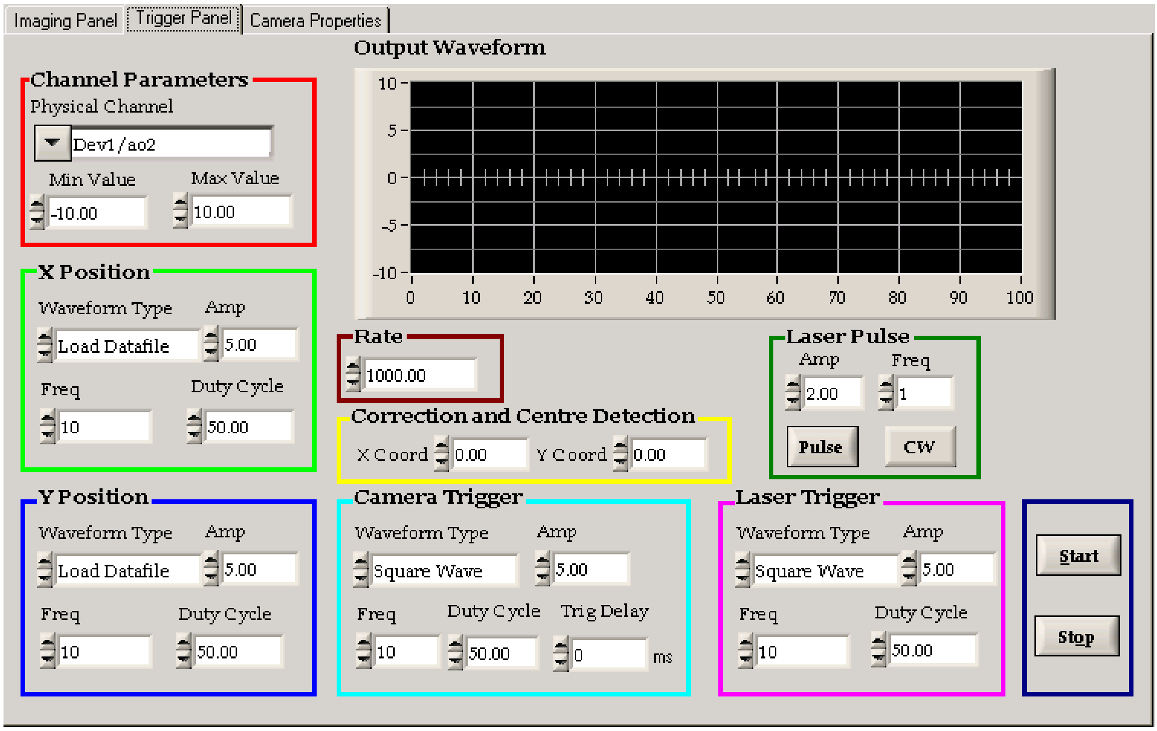
Figure 4.5: National Instruments vision tools popup window

The ‘Image Tools’ button displays various vision tools which are built in functions for examining some basic properties of the acquired images required for the efficient processing of the images. The vision tools appear as a pop up as shown in Figure 4.5. When images are available in memory the ‘Image Tools’ button is activated otherwise it is deactivated. The *X* position, *Y* Position and the pixel brightness values are displayed when the selection tool is used on the acquired image. The zoom in and zoom out tools can be used to get the desired magnification of the displayed image. The use of these tools only affects the display of the image. Any changes made on the displayed image does not reflect on the images when they are saved using the ‘Save’ button in snap area or ‘Save AVI’ button in the sequence area.

4.2.3.2 Trigger panel

The second sub panel of the command panel is the ‘Trigger Panel’. This panel controls the trigger signal for the laser scanner *X* position, *Y* position, the camera trigger and the laser trigger. The trigger panel is divided into different areas containing a set of controls for the triggering of a particular channel of the analogue output module described in section 3.7.7. All the controls confined in a colored enclosure are used to control designated parameters of the same channel. The trigger panel is shown in Figure 4.6.

Each of the four channels of the analogue output module (described in section 3.7.7) is used for controlling the laser scanner *X* position, laser scanner *Y* position, the camera triggering and the laser trigger. The designation of each channel to the actual hardware component it controls is given in Table 4-1. Each component can be triggered individually as well as in combination with any of the components. The controls in the ‘Channel Parameters’ enclosure are used to select the desired channels and to specify the minimum and maximum range of the amplitude of the output signal. The minimum and maximum amplitude of all the channels is collectively defined by selecting the values in the ‘Min Value’ and ‘Max Value’ dialogues. The minimum and maximum limit used here is -10 Volts and 10 Volts respectively matching the min/max limit of the analogue output module.



Channel Parameters
Physical Channel Dev1/ao2
Min Value -10.00
Max Value 10.00

X Position
Waveform Type Load Datafile
Amp 5.00
Freq 10
Duty Cycle 50.00

Y Position
Waveform Type Load Datafile
Amp 5.00
Freq 10
Duty Cycle 50.00

Rate 1000.00

Correction and Centre Detection
X Coord 0.00
Y Coord 0.00

Laser Pulse
Amp 2.00
Freq 1

Camera Trigger
Waveform Type Square Wave
Amp 5.00
Freq 10
Duty Cycle 50.00
Trig Delay 0 ms

Laser Trigger
Waveform Type Square Wave
Amp 5.00
Freq 10
Duty Cycle 50.00

Figure 4.6: The triggering panel, a sub panel of the command panel, for controlling the laser scanner X position, laser scanner Y position, the camera trigger and the laser trigger.

Table 4-1: Designation of channels of the analogue output module to the hardware component each channel controls.

Analogue Output Physical Channel	Actual Component/ Control Enclosure
Dev1/ao0	Laser scanner X position
Dev1/ao1	Laser scanner Y position
Dev1/ao2	Camera trigger
Dev1/ao3	Laser trigger

Single or multiple channels can be selected from the ‘Physical Channel’ drop down list as shown in Figure 4.7. If only one channel is selected the controls in the ‘*X* Position’ enclosure are used to control the output parameters. A designation of the control enclosures with respect to the number of selected channels is given in order in Table 4-2. For example, the 3rd channel (Dev1/ao2) controls the triggering of the camera. If the 3rd channel is the only selected channel, the controls within the ‘*X* Position’ enclosure will be used to control the output signal instead of the ‘camera trigger’ enclosure. Similarly if the 3rd and the 4th channels are selected they will be controlled using the ‘*X* Position’ and ‘*Y* Position’ enclosure controls respectively rather than the ‘Camera Trigger’ and ‘Laser Trigger’ enclosure controls. Multiple channels can be selected using the browse option along with the Ctrl key from the keyboard.

Table 4-2: Designation of control enclosure to the channel it controls for different number of selected channels.

No. of Selected Channels	Channel Number	Control Enclosure
1	1st Channel	X Position
2	1st channel	X Position
	2nd Channel	Y Position
3	1st channel	X Position
	2nd Channel	Y Position
	3rd Channel	Camera Trigger
4	As shown in Table 4-1	

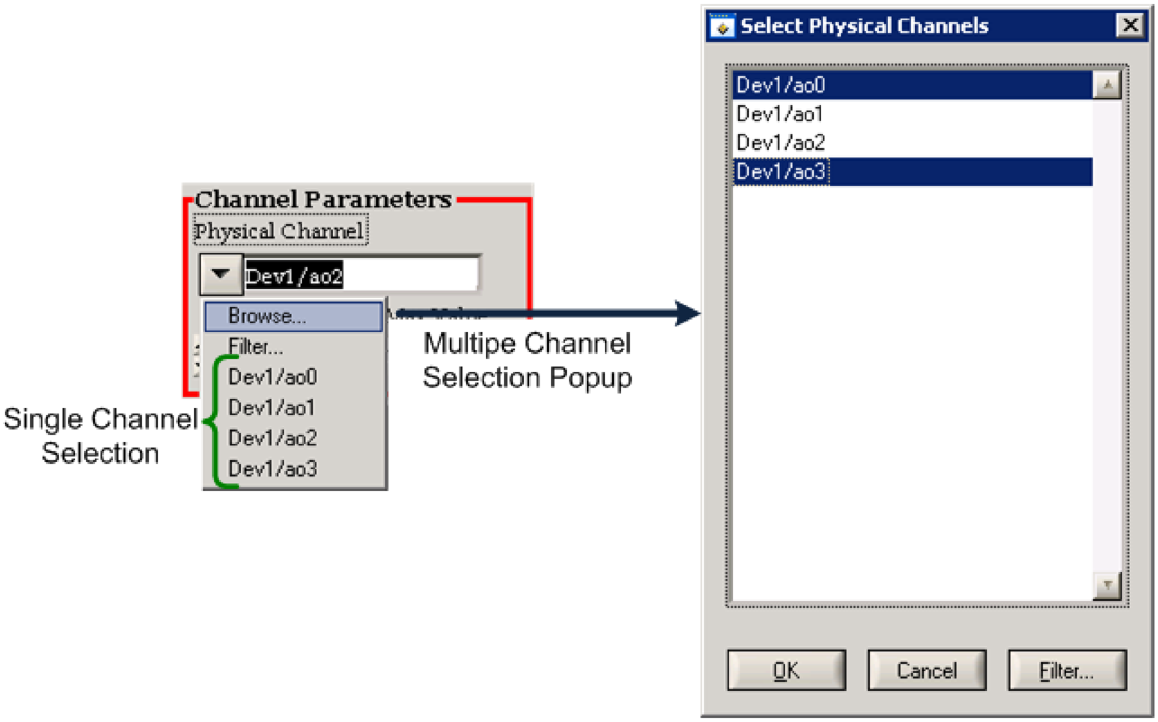


Figure 4.7: Single or multiple selection options from the trigger panel.

The ‘Rate’ dialogue is an important control for the signal generation because it controls the number of elements of the signal per second. It defines the sampling rate for the onboard sampling clock. A rate of 1000 translates into 1000 elements per second.

The common controls for output voltage in all the channel control enclosures are ‘Waveform Type’, ‘Amp’, ‘Freq’ and ‘Duty Cycle’. The ‘Waveform Type’ drop down control allows a selection from five different predefined waveforms along with an option of a user defined waveform. The options available in the ‘Waveform Type’ drop down menu are shown in Figure 4.8. The amplitude and frequency of the predefined waveforms can be specified using the ‘Amp’ and ‘Freq’ dialogues. The maximum and minimum value for the ‘Amp’ control cannot exceed those specified in the ‘Max Value’ and ‘Min Value’ dialogues respectively. The Duty Cycle option only controls the duty cycle of a square wave. None of these options affect the user defined waveform input. The ‘Trig Delay’ control in the ‘Camera Trigger’ enclosure controls the delay of the camera trigger signal. This displaces the generated camera trigger waveform by the specified time in milliseconds. This affects all waveform type options for the camera trigger including the user defined waveform.

The signal is generated upon depressing the ‘Start’ button. If a user defined waveform is selected a file selection popup will appear requiring the path of the user defined waveform. The user defined waveform is a Microsoft Excel file containing amplitude values with a number of elements equal to the rate specified in the ‘Rate’ dialogue. The frequency is of the user defined waveform is defined by the elements of the waveform. If the user defined waveform is selected for multiple channels, then the file selection popup will appear multiple times in an ascending order of the channels. If no file is selected no signal will be generated and the program will lead to an error. Once a signal is generated using the ‘Start’ button, a Microsoft Excel file containing the values of the signal is saved in the project folder. The number of elements of the signal is equal to the number of selected channels times the rate. The rate defines the values for a particular channel. For example, if 2 channels are selected and the signal is generated at a rate of 1000 per second then the saved signal file will contain a total of 2000 elements. The first 1000 values are the amplitudes for the first channel whereas the next 1000 values are amplitude values for the second selected channel.

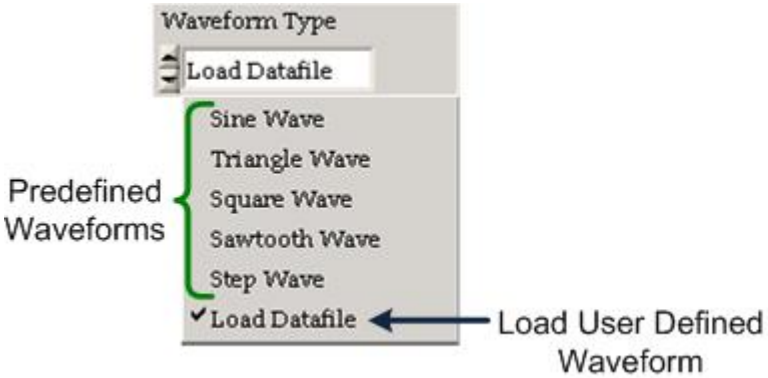


Figure 4.8: Available waveform selection from the ‘Waveform Type’ drop down control.

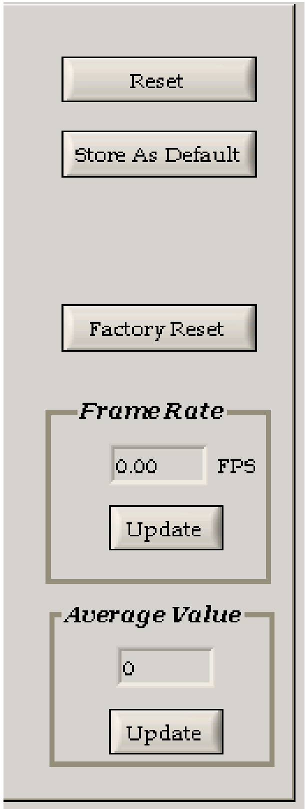
The controls in the ‘Laser Pulse’ enclosure are used to trigger either a single laser pulse or a continuous operation using the ‘Pulse’ or ‘CW’ control buttons respectively. The amplitude and frequency of the laser pulse can be defined using the ‘Amp’ and ‘Freq’ controls within the ‘Laser Pulse’ enclosure. The selection of the channels from the ‘Physical Channel’ drop down control is not required for this option. Signal for other channels can be simultaneously generated with the continuous option since continuous illumination will be required for techniques such as µ-PIV. When a continuous signal is generated using the ‘CW’ button it converts to a ‘Stop’ button which is used to stop the continuous signal. When the generated signal is stopped the button converts back to ‘CW’.

The ‘X coord’ and ‘Y Coord’ controls in the ‘Correction and Centre Detection’ enclosure are used to detect the approximate centre of the laser position in the field-of-view. These controls are also used to determine the limit of the laser scanner scanning system within the field-of-view. Similar to the ‘Laser Pulse’ option no channel selection is required for these controls. The ‘*X* Coord’ is used to move the laser scanner in the horizontal direction whereas the ‘*Y* Coord’ is used to move the laser scanner in the vertical direction. The maximum and minimum acceptable values in these controls are defined by the ‘Max Value’ and ‘Min Value’ dialogues respectively within the ‘Channel Parameters’ enclosure. These can be used in combination with the other signal generating controls. If it is used in combination with the ‘*X* position’ and ‘*Y* position’ enclosure controls the value of the ‘*X* Coord’ and ‘*Y* Coord’ is added to the amplitude of the respective enclosures. Alternatively, they can be set to ‘0’ and these values can be directly incorporated in the user defined waveform amplitude values.

The ‘Output Waveform’ , located in the top right of the ‘Trigger Panel’, is a graphical display of the signal generated per second for all the channels. Different colors are used to display the generated signal graphically. This gives a visual indication of the interaction of the components in terms of the generated signal. The amplitude of the generated signal is displayed on the vertical axis whereas the time in seconds is displayed on the horizontal axis of the ‘Output Waveform’ display.

4.2.3.3 Camera properties panel

The third sub panel of the command panel is the ‘Camera Properties Panel’. This panel contains the controls for varying the operational properties of the camera. Initially these controls are disabled which is indicated by the deactivated borders around the controls within each sub panel of the ‘Camera Properties Panel’. When the ‘Camera Attributes’ button is pressed from the camera control area the program acquires the available camera attributes with their current values or the last used values. The borders of the active controls within each sub panels are activated indicating the available controls. All the controls described in the sections below can be used to alter the camera operation in real time.



Frame Rate
0.00 FPS

Average Value
0

Figure 4.9: Main command buttons of the camera properties sub panel (a sub panel of the command panel).

The command buttons on the right hand side of the panel are the main buttons of the panel. Other property controls are grouped into seven different tabs. The main command buttons of the camera properties panel are shown in Figure 4.9.

The main buttons include ‘Reset’, ‘Store as Default’, ‘Factory Reset’, ‘Frame Rate’ and ‘Average Value’. The ‘Reset’ button is used to reset the camera to default configuration. The current configuration of the camera properties are saved to the flash memory of the camera using the ‘Store as Default’ button. These configurations are loaded upon using the ‘Reset’ button. The ‘Factory Reset’ button resets the camera with the factory default property values. The ‘Frame Rate’ display shows the actual rate of acquisition of the camera in frames per second (FPS) when the corresponding ‘Update’ button is pressed. The ‘Average Value’ display indicates the greyscale average of the actual image in 2 bits (0 to 4095) upon pressing the corresponding ‘Update’ button.

The camera properties panel is further divided into seven sub panels containing a group of common controls as shown in Figure 4.10. These sub panels include ‘Exposure’, ‘Window’, ‘Trigger’, ‘Data Output’, ‘Characteristics’, ‘Correction’ and ‘Info’. Each panel controls specific properties related to the operation of the camera. The name of the controls and the panels are selected to be as close as possible to the camera properties they control.

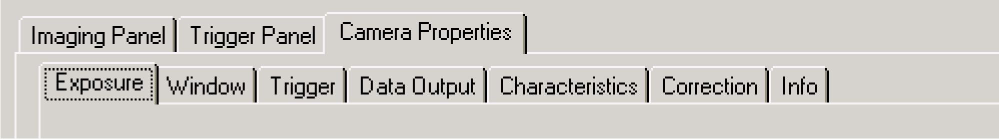
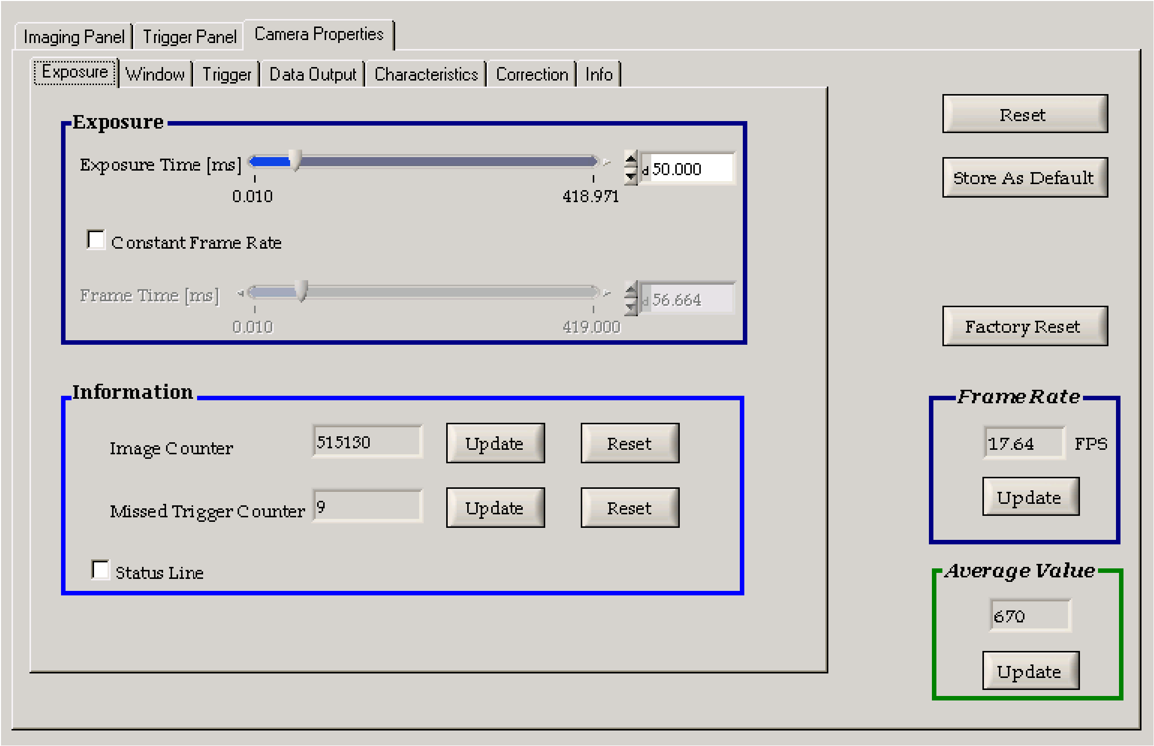


Figure 4.10: Sub panels of the camera properties panel (a sub panel of the command panel)

The ‘Exposure’ sub panel contain the controls for the exposure settings of the camera when the exposure time is defined by the camera. It is divided into two enclosures, exposure and information as shown in Figure 4.11. The camera exposure time is configured in milliseconds using the ‘Exposure Time [ms]’ slider or numeric control. The time required by the camera sensor to integrate the incoming light is defined by the exposure time. The ‘Frame Time [ms]’ slider and numeric control are disabled until the ‘Constant Frame Rate’ is not checked. However, in this case the frame rate is updated along with the exposure time at a predefined rate. The range of the exposure time and the frame time is 0.01 to ~419 msec. When the ‘Constant Frame Rate’ is checked, the ‘Frame Time [ms]’ control is used to control the frame time independently in milliseconds. However, when the constant frame rate option is used the camera may operate at a rate lower than the maximum possible value. When the ‘Constant Frame Rate’ is switched off, the camera delivers the images at the maximum possible rate which depends on the exposure time and the camera read out time.



Exposure

Exposure Time [ms] 50.000 (0.010 – 418.971)

□ Constant Frame Rate

Frame Time [ms] 56.664 (0.010 – 419.000)

Information

Image Counter 515130

Missed Trigger Counter 9

□ Status Line

Frame Rate

17.64 FPS

Average Value

670

Figure 4.11: Exposure panel controls for setting the camera exposure settings.

The information enclosure surrounds the controls that provide information about the acquired images. The ‘Image Counter’ displays the real time 24 bit count of the images and increments by 1 for each new acquired image. The ‘Missed Trigger Counter’ displays the count of the trigger pulses that are not utilized because they were received either during image exposure or image read out. When the camera is operating in ‘Free Running Mode’ all the pulses received from interface triggering or I/O triggering are counted as missed trigger. The values for both the aforementioned controls are updated using the corresponding ‘Update’ button. These controls can be reset using the corresponding ‘Reset’ buttons. When the ‘Status Line’ box is checked the last row of the image is replaced by the image information with each parameter coded into 4 pixels. The details of the parameters in the status line please refer to the camera manual (PhotonFocus MV-D1024E-160-CL manual).

The ‘Window’ sub panel contains the controls for the region of interest, multiple regions of interest and the decimation settings. These controls are not yet incorporated in the GUI. However, they will be included in the future which will drastically improve the control of the user over the performance of the camera in terms of the frame rate and region of interest settings.

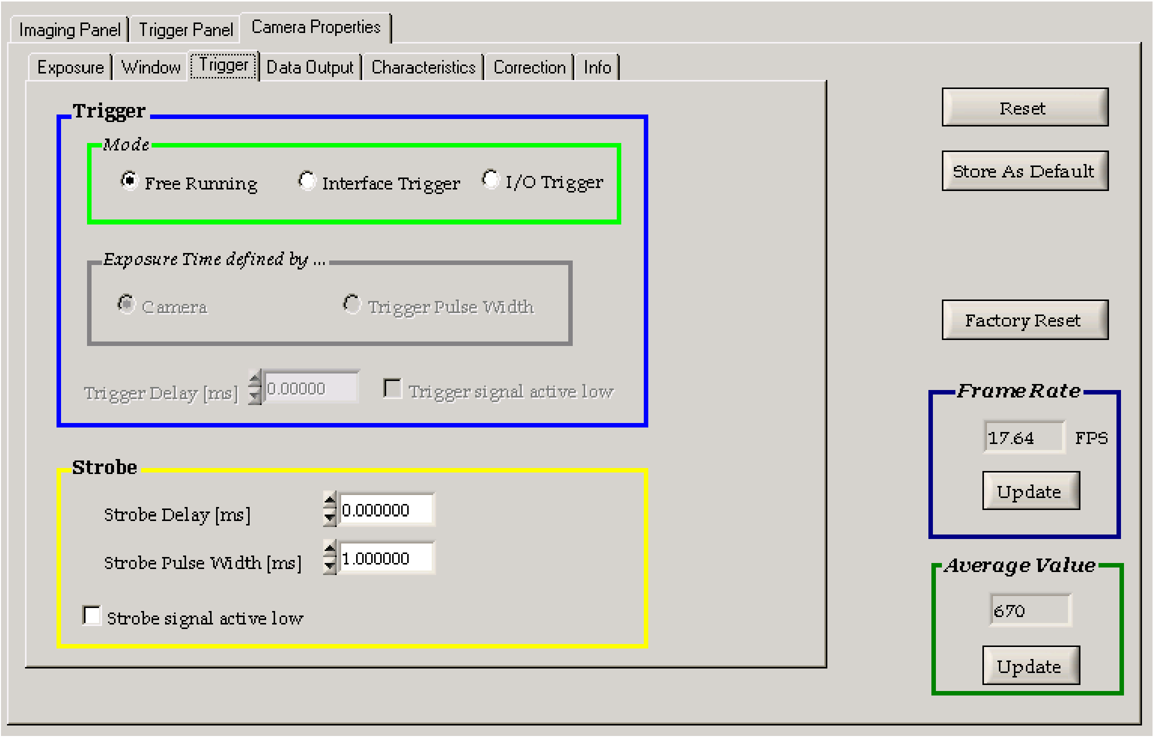
The ‘Trigger’ sub panel contains the controls for the trigger and strobe settings of the camera as shown in Figure 4.12. This sub panel is divided into two main enclosures, Trigger and Strobe. The Trigger enclosure contains the controls for the trigger mode, the exposure control and the trigger delay in milliseconds. One out of three trigger sources can be utilized for triggering the camera.

The trigger modes include ‘Free Running mode’, ‘Interface Trigger mode’, and ‘I/O Trigger mode’. In the ‘Free running mode’ the camera continuously acquires and delivers the images at a frame rate defined by the ‘Exposure Time [ms]’ in the ‘Exposure’ sub panel. The Interface Trigger mode is used when the trigger is applied by the CameraLink interface The I/O Trigger mode is used when the trigger is applied directly to the camera through the power supply connector. The trigger signal is typically in the form of a square wave. The exposure time for the later two trigger modes can either be defined by the camera or by the pulse width of the trigger signal. If the exposure time is defined by the camera then the image exposure is initiated at the incoming trigger active edge. The time of exposure is defined using the ‘Exposure Time [ms]’ control from the ‘Exposure’ sub panel. The exposure is not only initiated by the input trigger signal active edge but the pulse width of the trigger signal also defines the exposure time of the camera when the ‘Trigger Pulse Width’ option is selected from the ‘Exposure Time defined by’ enclosure. The exposure stops with the inactive edge of the trigger signal. The controls in the exposure enclosure of the ‘Exposure’ sub panel are disabled when the Interface trigger or the I/O trigger modes are used with the exposure time defined by the trigger pulse width.

The ‘Trigger Delay [ms]’ is used to introduce a programmable delay in milliseconds between the incoming trigger signal active edge and the start of image acquisition. This delay can be utilized to synchronise the firing of the laser to the triggering of the camera. When this delay is used it tends to limit the frame rate of the camera. Hence, it is used to find a suitable delay value (msec) to be used in the ‘Trig Delay’ control within the ‘Camera Trigger’ enclosure of the ‘Trigger Panel’. This introduces the delay in the generated signal and the camera can be operated with no trigger delay at the desired frame rate.

By default the camera exposure is initiated at the rising edge of the incoming trigger signal. However, this polarity can be reversed using the ‘Trigger signal active low’ checkbox. Likewise, for the exposure defined by the incoming trigger pulse width the lower width will be utilized for the exposure time of the camera. In order to use the camera in external trigger mode, ‘CC1’under the ‘Camera Control Lines Source’ should be set to external type with number 2 in the ‘Camera Attributes’ tab in Measurement and Automation Explorer (MAX). The ‘Use Default Sources’ checkbox under the ‘Camera Control Lines Source’ in the ‘Camera Attributes’ tab in MAX should also be unchecked.

The controls within the Strobe enclosure regulate the strobe signal delay, pulse width and polarity. A strobe output is an opto-isolated output which can be used to trigger a strobe and is located on the power supply connector. However, a separate power supply is required for the strobe output. The ‘Strobe Delay [ms]’ control defines the delay time in milliseconds between the input trigger signal active edge to the rising edge of the strobe output signal. The actual width of the strobe trigger output is defined in milliseconds using the ‘Strobe Pulse Width [ms]’ control. In order to disable the strobe signal output the ‘Strobe Pulse Width [ms]’ should be set to 0. By default the strobe signal output is active high. The active high or active low output of the strobe signal can be defined using the ‘Strobe signal active low’ checkbox.



Trigger

Mode

● Free Running ○ Interface Trigger ○ I/O Trigger

Exposure Time defined by ...

● Camera ○ Trigger Pulse Width

Trigger Delay [ms] 0.00000 □ Trigger signal active low

Strobe

Strobe Delay [ms] 0.000000

Strobe Pulse Width [ms] 1.000000

□ Strobe signal active low

Frame Rate

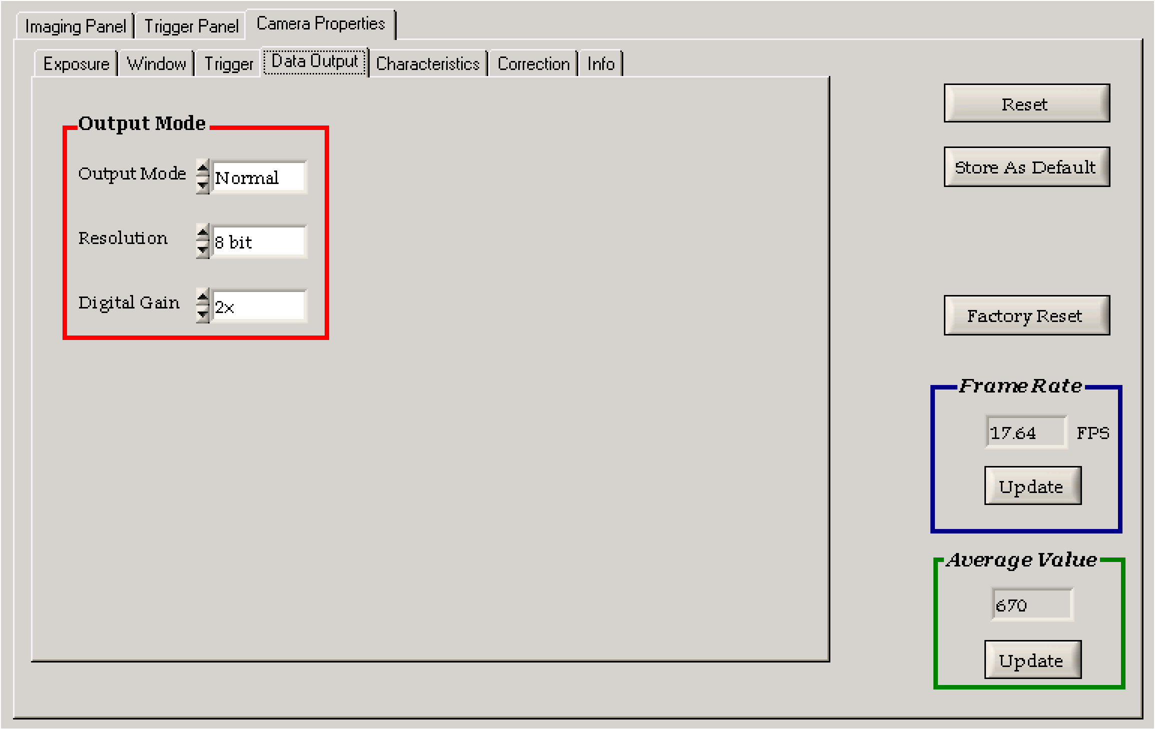
17.64 FPS

Average Value

670

Figure 4.12: Trigger sub panel controls for setting the camera triggering settings.

The ‘Data Output’ sub panel controls the image data settings in terms of the bit depth (resolution) and digital gain as shown in Figure 4.13. The options available for the Output Mode, Resolution and Digital Gain of the camera are shown in Figure 4.14 (a), (b) and (c) respectively.



Output Mode

Output Mode: Normal

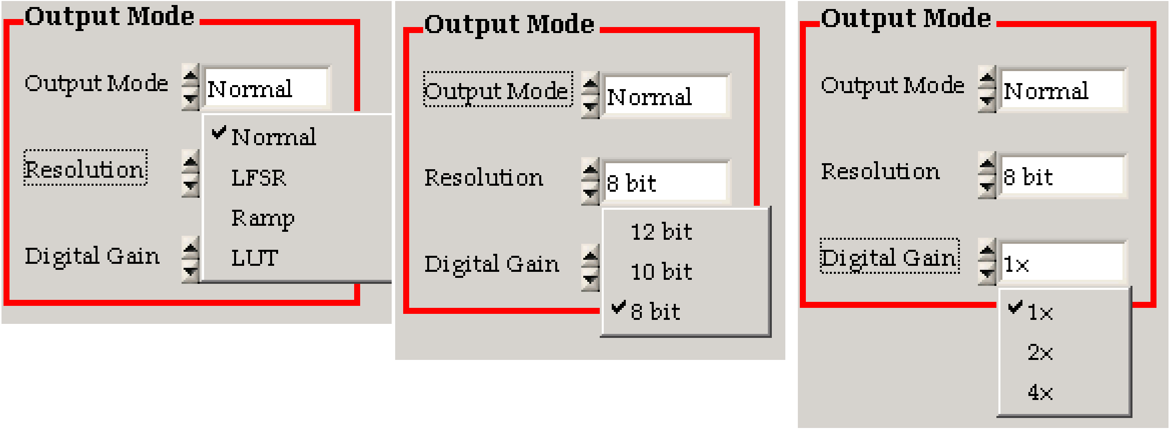
Resolution: 8 bit

Digital Gain: 2x

Frame Rate: 17.64 FPS

Average Value: 670

Figure 4.13: Data output sub panel controls for setting the output image data settings.



(a) (b) (c)

Figure 4.14: Data output sub panel control options for the (a) Output mode (b) Resolution and (c) Digital gain for the output image data settings.

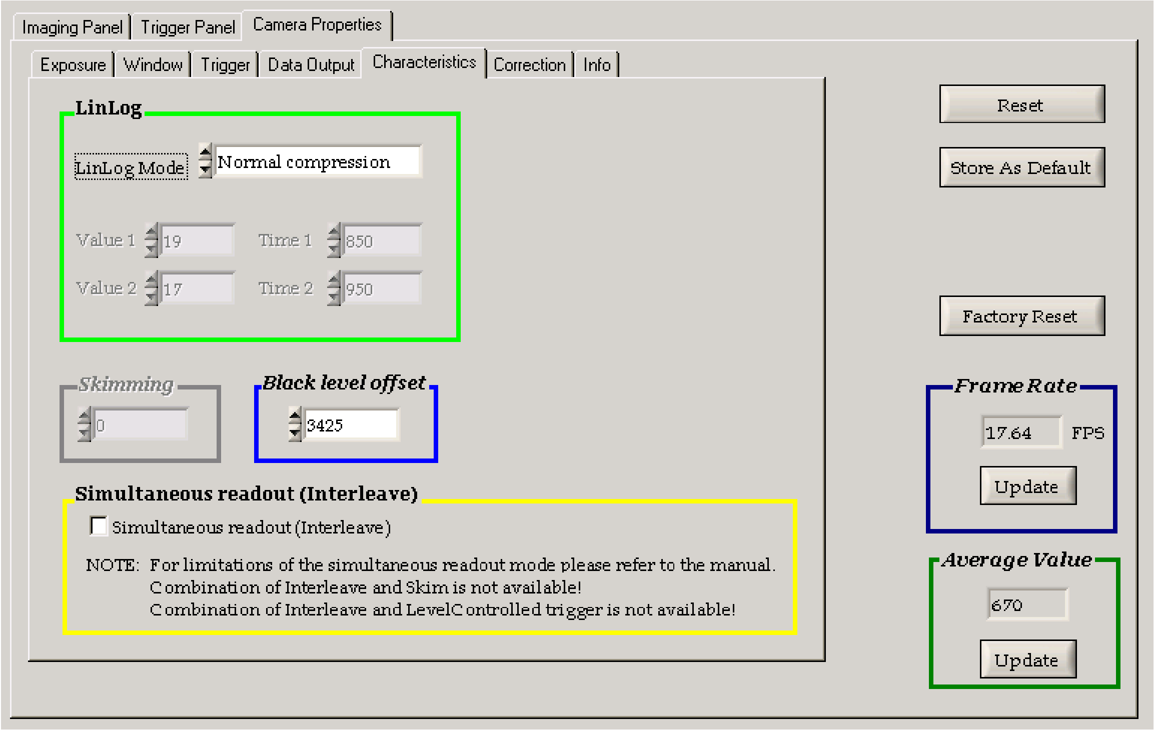
The Output mode has four options, Normal, LFSR, Ramp and LUT. The Normal mode is used for image acquisition. The LFSR and Ramp modes are used to generate test images. The test images are used to identify data transmission errors in the data transmission path from the camera to the frame grabber. A test image of a constant pattern containing every possible grey level repeated for every row of a pseudo-random gray level sequence is generated using the ‘LFSR’ (linear feedback shift register) output mode. The ‘Ramp’ output mode generates a test image containing a constant pattern, depending on the gray level resolution, with increasing grey level from left to right side.

The ‘LUT’ mode is basically a grey level transformation which remaps the grey level values of an input image to the new values. The grey scale value of each pixel in an image is converted into another grey value using the look-up table (LUT). For more information related to test images and the look up table please refer to the camera manual.

The camera can be operated at 8 bit, 10 bit and 12 bit resolution. The required bit depth can be selected from the ‘Resolution’ drop down list. In order to use the camera at the desired bit depth the corresponding bit depth should be selected in the camera attributes in Measurement and Automation Explorer (MAX).

A digital amplification is applied to the images within the camera using the three available values in the ‘Digital Gain’ drop down control. Digital gain 1x, 2x or 4x means that the digital image data is multiplied by a factor of 1, 2 or 4 respectively within the camera.

The ‘Characteristics’ sub panel contains the controls for modifying the response of the camera to the input light signal and the output grey level as shown in Figure 4.15. The default response of the camera to the input light and the output grey level is linear. However, this response can be modified by using the LinLog or Skimming options available in this sub panel. The LinLog and Skimming options both are PhotonFocus specific technologies. Skimming feature, a proprietary PhotonFocus technology, provides an adjustable range of in-pixel gain for low signal levels which can be used to enhance detail in dark areas of the image. Skimming can be used independently as well as in combination with the LinLog modes (Low and Normal Compression modes only). The average image value at no light intensity is the black level. This black level can be adjusted using the ‘Black level offset’ control which changes the overall brightness of the image.



LinLog

LinLog Mode: Normal compression

Value 1: 19 Time 1: 850

Value 2: 17 Time 2: 950

Skimming: 0

Black level offset: 3425

Simultaneous readout (Interleave)

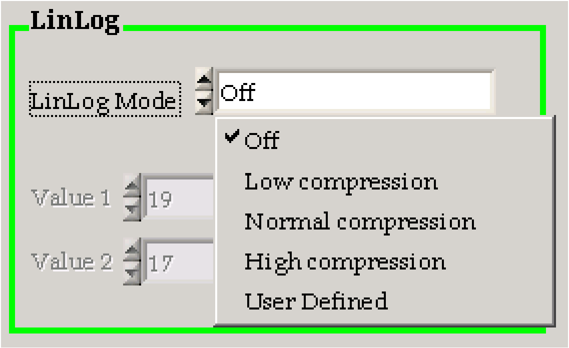
☐ Simultaneous readout (Interleave)

NOTE: For limitations of the simultaneous readout mode please refer to the manual.
Combination of Interleave and Skim is not available!
Combination of Interleave and LevelControlled trigger is not available!

Frame Rate: 17.64 FPS

Average Value: 670

Figure 4.15: Characteristics sub panel for controlling the camera response.



LinLog Mode: Off

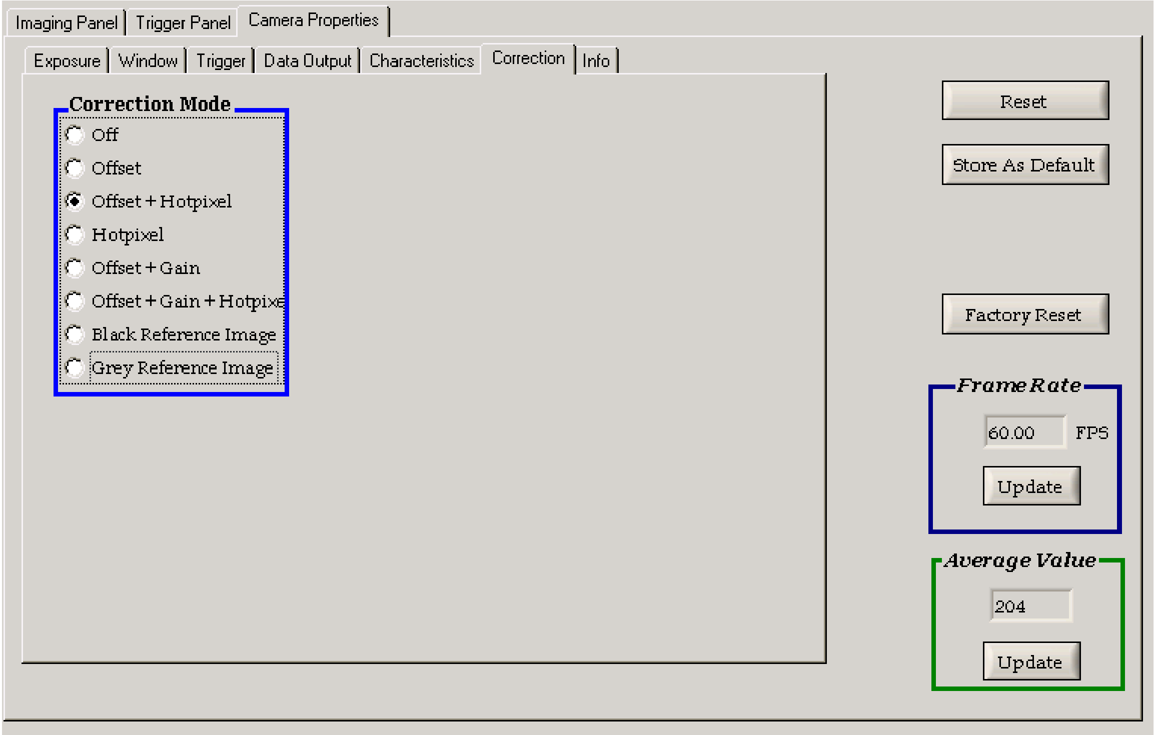
Value 1: 19

Value 2: 17

Figure 4.16: The available LinLog compression mode for the camera.

The camera is capable of operating in sequential read out mode and simultaneous read out mode. In the sequential read out mode, the next image acquisition will initiate only when the previous image has been completely read out. Hence, the frame time in the sequential read out mode is the sum of exposure time and read out time. When the camera is operated in the simultaneous read out mode, the acquisition of the next image depends on the maximum of the exposure time and the read out time. Hence, the frame time is determined by the maximum of the exposure time or the read out time. The 'Simultaneous readout (Interleave)' checkbox is used to switch between the two modes. When it is checked the camera operates in the simultaneous read out mode otherwise it operates in the simultaneous read out mode by default. The frame rate of the camera increases significantly by using the simultaneous read out mode if the frame time is less than or equal to the exposure time. However, the simultaneous read out mode is only available in the 'Free Running' trigger mode. For more information about the LinLog modes, Skimming, Black level offset and the read out modes please refer to the camera manual.

Some image pre-processing features for the compensation of non-uniformities caused by the sensor, the lens or the illumination are available for the camera. The improvement of image quality in this way is generally known as 'Shading Correction' or 'Flat Field Correction' and entails a combination of offset correction, gain correction and pixel interpolation. These image pre-processing features are grouped in the 'Correction' sub panel as shown in Figure 4.17. A black and a grey reference image are required for the automatic determination of correction values by the camera. The internal flash memory of the camera contains default factory preset correction values. However, the factory preset correction values can be replaced by new values calculated from obtained black and grey reference images. A black reference image is an image obtained at no illumination e.g. with the lens aperture close whereas the grey reference image is an image obtained at uniform illumination. The offset correction is based on the black reference images whereas the gain correction is based on the grey reference image. The fixed pattern noise of the CMOS sensor is reduced using the offset correction which subtracts a configurable positive or negative value from the live image. The error arising due to hot pixels is typically removed by interpolation. A certain threshold value applied to each pixel of the black reference image determines the hot pixels. The value of the hot pixel is replaced by the average of its neighbour when the hot pixel correction is applied. In order to flatten uneven illumination or to compensate shading effects of a lens the gain correction is used. The offset and gain correction are applied to individual pixels separately. The correction modes can be switched separately with a number of available combinations as shown in Figure 4.17. The black reference image and the grey reference image stored in the camera RAM can be output using the last two options respectively. Please refer to the camera manual for details about the generation and properties of the reference images and the operation of the offset and gain correction algorithm.



Correction Mode
○ Off
○ Offset
● Offset + Hotpixel
○ Hotpixel
○ Offset + Gain
○ Offset + Gain + Hotpixe
○ Black Reference Image
○ Grey Reference Image

Frame Rate
60.00 FPS

Average Value
204

Figure 4.17: Correction sub panel for the application of the correction modes.

The ‘Info’ panel is the last sub panel of the camera properties panel. It contains information about the camera. It displays the camera model, serial number, max pixel clock values and the max available resolution.

4.3 *Photobleaching MTV scheme steps*

A number of parameters and a sequence of operations are involved in the image acquisition for the photobleaching MTV techniques. Similarly, typical parameter values and various steps are involved in processing the acquired images in order to reveal the tagged region diameter and the velocity of the tagged region. The following two sections contain a description of the sequence of operation and typical effective parameters involved in the image acquisition and processing of images for the photobleaching MTV approach.

4.3.1 *Image acquisition*

A sequence of operation utilizing the developed GUI is required to collect useful digital images for the photobleaching MTV technique. The steps involved in the image acquisition for the photobleaching MTV experiments are described in Figure 4.18. The steps are divided into three main groups including camera settings, trigger settings and image acquisition.

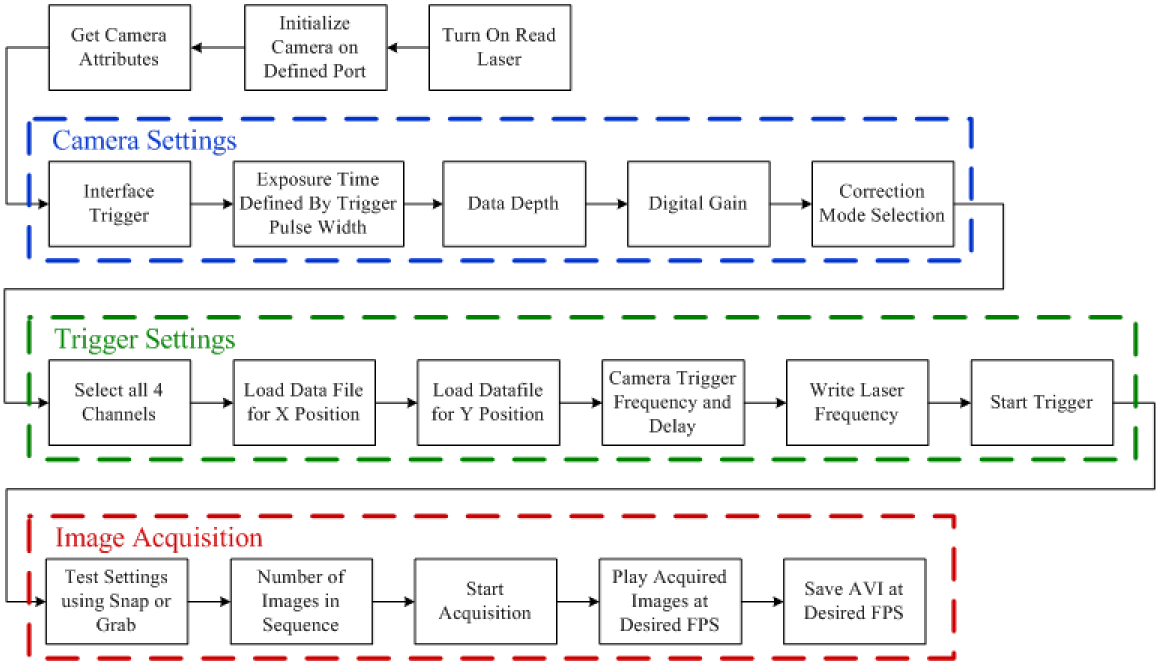

Turn On Read Laser
Initialize Camera on Defined Port
Get Camera Attributes
Camera Settings
Interface Trigger
Exposure Time Defined By Trigger Pulse Width
Data Depth
Digital Gain
Correction Mode Selection
Trigger Settings
Select all 4 Channels
Load Data File for X Position
Load Datafile for Y Position
Camera Trigger Frequency and Delay
Write Laser Frequency
Start Trigger
Image Acquisition
Test Settings using Snap or Grab
Number of Images in Sequence
Start Acquisition
Play Acquired Images at Desired FPS
Save AVI at Desired FPS

Figure 4.18: Flow chart indicating the sequence of steps involved in image acquisition for the photobleaching MTV technique.

The read laser is continuously required in order to generate the fluorescence signal as indicated in Figure 3.2. The camera is initialized on port 0 and the camera attributes are acquired. The settings related to the camera are grouped in camera settings. The trigger mode of the camera is set to interface trigger with the exposure time defined by the trigger pulse width. This enables the trigger signal to control the instant and duration of the exposure of the camera. A resolution of 8 bits is selected as the data depth of the images. A digital gain is selected based on the brightness of the fluorescence signal incident on the camera sensor. A correction mode of ‘Offset + Gain + Hotpixel’ is selected in order to obtain error free images for processing.

The trigger settings are used to generate the trigger signal for controlling the laser scanner position, the camera and the laser triggering. All four channels of the analog output are selected. Two Microsoft Excel files containing the *X*-Coordinates and *Y*-Coordinates of the tagging points are generated. Load Datafile option for the X-position and *Y*-Position waveform type are selected to use these coordinates. A square waveform is selected for both the camera and the laser trigger. The desired frequency, the calculated delay and the duty cycle are selected for the camera trigger whereas only the frequency and the duty cycle is selected for the laser trigger. Upon pressing the start button the two files containing the *X*-Coordinates and *Y*-Coordinates for the laser scanner are loaded. This generated the desired trigger signal suitable for the photobleaching MTV measurements.

The Snap or Grab button is used to test these settings. If these setting do not provide the images with the desired properties then these settings are changed and tested again until the desired quality and properties of the images are acquired. The number of images required is selected and the acquisition is started using the Sequence button. After the acquisition finishes the acquired images are played and checked at the selected speed (frames per second). Then the images are saved as a video file in AVI format at the selected frame rate. The images can be saved multiple times at multiple frame rates.

4.3.2 Image processing

The sequence of images is processed using the Particle Master Shadowgraphy module of commercially available software (LaVision, DaVis 7.4). A flow chart elaborating the sequence of operation for the image processing is presented in Figure 4.19. The set of raw images are imported into the image processing software (LaVision, DaVis 7.4). The number of images and their order for import can be selected here. The images are then stored as part of the particle sizing processing project. All the processing steps mention hereafter are performed in the batch processing mode.

For velocity calculation the separation between the points in the object plane need to be specified for the spatial calibration of the acquired images. The spatial calibration of the images is essential for velocity calculations as the required measured distance is in mm rather than in pixels. Since the camera is perpendicular to the object plane, a linear scaling of the camera can be applied. A linear relationship between pixel and physical space is used to determine a factor and an offset for the linear scaling of the camera. A calibration plate containing equidistant dots is used for the calibration of images. This calibration plate is divided into different areas. Each area contains equidistant dots with a distinct separation between them. The factor and the offset required for the linear scaling of the camera is obtained by imaging the suitable region of the calibration plate. This gives the pixel/mm ratio required for the calibration. Since the camera is perpendicular to the object plane hence the second calibration mode (define scale, no image distortion) is selected. In this mode the pixel/mm ratio is directly entered and the reference point for the scales is selected. Although it is possible to load the whole image set, typically one image is opened up using the load button. A preview of the selected image is available in the modal dialog. The pixel/mm ratio is input to calibrate the image to object space. The origin or the reference point is also determined in this step. The reference point is defined by entering the pixel number and the corresponding mm value each for the *x* and *y* axis. This calibration is then applied to the entire image set. This calibrates the all the images in the set to object space in mm.

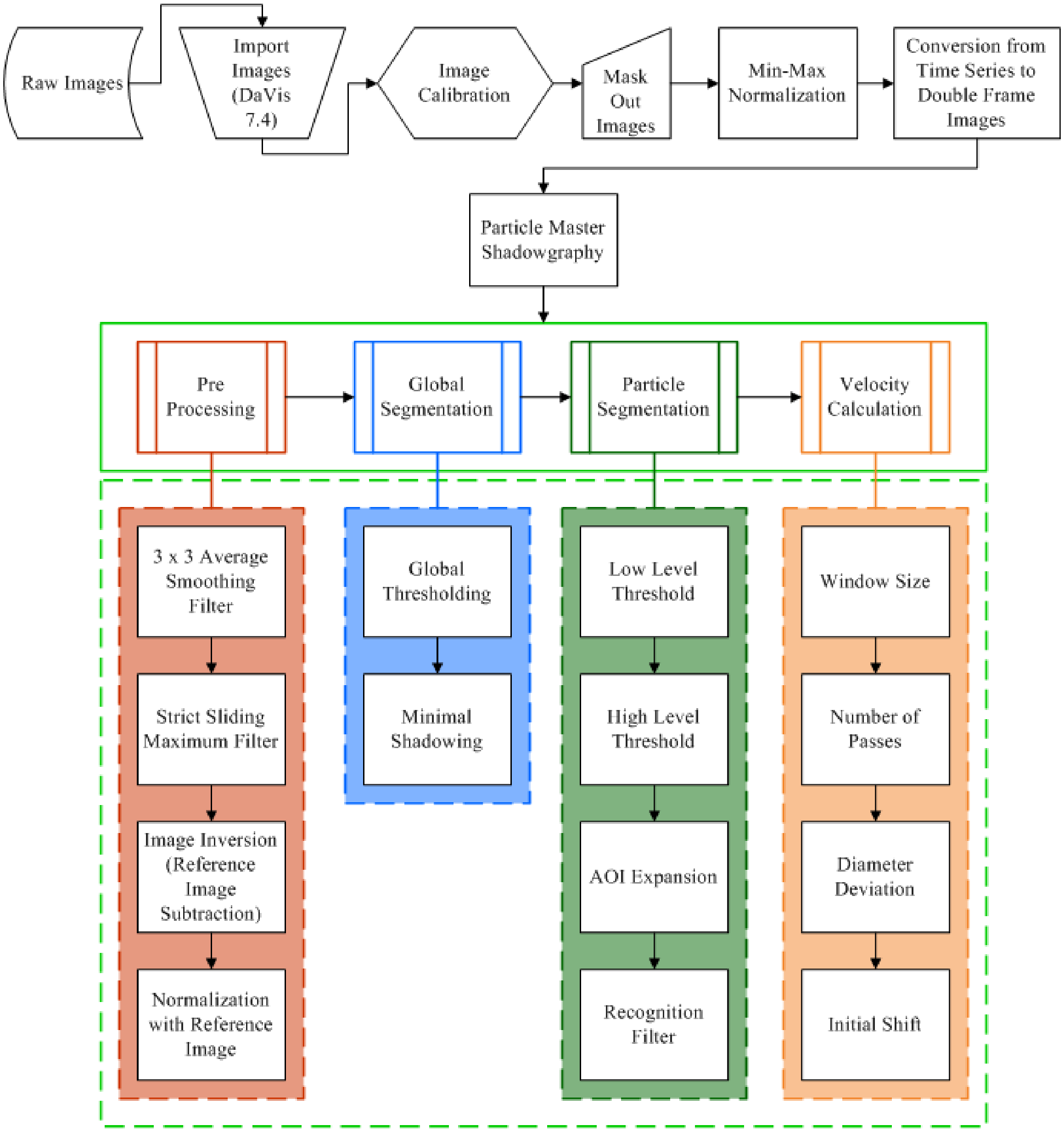


Figure 4.19: A flow chart of the sequence of image processing operations for Photobleaching MTV.

A geometric mask is applied in order to remove the undesired region of the images. The boundary of the flow region is defined using a geometric mask. A masked out operation from the masking function processing group is used to apply a frame dependent, enabled inside, geometric mask to the set of images. This results in the removal of the region outside the channel geometry. The region outside the mask is not included in further processing.

Due to the in-homogeneity in the tagged region diameter in successive images, a local tagged region intensity correction is required (Images shown in next chapter). A min/max intensity normalization filter from the non-linear filter group is applied which determines the local maximum and minimum intensity in a region defined by the scale length in pixels. A sliding minimum calculated over a window of the defined scale length is subtracted from the original image which results in the removal of the local background intensity level. A ratio of a local strict sliding maximum over a window of the specified scale length to a global maximum (sliding maximum over a window of 10 times the scale length) is multiplied with the result of the previous operation (removal of the local background intensity level). This results in homogeneous tagged region intensities without changing their location. A scale length of 100 pixels is used for the mix/max intensity normalization of the images.

The velocity calculation using the particle sizing is only possible for double frame images. Hence, the time series of images is required to be converted to double frame images. A create multi-frame buffer from time series operation from the copy and reorganize processing group is applied to convert the time series to double frame images. Consecutive images (1+2, 3+4 …) are used to generate the double frame images from the time series.

The images are now ready for the particle sizing and velocity calculation operations. These operations consist of four major steps. These steps include preprocessing, global segmentation, particle segmentation and velocity calculation. The actual parameters values used in particle sizing are mentioned in Appendix C-1. In the first step the number of 1st and 2nd frame from the double frame images need to be defined. The tagged region statistics are calculated only from the first frame whereas the second frame is only used for velocity calculations. The number of first frame is defined as “0”and the number of second frame is defined as “1”.

A reference image is required for shadowgraphy image processing in order to distinguish the dark tagged regions against the bright background light. A reference image may be recorded prior to tagging the desired regions. Alternatively it can be calculated from the source images. A reference image is simply an image of the bright background devoid of any tagged region with a constant distribution and illumination intensity. The selection of the reference image and a smoothing filter is performed in the image preprocessing step. A 3 x 3 average smoothing filter is used in order to remove the noise and smooth the edges of the tagged regions. A reference image is calculated for each source image. A strict sliding maximum filter, with a filter length of 100 pixels, and subtracting the minimum for each image aides in the calculation of the corresponding reference image. Subtracting the minimum deals with the tagged region which are not completely black. The strict sliding maximum filter generates a smooth reference image with intensity counts either equal to more than the source image. The filter length for the strict sliding maximum filter is required to be about 1.5 times the diameter of the biggest tagged region. The maximum tagged region diameter was around 70 pixels hence a strict sliding maximum filter length of 100 pixels is used. Inverted intensity images are obtained by subtracting the reference image from the corresponding tagged images. The inverted images have high intensities for the tagged region and low intensities for the background. In order to remove systematical errors due to inhomogeneous illumination, the difference between the tagged images and the reference image are normalized by the reference image in the particle recognition dialogue box. Normalizing with the reference image means that the inverted image is also divided by the reference image.

In the global segmentation step a user defined global threshold defines the area of the image for segmentation. The global threshold is a percentage of the difference between the maximum and minimum intensity of the image. The areas of the image above this global threshold value are segmented with a rectangle arranged above each segment. A pixel is determined to be in a certain segment if it satisfies two conditions. First it should be above the user defined global threshold and second it should be adjacent to another pixel of the same segment. A 50% global threshold is used for the global segmentation of the images. Minimal shadowing prevents the noise to be detected as a tagged region in case the image does not contain any tagged region. It is required if the images are not normalized with the reference image, hence it is not used here.

The tagged regions, their diameter, centre of mass and eccentricity are detected in the particle segmentation step. The particle segmentation is applied to all the segments of the image that are above the global threshold analyzing each segment separately. Two separate areas based on a low level and high level thresholds are calculated by counting the number of pixels above each threshold. Horizontal and vertical edges are arranged around the segments forming the smallest bounding box which is enlarged by the angle of incidence (AOI) expansion user defined percentage. The expansion of the area of interest relative to that resulting from global segmentation is defined by the AOI expansion option. The longest and shortest axis and a corresponding circle diameter are computed for the areas defined by each threshold. The average of the values obtained for both levels results in determining the tagged region diameter, the eccentricity and the centre of mass. A threshold of 50% and 20% is used for low level and high level respectively along with an AOI expansion of 30%. The fill particle option is not checked since none of the tagged regions have any holes in them resulting in faster computation.

Region of the inverted image that are above the global threshold are sometimes detected as tagged regions. Similarly, the diameters of highly diffused and defocused tagged areas may be exaggerated. The recognition filter is applied in order to filter out undesired areas to be detected as tagged regions. A maximal low level area filters out highly diffused and out of focus tagged regions resulting in exaggerated tagged region diameters. In the case of diffused and defocused tagged areas the low level diameter is greater than the high level diameter. A user defined value for the maximal low level area filter defines the percentage of the high level area that is acceptable for the low level regions. This value sets the level of the acceptable sharpness of the tagged regions. A minimal area filter is applied in order to remove noise being detected as a tagged region. The minimal area filter defines the lower limit to the diameter of the tagged regions in pixels. Any region detected below this limit will be ignored. Some unrealistic values for the tagged region diameter are filtered out by defining the minimum and maximum range of the height and width of the detected regions. Typical values used for the various filters in the recognition filter dialogue are reported in Appendix C-1.

The velocity of the tagged regions is calculated based on the window size, the diameter deviation and the initial shift. Prior to velocity calculation, the position and size of each tagged region is calculated and stored. The algorithm determines the pair of particles depending on the size and the allowed shift. The detection window, whose position is dictated by the initial shift, only accepts the particles which are within the defined diameter deviation range. The velocity field is finally calculated in two passes with a decreasing window size. A 0.1 x 0.1 mm window size is used for the first pass which reduces to 0.05 x 0.05 mm for the second pass. The sizing and velocity information of the tagged regions is displayed as an overlay on the raw images. Also a text file containing the location, size and velocity of the tagged region is stored within the project folder. The parameters used in the velocity calculation are mentioned in Appendix C-1. The display settings of the overlay information can be modified using the overlay setting sub menu in the output selection dialogue.

4.4 Steps involved in the µ-PIV/PTV scheme

A set of operations, different from photobleaching MTV, is involved in the image acquisition for the µ-PIV scheme. An image processing scheme specific to µ-PIV is involved in the processing of the acquired images in order to reveal particle velocities. The various steps involved in the image acquisition and processing along with typical parameter values are described in the following two sections.

4.4.1 Image acquisition

The steps involved in the image acquisition for micro-PIV are similar to the ones utilized for the acquisition of images for photobleaching MTV as shown in Figure 4.20. These steps are also divided into three main groups including camera settings, trigger settings and image acquisition. Only the trigger settings are different from those used for photobleaching MTV.

The description of the camera settings is the same as for the photobleaching MTV approach. However, typical values used may be different. The laser scanner is not utilized for micro PIV measurements and the laser is used in an epifluorescent configuration. Only one channels, Dev1/ao2, is selected because only the camera needs a trigger signal. Since only one channel is selected hence the controls within the *X* position enclosure will be used to set the camera trigger frequency and duty cycle. A square waveform is selected for the camera trigger signal. The laser operates in the continuous mode for µ-PIV measurements. Therefore, CW is selected from the laser pulse enclosure which operates the laser in continuous mode. Similar to the camera settings the description of the image acquisition steps involved for µPIV is the same as for the photobleaching MTV approach. A video file in AVI format containing the particle fluorescence images is saved for processing.

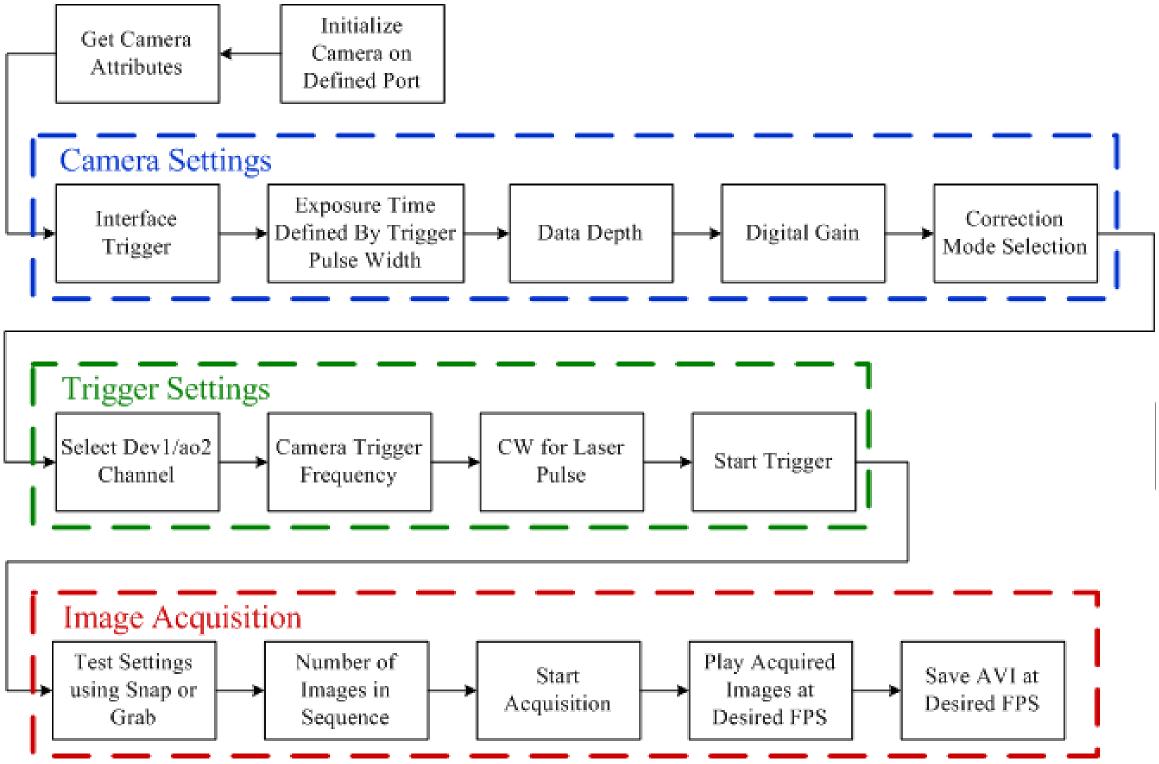


Figure 4.20: Flow chart indicating the sequence of steps involved in image acquisition for the µPIVtechnique.

4.4.2 Image processing

The sequence of images is processed using the particle image velocimetry module of commercially available software (LaVision, DaVis 7.4). A flow chart elaborating the sequence of operation for the image processing for particle image velocimetry is presented in Figure 4.21. The first two steps performed are the same as explained in section 4.3.1. All the processing steps mention hereafter are performed in the batch processing mode.

The µ-PIV images typically have variable background intensity due to the out of focus particles. In order to remove the intensity fluctuations a sliding minimum with a scale length of 5 pixels is subtracted from the images. A sliding minimum over time over 3 images distinguishes the particles against the uniform intensity of the background. The sliding minimum is calculated from 3 consecutive images and subtracted from the current source image. The signal intensity of the particles is increased by multiplying the source images with a constant factor of 5. This multiplication increases the intensity count of the images by a factor of 5.

After pre-processing, the image set is ready for vector field computation. A multi-pass iteration scheme with a decreasing overlapping window is used for velocity calculations for the micro particle image velocimetry images. Since, a time series of images is used for the vector calculation the cross-correlation mode is automatically selected. The cross-correlation calculates the velocity from two consecutive singly exposed images with a time difference between them. The correlation algorithm takes the first exposure as frame 0 and the second exposure as frame 1. Both the frames are divided into interrogation windows of specified size and cross-correlation is performed only on the intensities inside the corresponding windows of both images.

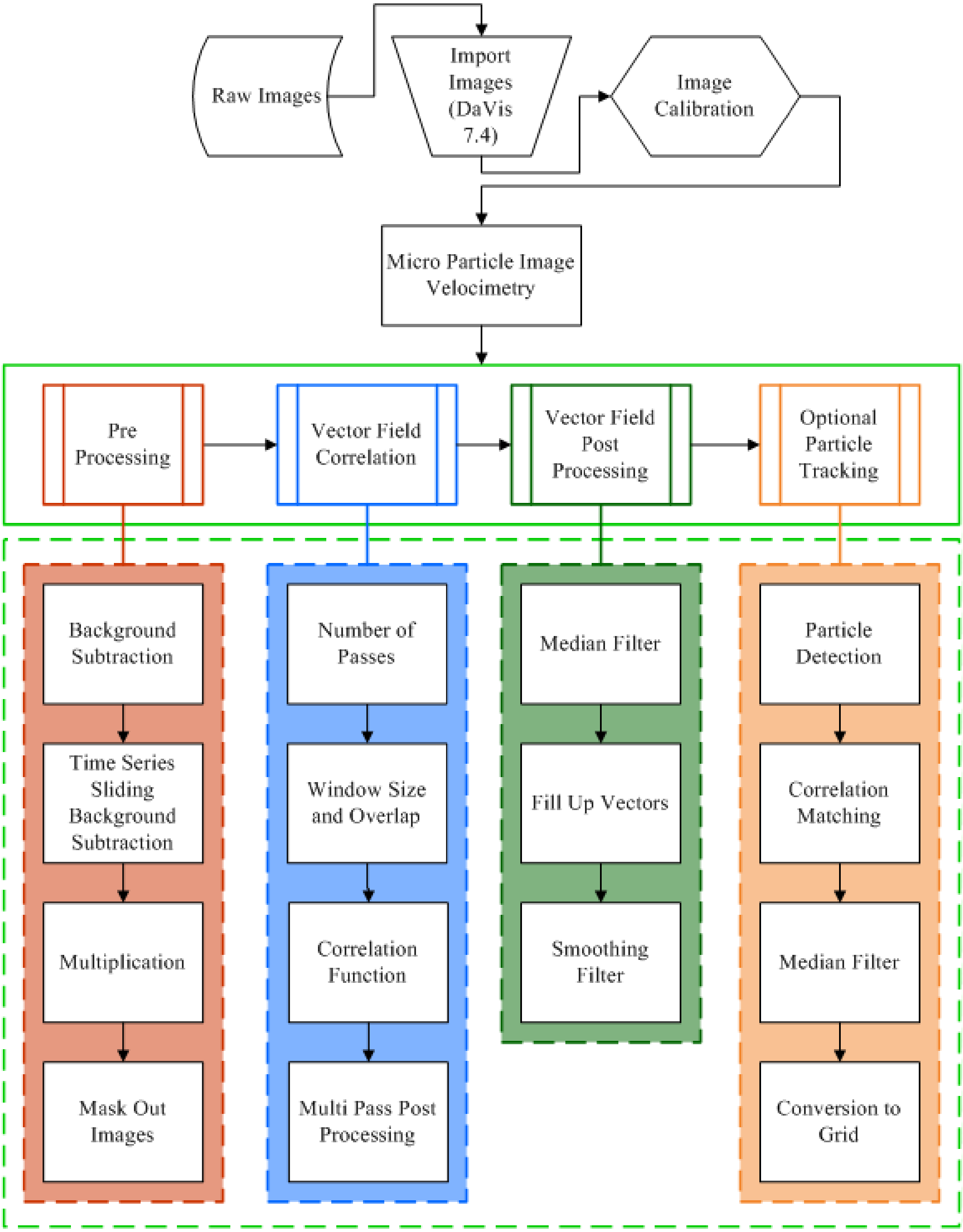


Figure 4.21: A flow chart of the sequence of image processing operations for micro particle image velocimetry..

The number of passes, the window size and the weighting function are specified for the interrogation of the images. In a multi-pass scheme a decreasing window size, the number of passes along with the interrogation window overlap percentage is defined. The height and width of the interrogation window is defined by the window size. A 64 x 64 pixel window with a 75% overlap with a 1:1 circular Gaussian weighting function is used for the first pass. A circular Gaussian weighting function is used because it ensures the highest accuracy. If a square weighting function is used it will result in an increase in the speed of processing at the expense of decreased accuracy. A 32 x 32 pixel window with 75% overlap and 1:1 circular Gaussian weighting function is used for the two second passes. The Gaussian weighting function reduces the speed of vector calculation with an increase in accuracy. The overlap between neighbouring interrogation windows is defined by the overlap resulting an increase in the grid size. The spatial resolution and the accuracy of the vector field are improved by using a multi-pass scheme with a decreasing window size. An adaptive window shift based on the vector calculation of the first pass is used for the second pass with the decreasing window size. This scheme allows for the use of a smaller window size than possible without an adaptive window shift. The image correction is not used in the processing of image.

A 'standard I1*I2 (via FFT, no zero-padding)' option is used as the correlation function selection for the vector calculation. This scheme makes use of a standard cyclic FFT based algorithm and is about a factor 5 faster than the 'normalized (I1-I1avg)/rms' scheme. This processing scheme has some inherent disadvantages which includes the bias towards lower displacements as a result of the introduction of a weighting of the correlation coefficient towards lower displacements. The calculated displacement is also shifted towards the next integer number due to a 'peak locking' bias effect. However, when the approximate vector length is know the weighting of the coefficients can be used to an advantage by defining a window shift corresponding to the estimate of the vector length. Reliable results have been obtained using the fast standard FFT-correlation with the adaptive multi pass scheme with three iterations.

A strongly remove and iteratively replace median filter is used in combination with an interpolation based fill up empty spaces option is used for the vector post processing step. A smoothing filter is applied in order to align the vectors to defined grid spacing.

Finally an optional particle tracking step can be applied to get vectors for individual particles. The vectors obtained as a result of the µPIV processing are taken as reference for the particle tracking calculations. The size range of the particles along with a correlation window size is defined in the particle detection and the correlation matching steps of the PTV scheme. The median filter is used to post process the vector field obtained by the particle tracking scheme. The obtained vectors are converted to a desired grid to get a uniform distribution of the vectors.

4.4.3 Caged fluorescence MTV

The caged fluorescence MTV can also be performed using the MTV setup configuration. The image acquisition and processing will incorporate similar steps, with different parameter values, for the caged fluorescence approach.

4.5 Conclusion

A sequential coordinated operation of all the hardware components is required for the successful operation of the scanning two dimensional MTV system both for the photobleaching approach and the caged fluorescence scheme. A graphical user interface is developed in house to achieve this controlled operation of the designed system. The controls of the designed GUI enable the user to change various parameters of the components to achieve the best raw images for each individual scheme. The same setup is used to perform MTV as well as µPIV measurements by varying the parameters using the developed GUI.

Different parameters of the designed setup are used to obtain raw images for the MTV and µ-PIV measurements. These images are processed to reveal the velocity information using specific parameters of commercially available software (DaVis 7.4). The raw images are pre-process to prepare them for vector processing. These pre-processed images are used to obtain the vector field utilizing the defined parameters of the software.

CHAPTER 5

COMPARISON OF μPIV AND MTV

5.1 Introduction

This chapter presents the results of the experiments conducted using the developed scanning two dimensional MTV system for micorscale flows. The experiments are performed in small regions of the flow cells described in section 3.8. The qualitative visualization of the flow through the flow cells is presented which illustrate the utility of the developed MTV approach as a semi-quantitative visualization approach. The flexibility of the laser scanning system in producing any desired pattern within the FOV is demonstrated. The flow measurements using the developed system based on both photobleaching MTV and μ-PIV/PTV are performed in order to provide a proof of concept and to evaluate the prospects of both the designed measurement setup and the measurements.

5.2 System performance

The various features of the designed two dimensional scanning MTV system can be used to define the performance of the system. The novelty of the two dimensional system is the use of a laser scanner instead of using a mask. The range of the laser scanning system within the FOV, the size and the number of tagged regions and the diverse tagged patterns can be used to determine the performance of the system.

5.2.1 Scanner range

The maneuverability of the write laser within the FOV is limited by the range of the laser scanner. The laser scanner is capable of moving the write laser to any desired location within the area under observation. The range of the laser scanner is demonstrated in Figure 5.1 in which the write laser is moved to 13 arbitrary locations within the FOV using the laser scanner.

The dark dots shown in Figure 5.1 depict the location of the write laser at a particular instant in time. The position of the write laser dots shows the maneuverability of the laser scanner system. The location of these dots demonstrates that the laser scanner is not only capable of tagging regions in the central region of the flow but also capable of tagging regions close to the boundary of the FOV. The write laser is moved to each location an image is captured. The write laser can easily be moved to any location at the points shown in the figure or any other arbitrary location between the shown points.

The write laser is focused onto a planer surface placed at the focal plane of the imaging system. The images are captured without the fluorescence filter or the high pass filter because the reflection of the write laser from the surface is captured instead of fluorescence. The final image is obtained by processing individual images and merging the images together to get an image with all the points. The area imaged by the imaging system shown in the figure is 1.5 mm x 1.5 mm. The laser scanner covers the entire range of the area imaged by the imaging system.

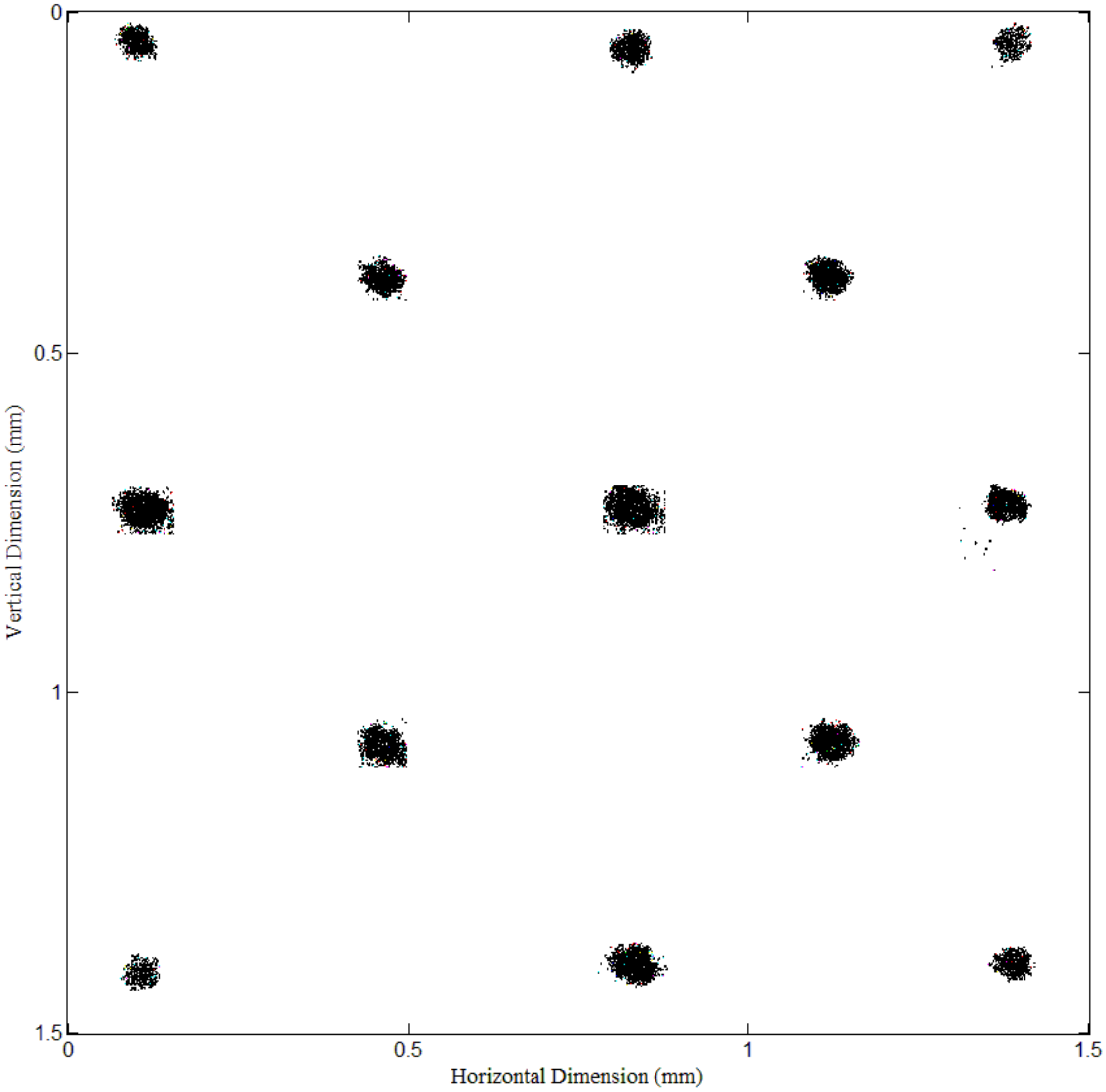


Figure 5.1: Demonstration of the maneuverability and the range of the laser scanner within the field of view.

5.2.2 Size of the tagged region

The size of the individual tagged dots and the density of the tagged dots within a pattern are some of the important parameter that needs to be considered in the analysis of the scanning MTV system. The size of the tagged region needs to be as small as possible in order to accommodate a higher density of tagged dots in a generated pattern. The density of the tagged region defines the number of available locations for instantaneous velocity measurements. A higher density of the tagged region will cover a greater number of interrogation points of the flow field which will enable the generation of an averaged velocity field from the instantaneous velocity measurements.

The tagged dots tend to disappear as a result of diffusion which can be observed in Figure 5.3. The higher density of the tagged region will result in tagging a successive region before the former tagged region disappears. This will allow a continuous instantaneous velocity measurement.

5.2.3 Tagging patterns

The scanning MTV measurement system is capable of generating any desired pattern for flow measurements. The tagging pattern is generated by defining the sequence of input voltage to the laser scanner system as described in Chapter 4. Therefore, any tagging pattern is easily generated by simply changing the voltage values in the input data files.

The capability of generating any desired pattern provides the flexibility to conduct non-intrusive flow measurements at any point in the flow within the FOV. In order to analyze the flow properties close to the wall, the tagging pattern can be generated close to the channel walls. Similarly if the flow behavior around a cylinder or behind the wake of a miniature obstacle is analyzed the tagging pattern will be generated along the surface where the measurements are desired. The flow profile can also be analyzed by simply generating a tagging pattern in the form of a continuous line perpendicular to the axial flow direction. These continuous lines can be generated at multiple locations of the channel to analyze the evolution of the flow profile at several locations within the FOV.

The tagging pattern can also be changed progressively in order to cover the entire flow region. The progressions should be such that each tagged region moves by a certain value to cover the regions missed by the previous tagging pattern. Finally the accumulation of the velocity at each individual tagged region will give an averaged velocity measurement.

5.3 Flow measurements

Flow measurements are carried out using the designed system. The measurement system is flexible to accommodate both μ-PIV/PTV measurements and scanning MTV measurements based on either caged fluorescence or photobleaching. The flow cells indicated in section 3.8 are used for μ-PIV and scanning MTV measurements. The measurements are carried out in a small section of the flow cells. The flow used for the experiments is a pressure driven flow through the flow cells. However, flow measurements using other flow mechanism such as electokinetic flows can also be performed using the developed system.

5.3.1 Scanning MTV measurements

The scanning MTV measurements based on photobleaching are performed in the flow cell consisting of the plastic cut out serpentine channel sandwiched between two standard glass slides described in section 3.8. This channel is used for scanning MTV measurements because of the flexibility of illumination from both sides. Optical access from both sides is required for scanning MTV measurements based on photobleaching because a back illumination configuration is employed. The illumination and imaging are performed from the opposite sides.

The scanning MTV measurements based on photobleaching are performed utilizing a homogeneous fluorescein sodium salt (Uranine) solution. Uranine was mixed in distilled water to a concentration of 1 mM. The flow through the test section is achieved using the syringe pump described in section 3.9. The syringe pump controls the quantity of fluid pumped into the flow cell. A volume flow rate of 10 μl/min is used for the scanning MTV measurements. The section of the flow cell used for scanning MTV measurements is indicated on the image of the flow cell shown in Figure 5.2. The selection of the test section is arbitrary. The flow is directed from the flow cell inlet and passes through the serpentine geometry. The flow direction in the test section is from bottom to top. After the 90°turn the direction of the flow is from the right side to left side of the test section. A detailed description of the flow cell is given in section 3.8 and its exploded view is shown in Figure 3.13.

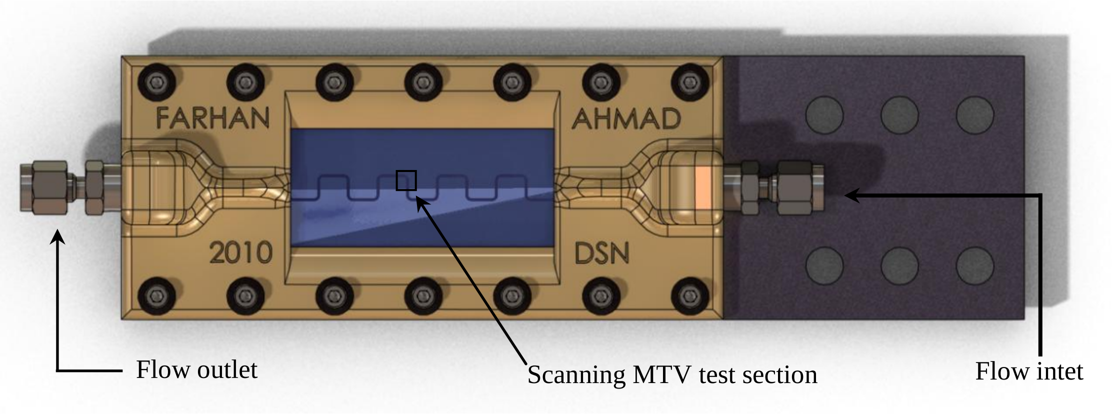


Figure 5.2: Test section for the scanning MTV measurements.

The images of the photobleached tagging sequence using the scanning MTV system are shown in Figure 5.3 (A-F). The geometry of the test section can be observed from these images. The boundary of the channel is not well defined. Therefore the images are masked out to specify the channel boundary for image processing. The size of the individual tagged region and the density of the tagging pattern can be observed from these images. The dark dots visible in the images are the photobleached tagged regions. The tagging pattern is in a reverse direction from the flow. The tagging pattern starts from the top left side. The write laser moves from the left towards the right and then moves from top to bottom. The images are frames extracted from a time series of images. The images shown in Figure 5.3 (A-F) are frames 19, 24, 27, 32, 35 and 40 extracted from a time series data of a sequence of scanning MTV images.

The tagged photobleached regions tend to disappear within several successive frames as a result of high rate of diffusion. However, the instantaneous velocity can be calculated using the frames where the photobleached regions have a significant contrast as compared to the background. The contrast of the images refers to the intensity difference between the dark tagged regions and the fluorescing background. The decrease in contrast of the tagged region can be attributed to diffusion. The usable duration of the tagged region is dependent on several parameters. These parameters include the flow velocity, the fluid temperature and the thickness of the flow cell. The practical utilizable duration of the tagged region decreases with an increase of any of the stated parameters. Hence, these parameters are inversely related to the functional duration of the tagged region.

A sequence of processed images is shown in Figure 5.4 (A-F). The calculated instantaneous velocity information and the diameter of the detected tagged region are displayed alongside each tagged region. The direction and magnitude of velocity are also demonstrated by the length and the direction of the velocity vector within the circle depicting the detected tagged region. The magnitude of velocity is very low that is why the displayed velocity vectors are not clearly visible in the images. The steps involved in the processing of images are explained in section 4.3.2. The parameters used in the processing are provided in C-1

The images of the scanning MTV measurements have a low contrast as discussed above. This low contrast can also be interpreted as a low signal-to-noise ratio (SNR). Measuring bright signal against a dark background is always more favorable than measuring a dark signal against a bright background. The low contrast or SNR can be attributed to several factors. One of the factors is the flow velocity. The velocity of the flow affects the fluorescence signal which leads to a varying contrast of the images as described in section 2.3.3.2.2. An increase in velocity results in an increase in the fluorescence signal whereas a decrease in flow velocity results in decreased fluorescence intensity. This variation in the fluorescence intensity leads to a low contrast or SNR.

Another cause of the low SNR is the use of a broad band LED (described in section 3.7.8) for bulk illumination instead of a laser. The LED used for bulk illumination does not provide a uniform intensity. The fluorescence intensity is directly related to the illumination intensity. The non uniform intensity gives rise to a varying contrast and

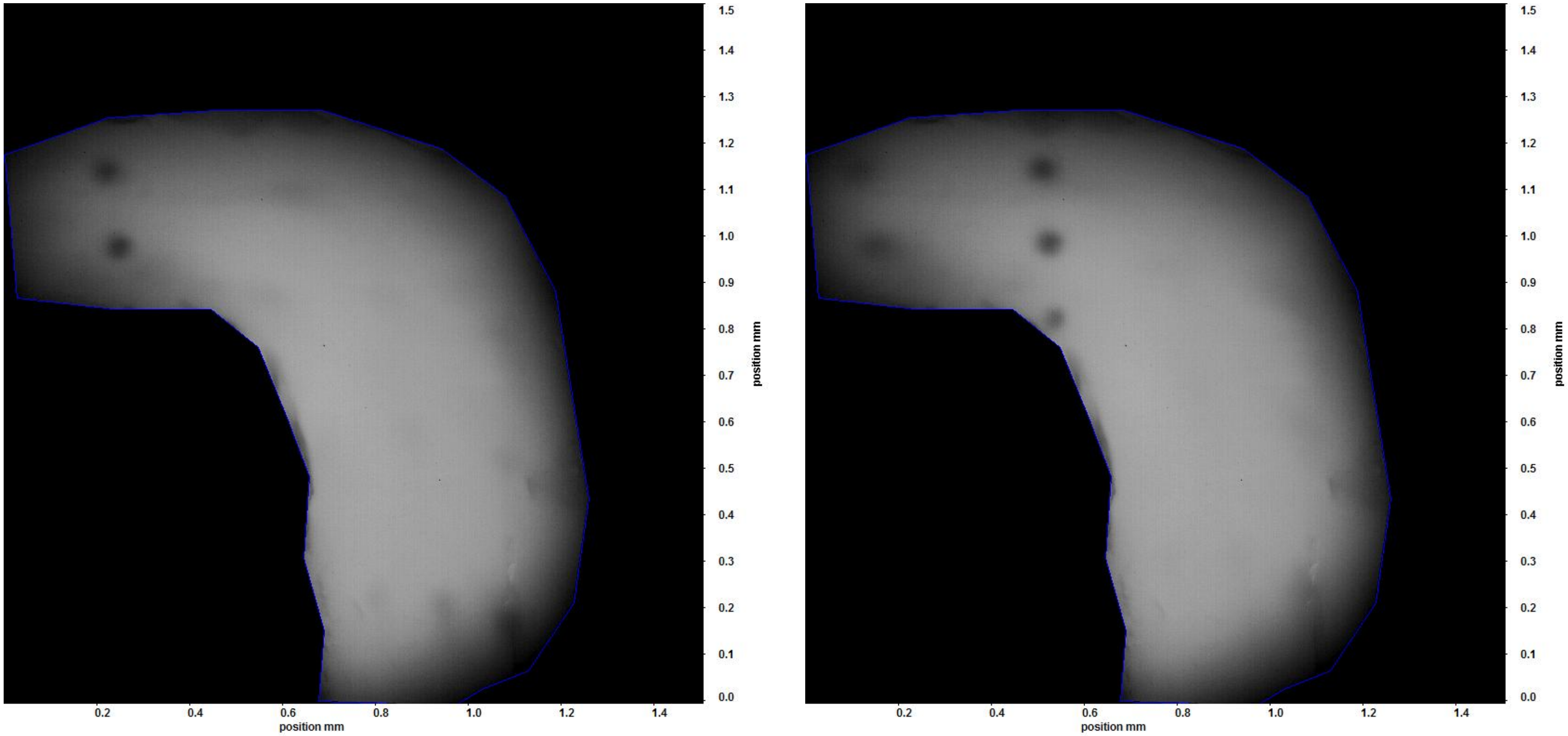
SNR. The use of a continuous light source capable of providing a uniform intensity illumination will significantly increase the contrast and SNR of the captured images.

The low SNR of the images results in the detection of tagged regions incorrectly. This may be referred to as the false detection of the tagged regions. The false detected tagged regions are indicated on the images in Figure 5.4. This false detection of the tagged regions can be prevented by improving the contrast of the captured images.

It can be observed from the images in Figure 5.4 that the calculated velocity is varying significantly. There are several factors that cause the variation in the calculated velocity of the tagged regions. Two major factors responsible for this variation in velocity are the use of a syringe pump and the boundary of the flow cell which is not well defined. The flow cell used for the experiments is custom made. The process of gluing the channel geometry to the glass slides is not yet perfected. The glue used for sealing the geometry does not wick properly to the channel boundary. This leads to a channel wall which is not well defined. The fluid used for the experiments flows outside of the expected channel boundary.

The syringe pump is used to generate a pressure driven flow through the channel. The syringe pump does not provide an accurate flow for fluid velocity measurements because of the sticky syringes which cause a variation in the flow velocity. The plunger of the syringe, driven by the syringe pump sticks to the wall of the syringe. The variation in the flow velocity leads to varying fluorescence signal which affects the SNR of the images.

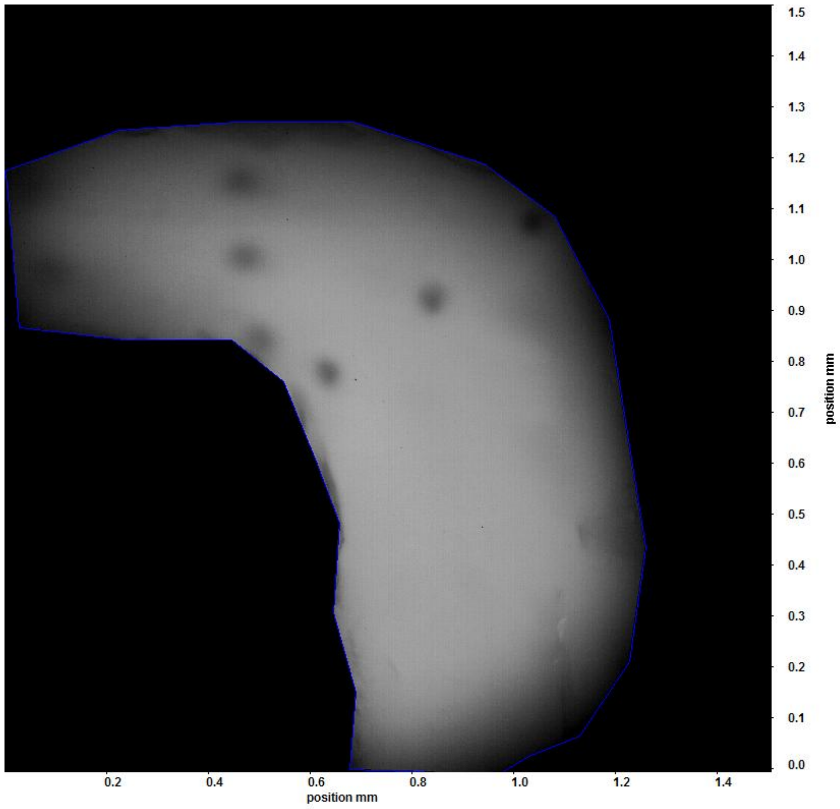
The processing of the images presents a challenge because the tagged regions tend to disappear rather quickly. The low contrast and SNR supplements the difficulty in processing the tagged images. The disappearing tagged regions make it difficult to tag a region and follow it indefinitely through the test section. The image processing is done using the images having a significant contrast of the tagged regions. The contrast of the images tends to decrease within 4 to 5 frames. These frames are used to process the tagged regions to reveal the instantaneous flow velocity.



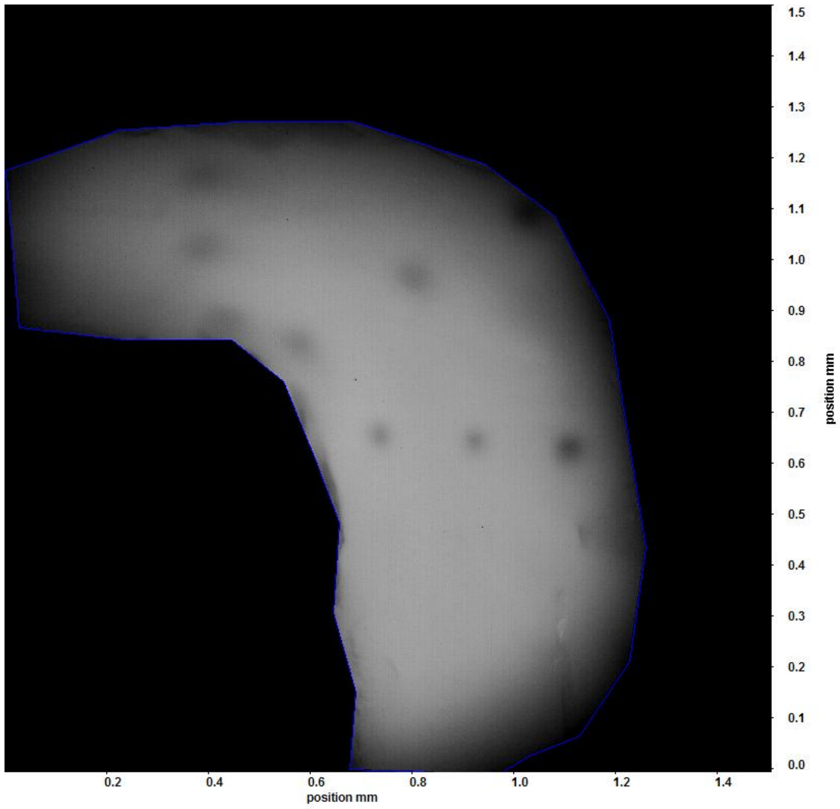
0.2
0.4
0.6
0.8
1.0
1.2
1.4
position mm
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5
position mm

(A)

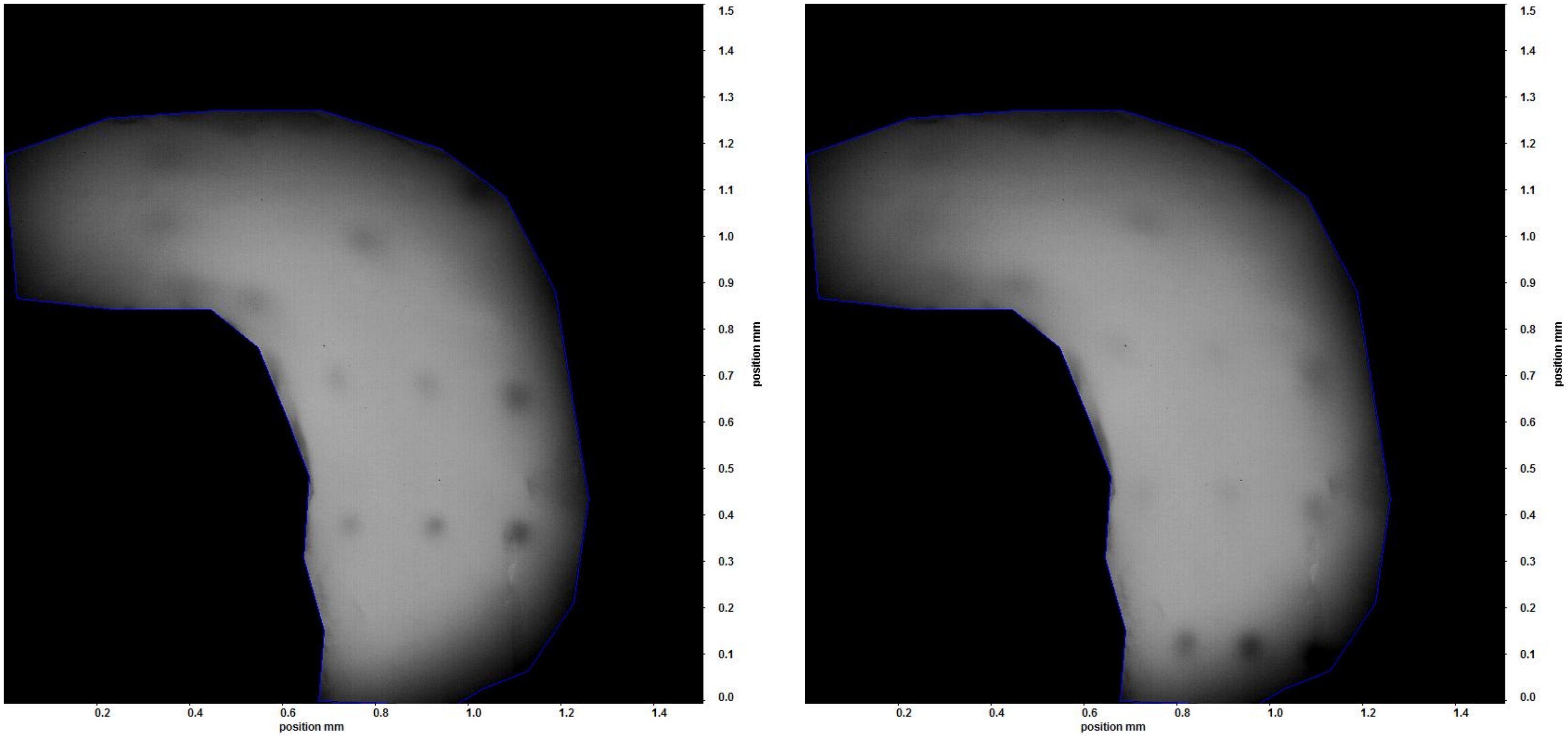
(B)


1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
position mm
0.2
0.4
0.6
0.8
1.0
1.2
1.4
position mm

(C)

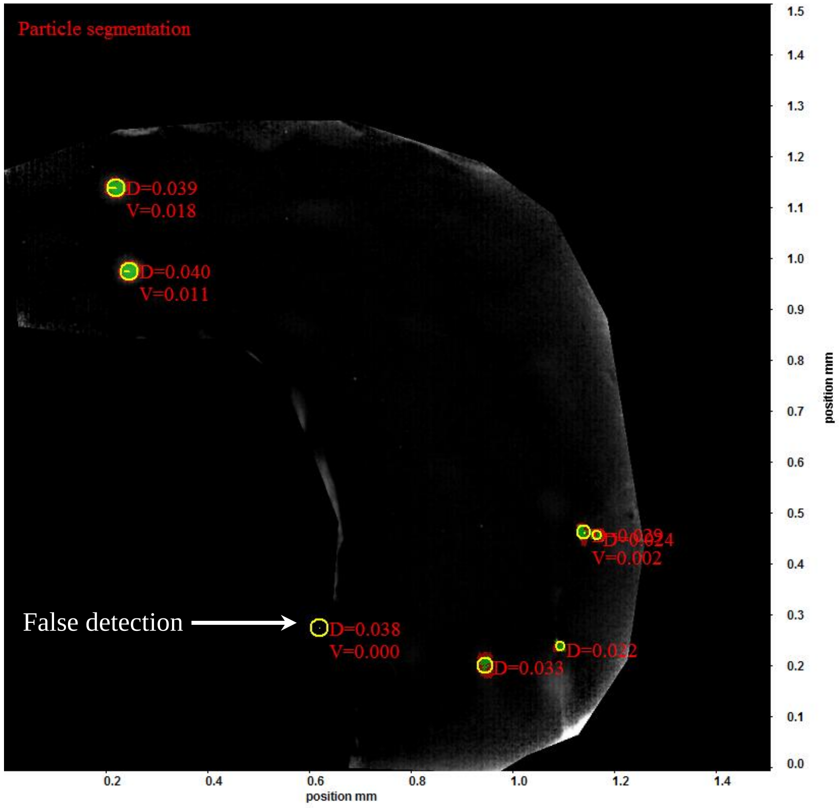

1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
position mm
0.2
0.4
0.6
0.8
1.0
1.2
1.4
position mm

(D)

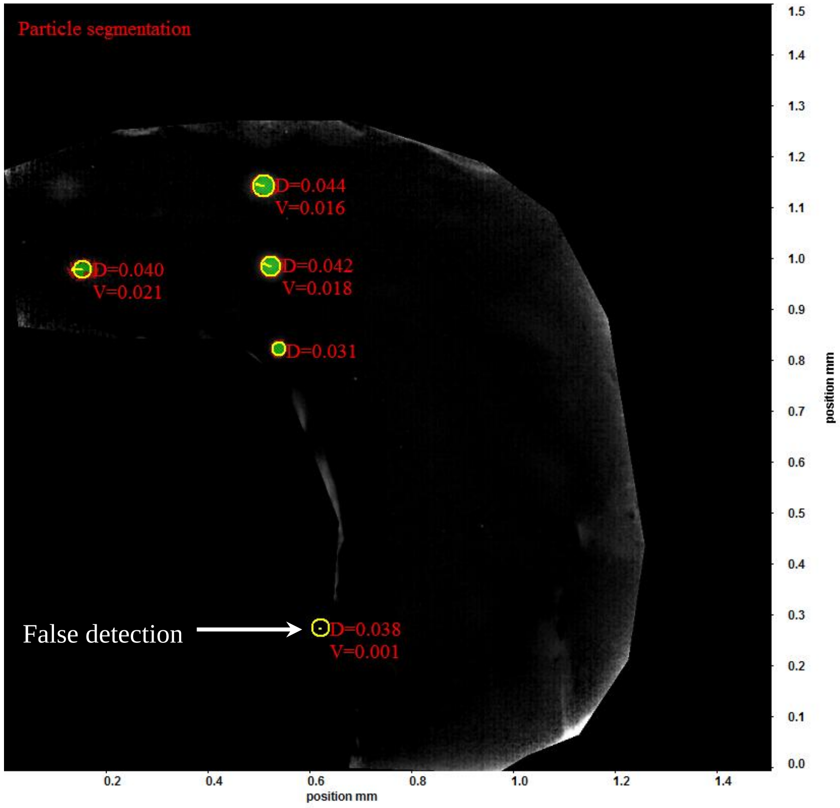


(E) **(F)**

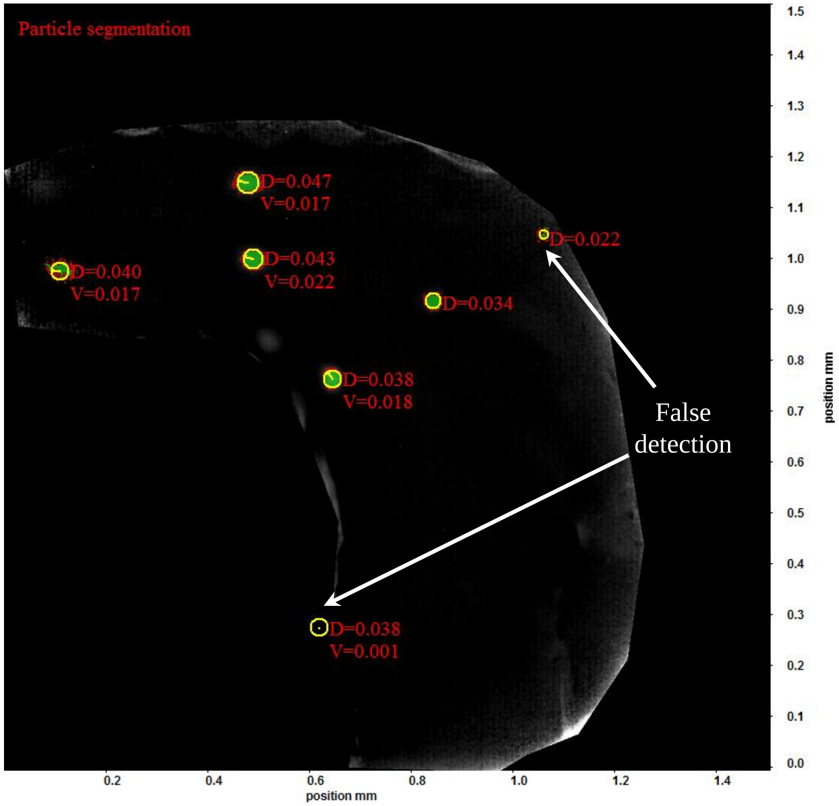
Figure 5.3: Image sequence of tagged region withing the flow using the scanning MTV system. The flow direction is from bottom to top and from right to left. The tagging sequence starts from the top left side and progresses to the right. It continues from the top to the bottom. The images are masked out in order to remove the undesired boundary. The images (A), (B), (C), (D), (E) and (F) are frames 19, 24, 27, 32, 35 and 40 from a times series data.


Particle segmentation
D=0.039
V=0.018
D=0.040
V=0.011
D=0.029
D=0.024
V=0.002
False detection
D=0.038
V=0.000
D=0.022
D=0.033
0.2
0.4
0.6
0.8
1.0
1.2
1.4
position mm
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
position mm

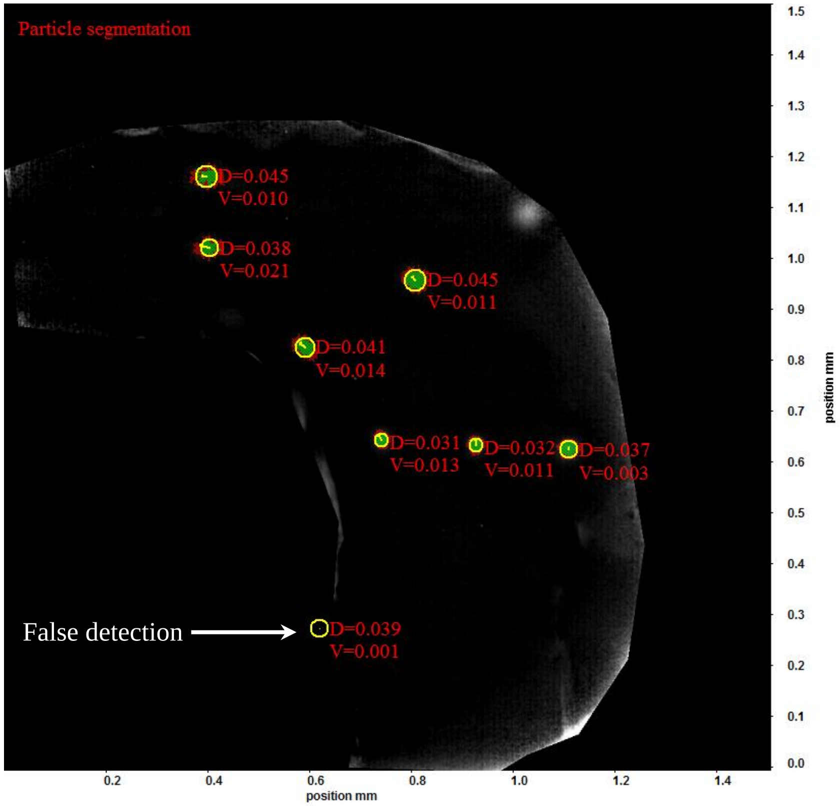
(A)


Particle segmentation
D=0.044
V=0.016
D=0.040
V=0.021
D=0.042
V=0.018
D=0.031
False detection
D=0.038
V=0.001
0.2
0.4
0.6
0.8
1.0
1.2
1.4
position mm
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
position mm

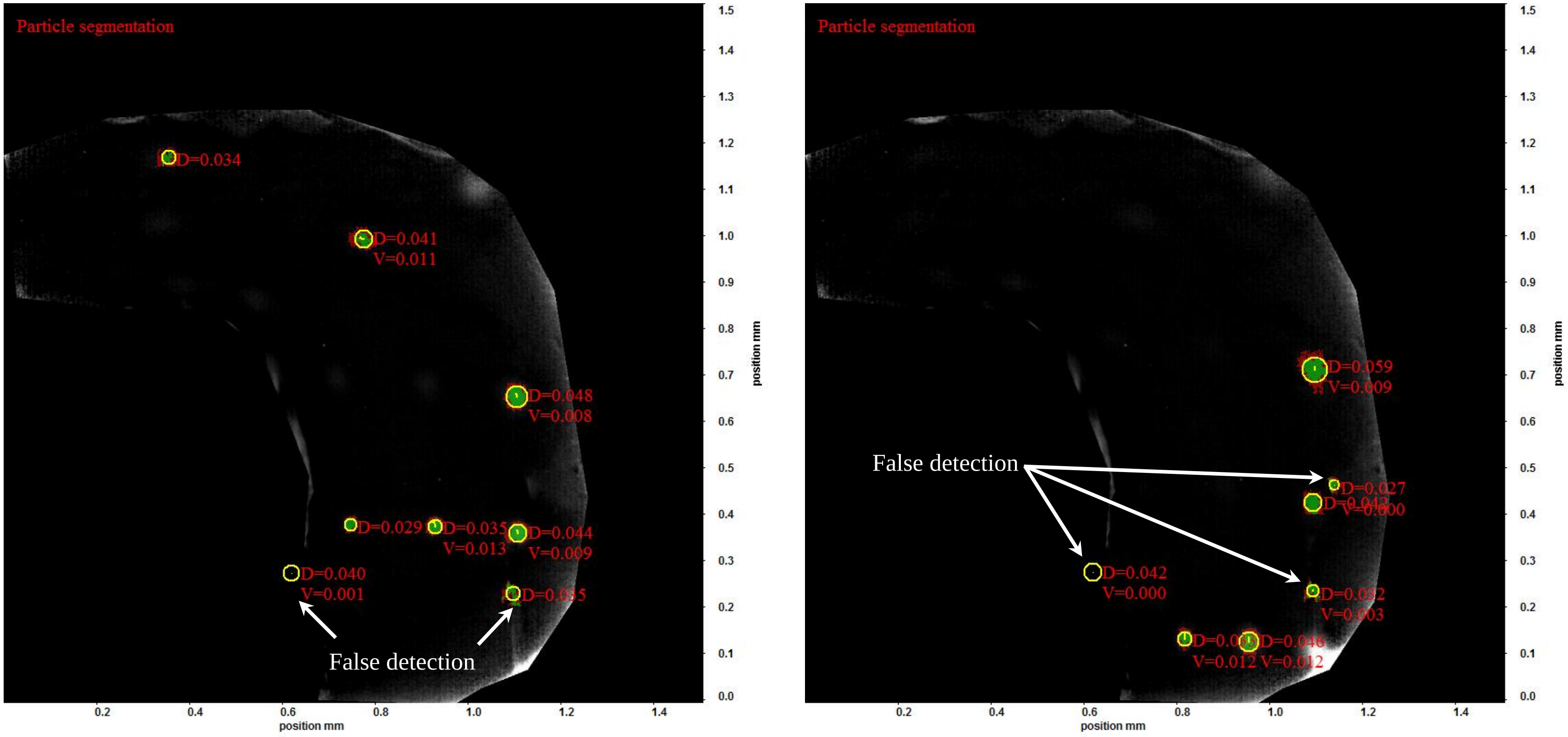
(B)


Particle segmentation
D=0.047
V=0.017
D=0.040
V=0.017
D=0.043
V=0.022
D=0.022
D=0.034
D=0.038
V=0.018
False
detection
D=0.038
V=0.001
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
position mm
0.2
0.4
0.6
0.8
1.0
1.2
1.4
position mm

(C)


Particle segmentation
D=0.045
V=0.010
D=0.038
V=0.021
D=0.045
V=0.011
D=0.041
V=0.014
D=0.031
V=0.013
D=0.032
V=0.011
D=0.037
V=0.003
False detection
D=0.039
V=0.001
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
position mm
0.2
0.4
0.6
0.8
1.0
1.2
1.4
position mm

(D)



(E) **(F)**

Figure 5.4: Processed scanning MTV measurement images indicating the evolution of velocity and the corresponding change in diameter of the tagged regions. The images (A), (B), (C), (D), (E) and (F) are frames 19, 24, 26, 31, 35 and 41 from a times series data of processed images.

5.3.2 μ-PIV/PTV measurements

The μ-PIV/PTV measurements are performed in a section of the each of the flow cells described in section 3.8. The micro particles described in section 3.6 are mixed homogenously in distilled water and are used for the μ-PIV/PTV measurements. A pressure driven flow of this mixture is generated using the syringe pump which is used for the experiments.

The test section used for μ-PIV/PTV measurements is indicated on the flow cell shown in Figure 5.5. This flow cell is suitable for μ-PIV measurements because an epifluorescence configuration is used. Optical access only from one side is acceptable because the illumination and the flow measurements are performed from the same side. This channel was designed and developed keeping in view the epifluorescent configuration used for μ-PIV/PTV measurements and scanning MTV measurements based on caged fluorescence. A detailed description of the flow cell has already been given in section 3.8 and its exploded view is shown in Figure 3.12. The direction of the flow is from the right side of the assembly to the left side. The test section is a section of a curve of the serpentine geometry embedded in the channel holder. The selection of the test section used for μ-PIV/PTV measurements is arbitrary.

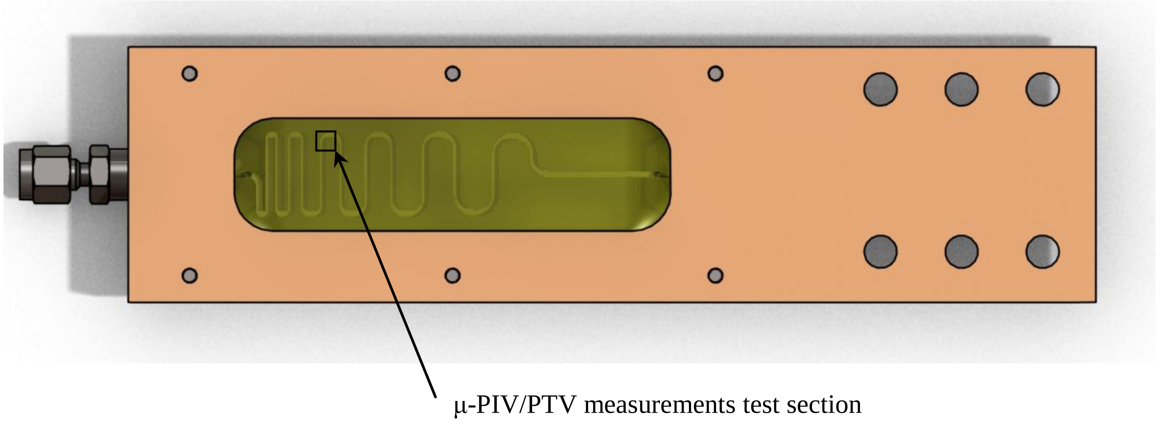


Figure 5.5: Test section for the first flow cell used for μ-PIV/PTV measurements.

The test section used for µ-PIV/PTV measurements is indicated on the flow cell shown in Figure 5.6. This flow cell was designed and developed specifically for scanning MTV measurements based on photobleaching because of a back illumination configuration which requires optical access from both sides. However, this flow cell can also be used for µ-PIV/PTV measurements utilizing an epifluorescent configuration. The direction of the flow is from the right side of the assembly to the left side as indicated in the figure. The selection of the test section used for µ-PIV/PTV measurements is arbitrary. A detailed description of the flow cell is given in section 3.8 and its exploded view is shown in Figure 3.13.

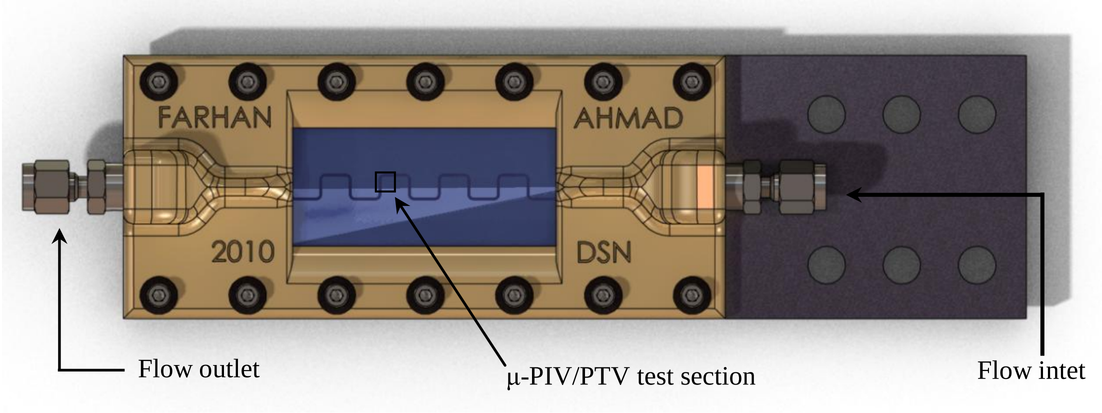


Figure 5.6: Test section for the second flow cell used for µ-PIV/PTV measurements.

The raw particle images captured for µ-PIV/PTV measurements for the test sections of the first and second flow cell test sections are shown in Figure 5.7 (A-D) and Figure 5.8 (A-D). The images shown in Figure 5.7 (A-D) are frames 55, 57, 66 and 74 extracted from a times series data of the particle seeded flow for the test section shown in Figure 5.5. Similarly the images shown in Figure 5.8 (A-D) are frames 11, 14, 32 and 44 extracted from a times series data of particle seeded flow for the test section shown in Figure 5.6.

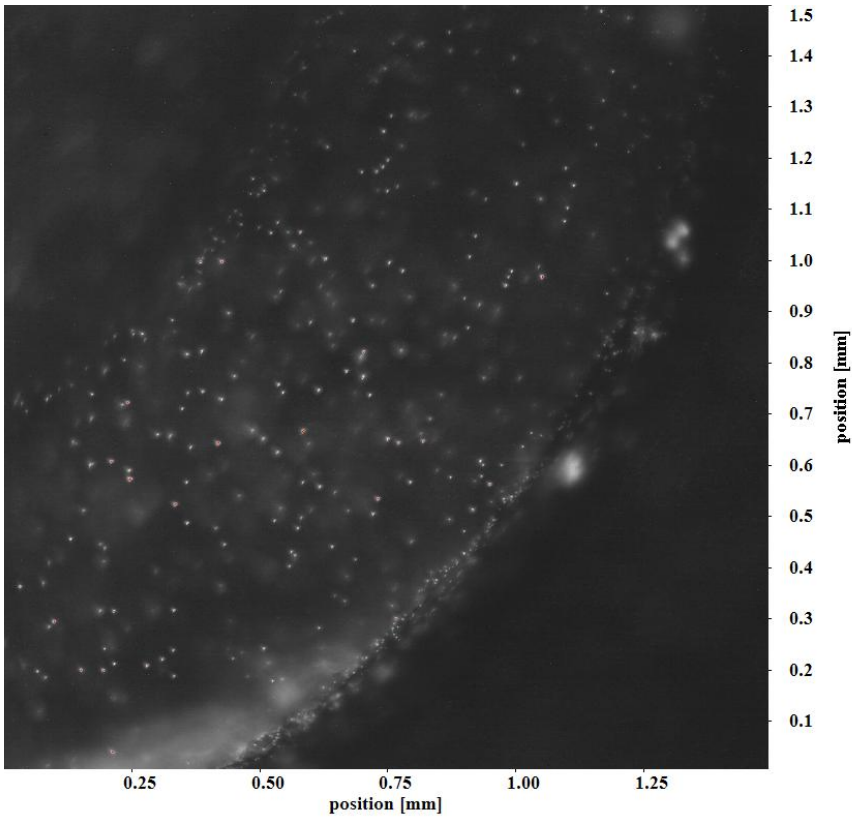
These images are shown to demonstrate the FOV, the contrast of the images, and the density of particles used for µ-PIV/PTV measurements. The geometry of the test section can be visualized from these images. The depth of the flow geometry of the first flow cell is greater than that of the second flow cell. The images in Figure 5.7 (A-D) show a greater background noise of out of focus particles as compared to the images shown in Figure 5.8 (A-D). The boundary of the channel geometry for the first flow cell is well defined. However, the boundary of the channel geometry is not well defined for the second flow cell. The particle mixed fluid seems to flow outside the presumed channel boundary as observed in Figure 5.8 (A-D).

The processed μ-PIV/PTV images for the test sections of both the flow cells are shown in Figure 5.10 (A-H). The steps involved in the processing of images are explained in section 4.3.2. The parameters used in the processing are provided in C-1

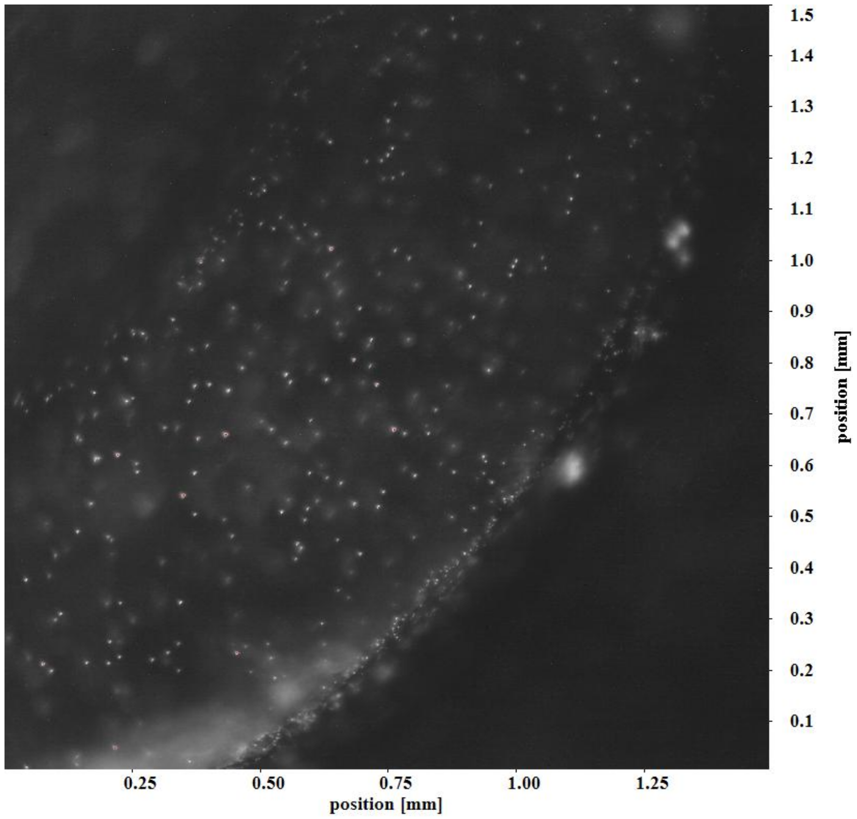
The velocity vectors display the magnitude and direction of the calculated velocity from the particle images. The magnitude of the velocity is demonstrated by the colors of the vectors. A color bar displayed alongside each image depicts the magnitude of velocity associated with the colors. The velocity field shown in the images is not the averaged velocity field. The motion of individual particles is tracked. Each vector displays the velocity of an individual particle.

The processed images shown in Figure 5.9 (A-D) display the two dimensional flow velocities ranging from 0 mm/sec to 1.5 mm/sec through the test section indicated in Figure 5.5. The processed images shown in Figure 5.10 (A-D) display the two dimensional flow velocities through the test section indicated in Figure 5.6. The velocity ranges from 0 μm/sec to 200 μm/sec. The particle density used in the experiments is not dense therefore particle tracking was possible. The different flow velocities captured using the designed system indicates that the system can be used for a varying range of flow velocities.

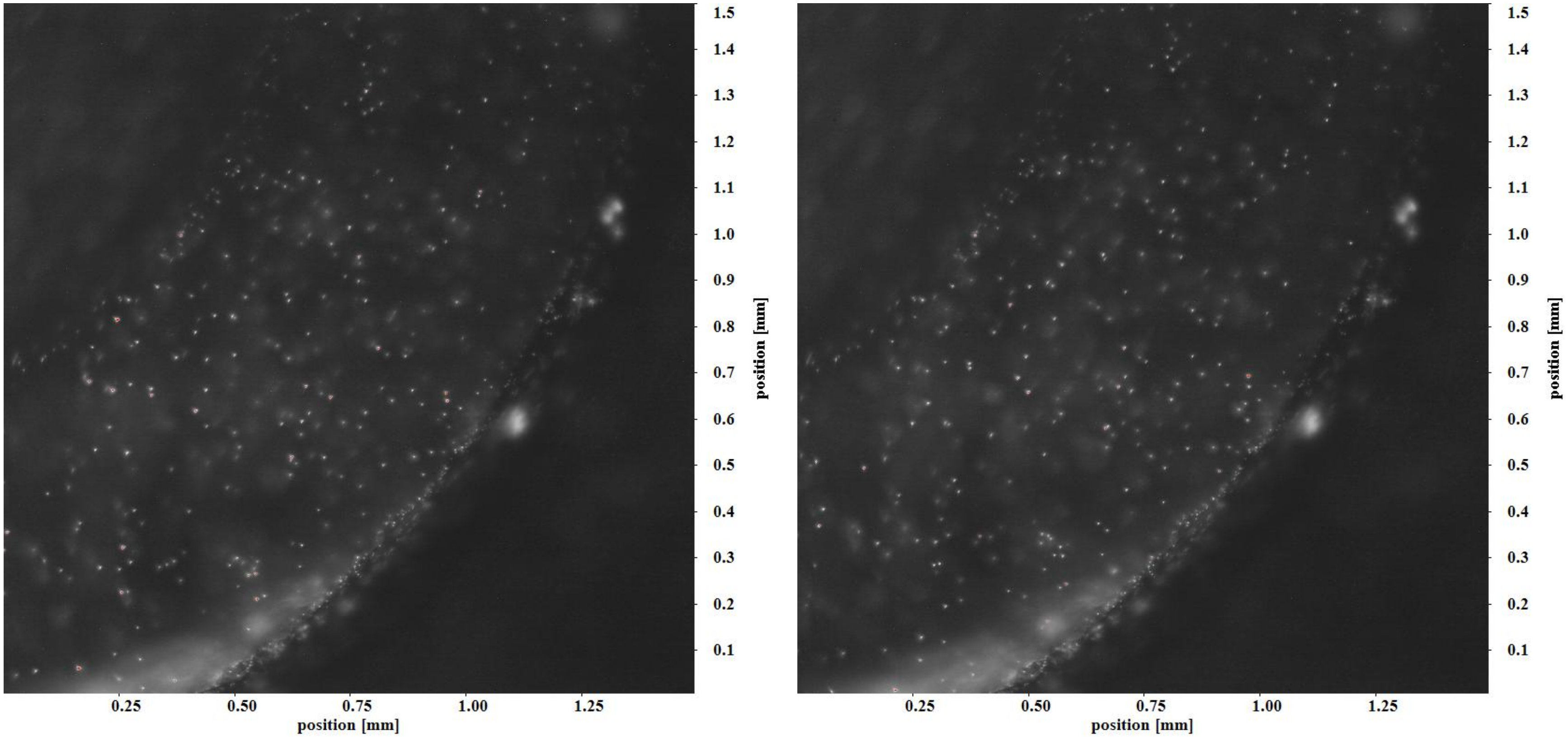
The velocities of the particles flowing outside the defined flow channel geometry have also been calculated as shown in Figure 5.10 (A-D). Flow measurements using μ-PIV/PTV are performed in this flow cell to display the flow outside the channel boundary. This flow effects the velocity measurements using the scanning MTV approach.


0.25
0.50
0.75
1.00
1.25
position [mm]
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
position [mm]

(A)

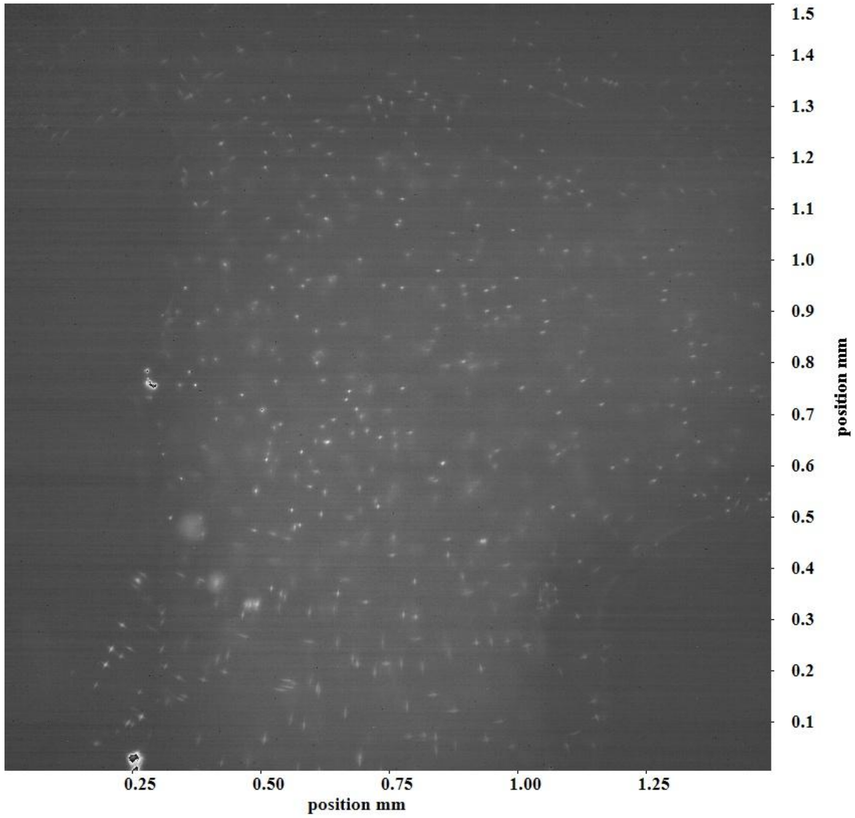

0.25
0.50
0.75
1.00
1.25
position [mm]
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
position [mm]

(B)

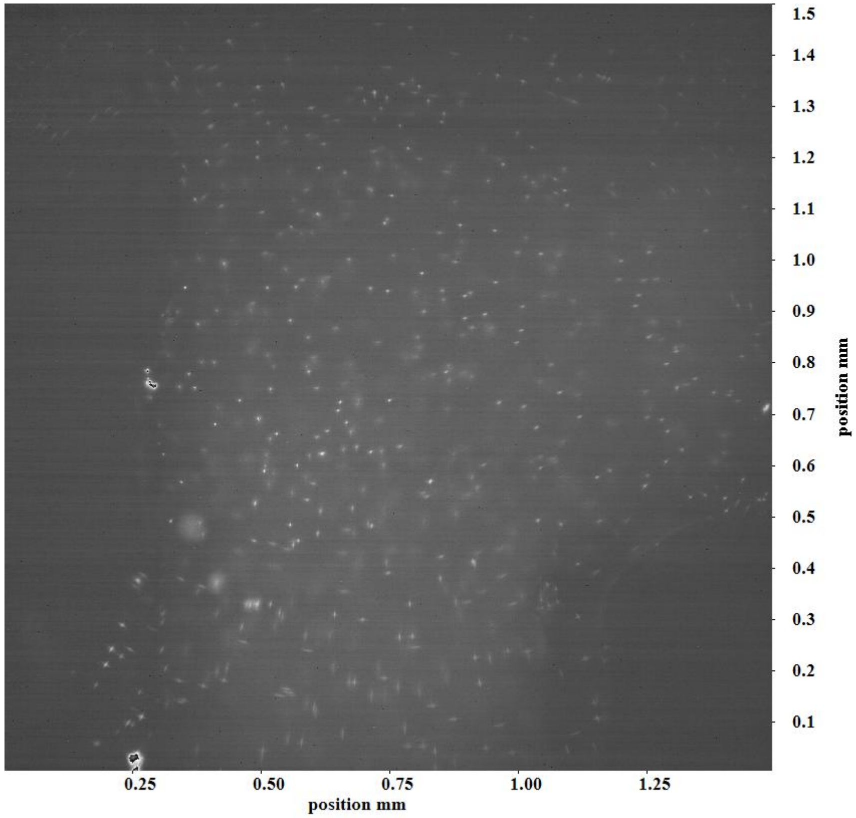


(C) (D)

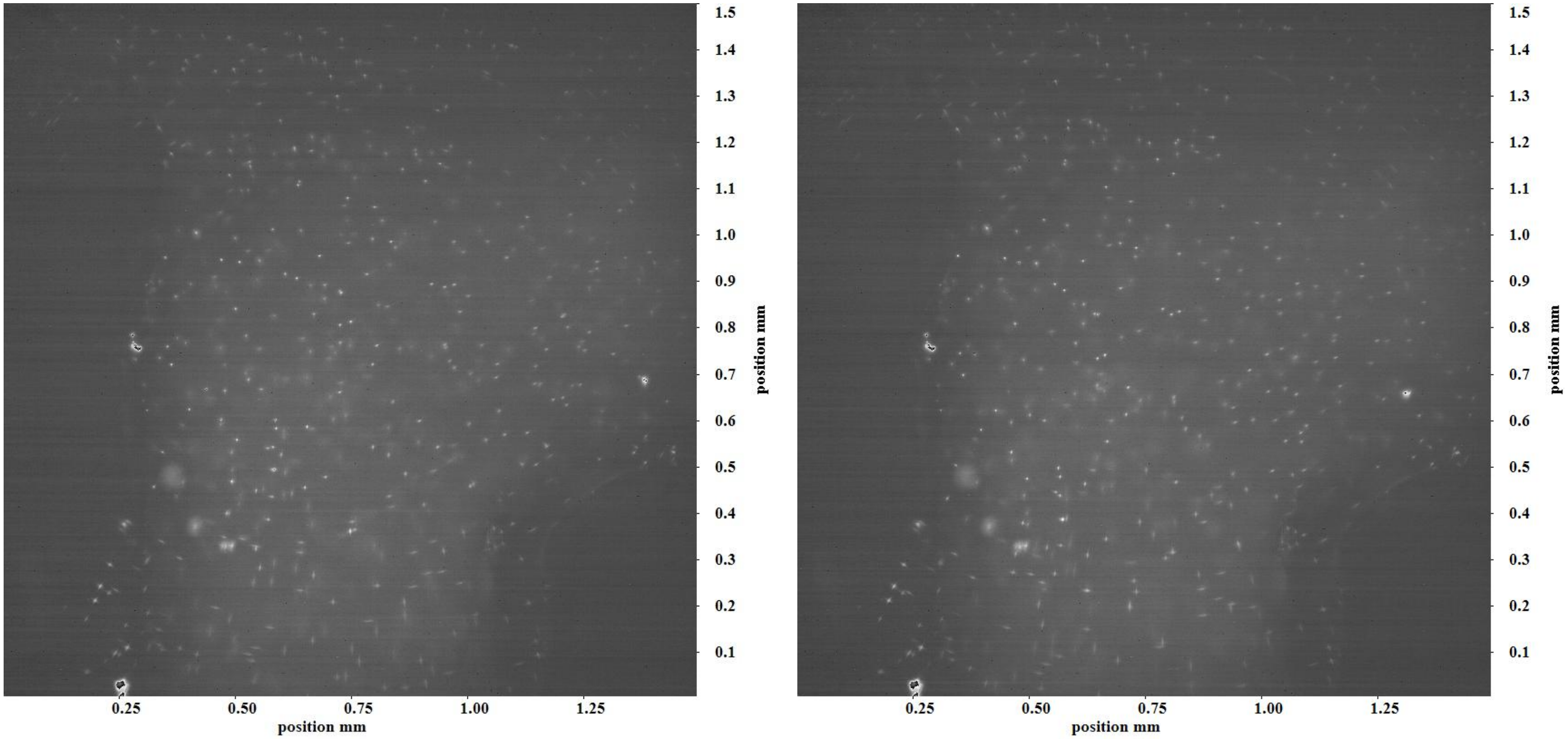
Figure 5.7: Raw image sequence of particle images used for μ-PIV/PTV measurements through the test section in Figure 5.5. Images (A), (B), (C), (D) are frames 55, 57, 66 and 74 respectively. The boundary of the test section and the particle density can be observed from these images. The flow direction is from the lower side to the upper side.


1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
position mm
0.25
0.50
0.75
1.00
1.25
position mm

(A)

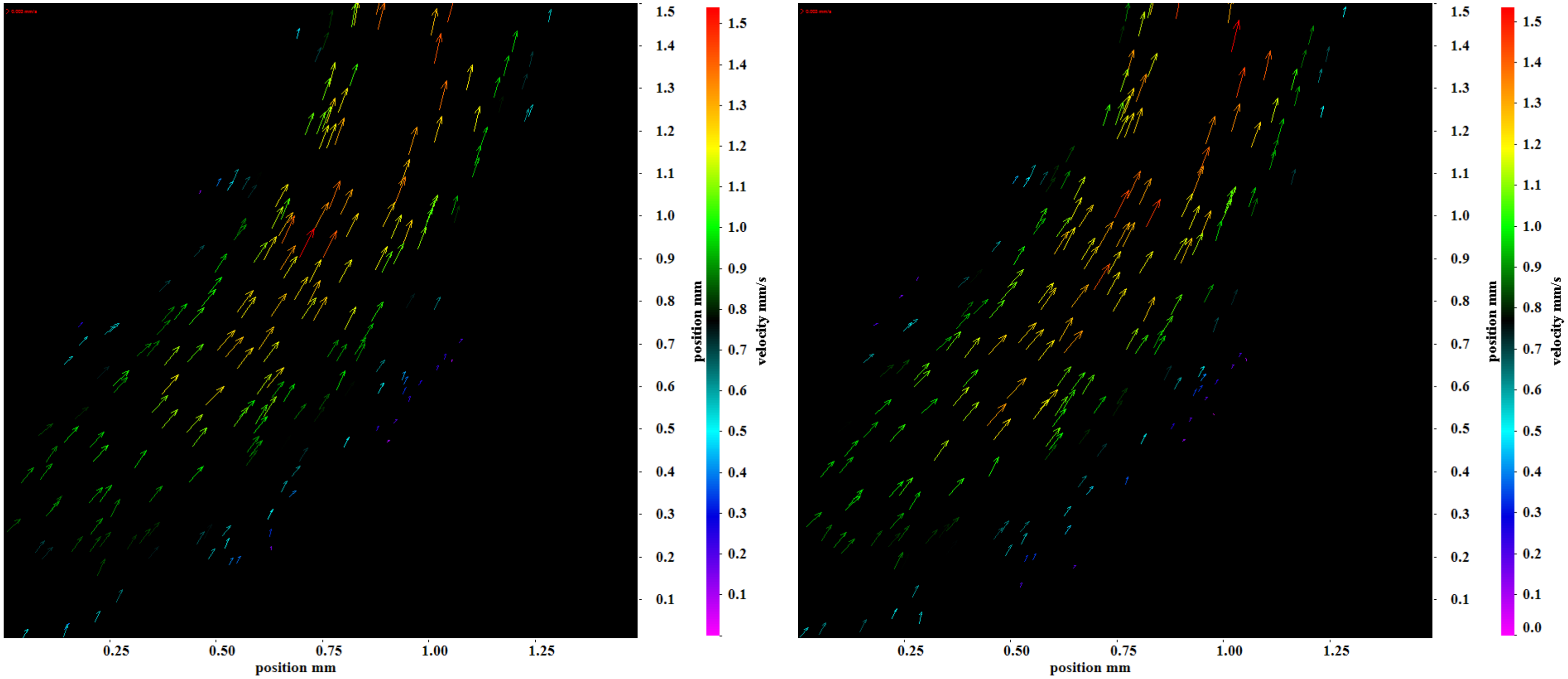

1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
position mm
0.25
0.50
0.75
1.00
1.25
position mm

(B)



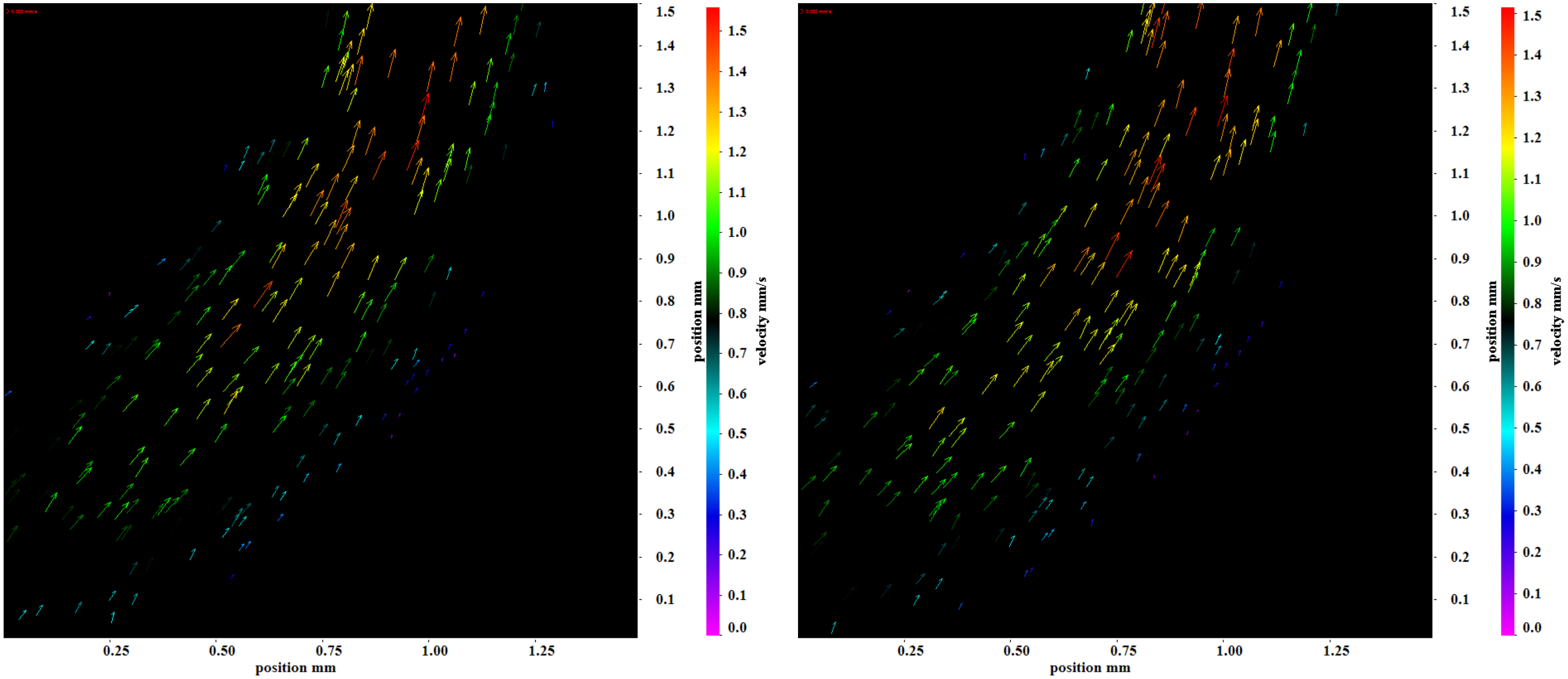
(C) (D)

Figure 5.8: Raw image sequence of particle images used for μ-PIV/PTV measurements through the test section shown in Figure 5.6. Images (A), (B), (C), (D) are frames 11, 14, 32 and 44 respectively. The test section channel boundary and the particle density can be observed from these images. The flow direction is from the right side to the left side and then from the top to the bottom. The fluid flowing out of the presumed boundary can be observed in the top left side of the images.


position mm
0.25
0.50
0.75
1.00
1.25
position mm
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5
velocity mm/s
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5

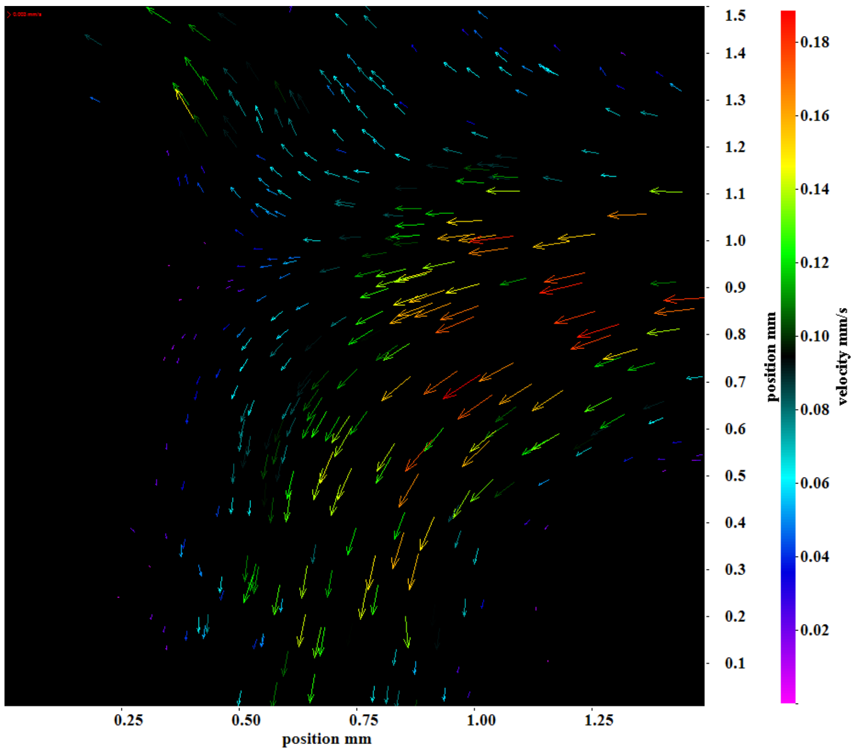
(A)

(B)

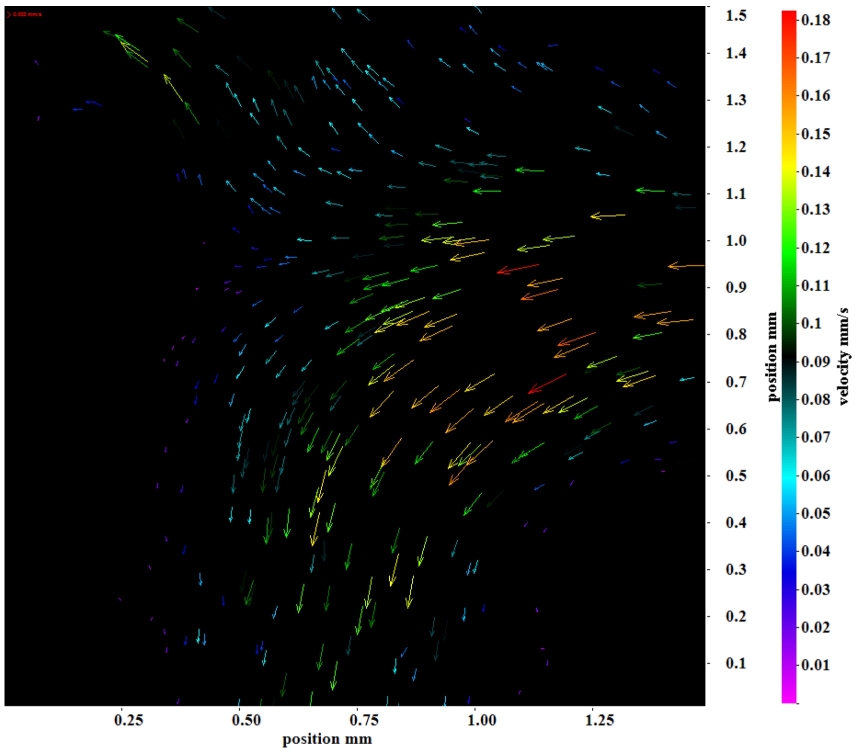


(C) **(D)**

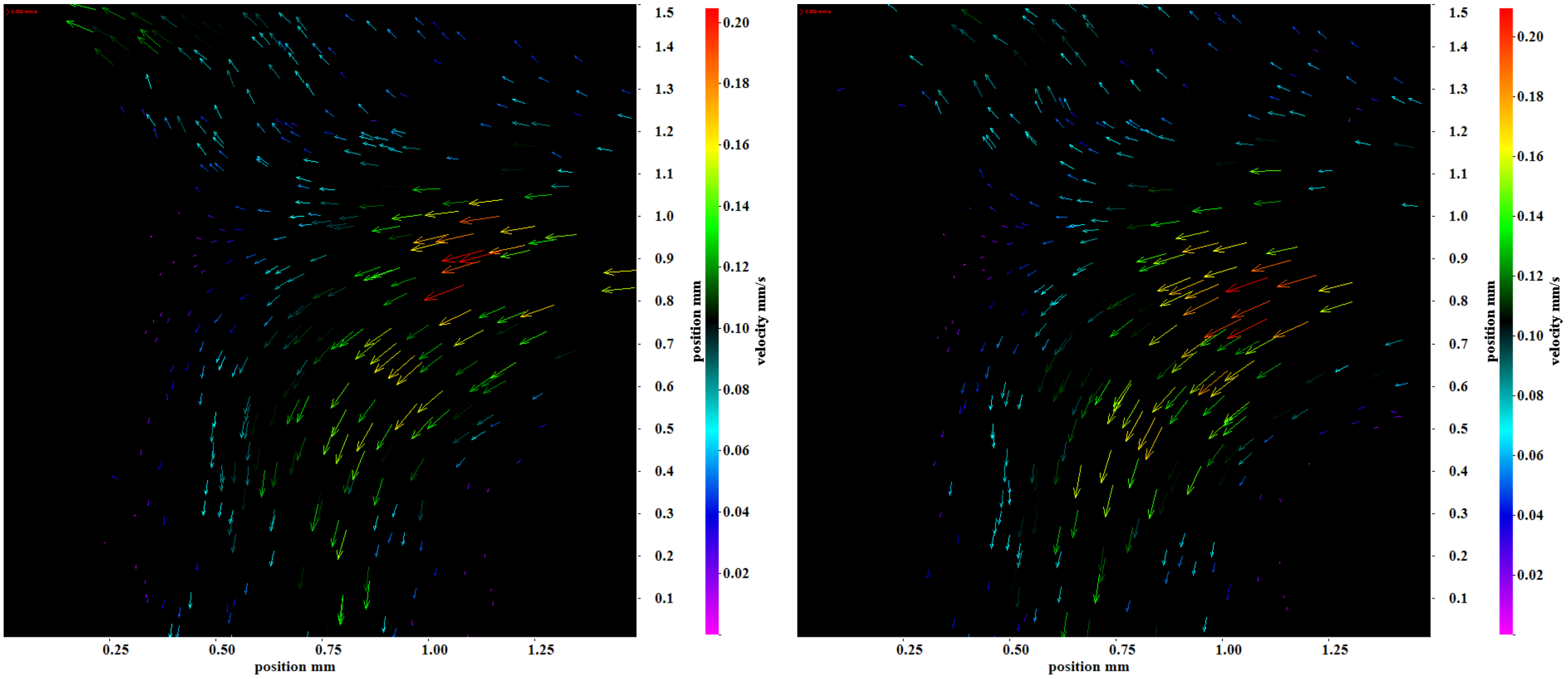
Figure 5.9: Processed vector images for μ-PIV/PTV measurement showing the evolution of velocity for particle images shown in Figure 5.7. The images (A), (B), (C), (D) are frames 55, 57, 66 and 74 from a times series data of processed μ-PIV/PTV images. The velocity ranges from 0 mm/sec to 1.5 mm/sec.


0.25
0.50
0.75
1.00
1.25
position mm
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
position mm
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
velocity mm/s

(A)


0.25
0.50
0.75
1.00
1.25
position mm
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
position mm
0.18
0.17
0.16
0.15
0.14
0.13
0.12
0.11
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
velocity mm/s

(B)



(C) (D)

Figure 5.10: Processed vector images for μ-PIV/PTV measurement showing the evolution of velocity for particle images shown in Figure 5.8. The images (A), (B), (C), (D) are frames 55, 57, 66 and 74 from a times series data of processed μ-PIV/PTV images. The velocity ranges from 0 μm/sec to 200 μm/sec. The flow velocity for the fluid flowing outside the presumed channel boundary can also be observed.

5.4 Conclusion

A proof of concept of the scanning MTV approach based on photobleaching has been successfully presented for microscale flows. The maneuverability of the laser scanner system has been demonstrated. The range of the laser scanner system covers the entire FOV. The write laser can be easily moved to any desired location within the area under observation. The particle master shadowgraphy module of a commercially available image processing software (DaVis 7.4) has been used to process the scanning MTV images to reveal the velocity information. The images of the tagged regions are effectively processed by the algorithm to reveal the two dimensional flow velocity information. The versatility of the designed imaging system has been demonstrated by successfully performing μ-PIV/PTV measurements for varying flow velocities.

The number of points where velocity is calculated using the scanning MTV approach is fewer than those for the μ-PIV/PTV measurements. The generation of a smaller tagged region will accommodate a denser tagging pattern. The contrast of the images is poor which leads to a poor SNR of the tagged images. This leads to a difficulty in processing the scanning MTV images. The images obtained for μ-PIV/PTV measurement have a good contrast which remains uniform throughout the measurements. The tagged regions tend to disappear quickly as a result of diffusion. Whereas the particles used for μ-PIV/PTV measurements do not disappear. The motion of a particle can be tracked throughout the test section geometry. The custom made flow cells, the use of syringe pump and the bulk illumination with a broad band LED are the cause of a non uniform or varying contrast of the scanning MTV images. Improvements addressing the issues discussed above with the scanning MTV system are proposed in the next chapter.

CHAPTER 6

FUTURE WORK

The proof of concept of the two dimensional scanning MTV system has been successfully presented. The imaging system has also been tested by capturing both photobleached images and particle images. The two dimensional scanning MTV system is still in the development phase. Some recommendations to improve the performance of this system are presented here.

6.1 Improved illumination

The back illumination using the broad band LED for the photobleaching approach is one of the reasons for the variation in the contrast of the images. The contrast of the images can be improved by using a continuous laser source for illumination instead of the LED. The continuous laser will provide uniform illumination throughout the FOV which will lead to a uniform contrast of the images. This will significantly improve the performance of the image processing algorithm.

6.2 Standard flow cell

A standard flow cell should be used for the measurements. This use of a standard flow cell will aid in determining the accuracy of the system through analytical calculations. The flow of the fluid outside the boundary of the channel causing the error in the current measurements will be prevented by the use of a standard flow cell.

6.3 Flow driving mechanism

One of the reasons of the variation in the measured flow velocities is the fluctuating flow provided by the syringe pump. These fluctuations in the flow can be avoided by the use of a gravity driven flow. The gravity driven flow can be generated by developing a setup which consists of a fluid reservoir placed at a certain height connected to the flow cell inlet. The height of the fluid reservoir can be calculated for the desired flow rate. The setup can be developed such that it can accommodate several height levels of the fluid reservoir to generated different flow velocities.

Alternatively an electrokinetic flow can be used with the setup which will provide a uniform flow field for the measurements. In order to utilize the electrokinetic flow mechanism, standard flow cells with embedded electrodes should be used.

6.4 Application of caged fluorescent dyes

The designed setup has only been used and tested with the photobleaching approach. The caged fluorescent dye should also be used to test the performance of the system. Measuring a bright signal against a dark background, as in the case of using a caged fluorescent dye, is always favorable than measuring the dark signal against a bright fluorescing background as in the case of photobleaching. The motion of the uncaged molecule can be tracked for a longer instance of time as compared to the photobleached tagged region which tend to disappear within 4 to 5 successive frames. However, the requirement of the ultraviolet permissible optics will complicate the measurements based on the caged fluorescent molecules.

6.5 Development of a dedicated algorithm

The images are processed using a commercially available particle tracking algorithm. Although the velocity measurements using this algorithm are satisfactory, the accuracy of processing can be further increased by developing and using a dedicated algorithm for this approach. An algorithm which can accommodate for the varying contrast and the decrease in size of the photobleached tagged regions while differentiating between closely marked tagged dots will significantly increase the capacity of the system in terms of image processing. The proposed algorithm should be capable of accommodating both the photobleaching approach and the caged fluorescence based approach of the two dimensional scanning MTV system.

6.6 Higher magnification objective

A higher magnification objective lens should be used for the measurements in order to accommodate smaller microchannels. The higher magnification objective when used in combination with the focusing lens will enable the generation of a smaller tagged region.

BIBLIOGRAPHY

Abgrall, P., & Nguyen, N.-T. (2009). *Nanofluidics.* Artech House, Boston, London.

Beebe, D., Mensing, G., & Walker, G. (2002). Physics and applications of microfluidics in biology. *Annual Review of Biomedical Engineering , 4*, 261-286.

Bernardinelli, Y., Haeberli, C., & Chatton, J.-Y. (2005). Flash photolysis using a light emitting diode: An efficient,compact,and affordable solution. *Cell Calcium , 37* (6), 565–572.

Beushausen, V., Roetmann, K., Schmunk, W., Wellhausen, M., Garbe, C., & Jähne, B. (2009). 2D-measurement technique for simultaneous quantitative determination of mixing ratio and velocity field in microfluidic applications. *Notes on Numerical Fluid Mechanics , 106*, 155-164.

Biage, M., & Campos, J. (2003). Visualization Study and Quantitative Velocity Measurements in Turbulent Taylor-Couette Flow by Phantomm Flow Tagging: A Description of the Transition to Turbulence. *Journal of the Brazilian Society of Mechanical Sciences and Engineering , 25* (4), 378–390.

Bohl, D. G., & Koochesfahani, M. M. (2004). Molecular tagging velocimetry measurements of axial flow in a concentrated vortex core. *Physics of fluids , 16 (11)*, 4185-4191.

Bohl, D., Koochesfahani, M., & Olson, B. (2001). Development of stereoscopic molecular tagging velocimetry. *Experiments in Fluids , 30* (3), 302-308.

CCD resolution calculation. (n.d.). Retrieved from Nikon microscopy U: http://www.microscopyu.com/tutorials/flash/pixelcalc/

Chu, C.-C., & Liao, Y.-Y. (1992). A quantitative study of the flow around an impulsively started circular cylinder. *Experiments in Fluids , 13* (2-3), 137-146.

Chu, C.-C., Wang, C.-T., & Hsieh, C.-S. (1993). An experimental investigation of vortex motions near surfaces. *Physics of Fluids, A , 5* (3), 662-676.

Chu, C.-C., Wang, C.-T., Chang, C.-C., Chang, R.-Y., & Chang, W.-T. (1995). Head-on collision of two coaxial vortex rings: experiment and computation. *Journal of Fluid Mechanics , 296*, 39-71.

Couch, G., Johnston, K., & Ojha, M. (1996). Full-field flow visualization and velocity measurement with a photochromic grid method. *Measurement Science and Technology , 7* (9), 1238-1246.

Cummings, E. B. (2000). An image processing and optimal nonlinear filtering technique for particle image velocimetry of microflows. *Experiments in Fluids , 29* (Suppl. S), S42--S50.

D’Arco, A., Charmet, J., & Cloitre, M. (1982). Nouvelle technique de marquage d’ecoulement par utilisation de molecules photochromes. *Applied Physics Reviews , 17*, 89-93.

F7103 Product Structure. (n.d.). Retrieved 03 12, 2009, from Invitrogen by Life Technologies: http://www.invitrogen.com/site/us/en/home/support/Product-Technical-Resources/Product-Structures.-07103.html

F7103 Spectrum data. (n.d.). Retrieved 01 2010, 31, from http://probes.invitrogen.com/media/spectra/data/1300ph9.txt

Falco, R., & Chu, C. (1987). Measurement of twodimensional fluid dynamic quantities using a photochromic grid tracing technique. *Proceedings of the International Society for Optics and Photonics , 814*, 706-710.

Feynman, R. P. (1992). There's plenty of room at the bottom. *Journal of Microelectromechanical Systems , 1* (1), 60-66.

Frantisak, F., De, A. P., Smith, J., & Hummel, R. (1969). Nondisturbing tracer technique for quantitative measurements in turbulent flow. *Industrial and Engineering Chemistry Fundamentals , 8* (1), 160-167.

Freudenthal, P., Pommer, M., Meinhart, C., & Piorek, B. (2007). Quantum nanospheres for sub-micron particle image velocimetry. *Experiments in Fluids , 43* (4), 525-533.

Gad-el-Hak, M. (1999). Fluid mechanics of microdevices-the freeman scholar lecture. *Journal of Fluids Engineering-Transactions of ASME , 121* (1), 5-33.

Gad-el-Hak, M. (2006). Gas and liquid transport at the microscale. *Heat Transfer Engineering , 27* (4), 13-29.

Garbe, C. S., Volker, K. R., & Jähne, B. (2006). An optical flow based technique for the non-invasive measurement of microfluidic flows. *12th International symposium on flow visualization,Göttingen, Germany, September 10-14* , 1-10.

Garbe, C. S., Volker, K. R., & Jähne, B. (2008). An optical flow MTV based technique for measuring microfluidic flow in the presence of diffusion and Taylor dispersion. *Experiments in fluids , 44*, 439-450.

Gendrich, C. P., Koochesfahani, M. M., & Nocera, D. G. (1997). Molecular tagging velocimetry and other novel applications of a new phosphorescent supramolecule. *Experiments in fluids , 23*, 361-372.

Gendrich, C., & Koochesfahani, M. (1996). A spatial correlation technique for estimating velocity fields using molecular tagging velocimetry (MTV). *Experiments in Fluids , 22* (1), 67-77.

Gravesen, P., Branebjerg, J., & Jensen, O. (1993). Microfluidics - A review. *Journal of Micromechanics and Microengineering , 3* (4), 168-182.

Guilkey, J., Gee, K., McMurtry, P., & Klewicki, J. (1996). Use of caged fluorescent dyes for the study of turbulent passive scalar mixing. *Experiments in Fluids , 21* (4), 237–242.

Heller, C. A., Henry, R. A., McLaughlin, B. A., & Bliss, D. E. (1974). Fluorescence spectra and quantum yields: Quinine, uranine, 9,10-diphenylanthracene, and 9,10-bis(phenylethynyl)anthracenes. *Journal of Chemical and Engineering Data , 19* (3), 214-219.

Hill, R., & Klewicki, J. (1996). Data reduction methods for flow tagging velocity measurements. *Experiments in Fluids , 20* (3), 142–152.

Hiller, B., Booman, R., Hassa, C., & Hanson, R. (1984). Velocity visualization in gas flows using laser-induced phosphorescence of biacetyl. *Review of Scientific Instruments , 55* (12), 1964–1967.

Ho, C.-M., & Tai, Y.-C. (1998). Micro-electro-mechanical-systems (MEMS) and fluid flows. *Annual Review of Fluid Mechanics , 30*, 579-612.

Hosokawa, S., Fukunaga, T., & Tomiyama, A. (2009). Application of photobleaching molecular tagging velocimetry to turbulent bubbly flow in a square duct. *Experiments in Fluids , 47* (4-5), 745-754.

Hrnčiřík, P., & Náhlík, J. (2010). Novel micro-scale analytical devices for on-line bioprocess monitoring: A review. *Chemical & Biochemical Engineering Quarterly , 24* (4), 489-500.

Hsu, A., Srinivasan, R., Bowersox, R., & North, S. (2009). Molecular tagging using vibrationally excited nitric oxide in an underexpanded jet flowfield. *AIAA J , 47* (11), 2597-2604.

Hu, H., & Koochesfahani, M. M. (2006). Molecular tagging velocimetry and thermometry and its application to the wake of a heated circular cylinder. *Measurements science and technology , 17*, 1269-1281.

Johnson, T., Ross, D., Gaitan, M., & Locascio, L. (2001). Laser modification of preformed polymer microchannels: Application to reduce band broadening around turns subject to electrokinetic flow. *Analytical Chemistry , 73* (15), 3656–3661.

Koochesfahani, M. M. (1999). Molecular Tagging Velocimetry (MTV): Progress and Applications. *30th AIAA Fluid Dynamics Conference.* Norfolk, VA.

Koochesfahani, M., & Nocera, D. (2001). Molecular tagging velocimetry maps fluid flows. *Laser Focus World , 37* (6), 103-108.

Koochesfahani, M., Cohn, R., & MacKinnon, C. (2000). Simultaneous whole-field measurements of velocity and concentration fields using a combination of MTV and LIF. *Measurement Science and Technology , 11* (9), 1289–1300.

Koochesfahani, M., Cohn, R., Gendrich, C., & Nocera, D. (1996). Molecular tagging diagnostics for the study of kinematics and mixing in liquid-phase flows. *8th International symposium on Applications of Laser Techniques to Fluid Mechanics (Plenary Session)*, (pp. 1.2.1–1.2.12). Lisbon, Portugal.

Korkotian, E., & Segal, M. (2007). Morphological constraints on calcium dependent glutamate receptor trafficking into individual dendritic spine. *Cell Calcium , 42* (1), 41–57.

Kuang, C., & Wang, G. (2010). A novel far-field nanoscopic velocimetry for nanofluidics. *Lab on a Chip - Miniaturisation for Chemistry and Biology , 10* (2), 240-245.

Kurada, S., Rankin, G., & Sridhar, K. (1994). Flow visualization using photochromic dyes: A review. *Optics and Lasers in Engineering , 20* (3), 177-192.

Lee, S. J., & Kim, S. (2009). Advanced particle-based velocimetry techniques for microscale flows. *Microfluid Nanofluid , 6* (5), 577-588.

Lempert, W. R., & Harris, S. R. (2000). Flow tagging velocimetry using caged dye photo-activated fluorophores. *Measurement in science and technology , 11*, 1251-1258.

Lempert, W., Boehm, M., Jiang, N., Gimelshein, S., & Levin, D. (2003). Comparison of molecular tagging velocimetry data and direct simulation Monte Carlo simulations in supersonic micro jet flows. *Experiments In Fluids , 34* (3), 403–411.

Lempert, W., Ronney, P., Magee, K., Gee, K., & Haugland, R. (1995). Flow tagging velocimetry in incompressible flow using photo-activated nonintrusive tracking of molecular motion (PHANTOMM). *Experiments in Fluids , 18* (4), 249–257.

Lim, D., Shelby, J., Kuo, J., & Chiu, D. (2003). Dynamic formation of ring-shaped patterns of colloidal particles in microfluidic systems. *Applied Physics Letters , 83* (6), 1145-1147.

Lindken, R., Rossi, M., Grooße, S., & Westerweel, J. (2009). Micro-Particle Image Velocimetry (PIV): Recent developments, applications, and guidelines. *Lab on a Chip-Miniaturisation for Chemistry and Biology , 9* (17), 2551-2567.

Liu, J. B., Pan, Q., Liu, C. S., & Shi, J. R. (1988). Principles Of Flow Field Diagnostics By Laser-Induced Biacetyl Phosphorescence. *Experiments In Fluids , 6* (8), 505–513.

Liu, J.-B., P, Q., L, C.-S., & Jie-Rong, S. (1988). Principles of flow field diagnostics by laser induced biacetyl phosphorescence. *Experiments in Fluids , 6* (8), 505-513.

Manz, A., Graber, N., & Widmer, H. (1990). Miniaturized total chemical analysis systems: A novel concept for chemical sensing. *Sensors and Actuators B: Chemical , 1* (1-6), 244-248.

Matching CCD resolution. (n.d.). Retrieved from http://www.photometrics.com.

Maynes, D., & Webb, A. R. (2002). Velocity profile characterization in sub millimeter diameter tubes using molecular tagging velocimetry. *Experiments in fluids , 32*, 3-15.

McCray, J., & Trentham, D. (1989). Properties and uses of photoreactive caged compounds. *Annu Rev Biophys Biophys Chem , 18*, 239–270.

McKeon, B. J., Comte-Bellot, G., Foss, J. F., Westerweel, J., Scarano, F., Tropea, C., et al. (2007). Molecular Tagging Velocimetry (MTV). In C. Tropea, A. Yarin, & J. F. Foss (Eds.), *Springer handbook of experimental fluid Mechanics* (pp. 362-382). Springer Berlin Heidelberg.

Meinhart, C. D., Wereley, S. T., & Santiago, J. G. (1999). PIV measurements of a microchannel flow. *Experiments in fluids , 27*, 414-419.

Meinhart, C., & Zhang, H. (2000). Flow structure inside a microfabricated inkjet printhead. *J Microelectromech Syst , 9* (1), 67–75.

Miles, R., & Lempert, W. (1997). Quantitative flow visualization in unseeded flows. *Annual Review of Fluid Mechanics , 29*, 285-326.

Miles, R., Connors, J., Markovitz, E., Howard, P., & Roth, G. (1989). Instantaneous profiles and turbulence statistics of supersonic free shear layers by Raman excitation plus laser-induced electronic fluorescence (Relief) velocity tagging of oxygen. *Experiments in Fluids , 8* (1-2), 17-24.

Miles, R., Grinstead, J., Kohl, R., & Diskin, G. (2000). The RELIEF flow tagging technique and its application in engine testing facilities and for helium-air mixing studies. *Measurement Science and Technology , 11* (9), 1272-1281.

Miller, S. (1967). *Photochemical reaction for the study of velocity patterns and profiles.* Toronto ON, Canada: University of Toronto.

Mitchison, T., Sawin, K., Theriot, J., Gee, K., & Mallavarapu, A. (1998). Caged fluorescent probes. In G. Marriot (Ed.), *Methods in Enzymology, Caged Compounds* (Vol. 291, pp. 63–78). Academic Press.

Mosier, B., Molho, J., & Santiago, J. (2002). Photobleached-fluorescence imaging of microflows. *Experiments in fluids , 33*, 545-554.

Nguyen, N.-T., & Wereley, S. T. (2006). *Fundamentals and applications of microfluidics.* Artech House, Boston, London.

Noullez, A., Wallace, G., Lempert, W., Miles, R., & Frisch, U. (1997). Transverse velocity increments in turbulent flow using the relief technique. *Journal of Fluid Mechanics , 339*, 287-307.

Ojha, M., Cobbold, R. S., Johnston, K., & Hummel, R. L. (1989). Pulsatile flow through constricted tubes: an experimental investigation using photochromic tracer methods. *Journal of Fluid Mechanics , 203*, 173-197.

Park, H., Moore, J., Trass, O., & Ojha, M. (1999). Laser photochromic velocimetry estimation of the vorticity and pressure field - Two-dimensional flow in a curved vessel. *Experiments in Fluids , 26* (1-2), 55-62.

Paul, P. H., Garguilo, M. G., & Rakestraw, D. J. (1998). Imaging of Pressure- and Electrokinetically Flows through Open Capillaries. *Analytical Chemistry , 70*, 2459-2467.

Pellicciol, A., & Wirz, J. (2002). Photoremovable protecting groups: Reaction mechanisms and applications. *Photochemical & Photobiological Sciences , 1* (7), 441–458.

PhotonFocus MV-D1024E-160-CL manual. (n.d.). Retrieved from http://www.photonfocus.com/upload/manuals/MAN028_e_V1_4_MVD1024E-series.pdf

Ponce, A., Wong, P. A., Way, J. J., & Nocera, D. G. (1993). Intense phosphorescence triggered by alcohols upon formation of a cyclodextrin ternary complex. *Journal of physical chemistry , 97*, 11137-11142.

Popovich, A., & Hummel, R. (1967). A new method for non-disturbing turbulent flow measurements very close to a wall. *Chemical Engineering Science , 22* (1), 21-25.

Pouya, S., Koochesfahani, M., Snee, P., Bawendi, M., & Nocera, D. (2005). Single quantum dot (QD) imaging of fluid flow near surfaces. *Experiments In Fluids , 39* (4), 784-786.

Prasad, A., Adrian, R., Landreth, C., & Offutt, P. (1992). Effect of resolution on the speed and accuracy of particle image velocimetry interrogation. *Experiments in Fluids , 13* (2-3), 105-116.

Raffel, M., Willert, C., Wereley, S. T., & Kompenhans, J. (2007). *Particle Image Velocimetry: A Practical Guide.* Springer Berlin Heidelberg.

Rička, J. (1987). Photobleaching velocimetry. *Experiments in Fluids , 5* (6), 381-384.

Roetmann, K., Schmunk, W., Garbe, C., & Beushausen, V. (2008). Micro-flow analysis by molecular tagging velocimetry and planar Raman-scattering. *Experiments in Fluids , 44* (3), 419–430.

Rondelez, Y., Tresset, G., Tabata, K., Arata, H., Fujita, H., Takeuchi, S., et al. (2005). Microfabricated arrays of femtoliter chambers allow single molecule enzymology. *Nature Biotechnology , 23* (3), 361–365.

Ross, D., & Locascio, L. (2003). Effect of caged fluorescent dye on the electroosmotic mobility in microchannels. *Analytical Chemistry , 75* (5), 1218–1220.

Santiago, J. G., Wereley, S. T., Meinhart, C. D., Beebe, D. J., & Adrian, R. J. (1998). A particle image velocimetry system for microfluidics. *Experiments in fluids , 25*, 316-319.

Sharp, K., & Adrian, R. (2005). On flow-blocking particle structures in microtubes. *Microfluidics and Nanofluidics , 1* (4), 376-380.

Shaughnessy, E. J., Katz, I. M., & Schaffer, J. P. (2005). *Introduction to fluid mechanics.* Oxford University Press, Inc.

Shelby, J. P., & Chiu, D. T. (2003). Mapping fast flows over micrometer length scales using flow tagging velocimetry and single molecule detection. *Analytical Chemistry , 75*, 1387-1392.

Shoji, S., & Esashi, M. (1994). Microflow devices and systems. *Journal of Micromechanics and Microengineering , 4* (4), 157-171.

Singh, A., Cummings, E., & Throckmorton, D. (2001). Fluorescent liposome flow markers for microscale particle-image velocimetry. *Analytical Chemistry , 73* (5), 1057-1061.

Sinton, D. (2004). Microscale flow visualization. *Microfluidics and Nanofluidics , 1*, 2-21.

Sinton, D., & Li, D. (2003). Electroosmotic velocity profiles in microchannels. *Colloids and Surfaces A: Physiochem. Eng. Aspects , 222*, 273-283.

Sinton, D., Escobedo-Canseco, C., Ren, L., & Li, D. (2002). Direct and Indirect Electroosmotic Flow Velocity Measurements in Microchannels. *Journal of colloids and interface science , 254*, 184-189.

Sinton, D., Xuan, X., & Li, D. (2004). Thermally induced velocity gradients in electroosmotic microchannel flows: The cooling influence of optical infrastructure. *Experiments in Fluids , 37* (6), 872–882.

Stier, B., & Koochesfahani, M. (1999). Molecular Tagging Velocimetry (MTV) measurements in gas phase flows. *Experiments in Fluids , 26* (4), 297–304.

Stone, H., & Kim, S. (2001). Microfluidics: Basic issues, applications, and challenges. *AIChE J. , 47* (6), 1250-1254.

Stone, H., Stroock, A., & Ajdari, A. (2004). Engineering flows in small devices: Microfluidics toward a lab-on-a-chip. *Annual Review of Fluid Mechanics , 36*, 381-411.

Thompson, B., Maynes, D., & Webb, B. (2005). Characterization of the hydrodynamically developing flow in a microtube using molecular tagging velocimetry. *Journal of Fluid Engineering , 127*, 1003-1012.

Thompson, B., Maynes, D., & Webb, B. (2002). Micro-scale velocity measurements using molecular tagging velocimetry: Methodology and uncertainty. *Proceedings of the 2002 ASME Joint U.S.-European Fluids Engineering Conference*, *257*, pp. 123-133.

Thorlabs DMLP425 plot data. (n.d.). Retrieved from ThorLabs: http://www.thorlabs.com/images/tabImages/DMLP425.xls

ThorLabs DMLP505 plot data. (n.d.). Retrieved from ThorLabs: http://www.thorlabs.com/images/tabImages/DMLP505.xls

Thurlow, E., & Klewicki, J. (2000). Experimental study of turbulent Poiseuille-Couette flow. *Physics of Fluids* , *12* (4), 865–875.

Wang, G. R. (2005). Laser induced fluorescence photobleaching anemometer for microfluidic devices. *Lab On A Chip* , *5* (4), 450-456.

Wang, G., & Fiedler, H. (2000). On high spatial resolution scalar measurement with LIF - Part 1: Photobleaching and thermal blooming. *Experiments in Fluids* , *29* (3), 257-264.

Went, F. W. (1968). The size of man. *American Scientist* , *56*, 400-13.

Wereley, S., & Meinhart, C. (2007). *Particle Image Velocimetry: A Practical Guide.* (M. Raffel, C. Willert, S. T. Wereley, & J. Kompenhans, Eds.) Springer Berlin Heidelberg.

Wereley, S., & Meinhart, C. (2010). Recent advances in micro-particle image velocimetry. *Annual Review of Fluid Mechanics* , *42*, 557-576.

Wereley, S., Gui, L., & Meinhart, C. (2002). Advanced algorithms for microscale particle image velocimetry. *AIAA J* , *40* (6), 1047-1055.

Whitesides, G. (2006). The origins and the future of microfluidics. *Nature* , *442* (7101), 368-373.

Yurechko, Y., & Ryazantsev, Y. (1991). Fluid motion investigation by the photochromic flow visualization technique. *Experimental Thermal and Fluid Science , 4* (3), 273-288.

Zalzal, P., Ojha, M., Ethier, C., Cobbold, R., & Johnston, K. (1994). Visualization of transitional pipe flow using the photochromic tracer method. *Physics of Fluids , 6* (6), 2003-2010.

Zhao, Y.-P. (2002). Morphological stability of epitaxial thin elastic films by van der Waals force. *Archive of Applied Mechanics , 72* (1), 77-84.

APPENDIX A

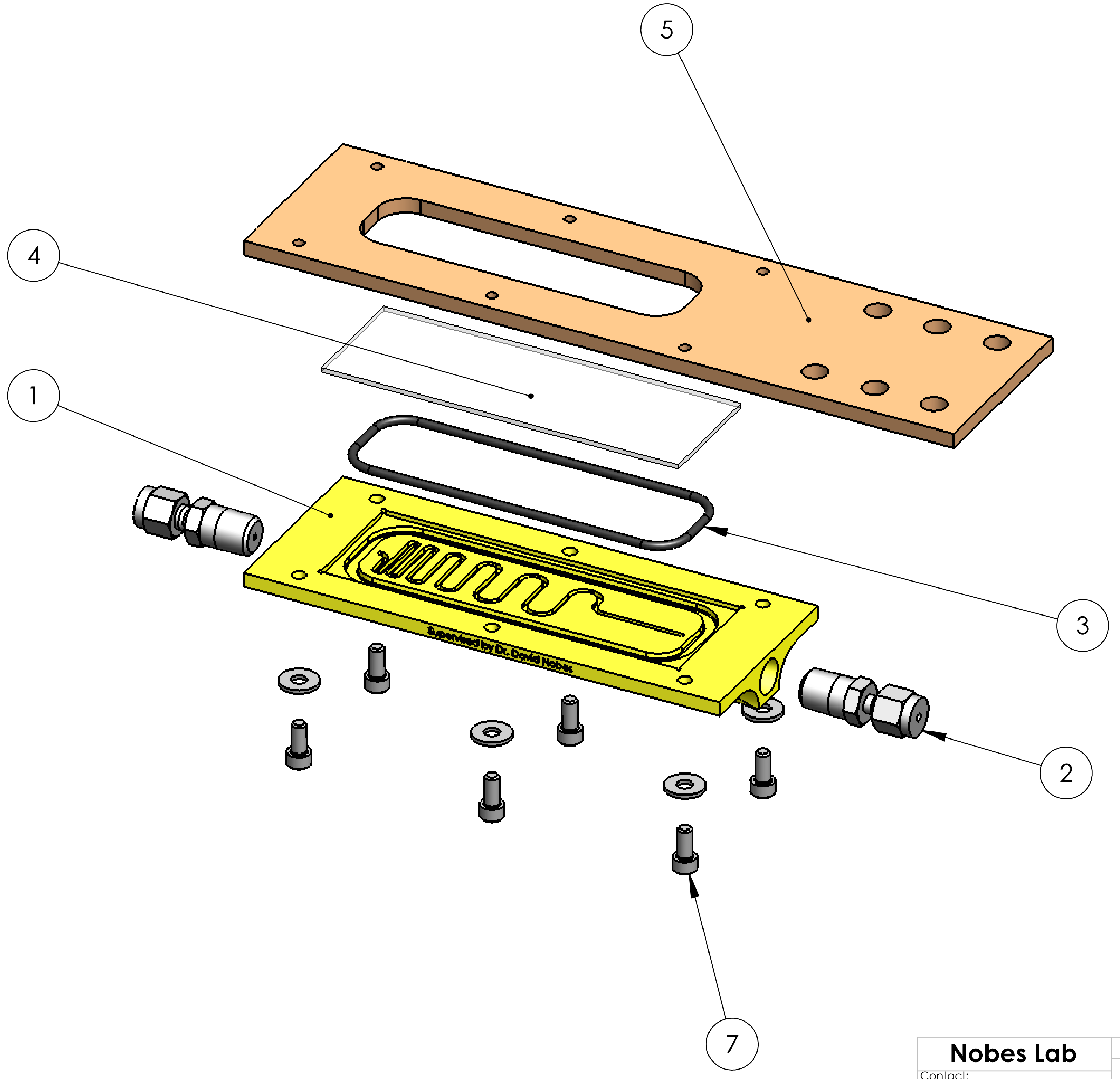
PARTS DRAWINGS

Some parts used in the assembly of the developed system were fabricated in the machine shop. The drawings of the fabricated parts indicating their dimensions are presented in this appendix. The drawings and dimensions of the different parts of the flow cell fabricated in the machine shop are also presented in this appendix.

The drawings of the different components fabricated for the rapid prototyped flow cell are shown in appendix A-1. The drawings of the assembly, channel holder, the custom made o-ring and the top plate are presented in appendix A-1.The drawings of the different components fabricated for the serpentine plastic cutout flow cell is shown in appendix A-2. The drawings of the assembly, the top cover, the glass slides with drilled holes, the serpentine cutout and the bottom plate are presented in appendix A-2.

The drawings of the assembly, the front plate and the side plate fabricated for the camera holder are presented in appendix A-3. Appendix A-4 presents the drawings of the assembly, the horizontal base and the vertical support fabricated for holding the write laser scanner. The drawings of the auxiliary components used in the assembly are provided in appendix A-5. The drawings of these auxiliary components include the blue laser holder, the pinhole base, the component used for raising the setup and the ultraviolet laser holder.

ITEM NO.	PART NUMBER	QTY.
1	Holder	1
2	Male Connector (Tappered Thread)	2
3	Custom O-Ring	1
4	GE-124 Quartz Microscopic Slide (3in x 1in)	1
5	Top Plate	1
6	Washers (Preferred Narrow FW 0.125)	6
7	Socket head cap screw	6



Nobes Lab

Contact: Farhan Ahmad
Email: farhan1@ualberta.ca
Phone: 780-492-6723
Comments:

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH µm 0.6

MATERIAL: Miscallenous

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Microchannel Assembly

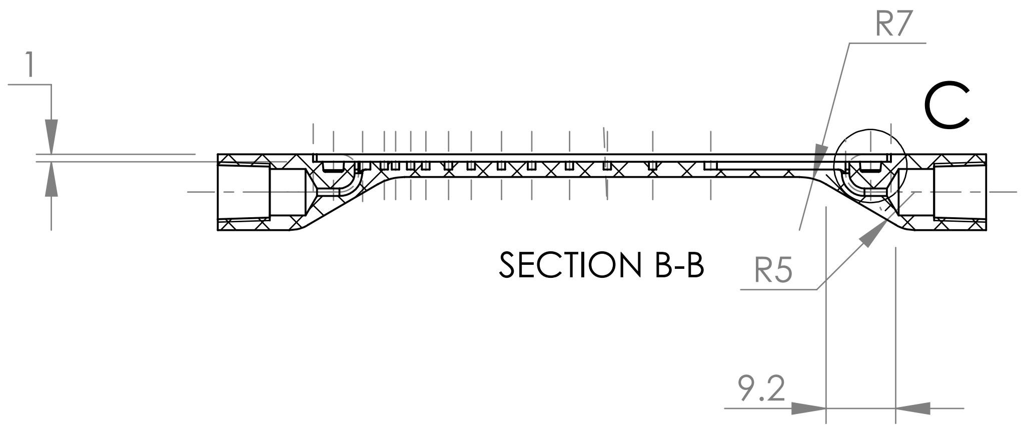
farhan1
8/24/2011 11:05:29 AM
Wednesday, August 24, 2011

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

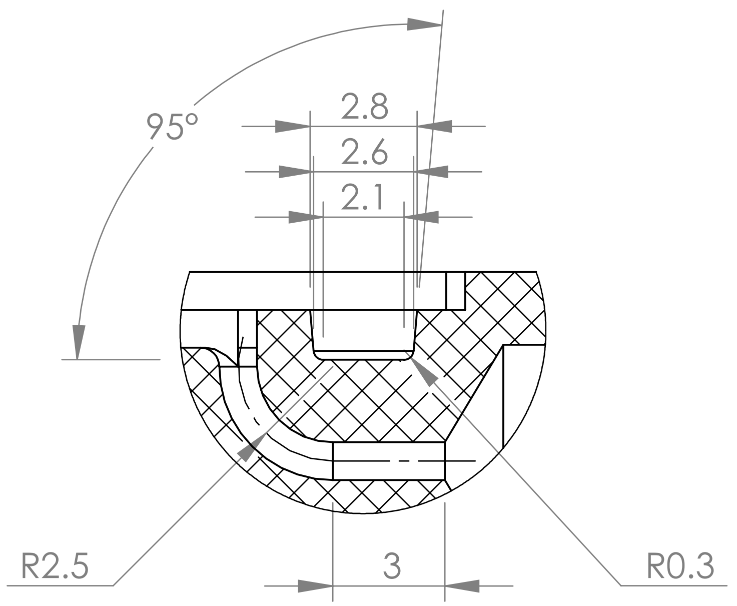
TITLE:
Rapid Prototype Flow Cell Assembly

SIZE **B**
Project:
REV 4

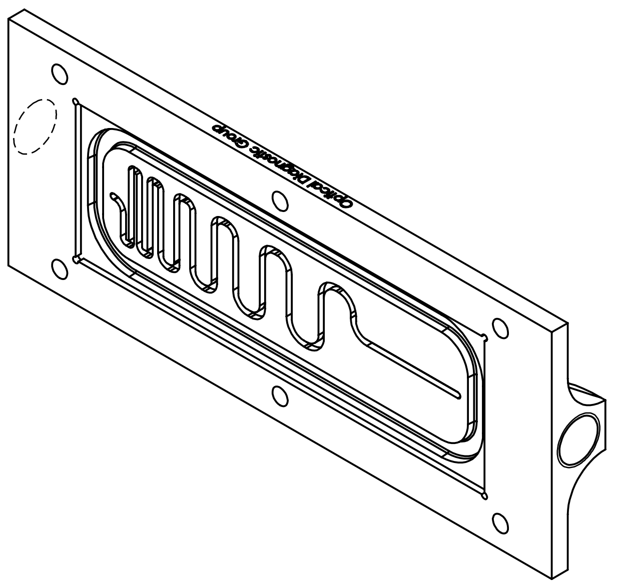
SCALE: 1:1
Mass: 71.38
SHEET 1 OF 1

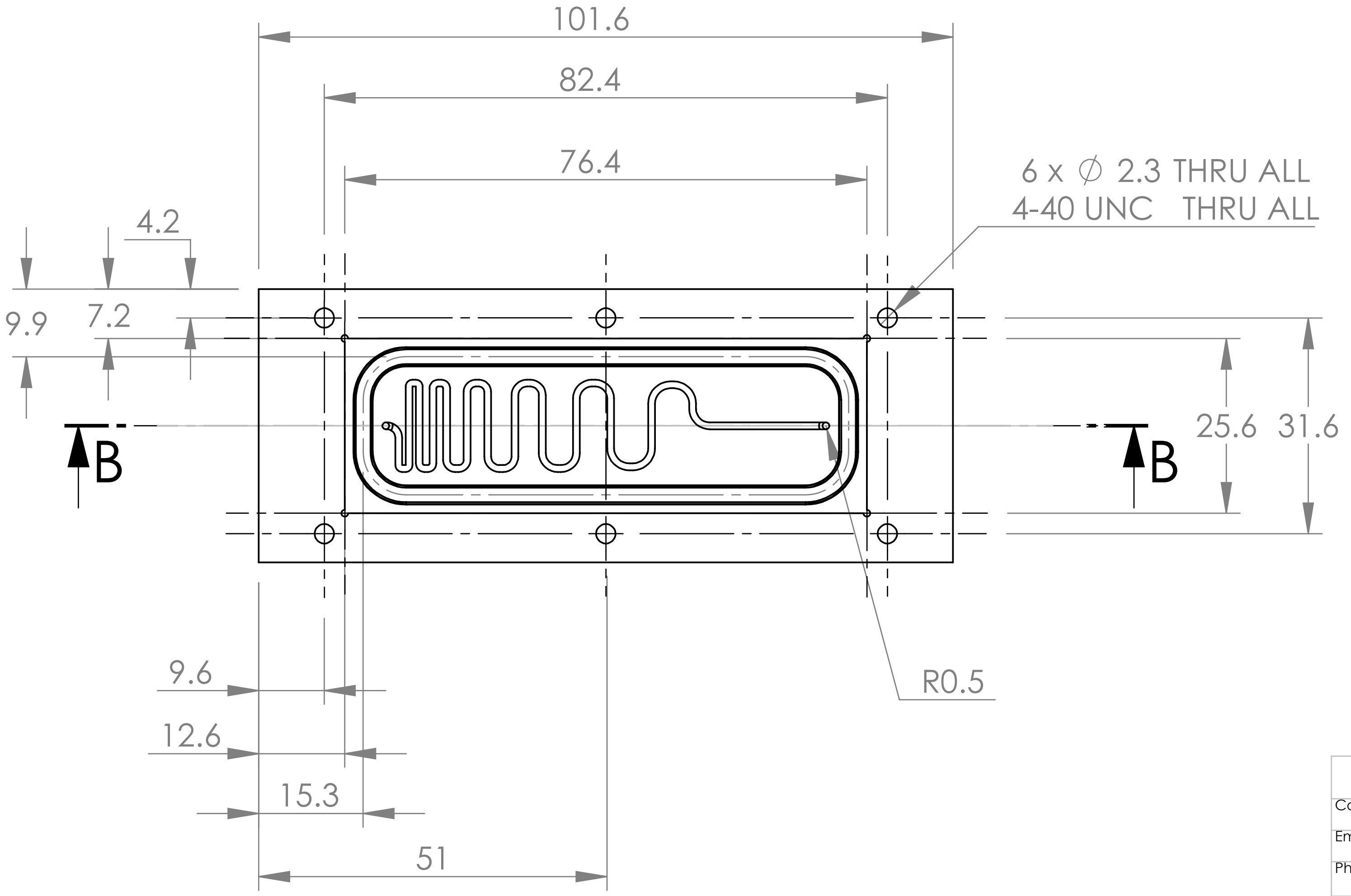


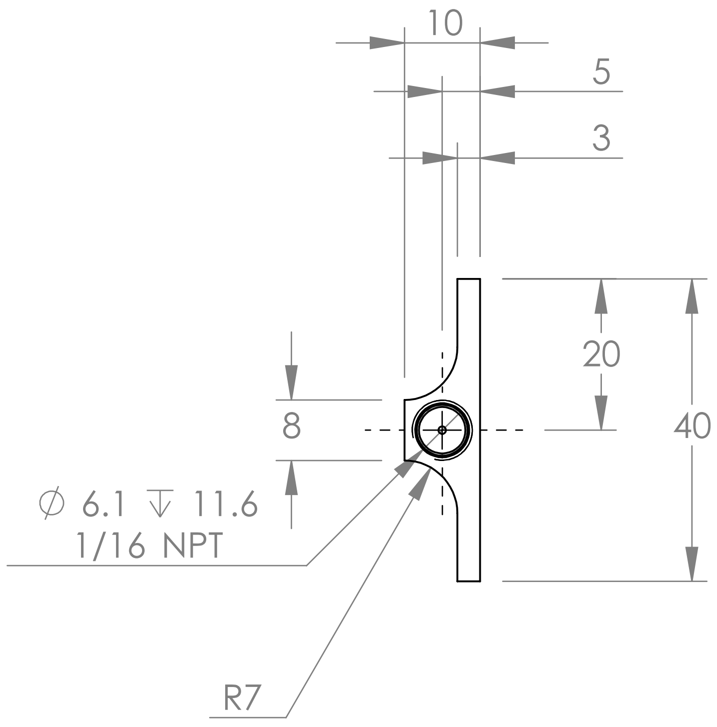
SECTION B-B



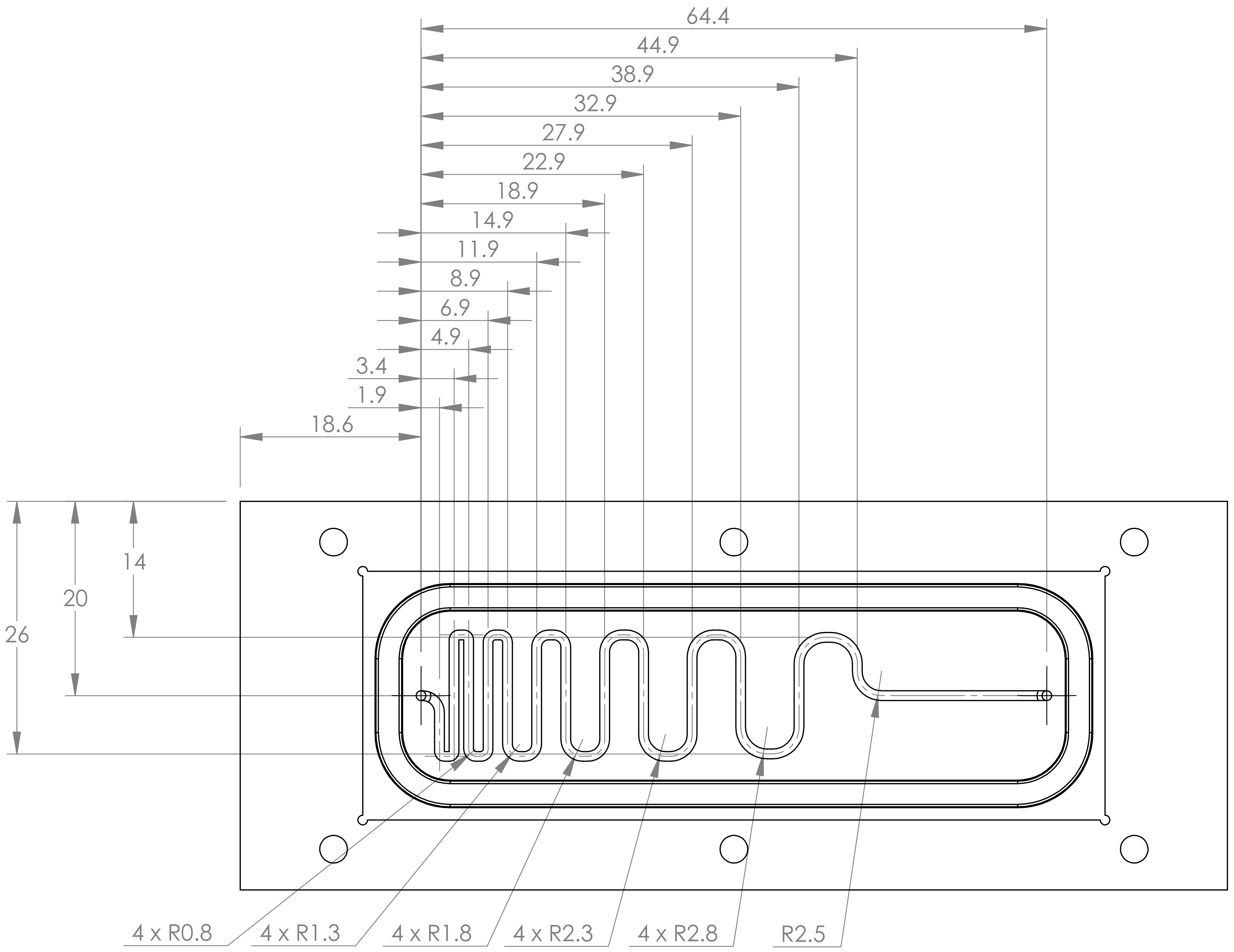
DETAIL C
SCALE 5 : 1







Nobes Lab	UNLESS OTHERWISE SPECIFIED:		NAME	The Department of Mechanical Engineeering UNIVERSITY OF ALBERTA	
Contact: Farhan Ahmad	DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° .X = ± 0.25 .XX = ± 0.05 .XX = ± 0.005	DRAWN	FARHAN	TITLE: The channel holder	
Email: farhan1@ualberta.ca		CHECKED	DSN		
Phone: 780-492-6723		ENG APPR.			
Comments:		Speed Code			
	SURFACE FINISH μm 0.6	MecE Job #			
	MATERIAL: Fulcure 720	File Name:	Holder	SIZE **B**	Project: REV 4
	Lab Location: MEC, 5-27	farhan1 8/24/2011 11:59:29 AM			
	DO NOT SCALE DRAWING	Wednesday, August 24, 201		SCALE: 1:1	Mass: 25.68 SHEET 1 OF 2



Nobes Lab

Contact: Farhan Ahmad

Email: farhan1@ualberta.ca

Phone: 780-492-6723

Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH 0.6 µm

MATERIAL: Fulcure 720

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Holder

farhan1
8/24/2011 11:59:29 AM
Wednesday, August 24, 201

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE: The channel holder

SIZE **B**

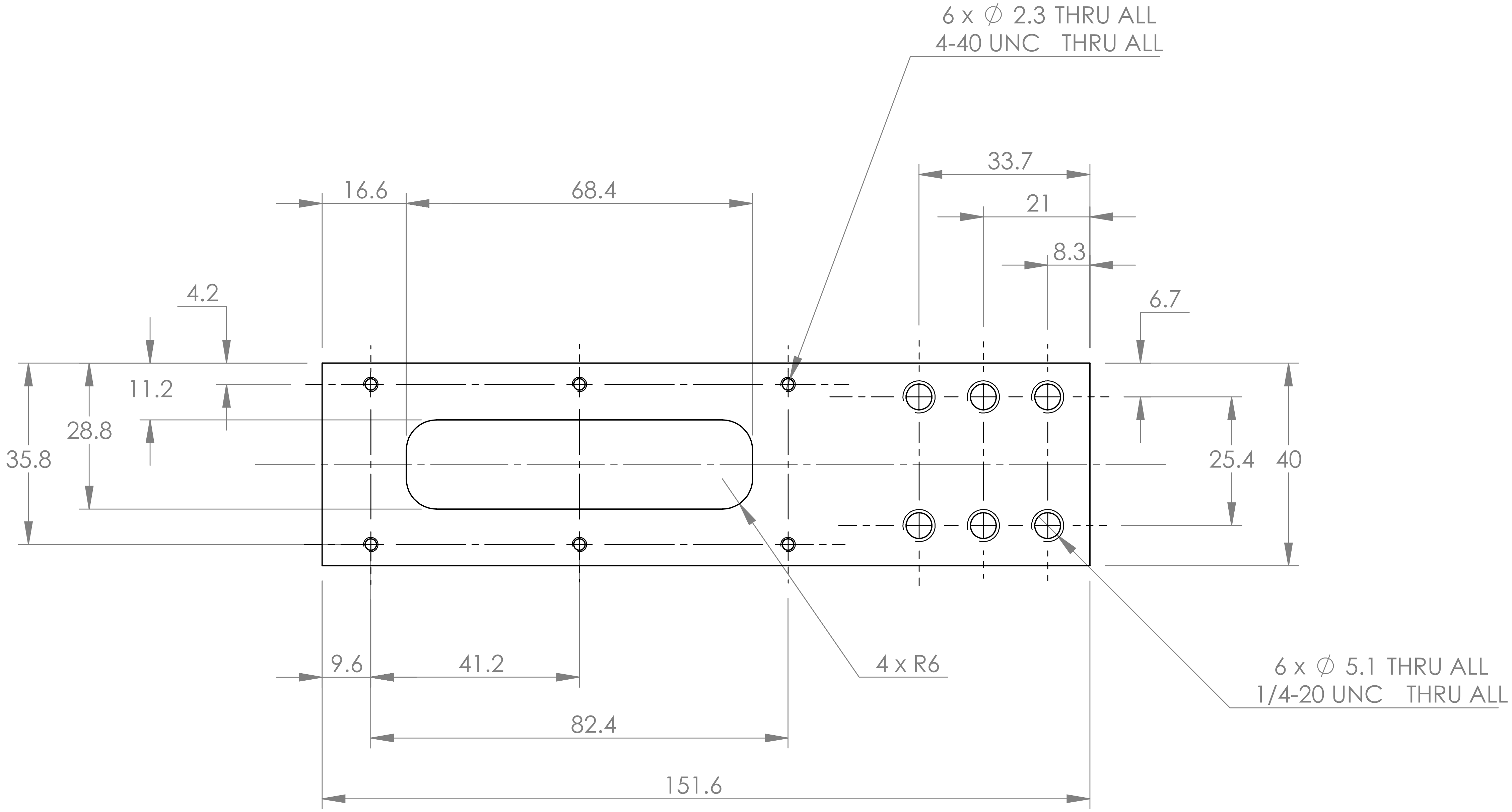
Project:

REV 4

SCALE: 2:1

Mass: 25.68

SHEET 2 OF 2



Note: Thickness is 3.2 mm

Nobes Lab

Contact: Farhan Ahmad
Email: farhan1@ualberta.ca
Phone: 780-492-6723
Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH 0.6 µm

MATERIAL: Aluminum

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Top Plate

farhan1
8/24/2011 11:31:16 AM
Wednesday, August 24, 201

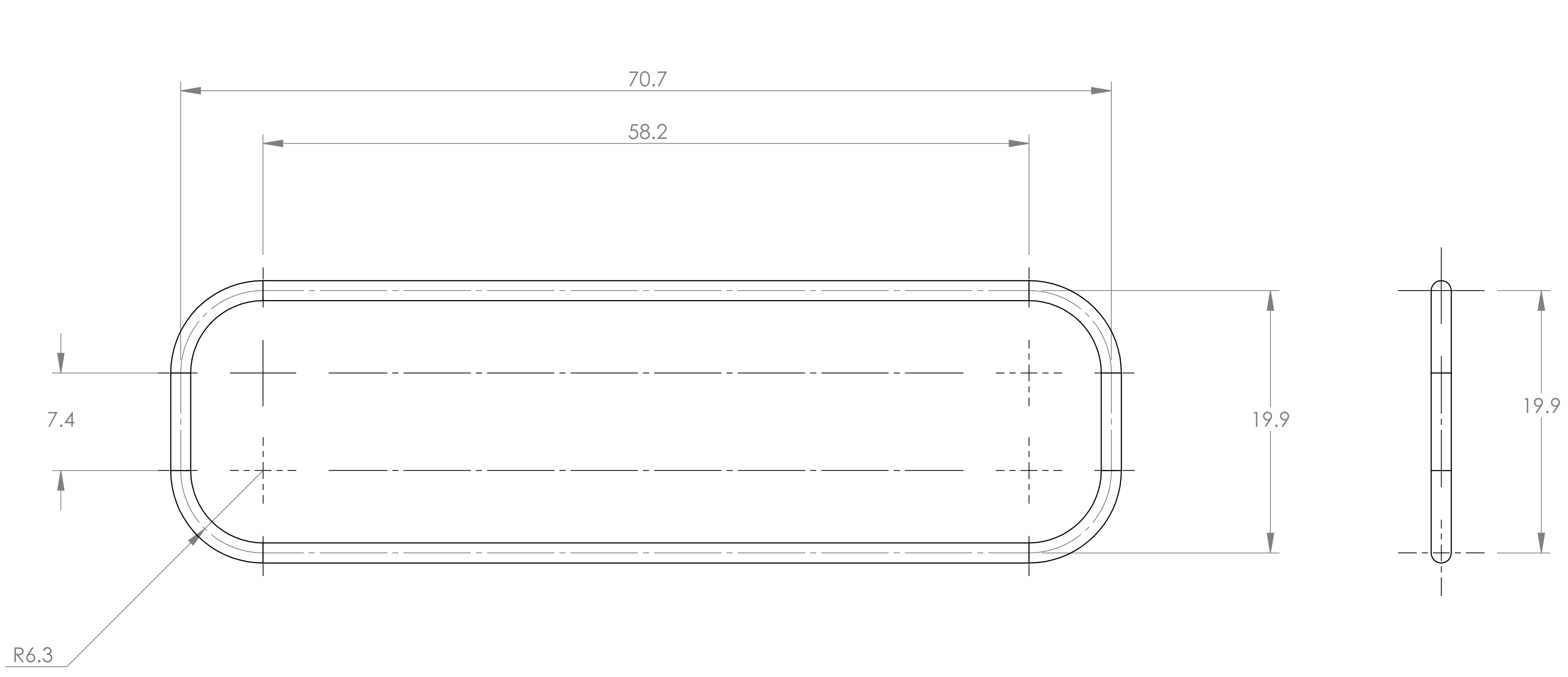
The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:

Top Plate

SIZE B
Project:
REV 4

SCALE: 1:1
Mass: 40.67
SHEET 1 OF 1



Note: Use 1.5mm diameter O-Ring

Nobes Lab

Contact: Farhan Ahmad

Email: farhan1@ualberta.ca

Phone: 780-492-6723

Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH µm 0.6

MATERIAL: Buna N (Nitrile)

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	O-Ring AS-568-034

farhan1
8/24/2011 11:07:54 AM
Wednesday, August 24, 201

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE: Custom O-Ring

SIZE **B**

Project:

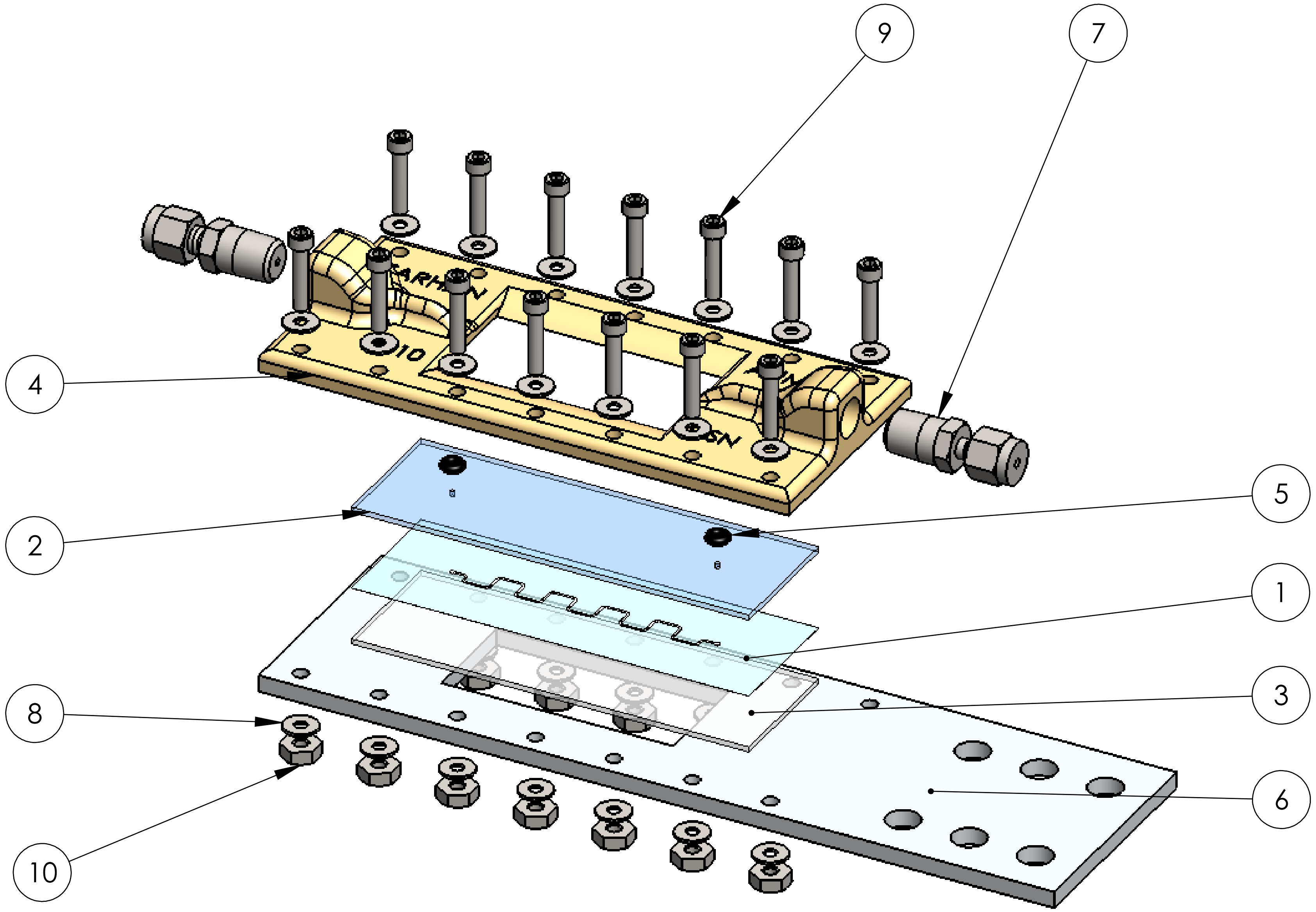
REV 3

SCALE: 3:1

Mass: 0.31

SHEET 1 OF 1

ITEM NO.	PART NUMBER	QTY.
1	Serpentine Channel Plastic Cut Out	1
2	Quartz Microscopic Slide with Holes (3in x 1in)	1
3	Quartz Microscopic Slide	1
4	Attachment Cell	1
5	O-Rings (AS568 - 003)	2
6	Aluminum Bottom Plate	1
7	1/16" NPT Fitting	2
8	Washers (Preferred Narrow FW 0.094)	28
9	Socket Head Cap Screws (HX-SHCS 0.099-48x0.5x0.5-C)	14
10	Hex Nuts (MSHXNUT 0.112-40-S-C)	14



Nobes Lab

Contact: Farhan Ahmad
Email: farhan1@ualberta.ca
Phone: 780-492-6723
Comments:

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH µm 0.6

MATERIAL: Miscalleneous

Lab Location: Mec, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Shim Cut Out Assm

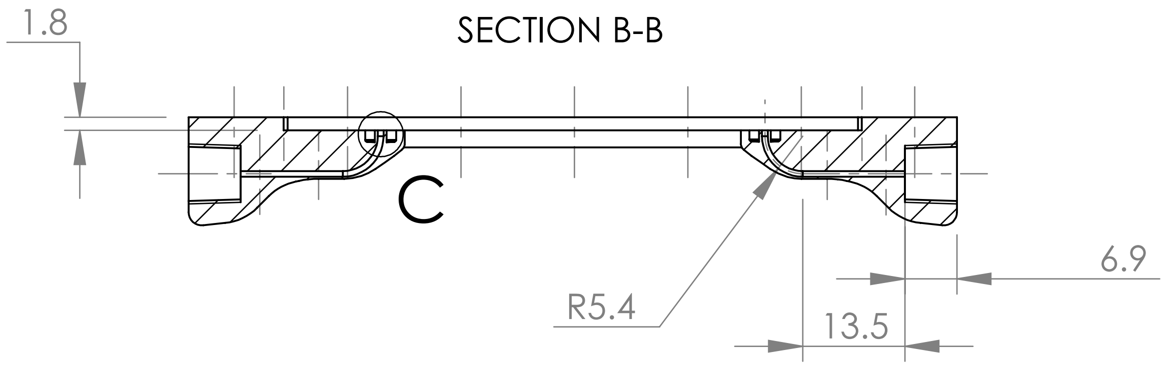
farhan1
8/24/2011 11:04:24 AM
Wednesday, August 24, 201

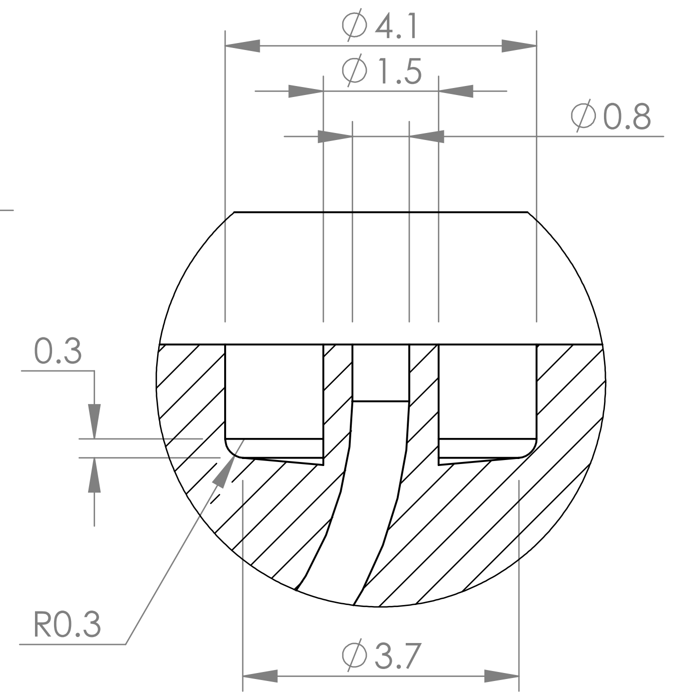
The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:
Plastic Cut Out Flow Cell Assembly

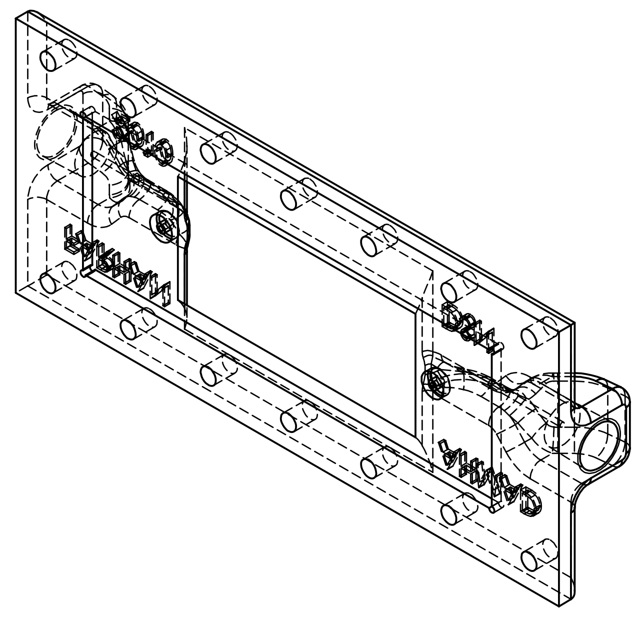
SIZE **B**
Project:
REV 2
SCALE: 1:1
Mass: 40.74
SHEET 1 OF 1

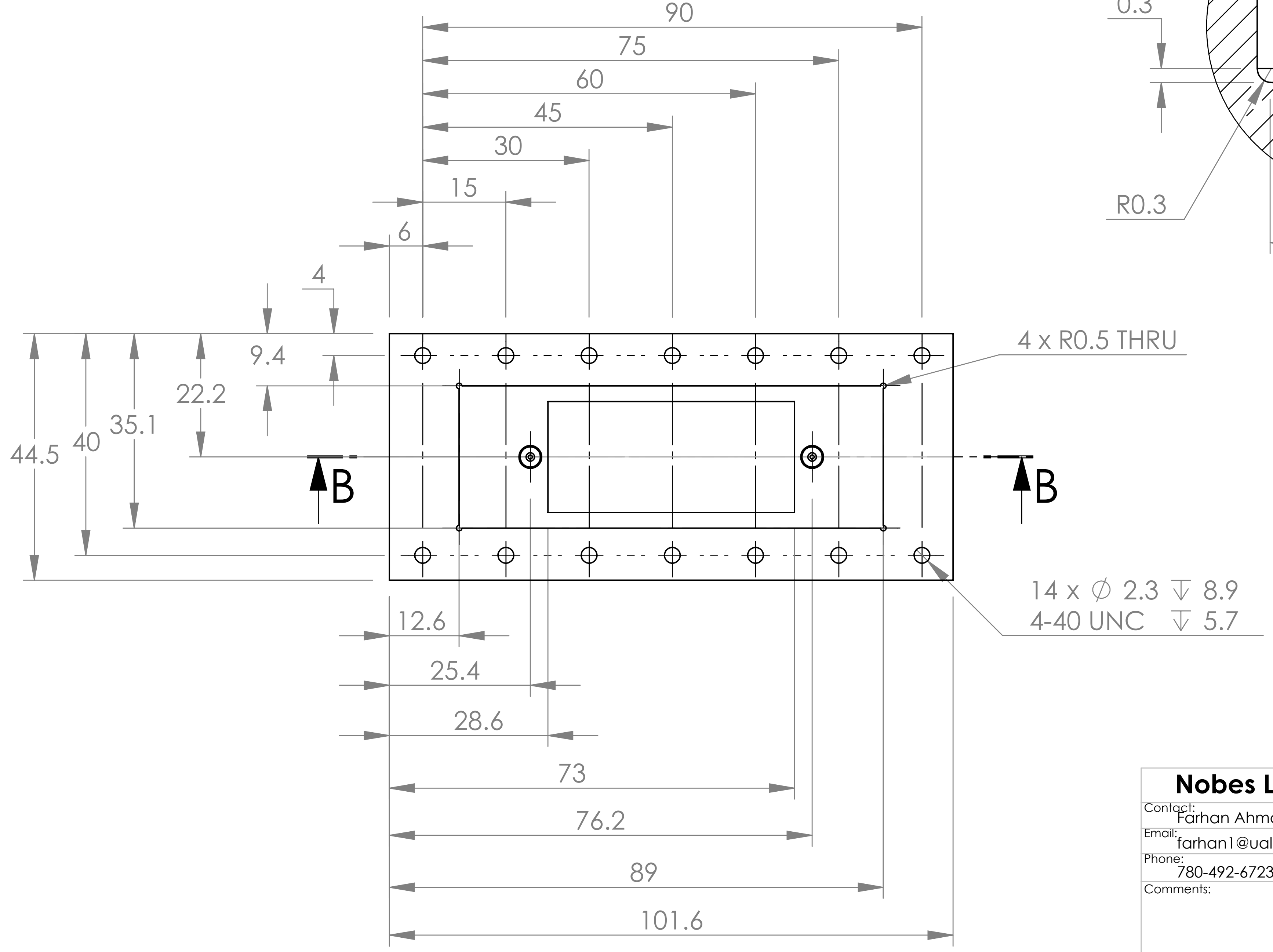
SECTION B-B

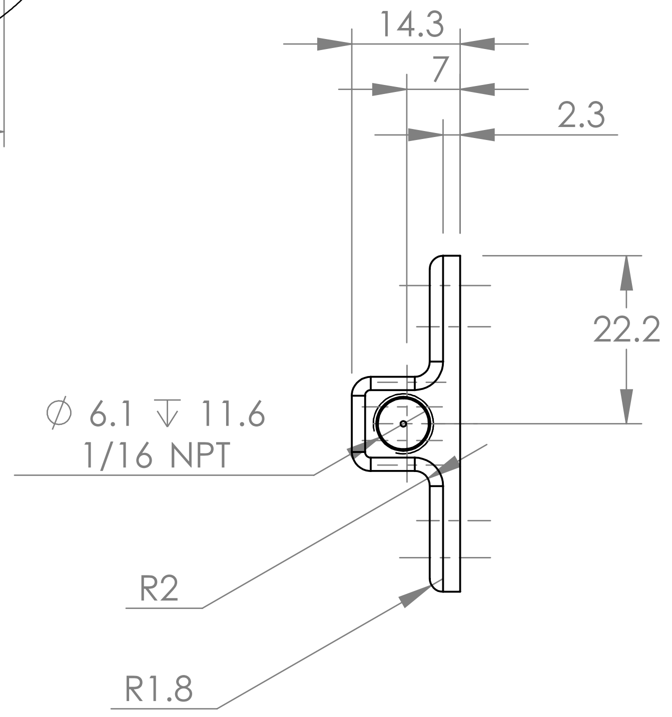




DETAIL C
SCALE 10 : 1







Nobes Lab

Contact: Farhan Ahmad
Email: farhan1@ualberta.ca
Phone: 780-492-6723
Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH 0.6 µm

MATERIAL:
Fulcure 720

Lab Location:
MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Attachment Cell

farhan1
8/24/2011 10:37:01 AM
Wednesday, August 24, 2011

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:

Attachment Cell

SIZE **B**

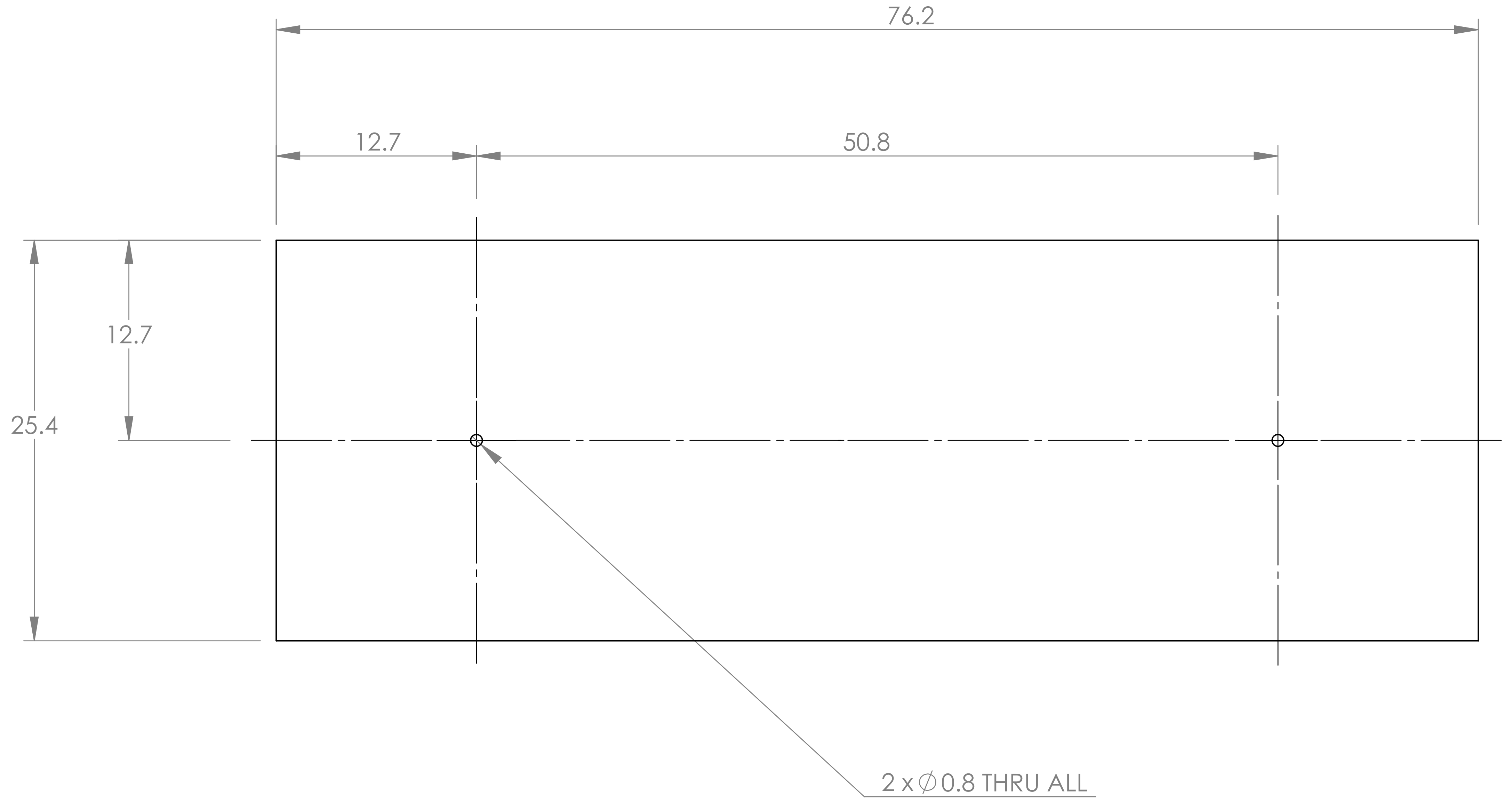
Project:

REV 6

SCALE: 1:1

Mass: 14.15

SHEET 1 OF 1



Note: Thickness is 1 mm

Nobes Lab

Contact: Farhan Ahmad
Email: farhan1@ualberta.ca
Phone: 780-492-6723
Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH µm 0.6

MATERIAL: Quartz

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Slide with Holes

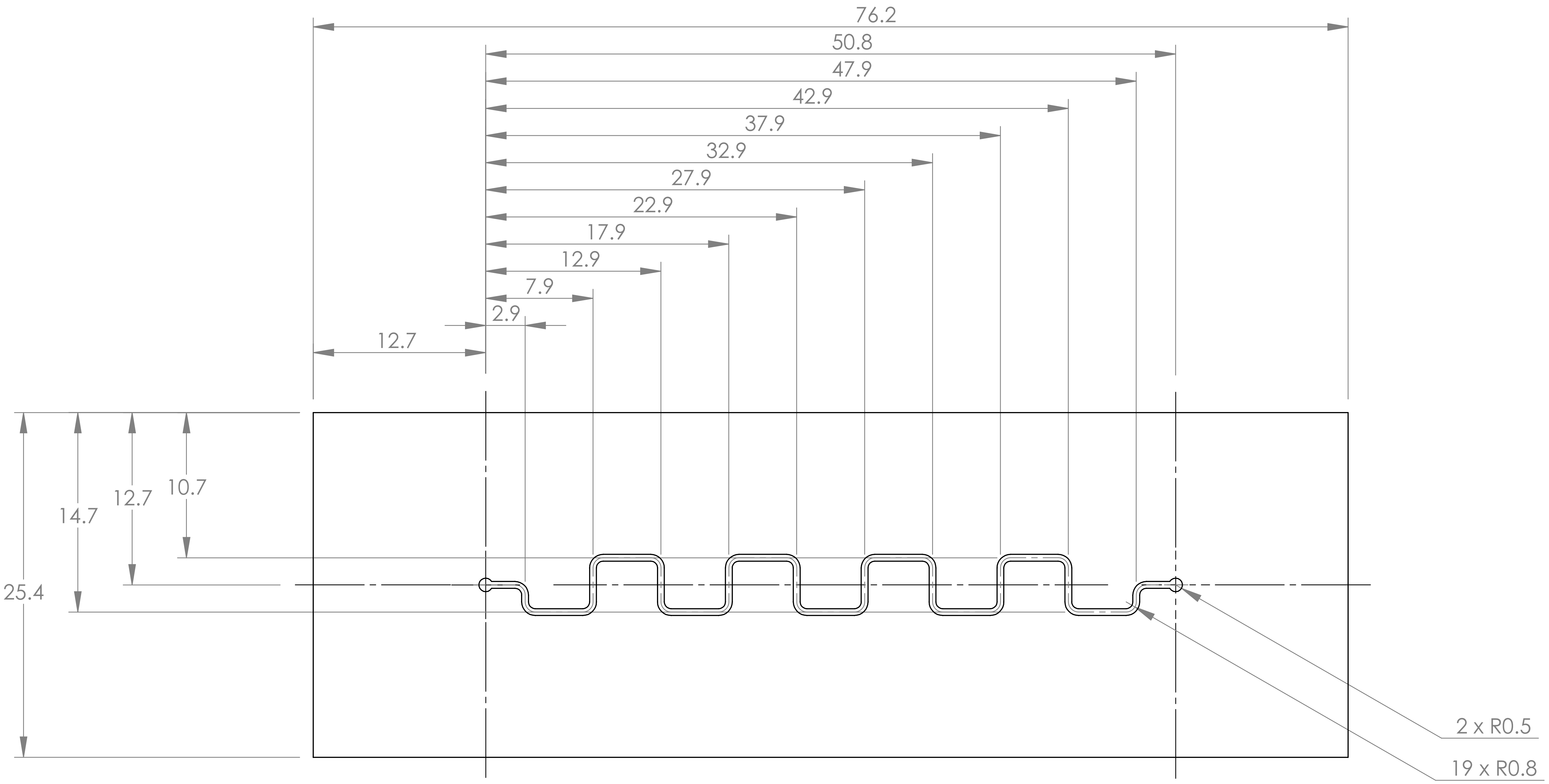
farhan1
8/24/2011 10:59:47 AM
Wednesday, August 24, 2011

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:
Quartz Microscope Slide with Holes

SIZE **B**   Project: REV 1

SCALE: 3:1   Mass: 1.93   SHEET 1 OF 1



Note: Thickness is 300 micrometers

Nobes Lab

Contact: Farhan Ahmad
Email: farhan1@ualberta.ca
Phone: 780-492-6723
Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH µm 0.6

MATERIAL: Plastic

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Plastic Cut Out

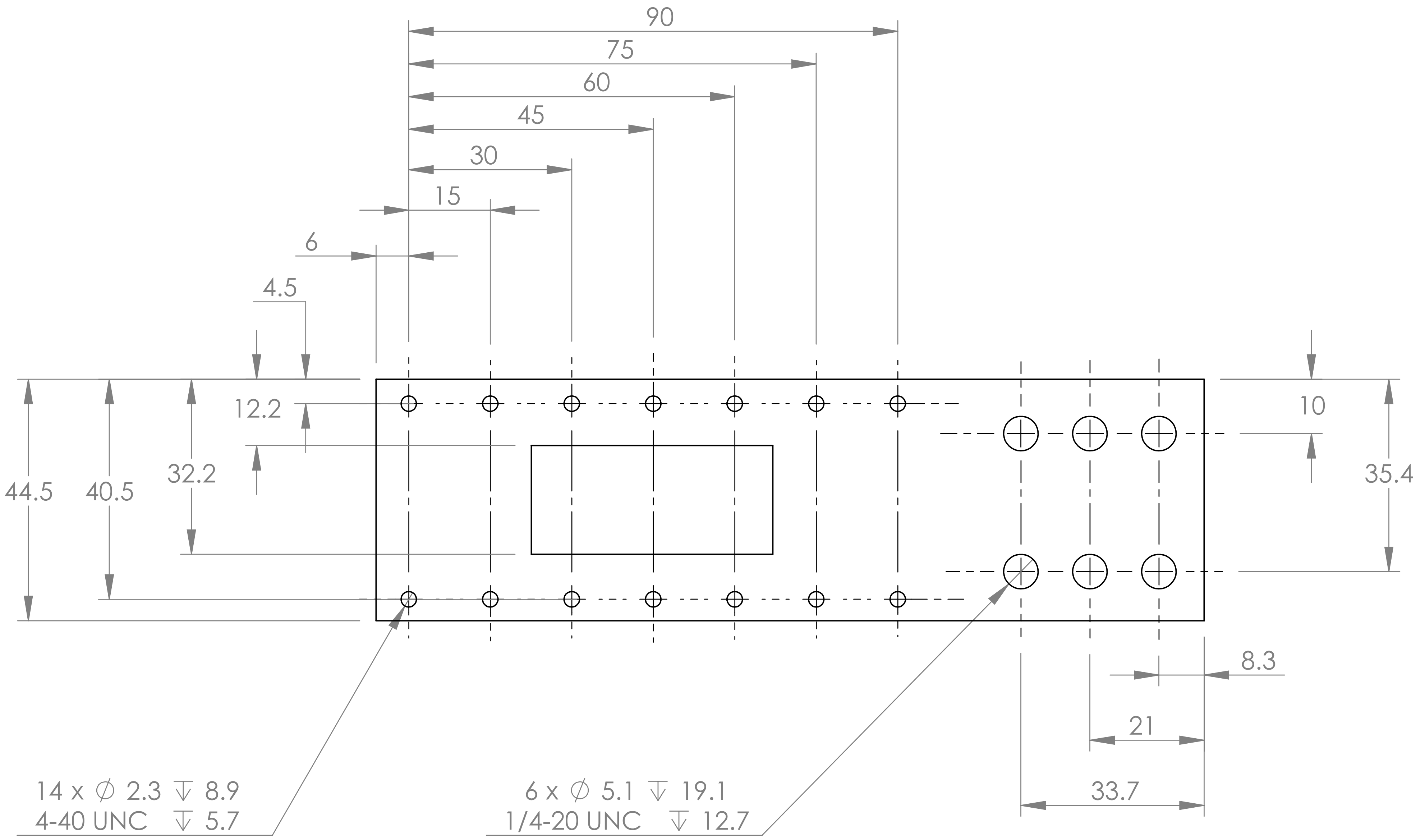
farhan1
8/24/2011 11:12:42 AM
Wednesday, August 24, 201

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:
Serpentine Channel Plastic Cut Out

SIZE **B** Project: REV 2

SCALE: 3:1 Mass: 0.06 SHEET 1 OF 1



Note: Thickness is 3.2 mm

Nobes Lab

Contact: Farhan Ahmad
Email: farhan1@ualberta.ca
Phone: 780-492-6723
Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH µm 0.6

MATERIAL: Aluminum

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Support Plate

farhan1
8/24/2011 11:03:33 AM
Wednesday, August 24, 2011

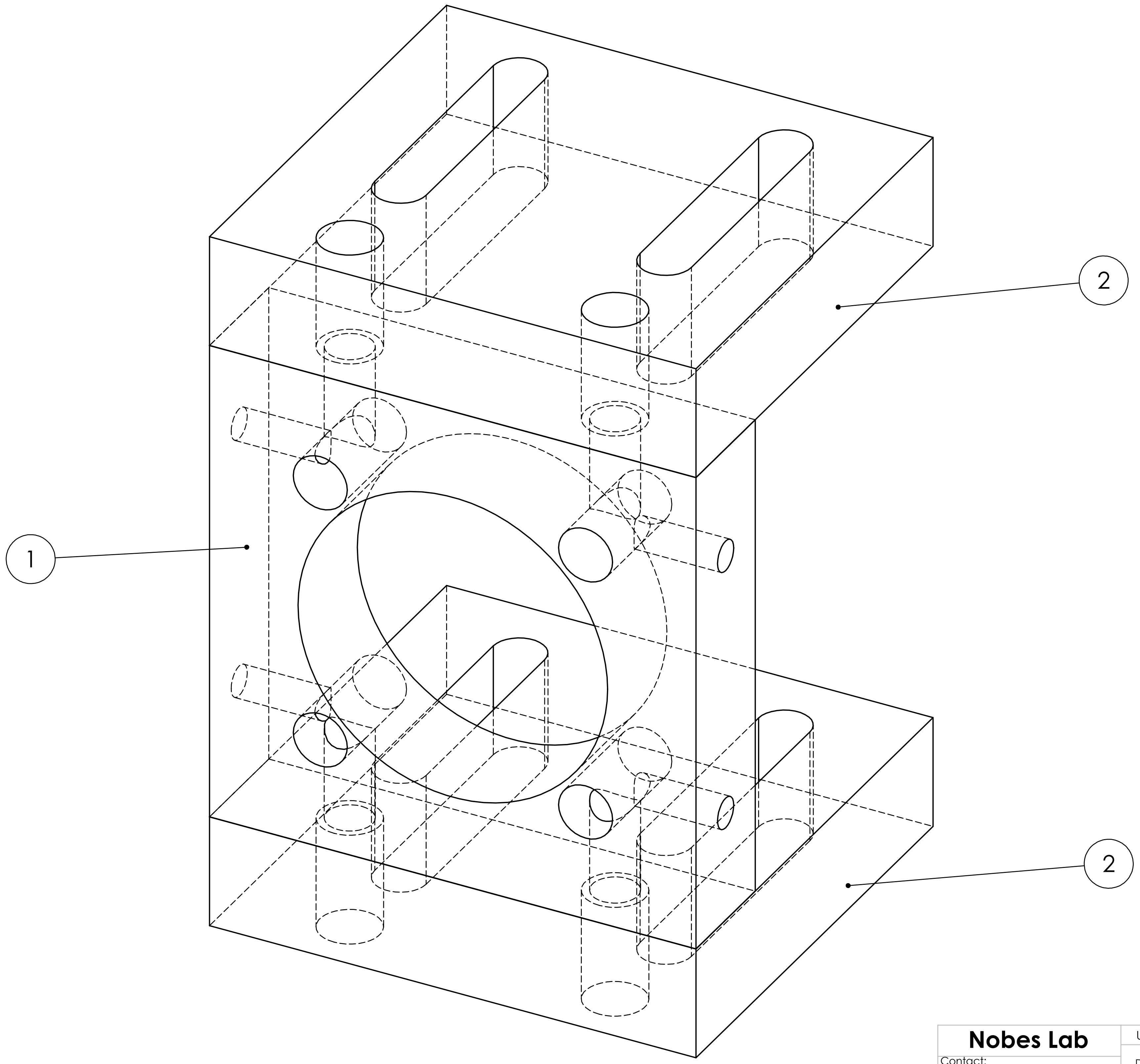
The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:
Aluminum Bottom Plate

SIZE **B**
Project:
REV 2

SCALE: 1:1
Mass: 17.80
SHEET 1 OF 1

ITEM NO.	PART NUMBER	QTY.
1	Front Plate	1
2	Side Plate	2



Nobes Lab
Contact: Farhan Ahmad
Email: farhan1@ualberta.ca
Phone: 780-492-6723
Comments:

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH µm 0.6 ✓

MATERIAL: Aluminum

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

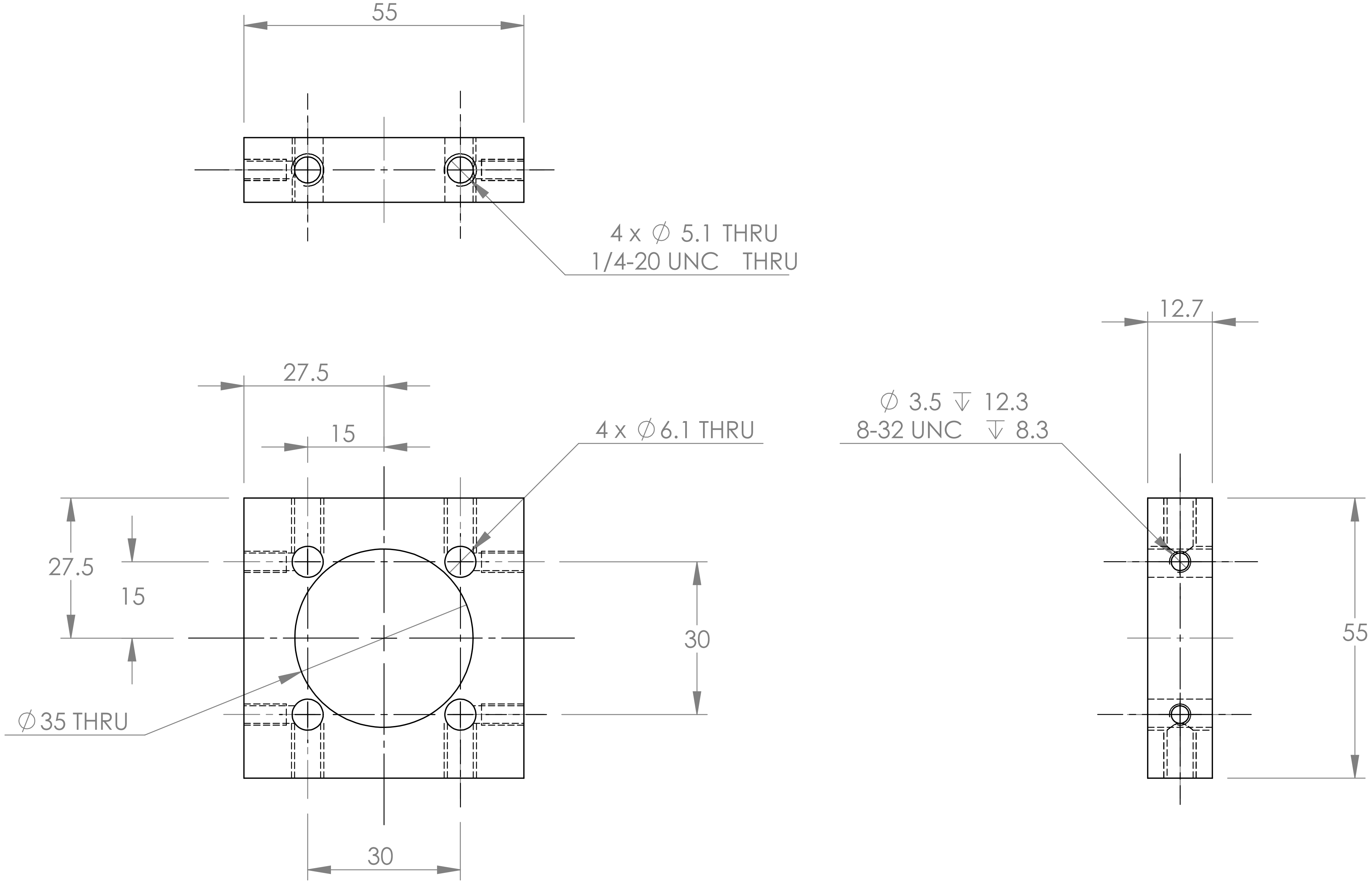
	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Cam Holder Assm

farhan1
8/24/2011 10:58:12 AM
Wednesday, August 24, 201

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:
Camera Holder Assembly

SIZE **B**
Project:
REV 3
SCALE: 2:1
Mass: 339.18
SHEET 1 OF 1



Nobes Lab

Contact: Farhan Ahmad
Email: farhan1@ualberta.ca
Phone: 780-429-6723
Comments:

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH 0.6 µm

MATERIAL: Aluminum

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Camera Holder

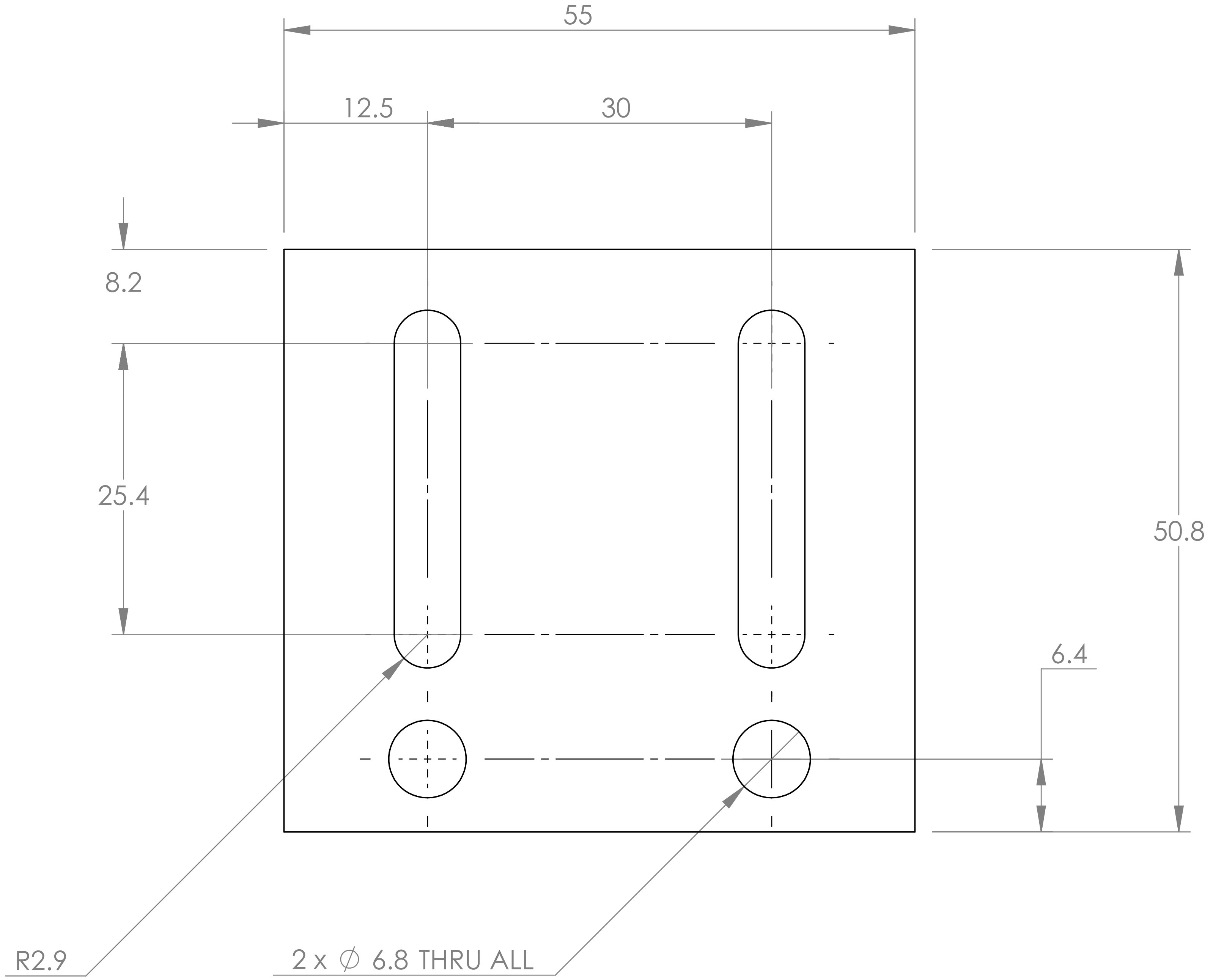
farhan1
8/24/2011 10:54:46 AM
Wednesday, August 24, 201

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:
Camera Holder Front Plate

SIZE **B**
Project:
REV 2

SCALE: 1:1
Mass: 0.14
SHEET 1 OF 1



Note: Thickness is 12.7 mm

Nobes Lab

Contact: Farhan Ahmad
Email: farhan1@ualberta.ca
Phone: 780-492-6723
Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH 0.6 µm

MATERIAL: Aluminum

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Camera Holder Part 2

farhan1
8/24/2011 10:49:51 AM
Wednesday, August 24, 2011

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:
Camera Holder Side Plate

SIZE **B**

Project:

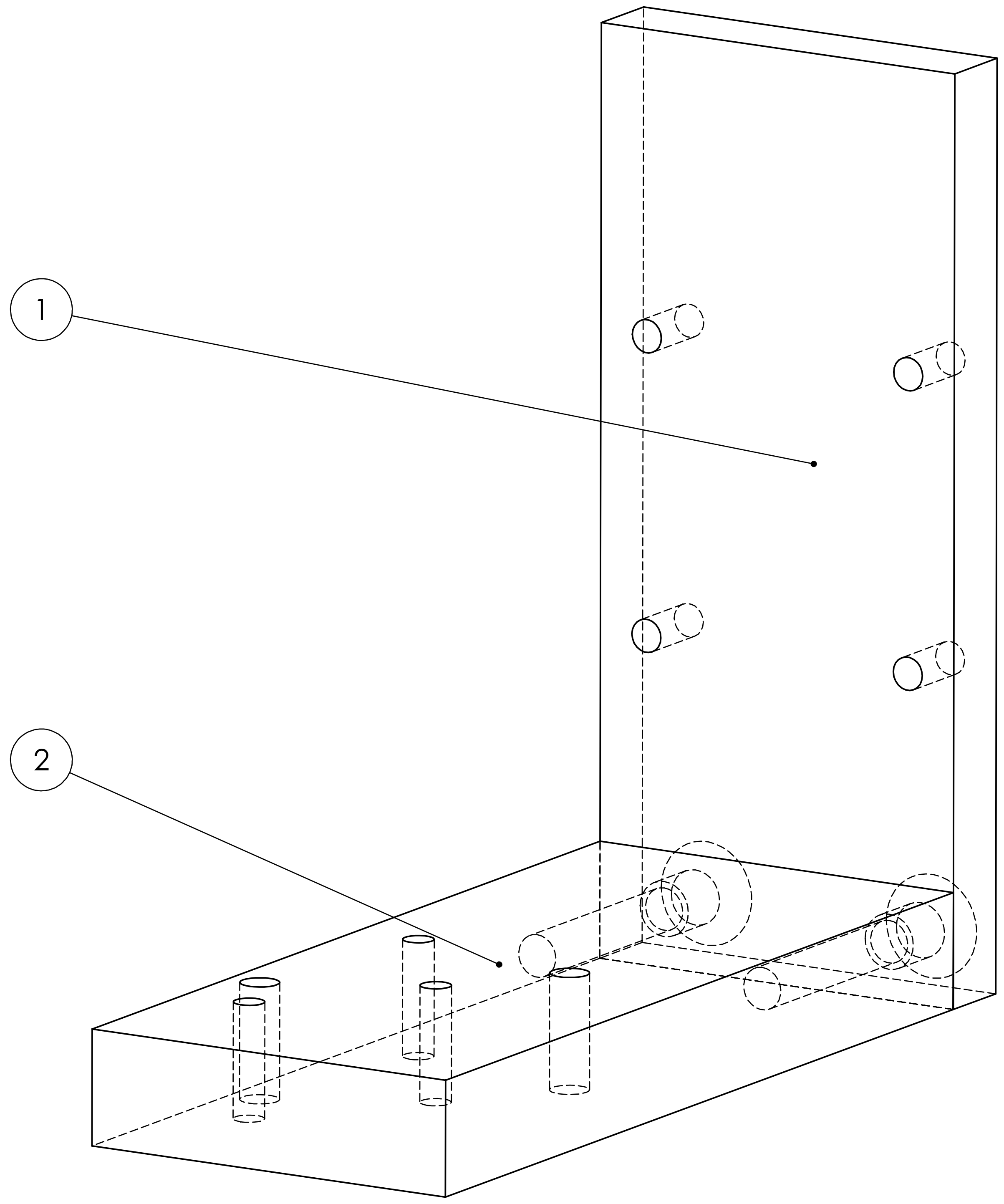
REV 2

SCALE: 2:1

Mass: 81.43

SHEET 1 OF 1

ITEM NO.	PART NUMBER	QTY.
1	Write Lase Scanner Vertical Support	1
2	Write Laser Scanner Horizontal Base	1



Nobes Lab

Contact: Farhan Ahmad

Email: farhan1@ualberta.ca

Phone: 780-492-6723

Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH µm 0.6

MATERIAL: Aluminum

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	Farhan
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Turning Platform

farhan1
8/24/2011 11:15:51 AM
Wednesday, August 24, 201

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:

Write Laser Scanner Holder

SIZE **B**

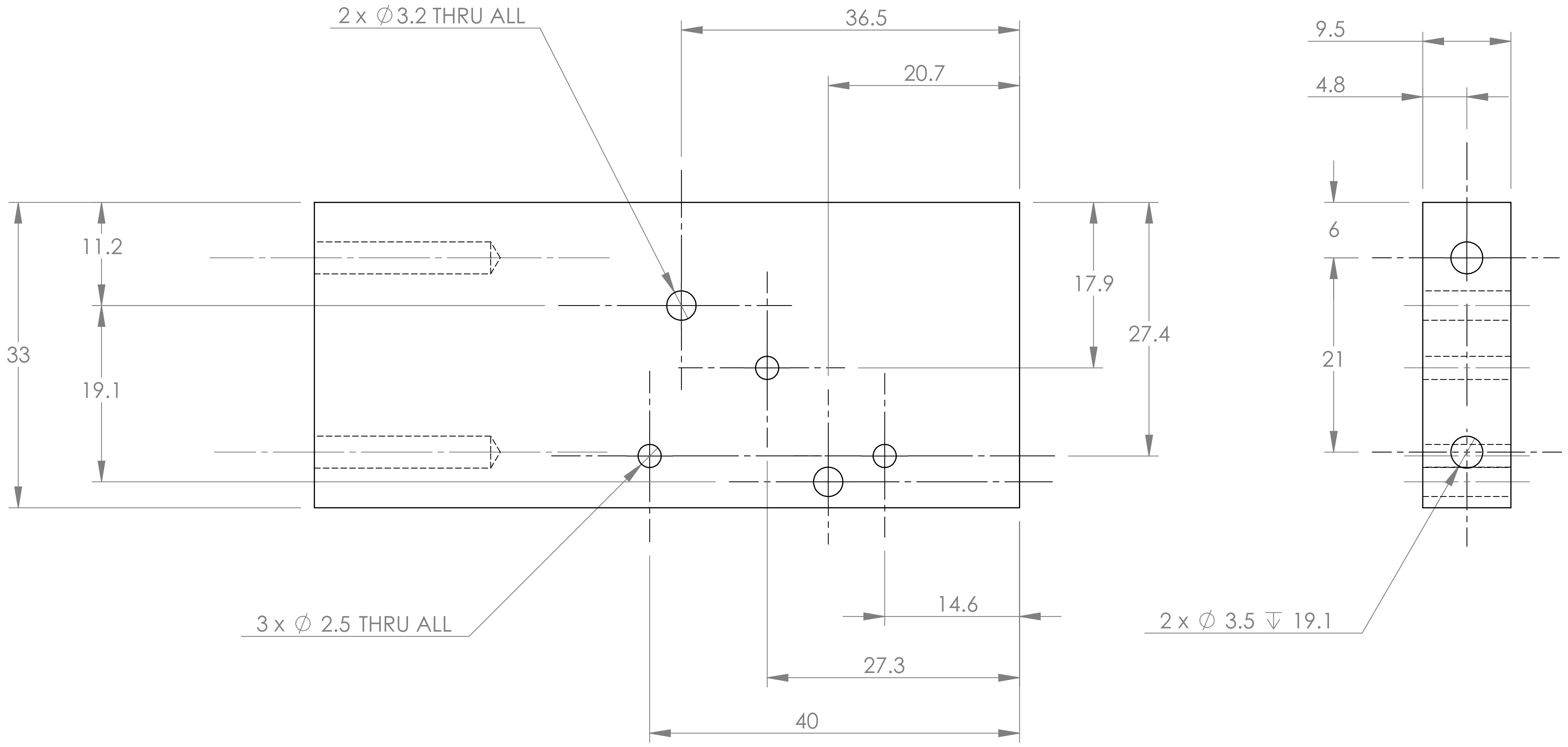
Project:

REV 2

SCALE: 2:1

Mass: 38.86

SHEET 1 OF 1



Nobes Lab

Contact: Farhan Ahmad

Email: farhan1@ualberta.ca

Phone: 780-492-6723

Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH 0.6 µm

MATERIAL: Aluminum

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Turning Platform Rev 2 Part 1

farhan1
8/24/2011 11:17:21 AM
Wednesday, August 24, 2011

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:

Write Laser Scanner
Horizontal Base

SIZE **B**

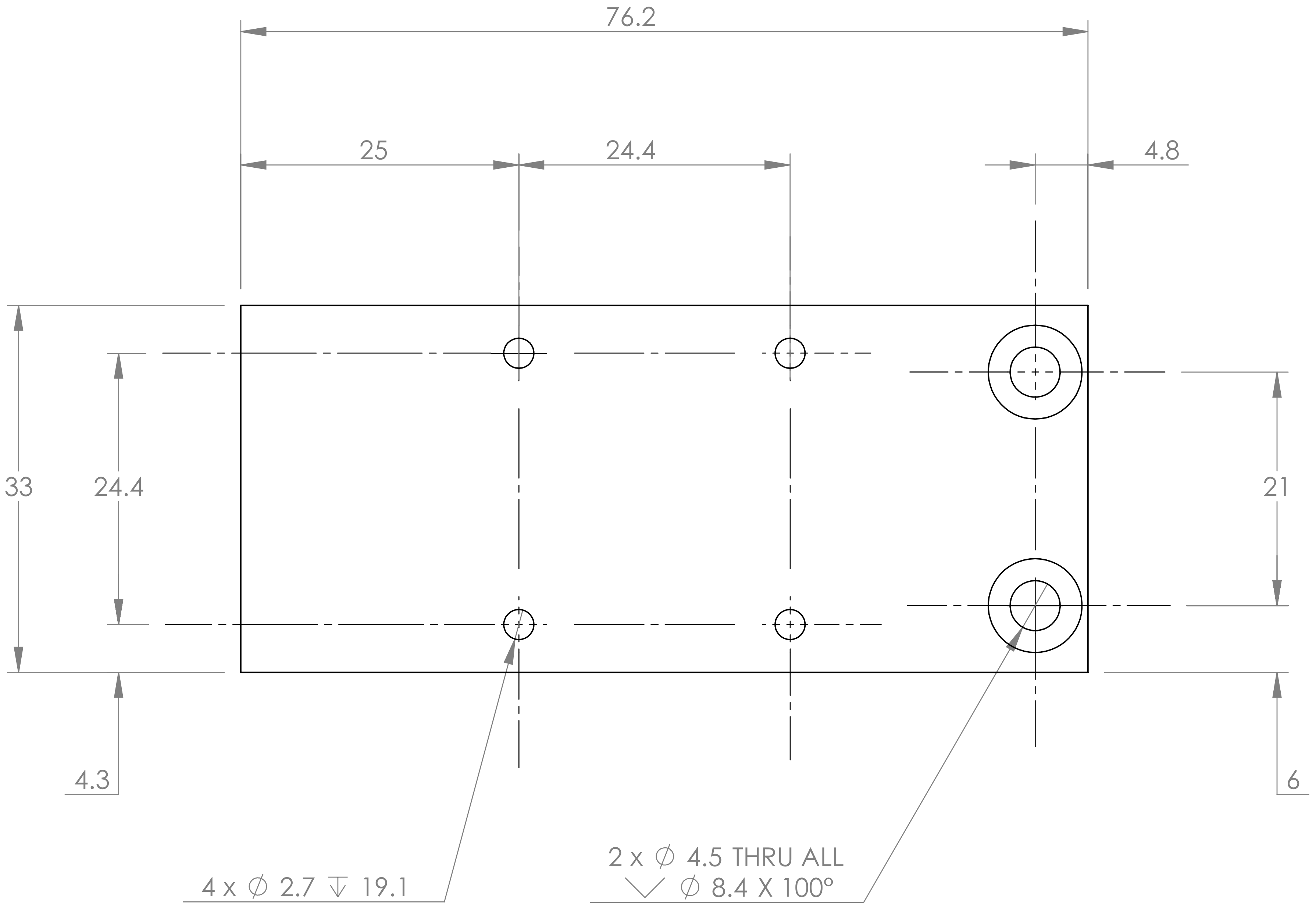
Project:

REV 2

SCALE: 2:1

Mass: 23.30

SHEET 1 OF 1



Note: Thickness is 6.4 mm

Nobes Lab

Contact: Farhan Ahmad

Email: farhan1@ualberta.ca

Phone: 780-492-6723

Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH µm 0.6

MATERIAL: Aluminum

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Vertical Support

farhan1
8/24/2011 11:20:28 AM
Wednesday, August 24, 201

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:

Write Laser Scanner
Vertical Support

SIZE **B**

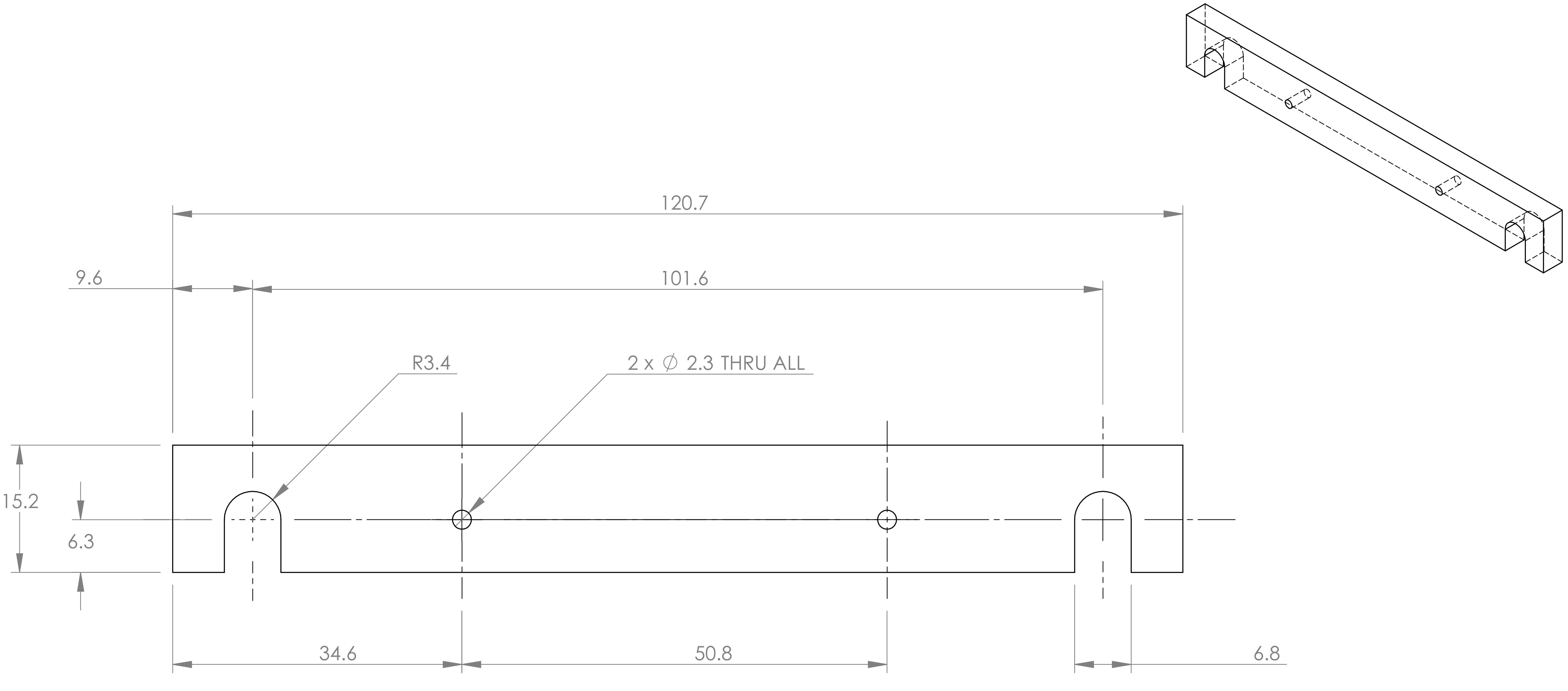
Project:

REV 2

SCALE: 2:1

Mass: 15.56

SHEET 1 OF 1



Note: Thickness is 6.4 mm

Nobes Lab

Contact: Farhan Ahmad

Email: farhan1@ualberta.ca

Phone: 780-492-6723

Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH µm 0.6

MATERIAL: Aluminum

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Custom LCPB1 v1

farhan1
8/24/2011 10:59:50 AM
Wednesday, August 24, 201

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:

Custom CLCPB1 for raising the setup

SIZE **B**

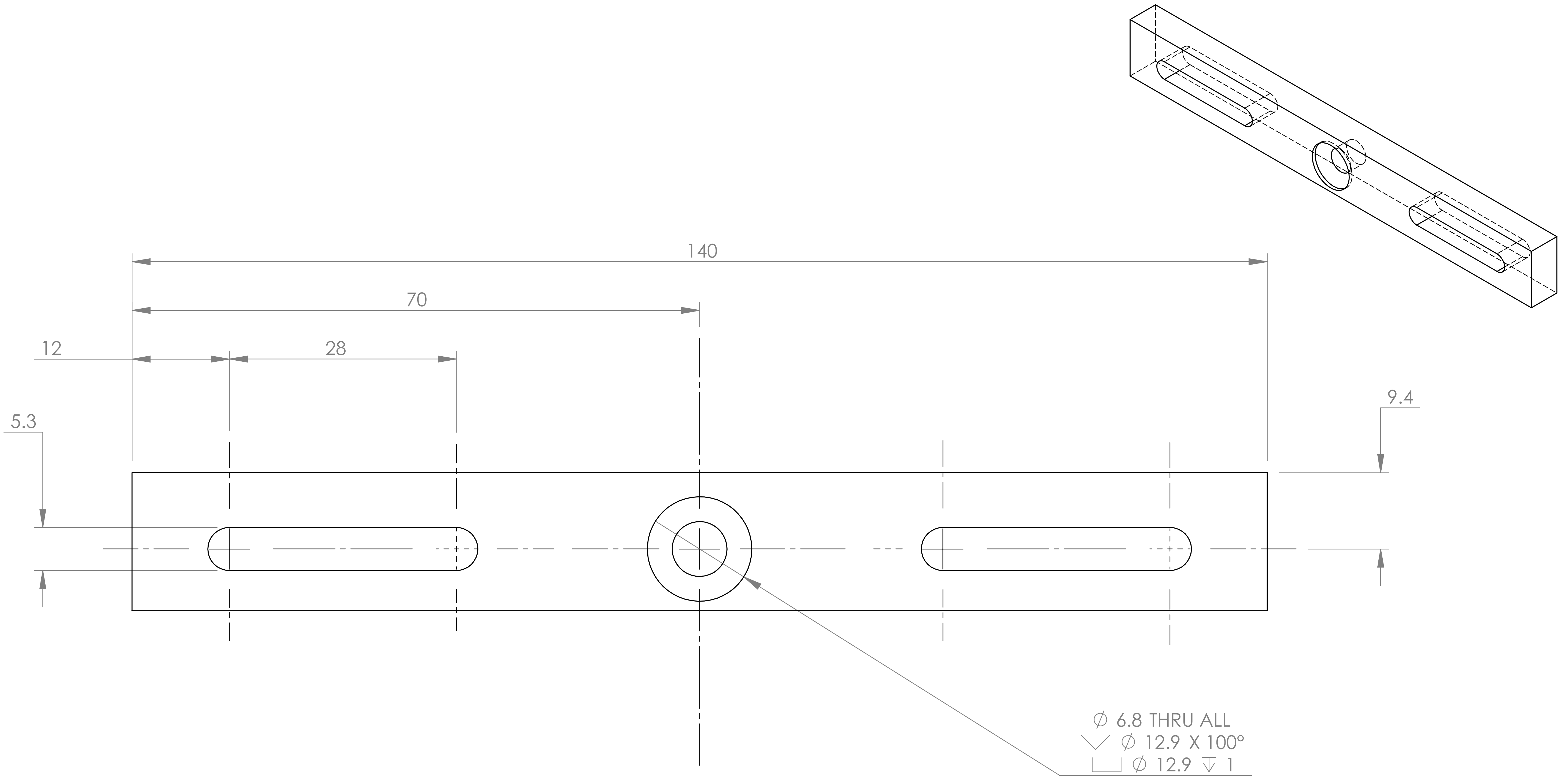
Project:

REV 1

SCALE: 2:1

Mass: 10.83

SHEET 1 OF 1



Note: Thickness is 8.9 mm

Nobes Lab

Contact: Farhan Ahmad

Email: farhan1@ualberta.ca

Phone: 780-492-6723

Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH µm 0.6 √

MATERIAL: Aluminum

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	FARHAN
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Pinhole Base Rev 1

farhan1
8/24/2011 11:09:53 AM
Wednesday, August 24, 2011

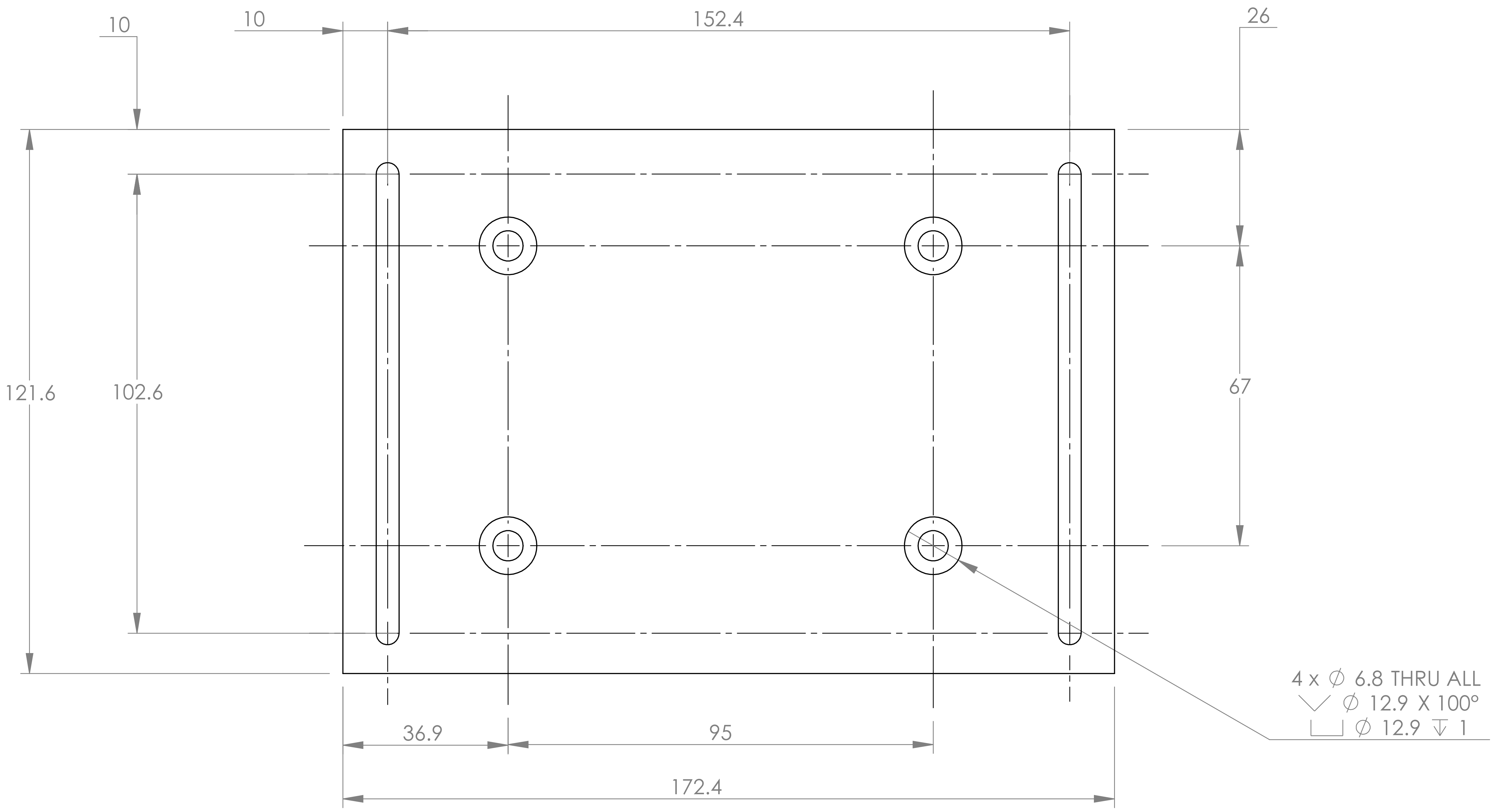
The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE:

Pinhole Base Holder

SIZE	Project:	REV
B		1

SCALE: 2:1 | Mass: 17.61 | SHEET 1 OF 1



Note: Thickness is 6.4 mm

Nobes Lab

Contact: Farhan Ahmad
Email: farhan1@ualberta.ca
Phone: 780-492-6723
Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
.X = ± 0.25
.XX = ± 0.05
.XX = ± 0.005

SURFACE FINISH µm 0.6

MATERIAL: Aluminum

Lab Location: MEC, 5-27

DO NOT SCALE DRAWING

	NAME
DRAWN	Farhan
CHECKED	DSN
ENG APPR.	
Speed Code	
MecE Job #	
File Name:	Blue Laser Holder

farhan1
8/24/2011 10:57:43 AM
Wednesday, August 24, 201

The Department of Mechanical Engineeering
UNIVERSITY OF ALBERTA

TITLE: Blue Laser Holder

SIZE **B** | Project: | REV 2

SCALE: 1:1 | Mass: 124.48 | SHEET 1 OF 1

APPENDIX B

SPECIFICATION TABLES

The detailed specifications of the equipment used for the construction and the operation of the experimental setup are provided in this appendix. Each table in the appendix presents the detailed specification of an individual component.

Table B-1: Properties of fluorescein bis-(5-carboxymethoxy-2-nitrobenzyl) ether, dipotassium salt (CMNB caged fluorescein)

Manufacturer	Molecular Probes Inc. (Eugene, OR, USA)
Catalogue No:	F7103
Chemical Formula	$C_{38}H_{24}K_2N_2O_{15}$
Molecular Weight	826.81
Solubility	High in water
Form	Solid
Package size	5 mg
Uncaging laser	~ 310 nm
Absorption maxima	~ 488 nm
Emission maxima	~ 520 nm
Storage temperature	≤-20°C

Table B-2: Properties of Uranine (Fluorescein Sodium Salt)

Manufacturer	Fisher Scientific
Product Name	Uranine
Synonyms	Fluorescein sodium salt
Chemical Formula	$C_{20}H_{10}O_5$.2Na
Molecular Weight	376.27
Solubility	Water soluble
Physical State	Powder
Odor	Odorless
Appearance	Orange

Table B-3: Properties of μ-PIV particles (Fluoro max dyed aqueous green fluorescing particles)

Manufacturer	Thermo Scientific
Product Name	Fluoro-Max dyed green aqueous fluorescent particles
Catalogue number	G0200
Dye type	Firefli fluorescent green (468/508 nm)
Size	1.9 μm
Density (Particle)	1.05 g/cm^3
Refractive index	1.59 at 589 nm (25°C)
pH	5-8
Solubility	Water soluble
Physical state	Suspension in water
Odor	Odorless
Appearance	White or colored liquid

Table B-4: Uncaging laser specification for MTV scanning system.

Manufacturer	Stanford Research Systems
Laser Model	NL100
Main Wavelength	337 nm
Laser Class	3B
Frequency	1-20 Hz for internal trigger
	0-20 Hz for external trigger
Maximum Rated Energy	170 µJ per pulse at 10Hz
Pulse Width	<3.5 ns at FWHM
Beam Size	3 x 7 mm
Beam Divergence	5 x 8 mrad
Peak Power	45 kW
Dimensions	27.9 x 9.5 x 9.5 cm

Table B-5: Fluorescing laser specification for MTV scanning system.

Manufacturer	Laserglow Technologies
Laser Model	LRS-0473-PFM-0200-05
Main Wavelength	473 nm
Beam Diameter	~2.0 mm
Peak Power	>200 mW
Beam Divergence	<1.5 mrad
M^2 Factor	<1.2
Transverse mode	TEM_{00}
MTTF	>10,000 hours

Table B-6: Ultraviolet laser scanner specification for MTV scanning system.

Manufacturer	Edmund Optics
Scanner Model	NT59-874
Mirror Substrate	1000Å thick low pinhole protected Aluminum
Mirror Aperture	3 mm
Mirror Damage Threshold	100-150 Watts/cm^2
Angular Excursion	±20º (Mechanical); ±40º (Optical)
Small Angle (0.1º) Step Response Time	160 µsec
Small Angle (20º) Step Response Time	800 µsec
Repeatability (Short term)	8 µrad
Size	3.95” L x 3.05” W x 1.21” H

Table B-7: Camera specification for MTV scanning system.

Manufacturer	PhotonFocus
Camera Model	MV-D1024E-160-CL-12
Interface	CameraLink base configuration
Technology	Progressive scan CMOS active pixel
Color Format	Monochrome
Array Size (Pixel)	1024 x 1024
Pixel Size	10.6 µm x 10.6µm
Active Optical Area	10.9 mm x 10.9 mm
Full Well Capacity	200 ke$^-$
Grey Scale Resolution	12 bit/ 10bit/ 8bit
Digital Gain	x1/ x2/ x4
Spectral Range	400 nm – 900 nm
Maximum Frame Rate	149 FPS at 10µsec exposure time.
Optical Fill Factor	35%
Characteristic Curve	Linear, LinLog, Skimming
Shutter Mode	Global shutter
Trigger Modes	Interface Trigger / I/O Trigger

Table B-8: Microscope objective specification for MTV scanning system.

Manufacturer	ThorLabs
Model	LMU-15X-NUV
Objective Type	Achromatic
Magnification	15x
Working Distance	8.5 mm
Numerical Aperture	0.32
Effective Focal Length	13 mm
Entrance Aperture	8.5 mm
Anti-reflection Coating	325-500 nm
Damage Threshold	50 MW/cm^2

Table B-9: Beam expander specifications for MTV scanning system.

Manufacturer	Laserglow Technologies
Model	X10DR-H (Hercules)
Expansion	10x
Maximum Input Diameter	2 mm
Output Aperture	20 mm
Antireflection Coating	Transmission = multi-coat 99.8%
Antireflection Coated Surface	All
Outer Dimensions	68 mm long x 60.3 mm Diameter

Table B-10: Analog output module specification for MTV scanning system.

Manufacturer	National Instruments
Model	USB-9263
Analog Output Channels	4
Measurement Type	Voltage
Maximum Voltage	10 V
Maximum Voltage Range	-10 V – 10 V
Current Drive Single	1 mA
Current Drive All	4 mA
Update Rate	100 kS/s

Table B-11: 470 nm high power LED specification for MTV scanning system.

Manufacturer	ThorLabs
Model	M470L2
Nominal Wavelength	470 nm
LED Output Beam Power	830 – 1560 mW
Collimated Beam Power	400 mW
Maximum LED Current	1600 mA
Lifetime	>50,000 hours

Table B-12: Aspheric lens specification for the LED assembly.

Manufacturer	ThorLabs
Model	ACL5040-A
Diameter	50 mm
Focal Length	40 mm
Clear Aperture	>90%
Numerical Aperture	0.554
Antireflection Coating	350 – 700 nm
Design Wavelength Tolerance	±5%

Table B-13: Excitation filter specifications for the LED assembly.

Manufacturer	Chroma Technology Corporation
Model	ET465/50m
Centre Wavelength	465 nm
FWHM	50
Diameter	25 mm R mm
Thickness	3.5 mm

Table B-14: DMLP425 Dichroic mirror specifications for the scanning MTV system.

Manufacturer	ThorLabs
Model	DMLP425
Design Wavelength	425 nm
Reflection Band	380 – 410 nm
Transmission Band	440 – 700 nm
Laser Damage Threshold	1.50 J/cm^2
Angle of Incidence	45º
Substrate Material	UV fused silica
Diameter	25.4 mm
Clear Aperture	23.0 mm
Thickness	3.2 mm

Table B-15: DMLP505 Dichroic mirror specifications for the scanning MTV system.

Manufacturer	ThorLabs
Model	DMLP505
Design Wavelength	505 nm
Reflection Band	380 – 490 nm
Transmission Band	520 – 700 nm
Laser Damage Threshold	1.50 J/cm^2
Angle of Incidence	45º
Substrate Material	UV fused silica
Diameter	25.4 mm
Clear Aperture	23.0 mm
Thickness	3.2 mm

Table B-16: E505LP long pass filter specifications for the scanning MTV system.

Manufacturer	Chroma Technology Corporation
Model	E505LP
Diameter	25 mm R mm
Thickness	5 mm

Table B-17: Field lens specification for the LED assembly.

Manufacturer	ThorLabs
Model	LB1676-A
Diameter	25.4 mm
Focal Length	100 mm
Clear Aperture	>90% of Diameter
Lens Shape	Convex/Convex
Design Wavelength	633 nm
Antireflection Coating	350 – 700 nm

Table B-18: Beam steering correction lens specification for the LED assembly.

Manufacturer	ThorLabs
Model	LB4330-UV
Diameter	25.4 mm
Focal Length	75 mm
Clear Aperture	>90% of Diameter
Lens Shape	Convex/Convex
Design Wavelength	588 nm
Antireflection Coating	290 – 370 nm

Table B-19: Specifications of the syringe pump responsible for the flow.

Manufacturer	Harvard Apparatus
Model	PHD 2000
Syringe	0.5 µl (minimum) – 140 ml (maximum)
Flow rate (with 0.5 µl syringe)	0.0001 µl/hr
Flow rate (with 140 ml syringe)	220.82 ml/min
Pusher travel rate	0.18 µl/min – 190.676 mm/min
Accuracy	±1%
Reproducibility	±0.1%
Linear force (max)	23 kg (standard) or 30 kg (high force)
Motor step/revolution of lead screw	800 – 12,800
Dimensions, H x W x D	15.9 x 22.8 x 27.9 cm
Weight	4.5 kg

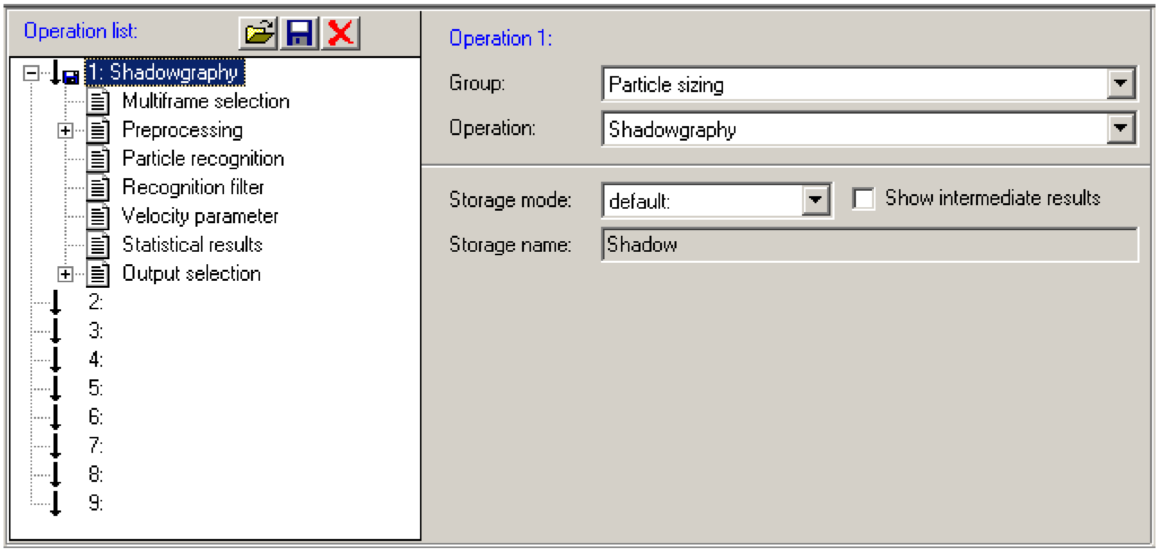
APPENDIX C

PROCESSING PARAMETERS

Appendix C-1provides the parameters used in the processing of the Photobleaching MTV images. The parameters used measures the area equivalent diameter and the velocity of the tagged regions.

Appendix C-2 provides the parameters used in the processing of Micro-Particle Image Velocimetry. The process calculates the velocity vectors based on a cross-correlation algorithm.

C-1 Photobleaching MTV Processing Parameters



Operation 1:

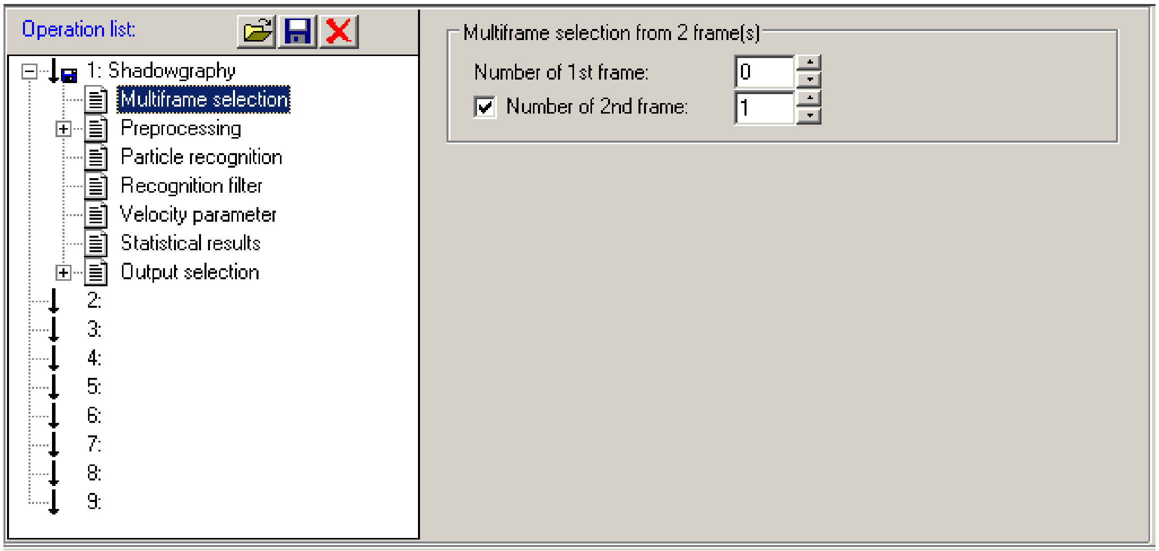
Group: Particle sizing

Operation: Shadowgraphy

Storage mode: default

□ Show intermediate results

Storage name: Shadow



Multiframe selection from 2 frame(s)

Number of 1st frame: 0

☑ Number of 2nd frame: 1

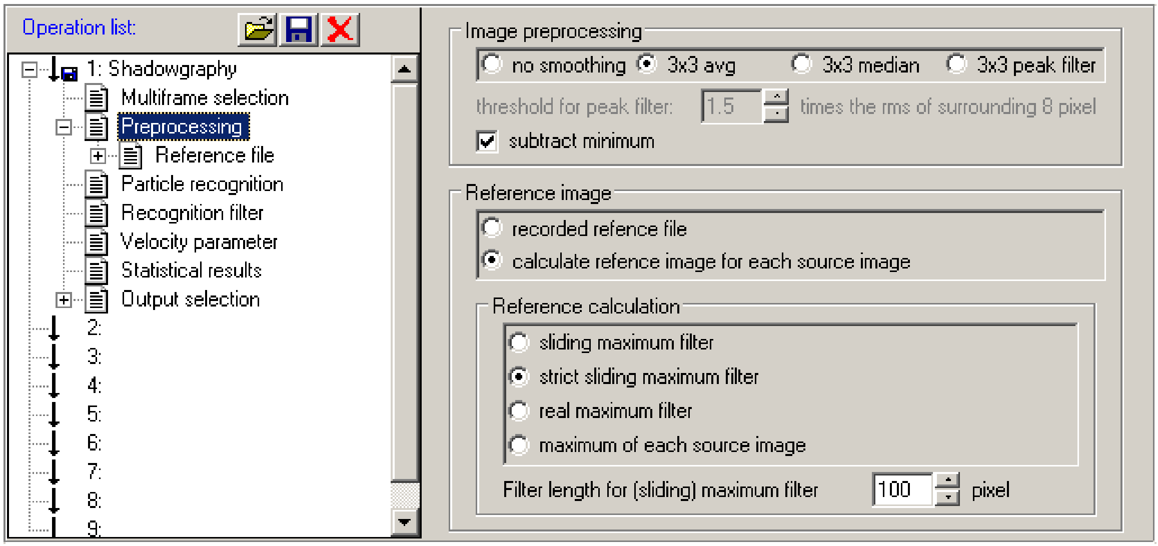


Image preprocessing

○ no smoothing ● 3x3 avg ○ 3x3 median ○ 3x3 peak filter

threshold for peak filter: 1.5 times the rms of surrounding 8 pixel

☑ subtract minimum

Reference image

○ recorded refence file

● calculate refence image for each source image

Reference calculation

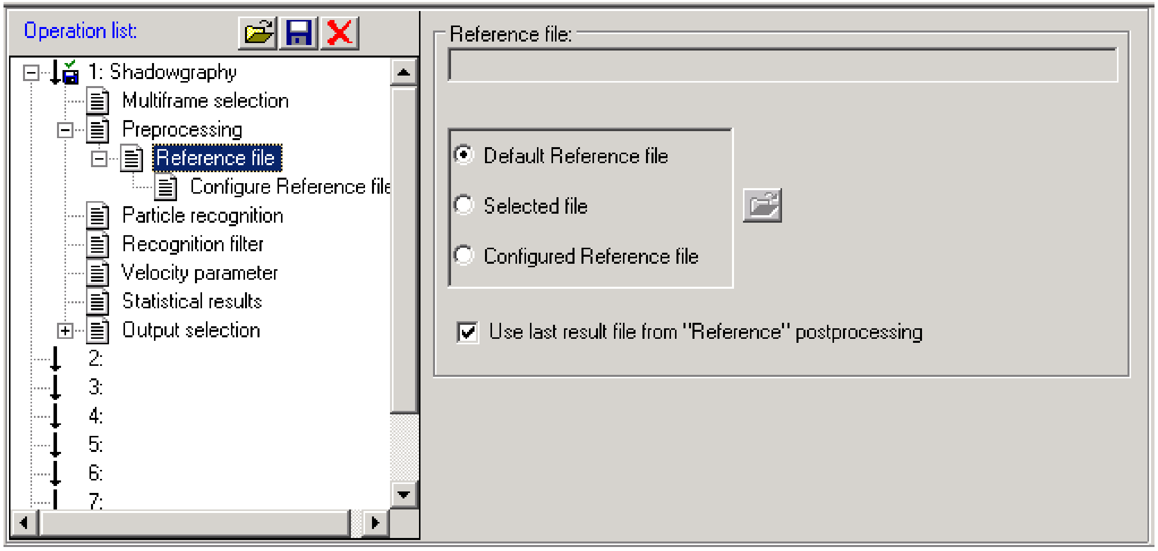
○ sliding maximum filter

● strict sliding maximum filter

○ real maximum filter

○ maximum of each source image

Filter length for (sliding) maximum filter 100 pixel



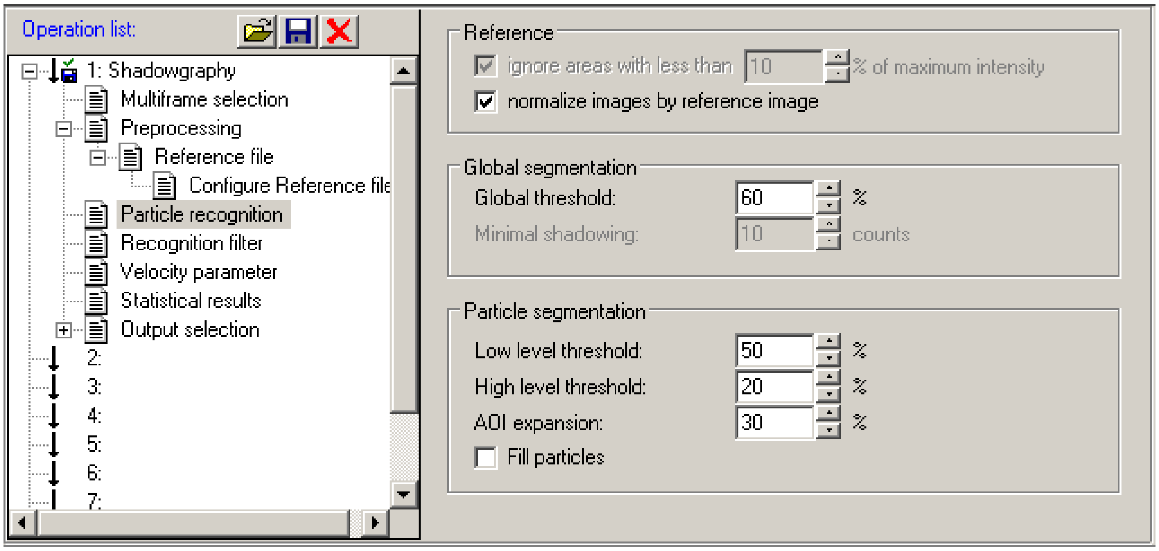
Reference file:

● Default Reference file

○ Selected file

○ Configured Reference file

☑ Use last result file from "Reference" postprocessing



Reference

☑ ignore areas with less than 10 % of maximum intensity

☑ normalize images by reference image

Global segmentation

Global threshold: 60 %

Minimal shadowing: 10 counts

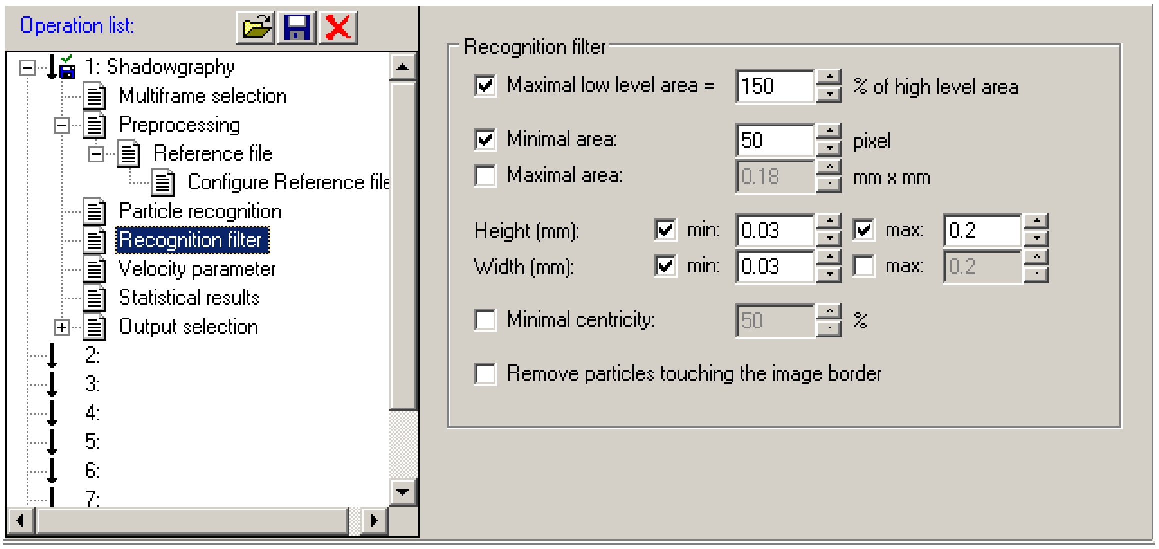
Particle segmentation

Low level threshold: 50 %

High level threshold: 20 %

AOI expansion: 30 %

□ Fill particles



Recognition filter

☑ Maximal low level area = 150 % of high level area

☑ Minimal area: 50 pixel

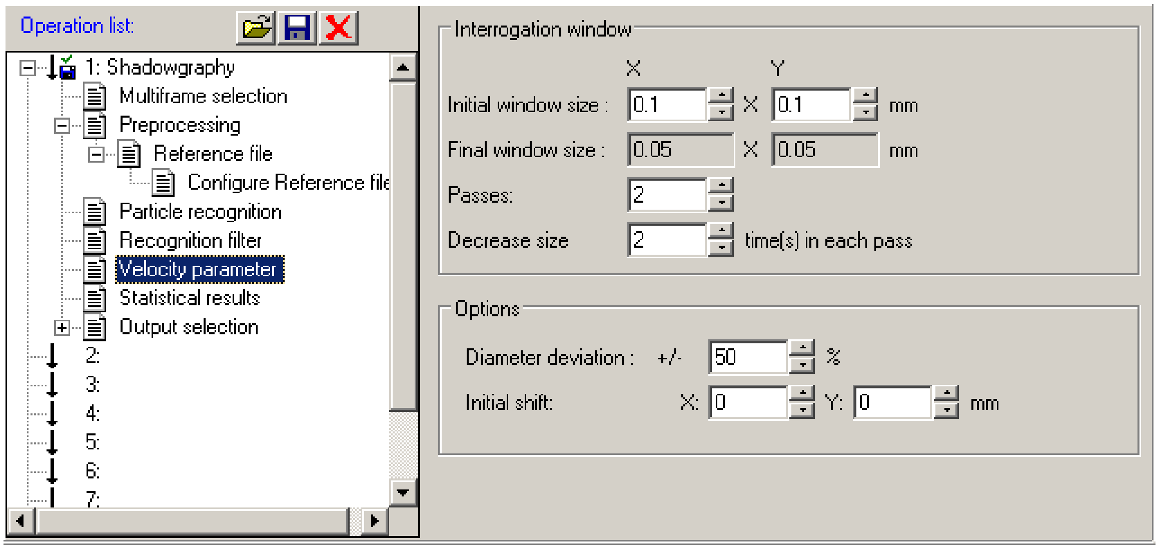
□ Maximal area: 0.18 mm x mm

Height (mm): ☑ min: 0.03 ☑ max: 0.2

Width (mm): ☑ min: 0.03 □ max: 0.2

□ Minimal centricity: 50 %

□ Remove particles touching the image border



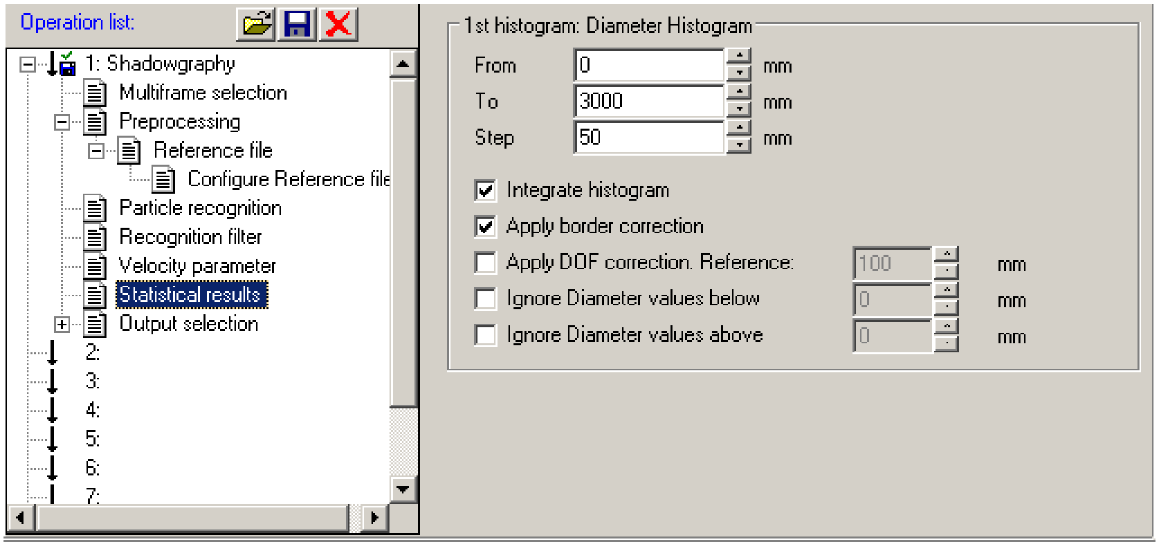
Interrogation window

	X		Y	
Initial window size :	0.1	X	0.1	mm
Final window size :	0.05	X	0.05	mm
Passes:	2			
Decrease size	2	time(s) in each pass		

Options

Diameter deviation : +/- 50 %

Initial shift: X: 0 Y: 0 mm



1st histogram: Diameter Histogram

From	0	mm
To	3000	mm
Step	50	mm

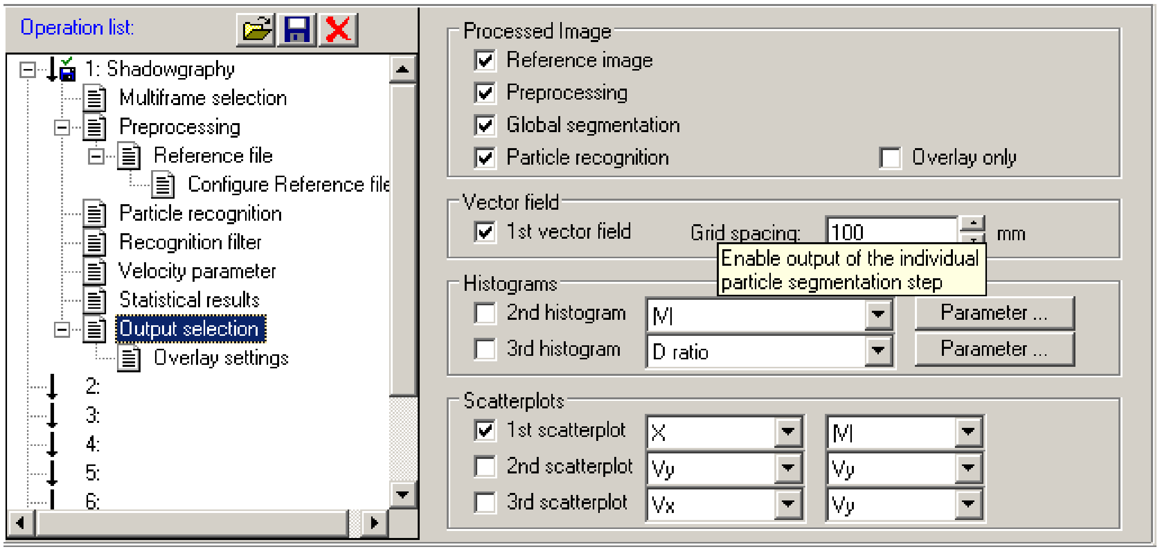
☑ Integrate histogram

☑ Apply border correction

☐ Apply DOF correction. Reference: 100 mm

☐ Ignore Diameter values below 0 mm

☐ Ignore Diameter values above 0 mm



Operation list:

- 1: Shadowgraphy
  - Multiframe selection
  - Preprocessing
    - Reference file
      - Configure Reference file
  - Particle recognition
  - Recognition filter
  - Velocity parameter
  - Statistical results
  - Output selection
    - Overlay settings
- 2:
- 3:
- 4:
- 5:
- 6:

Processed Image

☑ Reference image

☑ Preprocessing

☑ Global segmentation

☑ Particle recognition □ Overlay only

Vector field

☑ 1st vector field Grid spacing: 100 mm

Enable output of the individual particle segmentation step

Histograms

□ 2nd histogram M Parameter ...

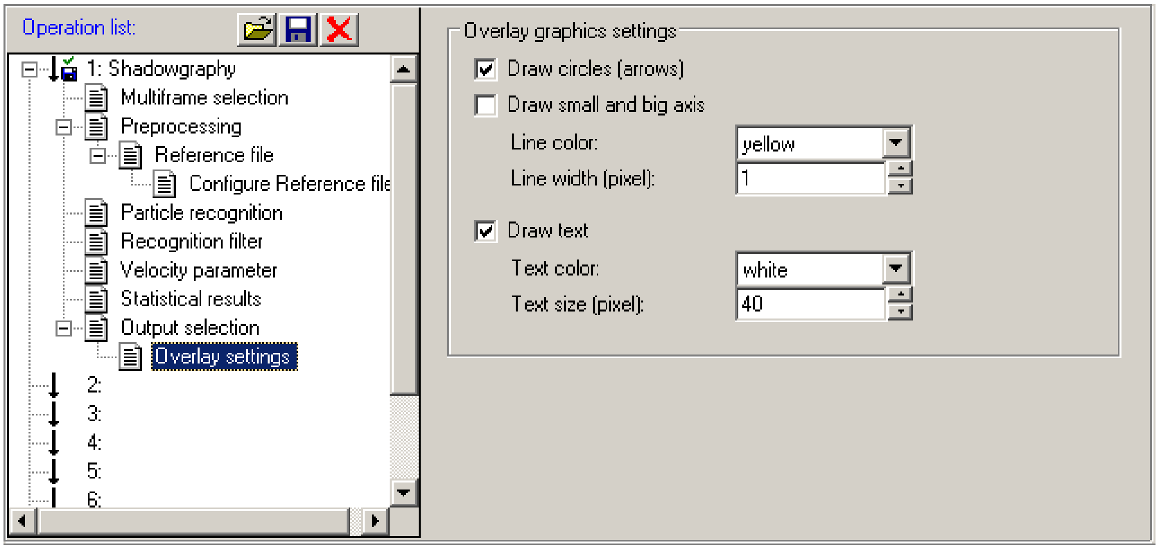
□ 3rd histogram D ratio Parameter ...

Scatterplots

☑ 1st scatterplot X M

□ 2nd scatterplot Vy Vy

□ 3rd scatterplot Vx Vy



Operation list:

- 1: Shadowgraphy
  - Multiframe selection
  - Preprocessing
    - Reference file
      - Configure Reference file
  - Particle recognition
  - Recognition filter
  - Velocity parameter
  - Statistical results
  - Output selection
    - Overlay settings
- 2:
- 3:
- 4:
- 5:
- 6:

Overlay graphics settings

☑ Draw circles (arrows)

□ Draw small and big axis

Line color: yellow

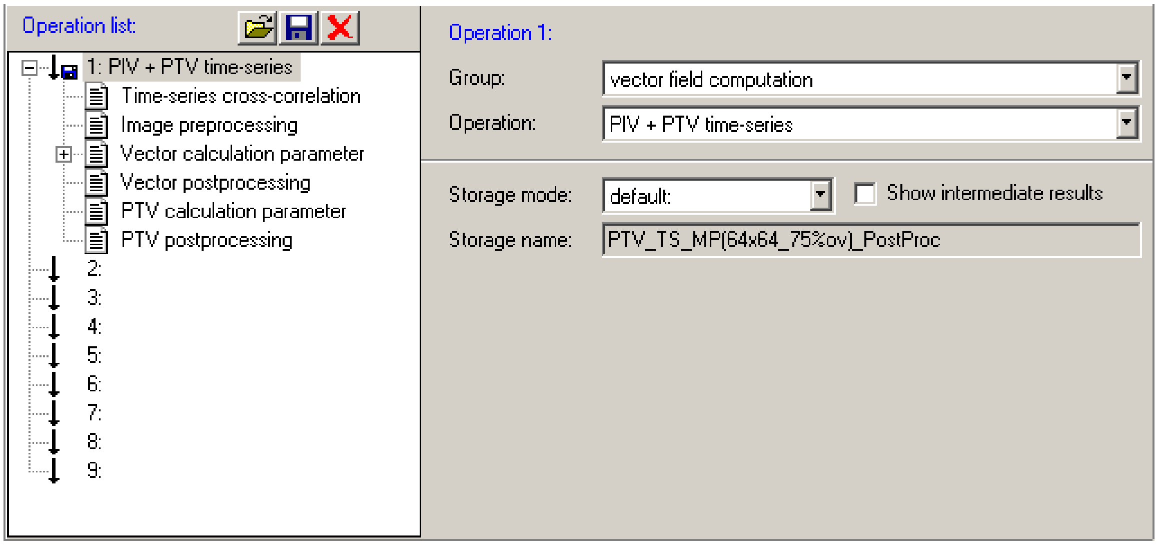
Line width (pixel): 1

☑ Draw text

Text color: white

Text size (pixel): 40

C-2 *μ- PIV/PTV Processing Parameters*



Operation list:

- 1: PIV + PTV time-series
  - Time-series cross-correlation
  - Image preprocessing
  - Vector calculation parameter
  - Vector postprocessing
  - PTV calculation parameter
  - PTV postprocessing
- 2:
- 3:
- 4:
- 5:
- 6:
- 7:
- 8:
- 9:

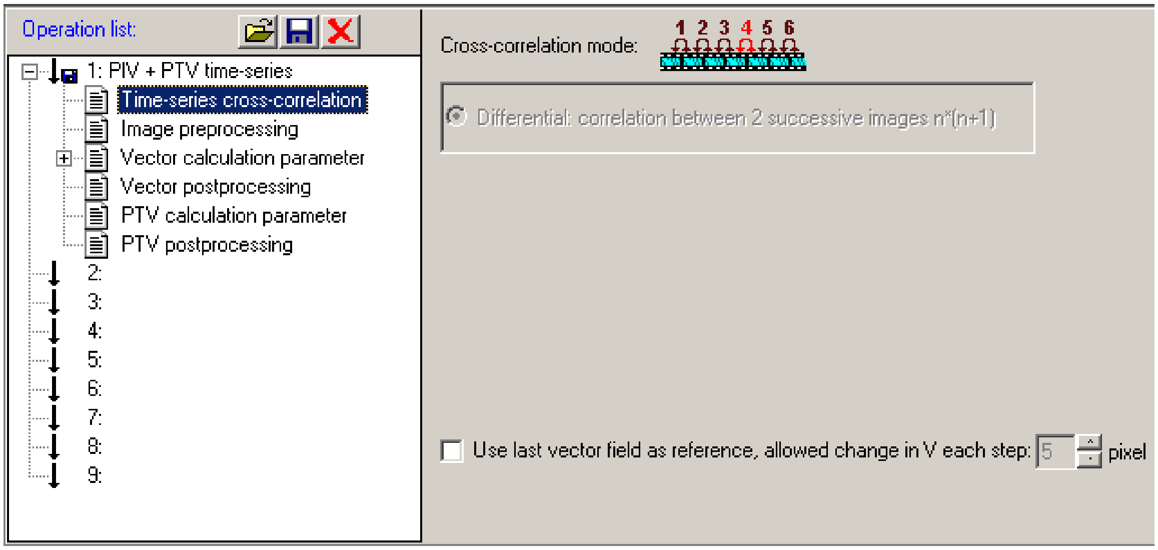
Operation 1:

Group: vector field computation

Operation: PIV + PTV time-series

Storage mode: default: ☐ Show intermediate results

Storage name: PTV_TS_MP(64x64_75%ov)_PostProc



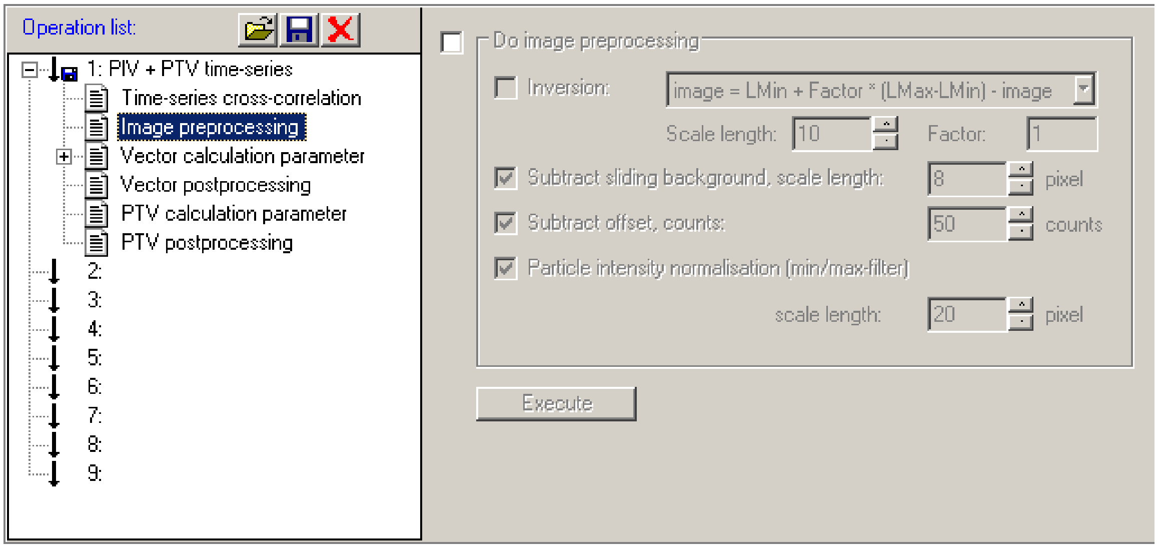
Operation list:

- 1: PIV + PTV time-series
  - Time-series cross-correlation
  - Image preprocessing
  - Vector calculation parameter
  - Vector postprocessing
  - PTV calculation parameter
  - PTV postprocessing
- 2:
- 3:
- 4:
- 5:
- 6:
- 7:
- 8:
- 9:

Cross-correlation mode:

● Differential: correlation between 2 successive images n*(n+1)

☐ Use last vector field as reference, allowed change in V each step: 5 pixel



Operation list:

- 1: PIV + PTV time-series
  - Time-series cross-correlation
  - Image preprocessing
  - Vector calculation parameter
  - Vector postprocessing
  - PTV calculation parameter
  - PTV postprocessing
- 2:
- 3:
- 4:
- 5:
- 6:
- 7:
- 8:
- 9:

□ Do image preprocessing

□ Inversion: image = LMin + Factor * (LMax-LMin) - image

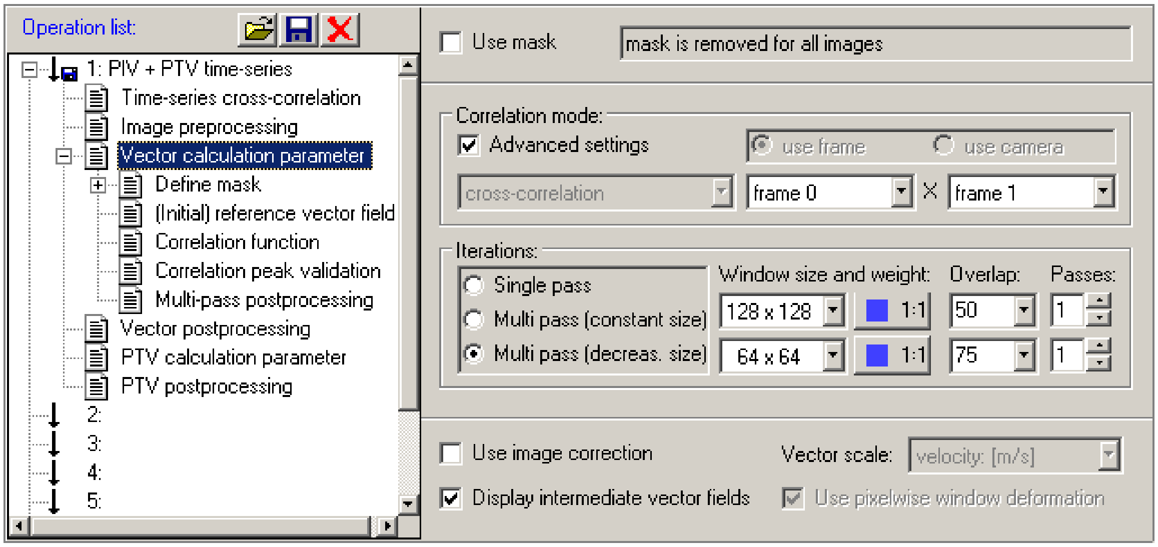
Scale length: 10 Factor: 1

☑ Subtract sliding background, scale length: 8 pixel

☑ Subtract offset, counts: 50 counts

☑ Particle intensity normalisation (min/max-filter)

scale length: 20 pixel



Operation list:

- 1: PIV + PTV time-series
  - Time-series cross-correlation
  - Image preprocessing
  - Vector calculation parameter
    - Define mask
    - (Initial) reference vector field
    - Correlation function
    - Correlation peak validation
    - Multi-pass postprocessing
  - Vector postprocessing
  - PTV calculation parameter
  - PTV postprocessing
- 2:
- 3:
- 4:
- 5:

□ Use mask mask is removed for all images

Correlation mode:

☑ Advanced settings ● use frame ○ use camera

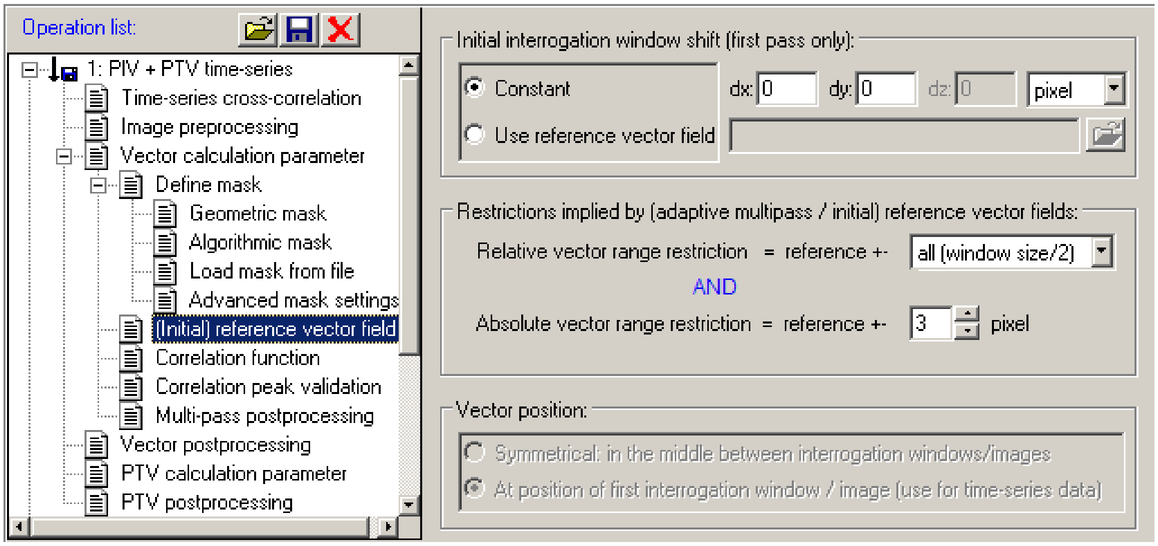
cross-correlation frame 0 X frame 1

Iterations:

	Window size and weight:		Overlap:	Passes:
○ Single pass				
○ Multi pass (constant size)	128 x 128	🟦 1:1	50	1
● Multi pass (decreas. size)	64 x 64	🟦 1:1	75	1

□ Use image correction Vector scale: velocity: [m/s]

☑ Display intermediate vector fields ☑ Use pixelwise window deformation



Initial interrogation window shift (first pass only):

● Constant dx: 0 dy: 0 dz: 0 pixel

○ Use reference vector field

Restrictions implied by (adaptive multipass / initial) reference vector fields:

Relative vector range restriction = reference +- all (window size/2)

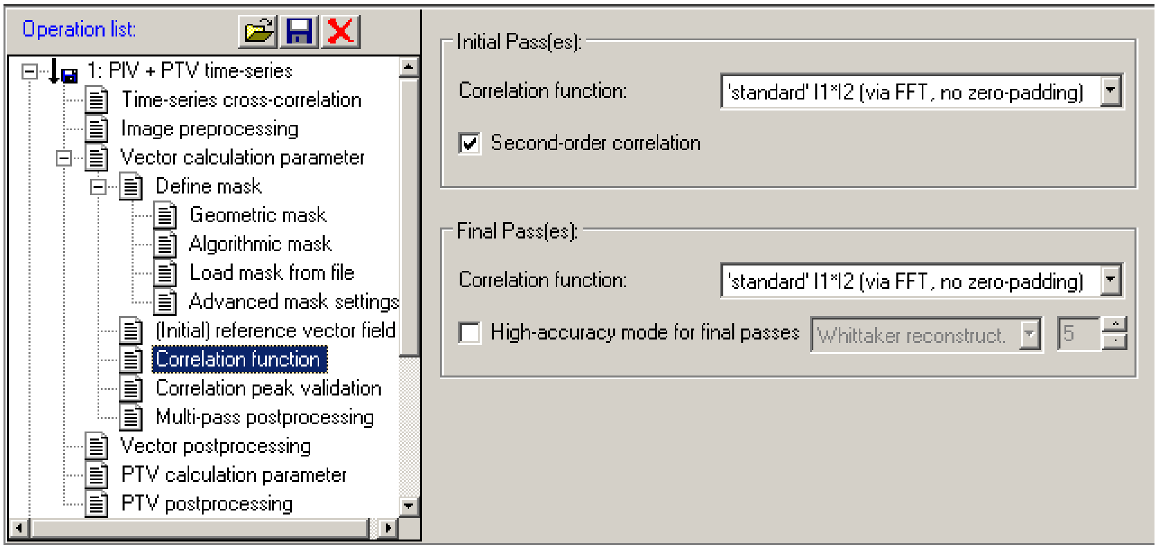
AND

Absolute vector range restriction = reference +- 3 pixel

Vector position:

○ Symmetrical: in the middle between interrogation windows/images

● At position of first interrogation window / image (use for time-series data)



Initial Pass(es):

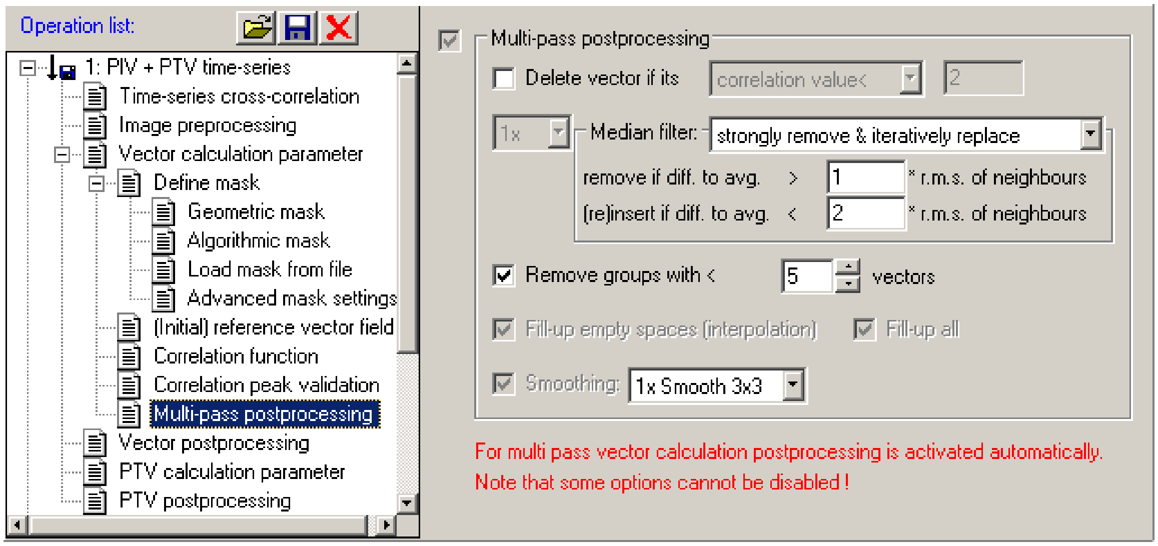
Correlation function: 'standard' I1*I2 (via FFT, no zero-padding)

☑ Second-order correlation

Final Pass(es):

Correlation function: 'standard' I1*I2 (via FFT, no zero-padding)

□ High-accuracy mode for final passes Whittaker reconstruct. 5



☑ Multi-pass postprocessing

☐ Delete vector if its | correlation value< | 2

1x | Median filter: | strongly remove & iteratively replace

remove if diff. to avg. > 1 * r.m.s. of neighbours

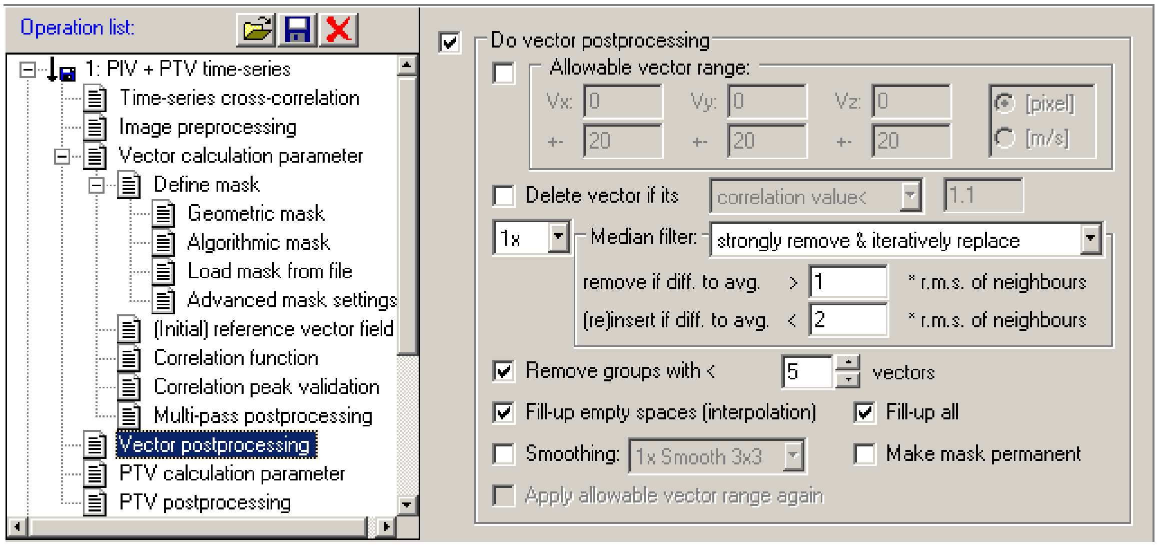
(re)insert if diff. to avg. < 2 * r.m.s. of neighbours

☑ Remove groups with < 5 vectors

☑ Fill-up empty spaces (interpolation) ☑ Fill-up all

☑ Smoothing: 1x Smooth 3x3

For multi pass vector calculation postprocessing is activated automatically.
Note that some options cannot be disabled !



☑ Do vector postprocessing

☐ Allowable vector range:

Vx: 0 Vy: 0 Vz: 0 ● [pixel]

+- 20 +- 20 +- 20 ○ [m/s]

☐ Delete vector if its | correlation value< | 1.1

1x | Median filter: | strongly remove & iteratively replace

remove if diff. to avg. > 1 * r.m.s. of neighbours

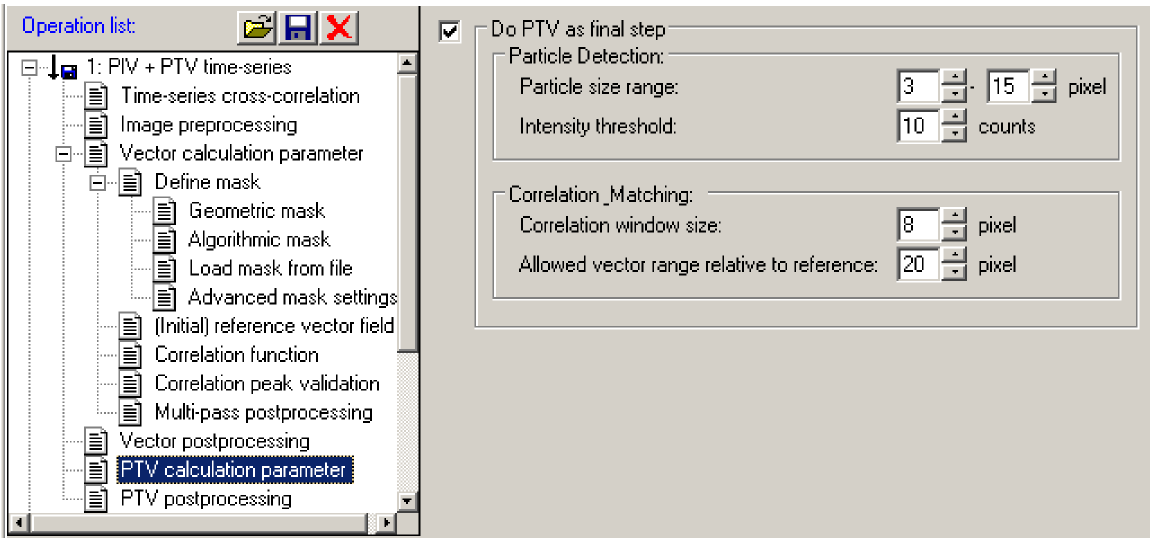
(re)insert if diff. to avg. < 2 * r.m.s. of neighbours

☑ Remove groups with < 5 vectors

☑ Fill-up empty spaces (interpolation) ☑ Fill-up all

☐ Smoothing: 1x Smooth 3x3 ☐ Make mask permanent

☐ Apply allowable vector range again



☑ Do PTV as final step

Particle Detection:

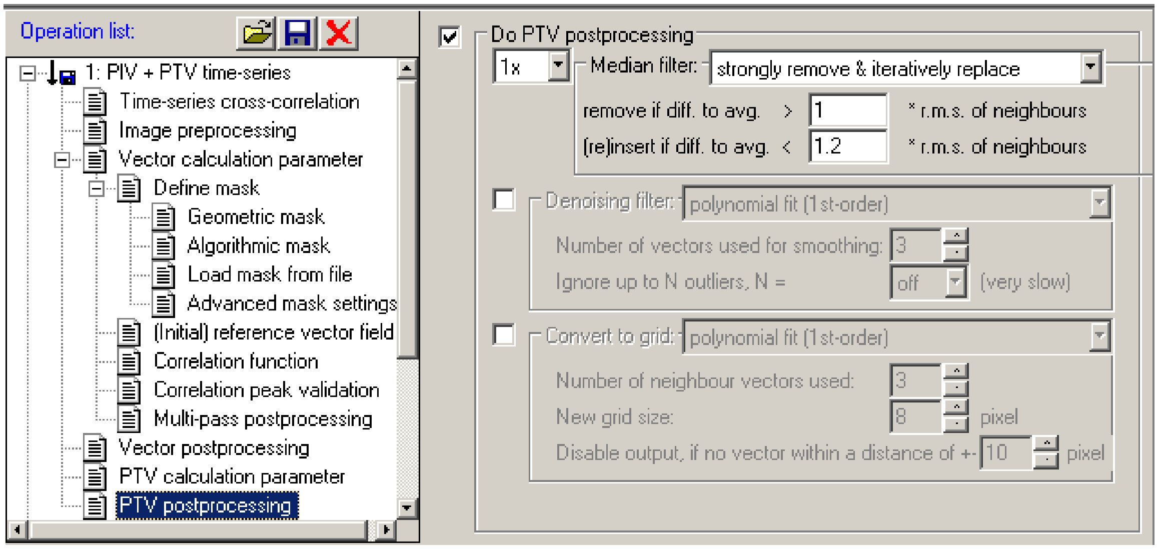
Particle size range: 3 - 15 pixel

Intensity threshold: 10 counts

Correlation Matching:

Correlation window size: 8 pixel

Allowed vector range relative to reference: 20 pixel



☑ Do PTV postprocessing

1x Median filter: strongly remove & iteratively replace

remove if diff. to avg. > 1 * r.m.s. of neighbours

(re)insert if diff. to avg. < 1.2 * r.m.s. of neighbours

☐ Denoising filter: polynomial fit (1st-order)

Number of vectors used for smoothing: 3

Ignore up to N outliers, N = off (very slow)

☐ Convert to grid: polynomial fit (1st-order)

Number of neighbour vectors used: 3

New grid size: 8 pixel

Disable output, if no vector within a distance of +- 10 pixel

APPENDIX D

LABWINDOWS CVI CODE

The LabWindows CVI code operating behind the graphical user interface (GUI) described in section 4.2 is provided in this appendix. The programming code is written based on C programming language.

```c
/************************************************************/
/*                                                          */
/* Main program file for the two-dimensional scanning       */
/* molecular tagging velocimetry for microscale flows       */
/* project                                                  */
/*                                                          */
/*Programmers: Farhan Ahmad and David S. Nobes              */
/*                                                          */
/*Date:              Version: Comment                       */
/*20 August 2009     V_9                                    */
/*                                                          */
/*                                                          */
/*Written to control sequential operation of                */
/*National Instruments NI-9263 (DAQ)                        */
/*PhotonFocus MV-D1024E-160-CL-CMOS (Camera)                */
/*Display and save image data                               */
/*                                                          */
/************************************************************/

/*********************************************************************************
*----------------------------------HEADER FILES----------------------------------*
**********************************************************************************/

#include "Windows.h"
#include <formatio.h>
#include <userint.h>
#include <cvirte.h>
#include <ansi_c.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdio.h>
#include <utility.h>
#include "niimaq.h"
#include "nivision.h"
#include "easytab.h"
#include "NIDAQmx.h"
#include "daqmxioctrl.h"
#include "sequence.h"
#include "api.h"
#include "pfcam.h"
#include "common.h"
#include "pftypes.h"
#include "platform.h"
#include <analysis.h>
#include "radioGroup.h"
#include "callserial.h"

#define PORT 0
#define NUMCHANS 4
#define DAQmxErrChk(functionCall) { if( DAQmxFailed(error=(functionCall)) )
        { goto Error; } }

/*********************************************************************************
*----------------------------------GLOBAL VARIABLES------------------------------*
**********************************************************************************/
static int panelHandle, imageHandle, trigHandle, cameraHandle,
           exposureHandle, windowHandle, triggerHandle, dataoutputHandle,
           characteristicsHandle, correctionHandle, infoHandle;
static const char * DaysOfWeek[] = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };
static INTERFACE_ID iid;
static SESSION_ID sid;
static Image** images = NULL;
static Image*  image_s = NULL;
static Image*  image_g = NULL;
static int imageArraySize = 0;
static AVISession aviDest;
static char proj_dir[MAX_PATHNAME_LEN];
static char file_name[MAX_PATHNAME_LEN];
static int gRunning;
static Image*  image_fun = NULL;
ToolWindowOptions myToolOptions;
HWND ImageWindow, ParentWindow;
int port, LinLog, Res, Gain, OutputMode, Value1, Value2, Time1, Time2,
    Skim, Trig, TrigLevel, AverageValue, CounterImageUpdate,
    MissedTriggerCounter, BlkOffset;
float ExpTime, FramTime, TrigDelay, FrameRateUpdate, StrobeDelay, StrobePulseWidth;

/*********************************************************************************
*----------------------------------GLOBAL ARRAYS----------------------------------*
**********************************************************************************/

static int ColorArray[16] = {VAL_RED, VAL_GREEN, VAL_BLUE, VAL_CYAN, VAL_MAGENTA,
                             VAL_YELLOW, VAL_DK_RED, VAL_DK_BLUE, VAL_DK_GREEN,
                             VAL_DK_CYAN,VAL_DK_MAGENTA, VAL_DK_YELLOW,
                             VAL_LT_GRAY, VAL_DK_GRAY, VAL_WHITE, VAL_GRAY};
static int ctl[] = { TRIGGER_FREE_RUNNING, TRIGGER_INTERFACE_TRIGGER, TRIGGER_IO_TRIGGER};
static int expdef[] = {TRIGGER_BY_CAMERA, TRIGGER_TRIG_PULSE};

/*********************************************************************************
*----------------------------------GLOBAL FUNCTIONS------------------------------*
**********************************************************************************/

static void GenStepWave(int numElements, double amplitude, double frequency,
                        double *phase, double stepWave[]);
static void LoadDataFile(int numElements, int numGroups, double XData[]);
void        IMAQ_CALLBACK visionWindowCallback (WindowEventType event, int windowNumber, Tool tool,
                                                Rect rect);

/*********************************************************************************
*----------------------------------MAIN PROGRAM----------------------------------*
**********************************************************************************/

int main (int argc, char *argv[])
{
    int     MainTabCtrl, CameraTabCtrl;
    if (InitCVIRTE (0, argv, 0) == 0)
        return -1;
    if ((panelHandle = LoadPanel (0, "sequence.uir", PANEL)) < 0)
        return -1;

    MainTabCtrl = EasyTab_ConvertFromCanvas(panelHandle, PANEL_CANVAS);
    EasyTab_LoadPanels(panelHandle, MainTabCtrl, 1, "sequence.uir",
                  __CVIUserHInst, IMG_PANEL, &imageHandle, TRIG_PANEL, &trigHandle, CAM_PANEL,
                  &cameraHandle, 0);

    CameraTabCtrl = EasyTab_ConvertFromCanvas(cameraHandle, CAM_PANEL_CAM_CANVAS);
    EasyTab_LoadPanels(cameraHandle, CameraTabCtrl, 1, "sequence.uir",
                  __CVIUserHInst, EXPOSURE, &exposureHandle, WINDOW, &windowHandle, TRIGGER,
                  &triggerHandle, DATAOUTPUT, &dataoutputHandle, CHARAC, &characteristicsHandle,
                  CORRECTION, &correctionHandle, INFO, &infoHandle, 0);

    NIDAQmx_NewPhysChanAOCtrl (trigHandle, TRIG_PANEL_CHANNEL, 1);
    ParentWindow = FindWindow (NULL, "Camera Panel");
    ImageWindow = (HWND)imaqGetSystemWindowHandle(0);
    SetWindowLong(ImageWindow, GWL_STYLE, WS_CHILDWINDOW);
    GetPanelAttribute (panelHandle, ATTR_SYSTEM_WINDOW_HANDLE, (int*)&ParentWindow);
    SetWindowLong(ImageWindow, GWL_STYLE, WS_CLIPSIBLINGS);
    SetParent((HWND) ImageWindow, (HWND) ParentWindow);

    SetCtrlAttribute(panelHandle, PANEL_DECORATION_BLUE, ATTR_FRAME_COLOR, VAL_BLUE);
    SetCtrlAttribute(trigHandle, TRIG_PANEL_DECORATION_1, ATTR_FRAME_COLOR, ColorArray[0]);
    SetCtrlAttribute(trigHandle, TRIG_PANEL_DECORATION_2, ATTR_FRAME_COLOR, ColorArray[1]);
    SetCtrlAttribute(trigHandle, TRIG_PANEL_DECORATION_3, ATTR_FRAME_COLOR, ColorArray[2]);
    SetCtrlAttribute(trigHandle, TRIG_PANEL_DECORATION_4, ATTR_FRAME_COLOR, ColorArray[3]);
    SetCtrlAttribute(trigHandle, TRIG_PANEL_DECORATION_5, ATTR_FRAME_COLOR, ColorArray[4]);
    SetCtrlAttribute(trigHandle, TRIG_PANEL_DECORATION_6, ATTR_FRAME_COLOR, ColorArray[5]);
    SetCtrlAttribute(trigHandle, TRIG_PANEL_DECORATION_7, ATTR_FRAME_COLOR, ColorArray[6]);
    SetCtrlAttribute(trigHandle, TRIG_PANEL_DECORATION_8, ATTR_FRAME_COLOR, ColorArray[7]);
    SetCtrlAttribute(trigHandle, TRIG_PANEL_DECORATION_9, ATTR_FRAME_COLOR, ColorArray[8]);
    SetPanelAttribute (panelHandle, ATTR_SCALE_CONTENTS_ON_RESIZE, 1);
    SetPanelAttribute (imageHandle, ATTR_SCALE_CONTENTS_ON_RESIZE, 1);
    SetPanelAttribute (trigHandle, ATTR_SCALE_CONTENTS_ON_RESIZE, 1);
    SetPanelAttribute (cameraHandle, ATTR_SCALE_CONTENTS_ON_RESIZE, 1);
    SetPanelAttribute (exposureHandle, ATTR_SCALE_CONTENTS_ON_RESIZE, 1);
    SetPanelAttribute (windowHandle, ATTR_SCALE_CONTENTS_ON_RESIZE, 1);
    SetPanelAttribute (triggerHandle, ATTR_SCALE_CONTENTS_ON_RESIZE, 1);
    SetPanelAttribute (dataoutputHandle, ATTR_SCALE_CONTENTS_ON_RESIZE, 1);
    SetPanelAttribute (characteristicsHandle, ATTR_SCALE_CONTENTS_ON_RESIZE, 1);
    SetPanelAttribute (correctionHandle, ATTR_SCALE_CONTENTS_ON_RESIZE, 1);
    SetPanelAttribute (infoHandle, ATTR_SCALE_CONTENTS_ON_RESIZE, 1);

    DisplayPanel (panelHandle);
    RunUserInterface ();
    return 0;
}

/*********************************************************************************
*----------------------------------DISPLAY TIME AND DATE-------------------------*
**********************************************************************************/

int CVICALLBACK TimerCallback (int panel, int control, int event, void *callbackData,
                               int eventData1, int eventData2)
{
    unsigned int        year, month, day, hour, min, sec, weekDay;
    char                Time[512], Date[512] ;
    CVIAbsoluteTime     absTime;

    switch (event) {
        case EVENT_TIMER_TICK:
            GetCurrentCVIAbsoluteTime(&absTime);
            CVIAbsoluteTimeToLocalCalendar(absTime, &year, &month, &day, &hour, &min, &sec, 0,
                                           &weekDay);
            sprintf(Time, "%02d:%02d:%02d", hour, min, sec);
            SetCtrlVal(panel, PANEL_CLOCK, Time);
            sprintf(Date, "%s %02d-%02d-%04d", DaysOfWeek[weekDay], month, day, year);
            SetCtrlVal(panel, PANEL_DATE, Date);

            break;
    }
    return 0;
}

/*********************************************************************************
*----------------------------------SEQUENTIAL ACQUISITION------------------------*
*                        Start the sequential acquisition of images              *
**********************************************************************************/

int CVICALLBACK Start (int panel, int control, int event, void *callbackData, int eventData1,
                       int eventData2)
{
    unsigned long   skipCount, nbOfImages, MaxImages;
    int             i;
    double          AcquisitionTime;

    switch (event) {
        case EVENT_COMMIT:

            if(image_g) {
                imaqDispose (image_g);
                free(image_g);
                image_g = NULL;
            }

            GetCtrlVal (imageHandle, IMG_PANEL_MAX_NUMBER_OF_IMAGES, &MaxImages);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DISPLAYED_IMAGE, ATTR_DIMMED, 1);
            SetCtrlAttribute (imageHandle, IMG_PANEL_START, ATTR_DIMMED, TRUE);
            SetCtrlAttribute (imageHandle, IMG_PANEL_STOP, ATTR_DIMMED, FALSE);
            SetCtrlAttribute (imageHandle, IMG_PANEL_SKIP_COUNT, ATTR_DIMMED, TRUE);
            SetCtrlAttribute (imageHandle, IMG_PANEL_INTERFACE_NAME, ATTR_DIMMED, TRUE);
            SetCtrlAttribute (imageHandle, IMG_PANEL_PLAY,ATTR_DIMMED, 0);
            SetCtrlAttribute (imageHandle, IMG_PANEL_SAVE_AVI,ATTR_DIMMED, 0);
            SetCtrlAttribute (imageHandle, IMG_PANEL_GRAB,ATTR_DIMMED, 1);
            SetCtrlAttribute (imageHandle, IMG_PANEL_SNAP,ATTR_DIMMED, 1);
            SetCtrlAttribute (imageHandle, IMG_PANEL_SAVE_SNAP,ATTR_DIMMED, 1);
            SetCtrlAttribute (imageHandle, IMG_PANEL_IMAGE_TOOLS,ATTR_DIMMED, 0);
            SetCtrlAttribute (panelHandle, PANEL_QUIT, ATTR_DIMMED, 1);
            SetCtrlVal (imageHandle, IMG_PANEL_LED, 1);
            GetCtrlVal (imageHandle, IMG_PANEL_NUMBER_OF_IMAGES, &nbOfImages);

            if (MaxImages<nbOfImages) {
                MessagePopup ("Error", "Number of Images cannot exceed the Maximum number of
```

```
                               available images.");
            }

            GetCtrlVal (imageHandle, IMG_PANEL_SKIP_COUNT, &skipCount);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DISPLAYED_IMAGE, ATTR_MIN_VALUE, 0);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DISPLAYED_IMAGE, ATTR_MAX_VALUE, nbOfImages-1);
            imageArraySize = nbOfImages;

            if(images) {
                for(i=0; i<imageArraySize; i++)
                    imaqDispose (images[i]);
                free(images);
            }

            images = (Image **) malloc(nbOfImages * sizeof(Image *));

            for(i=0; i<nbOfImages; i++){
                images[i] = imaqCreateImage (IMAQ_IMAGE_U8, 0);
            }

            imaqSetupSequence (sid, images, nbOfImages, skipCount, IMAQ_NO_RECT);
            imaqStartAcquisition (sid);
            SetCtrlAttribute (imageHandle, IMG_PANEL_TIMER, ATTR_ENABLED, TRUE);

            break;
    }
    return 0;
}

int CVICALLBACK CheckAcquistion (int panel, int control, int event, void *callbackData,
                                 int eventData1, int eventData2)
{
    char title[100];
    unsigned long acqInProgress, bufNum;

    switch (event) {
        case EVENT_TIMER_TICK:
            imgSessionStatus (sid, &acqInProgress, &bufNum);

            if (!acqInProgress) {
                SetCtrlAttribute (imageHandle, IMG_PANEL_STOP, ATTR_DIMMED, TRUE);
                SetCtrlAttribute (imageHandle, IMG_PANEL_START, ATTR_DIMMED, FALSE);
                SetCtrlAttribute (imageHandle, IMG_PANEL_TIMER, ATTR_ENABLED, FALSE);
                SetCtrlAttribute (imageHandle, IMG_PANEL_SKIP_COUNT, ATTR_DIMMED, FALSE);
                SetCtrlAttribute (imageHandle, IMG_PANEL_DISPLAYED_IMAGE, ATTR_DIMMED, 0);
                SetCtrlAttribute (imageHandle, IMG_PANEL_GRAB,ATTR_DIMMED, 0);
                SetCtrlAttribute (imageHandle, IMG_PANEL_SNAP,ATTR_DIMMED, 0);
                SetCtrlVal (imageHandle, IMG_PANEL_LED, 0);
                DisplayImage(imageHandle, IMG_PANEL_DISPLAYED_IMAGE, EVENT_VAL_CHANGED, NULL, 0, 0);
                         }
            break;
    }
    return 0;
}

int CVICALLBACK Quit (int panel, int control, int event, void *callbackData, int eventData1,
                      int eventData2)
{
    int     i, sts;

    switch (event) {
        case EVENT_COMMIT:

            if(images) {
                for(i=0; i<imageArraySize; i++)
                    imaqDispose (images[i]);
                free(images);
                images = NULL;
            }

            if(image_g) {
                imaqDispose (image_g);
                free(image_g);
                image_g = NULL;
```

```c
            }
            StopCallback(99,99,EVENT_COMMIT,0,99,99);
            QuitUserInterface (0);

            break;
    }
    return 0;
}

int CVICALLBACK Stop (int panel, int control, int event, void *callbackData, int eventData1,
                      int eventData2)
{
    int             Status;
    unsigned long   currBufNum;

    switch (event) {
        case EVENT_COMMIT:
            SetCtrlAttribute (imageHandle, IMG_PANEL_TIMER, ATTR_ENABLED, FALSE);
            SetCtrlAttribute (imageHandle, IMG_PANEL_START, ATTR_DIMMED, FALSE);
            SetCtrlAttribute (imageHandle, IMG_PANEL_SKIP_COUNT, ATTR_DIMMED, FALSE);
            SetCtrlAttribute (imageHandle, IMG_PANEL_INTERFACE_NAME, ATTR_DIMMED, FALSE);
            SetCtrlAttribute (imageHandle, IMG_PANEL_GRAB,ATTR_DIMMED, 0);
            SetCtrlAttribute (imageHandle, IMG_PANEL_SNAP,ATTR_DIMMED, 0);
            SetCtrlVal (imageHandle, IMG_PANEL_LED, 0);

            if (images) {
                SetCtrlAttribute (imageHandle, IMG_PANEL_PLAY,ATTR_DIMMED, 0);
                SetCtrlAttribute (imageHandle, IMG_PANEL_IMAGE_TOOLS,ATTR_DIMMED, 0);
            }

            imgSessionStatus (sid, &Status, &currBufNum);

            if (Status) {
                imgSessionAbort(sid, &currBufNum);
            }

            break;
    }
    return 0;
}

/*********************************************************************************
*----------------------------------DISPLAY IMAGE---------------------------------*
*                     Display the acquired sequence of images                    *
**********************************************************************************/

int CVICALLBACK DisplayImage (int panel, int control, int event,void *callbackData,
                              int eventData1, int eventData2)
{
    int             DisplayWidth, DisplayHeight, DisplayTop, DisplayLeft, ImageWidth, ImageHeight,
                    ImageTop, ImageLeft;
    unsigned int    imageToDisplay;

   switch (event) {
      case EVENT_VAL_CHANGED:
         if(images) {
             GetCtrlVal (imageHandle, IMG_PANEL_DISPLAYED_IMAGE, &imageToDisplay);
             GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_LEFT, &DisplayLeft);
             GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_TOP, &DisplayTop);
             GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_HEIGHT, &DisplayHeight);
             GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_WIDTH, &DisplayWidth);

             ImageLeft = DisplayLeft + 5;
             ImageTop = DisplayTop + 5;
             ImageHeight = DisplayHeight - 10;
             ImageWidth = DisplayWidth - 10;
             imaqZoomWindow (0, -2, -2, IMAQ_NO_POINT);
             imaqMoveWindow(0, MakePoint(ImageLeft,ImageTop));
             imaqSetWindowSize (0, ImageWidth, ImageHeight);
             imaqDisplayImage (images[imageToDisplay], 0, FALSE);
             imaqShowScrollbars (0, TRUE);
         }

         break;
```

```c
   }
   return 0;
}

/*********************************************************************************
*----------------------------------SAVE ACQUISITION------------------------------*
*                         Save the acquired sequence of images                   *
**********************************************************************************/

int CVICALLBACK SaveAVI (int panel, int control, int event, void *callbackData, int eventData1,
                         int eventData2)
{
    int             frame, FramesPerSecond;
    unsigned int    maxImage=0;
    char            AVIFile[MAX_PATHNAME_LEN] = "test";
    char            buffer[256]="";

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (imageHandle, IMG_PANEL_NUMBER_OF_IMAGES, &maxImage);
            GetCtrlVal (imageHandle, IMG_PANEL_AVI_FPS,&FramesPerSecond);

            if (FileSelectPopup ("", "*.avi", "*.avi", "Name of File to Save",
                VAL_SAVE_BUTTON, 0, 1, 1, 0, AVIFile) > 0)
                aviDest = imaqCreateAVI (AVIFile, NULL, IMAQ_USE_DEFAULT_QUALITY, FramesPerSecond,
                                         10000);
            for (frame = 0; frame < maxImage; ++frame) {
                imaqWriteAVIFrame (images[frame], aviDest, buffer, 0);
                ProcessSystemEvents();
            }
            imaqCloseAVI (aviDest);

            break;
    }
    return 0;
}

/*********************************************************************************
*----------------------------------PLAY ACQUISITION------------------------------*
*                         Play the acquired sequence of images                   *
**********************************************************************************/

int CVICALLBACK Play (int panel, int control, int event, void *callbackData, int eventData1,
                      int eventData2)
{
    int     test;
    double  time;

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (imageHandle, IMG_PANEL_PLAY, &test);

            if(test) {
                SetCtrlVal (imageHandle, IMG_PANEL_LED, 1);
                GetCtrlVal (imageHandle, IMG_PANEL_SPEED, &time);
                SetCtrlAttribute (imageHandle, IMG_PANEL_PLAY_TIMER, ATTR_INTERVAL, 1.0/time);
                SetCtrlAttribute (imageHandle, IMG_PANEL_PLAY_TIMER, ATTR_ENABLED , 1);
            }
            else {
                SetCtrlVal (imageHandle, IMG_PANEL_LED, 0);
                SetCtrlAttribute (imageHandle, IMG_PANEL_PLAY_TIMER, ATTR_ENABLED , 0);
            }

            break;
    }
    return 0;
}

int CVICALLBACK PlayTimer (int panel, int control, int event, void *callbackData, int eventData1,
                           int eventData2)
{
    int             DisplayWidth, DisplayHeight, DisplayTop, DisplayLeft, ImageWidth, ImageHeight,
                    ImageTop, ImageLeft;
    unsigned int    imageToDisplay, maxImage;
    switch (event) {
        case EVENT_TIMER_TICK:
            GetCtrlVal (imageHandle, IMG_PANEL_NUMBER_OF_IMAGES, &maxImage);
            GetCtrlVal (imageHandle, IMG_PANEL_DISPLAYED_IMAGE, &imageToDisplay);

            if ( imageToDisplay < maxImage-1) {
                GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_LEFT, &DisplayLeft);
                GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_TOP, &DisplayTop);
                GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_HEIGHT, &DisplayHeight);
                GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_WIDTH, &DisplayWidth);

                ImageLeft = DisplayLeft + 5;
                ImageTop = DisplayTop + 5;
                ImageHeight = DisplayHeight - 10;
                ImageWidth = DisplayWidth - 10;
                imaqZoomWindow (0, -2, -2, IMAQ_NO_POINT);
                imaqMoveWindow(0, MakePoint(ImageLeft,ImageTop));
                imaqSetWindowSize (0, ImageWidth, ImageHeight);
                imaqDisplayImage (images[imageToDisplay], 0, FALSE);
                imaqShowScrollbars (0, TRUE);
                SetCtrlVal (imageHandle, IMG_PANEL_DISPLAYED_IMAGE, imageToDisplay+1);
            }
            else {
                SetCtrlVal (imageHandle, IMG_PANEL_DISPLAYED_IMAGE, 0);
            }

            break;
    }
    return 0;
}

/*********************************************************************************
*----------------------------------VIDEO ACQUISITION------------------------------*
*                  Grab and play the continuous acquisition of images             *
**********************************************************************************/

int CVICALLBACK Start_Grab (int panel, int control, int event, void *callbackData, int eventData1,
                            int eventData2)
{
    int     i, status;
    double  time;

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (imageHandle, IMG_PANEL_GRAB, &status);

            if(status) {
                SetCtrlAttribute (imageHandle, IMG_PANEL_GRAB_TIMER, ATTR_INTERVAL, 1.0/10);
                SetCtrlVal (imageHandle, IMG_PANEL_LED, 1);

                if(image_g) {
                    imaqDispose (image_g);
                    free(image_g);
                }

                image_g = imaqCreateImage (IMAQ_IMAGE_U8, 0);
                imaqSetupGrab (sid, IMAQ_NO_RECT);
                SetCtrlAttribute (imageHandle, IMG_PANEL_GRAB_TIMER, ATTR_ENABLED, TRUE);
                SetCtrlAttribute (imageHandle, IMG_PANEL_IMAGE_TOOLS,ATTR_DIMMED, 0);
                SetCtrlAttribute (imageHandle, IMG_PANEL_SNAP, ATTR_DIMMED, 1);
                SetCtrlAttribute (imageHandle, IMG_PANEL_SAVE_SNAP,ATTR_DIMMED, 1);
                SetCtrlAttribute (imageHandle, IMG_PANEL_PLAY, ATTR_DIMMED, 1);
                SetCtrlAttribute (imageHandle, IMG_PANEL_SAVE_AVI, ATTR_DIMMED, 1);
                SetCtrlAttribute (imageHandle, IMG_PANEL_START, ATTR_DIMMED, 1);
                SetCtrlAttribute (imageHandle, IMG_PANEL_STOP, ATTR_DIMMED, 1);
                SetCtrlAttribute (panelHandle, PANEL_QUIT, ATTR_DIMMED, 1);
            }
            else {
                SetCtrlAttribute (imageHandle, IMG_PANEL_GRAB_TIMER, ATTR_ENABLED, FALSE);
                SetCtrlAttribute (imageHandle, IMG_PANEL_SNAP, ATTR_DIMMED, 0);
                SetCtrlAttribute (imageHandle, IMG_PANEL_START, ATTR_DIMMED, 0);
                SetCtrlAttribute (imageHandle, IMG_PANEL_STOP, ATTR_DIMMED, 0);
                SetCtrlVal (imageHandle, IMG_PANEL_LED, 0);
```

```
                if (images) {
                    SetCtrlAttribute (imageHandle, IMG_PANEL_PLAY, ATTR_DIMMED, 0);
                SetCtrlAttribute (imageHandle, IMG_PANEL_SAVE_AVI, ATTR_DIMMED, 0);
                SetCtrlAttribute (imageHandle, IMG_PANEL_IMAGE_TOOLS,ATTR_DIMMED, 0);
                }

                imaqStopAcquisition (sid);

                if(image_g) {
                    imaqDispose(image_g);
                    image_g = NULL;
                }
            }

            break;
    }
    return 0;
}

int CVICALLBACK Imaq_Loop (int panel, int control, int event, void *callbackData, int eventData1,
                           int eventData2)
{
    int         DisplayWidth, DisplayHeight, DisplayTop, DisplayLeft, ImageWidth, ImageHeight,
                ImageTop, ImageLeft;
    static int  nbFrame, t;
    char        data[64];

    switch (event) {
        case EVENT_TIMER_TICK:
            t = Timer();
            image_g = imaqGrab (sid, image_g, FALSE);
            GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_LEFT, &DisplayLeft);
            GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_TOP, &DisplayTop);
            GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_HEIGHT, &DisplayHeight);
            GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_WIDTH, &DisplayWidth);

            ImageLeft = DisplayLeft + 5;
            ImageTop = DisplayTop + 5;
            ImageHeight = DisplayHeight - 10;
            ImageWidth = DisplayWidth - 10;
            imaqZoomWindow (0, -2, -2, IMAQ_NO_POINT);
            imaqMoveWindow(0, MakePoint(ImageLeft,ImageTop));
            imaqSetWindowSize (0, ImageWidth, ImageHeight);
            imaqDisplayImage (image_g, 0, FALSE);
            imaqShowScrollbars (0, TRUE);
            nbFrame++;

            if (nbFrame>10) {
                GetPropertyString(port,"FrameRate", data);
                FrameRateUpdate=atof(data);
                SetCtrlVal(imageHandle, IMG_PANEL_ACTUAL_FPS, FrameRateUpdate);
                nbFrame=0;
            }

            break;
    }
    return 0;
}

/*********************************************************************************
*----------------------------------SNAP ACQUISITION------------------------------*
*                   Grab, dispalay and save the single acquired images           *
*********************************************************************************/

int CVICALLBACK Snap (int panel, int control, int event, void *callbackData, int eventData1,
                      int eventData2)
{
    int     DisplayWidth, DisplayHeight, DisplayTop, DisplayLeft ImageWidth, ImageHeight,
            ImageTop, ImageLeft;

    switch (event) {
        case EVENT_COMMIT:
            image_s = imaqCreateImage (IMAQ_IMAGE_U8, 0);
            image_s = imaqSnap (sid, image_s, IMAQ_NO_RECT);

            if (image_s)
                SetCtrlAttribute (imageHandle, IMG_PANEL_SAVE_SNAP,ATTR_DIMMED, 0);
                SetCtrlAttribute (imageHandle, IMG_PANEL_IMAGE_TOOLS,ATTR_DIMMED, 0);
                GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_LEFT, &DisplayLeft);
                GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_TOP, &DisplayTop);
                GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_HEIGHT, &DisplayHeight);
                GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_WIDTH, &DisplayWidth);

                ImageLeft = DisplayLeft + 5;
                ImageTop = DisplayTop + 5;
                ImageHeight = DisplayHeight - 10;
                ImageWidth = DisplayWidth - 10;
                imaqZoomWindow (0, -2, -2, IMAQ_NO_POINT);
                imaqMoveWindow(0, MakePoint(ImageLeft,ImageTop));
                imaqSetWindowSize (0, ImageWidth, ImageHeight);
                imaqDisplayImage (image_s, 0, FALSE);
                imaqShowScrollbars (0, TRUE);

                break;
    }
    return 0;
}

int CVICALLBACK Save_Snap (int panel, int control, int event, void *callbackData, int eventData1,
                           int eventData2)
{
    char        SNAPFile[MAX_PATHNAME_LEN]="test";

    switch (event) {
        case EVENT_COMMIT:

            if (FileSelectPopup ("", "Reference Image.jpg", "*.jpg", "Name of File to Save",
                                 VAL_SAVE_BUTTON, 0, 1, 1, 0, SNAPFile) > 0)
                imaqWriteJPEGFile (image_s, SNAPFile, IMAQ_USE_DEFAULT_QUALITY, NULL);

            break;
    }
    return 0;
}

/*********************************************************************************
*----------------------------------IMAGE TOOLS-----------------------------------*
**********************************************************************************/

int CVICALLBACK Image_Tools (int panel, int control, int event, void *callbackData, int eventData1,
                             int eventData2)
{
    switch (event) {
        case EVENT_COMMIT:
            myToolOptions.showSelectionTool = TRUE;
            myToolOptions.showZoomTool = TRUE;
            myToolOptions.showPanTool = TRUE;
            myToolOptions.showFreehandTool = TRUE;
            myToolOptions.showPointTool = TRUE;
            myToolOptions.showLineTool = TRUE;
            myToolOptions.showRectangleTool = TRUE;
            myToolOptions.showOvalTool = TRUE;
            myToolOptions.showPolygonTool = TRUE;
            myToolOptions.showClosedFreehandTool = TRUE;
            myToolOptions.showPolyLineTool = TRUE;
            imaqSetupToolWindow (TRUE, 5, &myToolOptions);
            imaqMoveToolWindow (MakePoint(650,250));
            imaqSetCurrentTool (IMAQ_RECTANGLE_TOOL);

            if(images)
                imaqShowToolWindow (TRUE);
            else if(image_g)
                imaqShowToolWindow (TRUE);
            else if(image_s)
                imaqShowToolWindow (TRUE);
            else
                MessagePopup ("Info Message", "No images in memory to use the image tools.");
```

```
            break;
    }
    return 0;
}

/*********************************************************************************
*----------------------------------CAMERA PROPERTIES-----------------------------*
*                   Initialize, obtain and change camera properties              *
**********************************************************************************/

int CVICALLBACK Initialize_Camera (int panel, int control, int event, void *callbackData,
                                   int eventData1, int eventData2)
{
    char    interfaceName[64], data[64], name[50], manu[50], message[1000];
    int     i, error, numOfPorts, mBytes, nBytes, version, type, status, PixelClock, XResol,
            YResol, Serial;
    char*   FileName = "DSC_0316.JPEG";
    int     DisplayWidth, DisplayHeight, DisplayTop, DisplayLeft, ImageWidth, ImageHeight,
            ImageTop, ImageLeft;
    void*   serialRef=0;
    UINT32* baudRates=0;

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (imageHandle, IMG_PANEL_INTERFACE_NAME, interfaceName);
            imgInterfaceOpen (interfaceName, &iid);
            imgSessionOpen (iid, &sid);

            SetCtrlAttribute (panelHandle, PANEL_INITIALIZE_CAMERA, ATTR_DIMMED, 1);
            SetCtrlAttribute (panelHandle, PANEL_PORT_INFO, ATTR_DIMMED, 0);
            SetCtrlAttribute (panelHandle, PANEL_CLOSE_CAMERA, ATTR_DIMMED, 0);
            SetCtrlAttribute (imageHandle, IMG_PANEL_START, ATTR_DIMMED, 0);
            SetCtrlAttribute (imageHandle, IMG_PANEL_GRAB, ATTR_DIMMED, 0);
            SetCtrlAttribute (imageHandle, IMG_PANEL_SNAP, ATTR_DIMMED, 0);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DECORATION_1, ATTR_FRAME_COLOR, VAL_BLUE);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DECORATION_2, ATTR_FRAME_COLOR, VAL_RED);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DECORATION_3, ATTR_FRAME_COLOR, VAL_GREEN);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DECORATION_4, ATTR_FRAME_COLOR, VAL_YELLOW);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DECORATION_5, ATTR_FRAME_COLOR, VAL_DK_BLUE);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DECORATION_6, ATTR_FRAME_COLOR, VAL_DK_GREEN);
            ResetTextBox (panelHandle, PANEL_DISPLAY_PORTS,"Reseting and Initializing");

            numOfPorts = 0;
            pfPortInit(&numOfPorts);
            SetCtrlVal (panelHandle, PANEL_PORT_COUNT, numOfPorts);

            for(i=0; i<numOfPorts; i++){
                pfPortInfo(i, manu, &mBytes, name, &nBytes, &version, &type);
                sprintf(message, "Port %d: Manufacturer: %s, %s, mBytes %d, nBytes %d, ver. %d,
                                  type %d", i, manu, name, mBytes, nBytes, version, type);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
            }

            port = PORT;
            SetCtrlVal(panelHandle, PANEL_PORT_NO, port);
            SetCtrlVal(panelHandle, PANEL_PORT_NO_1, port);
            pfPortInfo(port, manu, &mBytes, name, &nBytes, &version, &type);
            sprintf(message,"Opening camera on port %s, %s...", manu, name);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
            error = pfDeviceOpen(port);

            if(error < 0){
                handleError(error);
                return -1;
            }
            error = pfIsBaudRateSupported(port, 115200);

            if(error == 1){
                if (ConfirmPopup("Use camera at 115200 baud?", "The camera and the file grabber
                    supports\na higher baud rate. Default is 9600 baud\nshould camera start at 57600
                    baud")>0) {

                    error = pfSetBaudRate(port, 115200);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1,
                        "The camera and frame grabber supports a baud rate of 115200.\n
                         Starting camera with a baud rate of 115200");

                         if(error < 0){
                            handleError(error);
                            pfDeviceClose(port);

                            return 0;
                    }
            }
    }

    error = pfIsBaudRateSupported(port, 57600);

    if(error == 1){
        if (ConfirmPopup("Use camera at 57600 baud?", "The camera and the file grabber supports\n
            a higher baud rate.Default is 9600 baud\nshould camera start at 57600 baud")>0) {

            error = pfSetBaudRate(port, 57600);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "The camera and frame grabber
                               supports a baud rate of 57600.\nStarting camera with a baud rate of
                               57600");

            if(error < 0){
                handleError(error);
                pfDeviceClose(port);

                return 0;
            }
    }
}

    clGetSupportedBaudRates(serialRef, baudRates);
    GetPropertyString (port, "CameraName", data);
    GetPropertyInt(port, "Header.PixelClock", &PixelClock);
    GetPropertyInt(port, "Header.ResolutionX", &XResol);
    GetPropertyInt(port, "Header.ResolutionY", &YResol);
    SetCtrlVal(infoHandle, INFO_CAM_NAME, data);
    SetCtrlVal (infoHandle, INFO_PIXEL_CLOCK, PixelClock);
    SetCtrlVal (infoHandle, INFO_X_RESOL, XResol);
    SetCtrlVal (infoHandle, INFO_Y_RESOL, YResol);

    imgGetAttribute (sid, IMG_ATTR_GETSERIAL, &Serial);
    SetCtrlVal (infoHandle, INFO_SERIAL, Serial);
    sprintf(message,"%s with a maximum resolution of %d x%d Pixels", data, XResol, YResol);
    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
    sprintf(message,"Pixel clock of the camera = %d MHz", PixelClock/1000000);
    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
    image_fun = imaqCreateImage (IMAQ_IMAGE_U8, 0);
    imaqReadFile (image_fun, FileName, NULL, NULL);
    GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_LEFT, &DisplayLeft);
    GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_TOP, &DisplayTop);
    GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_HEIGHT, &DisplayHeight);
    GetCtrlAttribute (panelHandle, PANEL_IMAGE_WINDOW, ATTR_WIDTH, &DisplayWidth);
    SetCtrlAttribute (panelHandle, PANEL_QUIT, ATTR_DIMMED, 1);

    ImageLeft = DisplayLeft + 5;
    ImageTop = DisplayTop + 5;
    ImageHeight = DisplayHeight - 10;
    ImageWidth = DisplayWidth - 10;
    imaqZoomWindow (0, -2, -2, IMAQ_NO_POINT);
    imaqMoveWindow(0, MakePoint(ImageLeft,ImageTop));
    imaqSetWindowSize (0, ImageWidth, ImageHeight);
    imaqDisplayImage (image_fun, 0, FALSE);
    imaqShowScrollbars (0, TRUE);

    break;
    }
return 0;
}

int CVICALLBACK Close_Camera (int panel, int control, int event, void *callbackData,
                              int eventData1, int eventData2)
```

```
{
    char    message[100];

    switch (event) {
        case EVENT_COMMIT:

            pfDeviceClose(port);
            SetCtrlAttribute(panelHandle, PANEL_INITIALIZE_CAMERA, ATTR_DIMMED, 0);
            SetCtrlAttribute (panelHandle, PANEL_CLOSE_CAMERA, ATTR_DIMMED, 1);
            SetCtrlAttribute (panelHandle, PANEL_PORT_INFO, ATTR_DIMMED, 1);
            sprintf(message,"Close camera on port %d...", port);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            Stop(99,99,EVENT_COMMIT,0,99,99);

            if(image_g) {
                SetCtrlAttribute (imageHandle, IMG_PANEL_GRAB, ATTR_CTRL_VAL, 0);
                Start_Grab(99,99,EVENT_COMMIT,0,99,99);
            }

            SetCtrlAttribute (cameraHandle, CAM_PANEL_DECORATION_1, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (cameraHandle, CAM_PANEL_DECORATION_2, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DECORATION_1, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DECORATION_2, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DECORATION_3, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DECORATION_4, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DECORATION_5, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (imageHandle, IMG_PANEL_DECORATION_6, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (exposureHandle, EXPOSURE_DECORATION_1, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (exposureHandle, EXPOSURE_DECORATION_2, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_1, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_2, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_4, ATTR_FRAME_COLOR, VAL_DK_GRAY);
            SetCtrlAttribute (dataoutputHandle, DATAOUTPUT_DECORATION_1, ATTR_FRAME_COLOR,
                              VAL_DK_GRAY);
            SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_1, ATTR_FRAME_COLOR,
                              VAL_DK_GRAY);
            SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR,
                              VAL_DK_GRAY);
            SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_3, ATTR_FRAME_COLOR,
                              VAL_DK_GRAY);
            SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR,
                              VAL_DK_GRAY);
            SetCtrlAttribute (correctionHandle, CORRECTION_DECORATION_1, ATTR_FRAME_COLOR,
                              VAL_DK_GRAY);
            SetCtrlAttribute (panelHandle, PANEL_QUIT, ATTR_DIMMED, 0);
            SetCtrlAttribute (imageHandle, IMG_PANEL_START, ATTR_DIMMED, 1);
            SetCtrlAttribute (imageHandle, IMG_PANEL_GRAB, ATTR_DIMMED, 1);
            SetCtrlAttribute (imageHandle, IMG_PANEL_SNAP, ATTR_DIMMED, 1);
            SetCtrlAttribute (imageHandle, IMG_PANEL_STOP, ATTR_DIMMED, 1);

            if(images)
                SetCtrlAttribute (imageHandle, IMG_PANEL_IMAGE_TOOLS,ATTR_DIMMED, 0);
            else if(image_s)
                SetCtrlAttribute (imageHandle, IMG_PANEL_IMAGE_TOOLS,ATTR_DIMMED, 0);
            else
                SetCtrlAttribute (imageHandle, IMG_PANEL_IMAGE_TOOLS,ATTR_DIMMED, 1);

            imgClose (sid, TRUE);
            imgClose (iid, TRUE);

            break;
    }
    return 0;
}


int CVICALLBACK Port_Info (int panel, int control, int event, void *callbackData, int eventData1,
                           int eventData2)
{
    float   f;
    int     i, sts, error, OptionIndex;
    char    data[64], message[1000];
```

```
switch (event) {
    case EVENT_COMMIT:
        SetCtrlAttribute (cameraHandle, CAM_PANEL_DECORATION_1, ATTR_FRAME_COLOR, VAL_DK_BLUE);
        SetCtrlAttribute (cameraHandle, CAM_PANEL_DECORATION_2, ATTR_FRAME_COLOR, VAL_DK_GREEN);
        SetCtrlAttribute (exposureHandle, EXPOSURE_DECORATION_1, ATTR_FRAME_COLOR, VAL_DK_BLUE);
        SetCtrlAttribute (exposureHandle, EXPOSURE_DECORATION_2, ATTR_FRAME_COLOR, VAL_BLUE);
        SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_1, ATTR_FRAME_COLOR, VAL_BLUE);
        SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_2, ATTR_FRAME_COLOR, VAL_GREEN);
        SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR, VAL_RED);
        SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_4, ATTR_FRAME_COLOR, VAL_YELLOW);
        SetCtrlAttribute (dataoutputHandle, DATAOUTPUT_DECORATION_1, ATTR_FRAME_COLOR, VAL_RED);
        SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_1, ATTR_FRAME_COLOR,
                          VAL_GREEN);
        SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR,
                          VAL_RED);
        SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_3, ATTR_FRAME_COLOR,
                          VAL_BLUE);
        SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR,
                          VAL_YELLOW);
        SetCtrlAttribute (correctionHandle, CORRECTION_DECORATION_1, ATTR_FRAME_COLOR,
                          VAL_BLUE);


        GetPropertyString(port,"FrameRate", data);
        FrameRateUpdate=atof(data);
        SetCtrlVal(cameraHandle, CAM_PANEL_FRAME_UPDATE, FrameRateUpdate);

        GetPropertyInt(port,"Average.Value", &AverageValue);
        SetCtrlVal(cameraHandle, CAM_PANEL_AVERAGE_VALUE, AverageValue);

        GetPropertyString(port, "ExposureTime", data);
        sprintf(message,"Exposure Time: %s", data);
        InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
        ExpTime = atof(data);
        SetCtrlVal (exposureHandle, EXPOSURE_EXPOSURETIME, ExpTime);

        GetPropertyString(port, "FrameTime", data);
        sprintf(message,"Frame Time: %s", data);
        InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
        FramTime = atof(data);
        SetCtrlVal (exposureHandle, EXPOSURE_FRAMETIME, FramTime);

        GetPropertyString(port, "Trigger.CFR", data);
        sts=atoi(data);
        if (sts<1){
            SetCtrlAttribute(exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 1);
            SetCtrlVal(exposureHandle, EXPOSURE_CONST_FRAME_RATE, 0);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Constant frame rate:
                               Disabled");
        }
        else {
            SetCtrlAttribute(exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 0);
            SetCtrlVal(exposureHandle, EXPOSURE_CONST_FRAME_RATE, 1);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Constant frame rate:
                               Enabled");
        }

        GetPropertyInt(port,"Counter.Image", &CounterImageUpdate);
        SetCtrlVal(exposureHandle, EXPOSURE_IMG_COUNTER, CounterImageUpdate);

        GetPropertyInt(port,"Counter.MissedTrigger", &MissedTriggerCounter);
        SetCtrlVal(exposureHandle, EXPOSURE_MISS_TRIG_COUNTER, MissedTriggerCounter);

        GetPropertyString(port, "EnStatusLine", data);
        sts=atoi(data);
        if (sts<1){
            SetCtrlVal (exposureHandle, EXPOSURE_STATUS_LINE, sts);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Status Line: Disabled");
        }
        else {
            SetCtrlVal (exposureHandle, EXPOSURE_STATUS_LINE, sts);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Status Line: Enabled");
        }
```

```
        GetPropertyInt (port, "Trigger.Source", &Trig);
        GetPropertyInt (port, "Trigger.LevelControlled", &TrigLevel);
        GetPropertyFloat(port, "Trigger.Delay", &TrigDelay);
        GetPropertyString(port, "Trigger.Invert", data);
        sts=atoi(data);
        if (Trig==0) {
            SetCtrlVal (triggerHandle, TRIGGER_FREE_RUNNING, 1);
            SetCtrlVal (triggerHandle, TRIGGER_BY_CAMERA, 1);
            SetCtrlAttribute (exposureHandle, EXPOSURE_DECORATION_1, ATTR_FRAME_COLOR,
                              VAL_DK_BLUE);
            SetCtrlAttribute (exposureHandle, EXPOSURE_EXPOSURETIME, ATTR_DIMMED, 0);
            SetCtrlAttribute (exposureHandle, EXPOSURE_CONST_FRAME_RATE, ATTR_DIMMED, 0);
            GetCtrlVal (exposureHandle, EXPOSURE_CONST_FRAME_RATE, &sts);

            if (sts>0){
                SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 0);
            }
            else {
                SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 1);
            }

            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
            SetCtrlAttribute (triggerHandle, TRIGGER_TRIGGER_DELAY, ATTR_DIMMED, 1);
            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
                              VAL_DK_GRAY);
            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_SIG_ACTIVE, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_LINLOG, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_1, ATTR_FRAME_COLOR,
                              VAL_GREEN);
            SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR,
                              VAL_YELLOW);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Trigger Source:
                               Free Running");
        }
        else if (Trig==1) {
            SetCtrlVal (triggerHandle, TRIGGER_INTERFACE_TRIGGER, 1);
            GetPropertyString(port, "Trigger.Interleave", data);
            sts=atoi(data);

            if (sts>0){
                SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
                                  VAL_DK_GRAY);
            }
            else {
                SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 0);
                SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 0);
                SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
                                  VAL_RED);
            }

            SetCtrlAttribute (triggerHandle, TRIGGER_TRIGGER_DELAY, ATTR_DIMMED, 0);
            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_SIG_ACTIVE, ATTR_DIMMED, 0);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Trigger Source:
                               Interface Trigger");

            if (TrigLevel==0) {
                SetCtrlVal(triggerHandle, TRIGGER_BY_CAMERA, 1);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Exposure Time
                                   defined by: Camera");
                SetCtrlAttribute (exposureHandle, EXPOSURE_EXPOSURETIME, ATTR_DIMMED, 0);
                SetCtrlAttribute (exposureHandle, EXPOSURE_CONST_FRAME_RATE, ATTR_DIMMED, 0);
                GetCtrlVal (exposureHandle, EXPOSURE_CONST_FRAME_RATE, &sts);
                if (sts>0){
                    SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 0);
                }
```

```
                    else {
                        SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 1);
                    }

                    SetCtrlAttribute (exposureHandle, EXPOSURE_DECORATION_1, ATTR_FRAME_COLOR,
                                      VAL_DK_BLUE);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_LINLOG, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_1, ATTR_FRAME_COLOR,
                                      VAL_GREEN);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR,
                                      VAL_YELLOW);
                }
                else {
                    SetCtrlVal(triggerHandle, TRIGGER_TRIG_PULSE, 1);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Exposure Time
                                       defined by: Trigger Pulse Width");
                    SetCtrlAttribute (exposureHandle, EXPOSURE_EXPOSURETIME, ATTR_DIMMED, 1);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 1);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_CONST_FRAME_RATE, ATTR_DIMMED, 1);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_DECORATION_1, ATTR_FRAME_COLOR,
                                      VAL_DK_GRAY);
                    SetCtrlIndex (characteristicsHandle, CHARAC_LINLOG, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_LINLOG, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_1, ATTR_FRAME_COLOR,
                                      VAL_DK_GRAY);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR,
                                      VAL_RED);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR,
                                      VAL_DK_GRAY);
                }

                SetCtrlVal (triggerHandle, TRIGGER_TRIGGER_DELAY, TrigDelay);
                sprintf(message,"Trigger Delay: %f", TrigDelay);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

                if (sts<1){
                    SetCtrlVal(triggerHandle, TRIGGER_TRIG_SIG_ACTIVE, 0);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Trigger signal active l
ow: Disabled");
                }
                    else {
                        SetCtrlVal(triggerHandle, TRIGGER_TRIG_SIG_ACTIVE, 1);
                        InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Trigger signal acti
ve low: Enabled");
                    }
            }
            else {
                SetCtrlVal (triggerHandle, TRIGGER_IO_TRIGGER, 1);
                GetPropertyString(port, "Trigger.Interleave", data);
                sts=atoi(data);
                    if (sts>0){
                        SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                        SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                        SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR, VAL_
DK_GRAY);
                    }
                    else {
                        SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 0);
                        SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 0);
                        SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR, VAL_
RED);
                    }
                SetCtrlAttribute (triggerHandle, TRIGGER_TRIGGER_DELAY, ATTR_DIMMED, 0);
```

```
                SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_SIG_ACTIVE, ATTR_DIMMED, 0);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Trigger Source: I/O Trigger
");
                if (TrigLevel==0) {
                    SetCtrlVal(triggerHandle, TRIGGER_BY_CAMERA, 1);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Exposure Time defined b
y: Camera");
                    SetCtrlAttribute (exposureHandle, EXPOSURE_EXPOSURETIME, ATTR_DIMMED, 0);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_CONST_FRAME_RATE, ATTR_DIMMED, 0);
                    GetCtrlVal (exposureHandle, EXPOSURE_CONST_FRAME_RATE, &sts);
                    if (sts>0){
                        SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 0);
                    }
                    else {
                        SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 1);
                    }
                    SetCtrlAttribute (exposureHandle, EXPOSURE_DECORATION_1, ATTR_FRAME_COLOR, VAL_DK
_BLUE);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_LINLOG, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_1, ATTR_FRAME_COLOR, V
AL_GREEN);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR, V
AL_YELLOW);
                }
                else {
                    SetCtrlVal(triggerHandle, TRIGGER_TRIG_PULSE, 1);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Exposure Time defined b
y: Trigger Pulse Width");
                    SetCtrlAttribute (exposureHandle, EXPOSURE_EXPOSURETIME, ATTR_DIMMED, 1);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 1);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_CONST_FRAME_RATE, ATTR_DIMMED, 1);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_DECORATION_1, ATTR_FRAME_COLOR, VAL_DK
_GRAY);
                    SetCtrlIndex (characteristicsHandle, CHARAC_LINLOG, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_LINLOG, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_1, ATTR_FRAME_COLOR, V
AL_DK_GRAY);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR, V
AL_RED);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR, V
AL_DK_GRAY);
                }
                SetCtrlVal (triggerHandle, TRIGGER_TRIGGER_DELAY, TrigDelay);
                sprintf(message,"Trigger Delay: %f", TrigDelay);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
                    if (sts<1){
                        SetCtrlVal(triggerHandle, TRIGGER_TRIG_SIG_ACTIVE, 0);
                        InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Trigger signal acti
ve low: Disabled");
                    }
                    else {
                        SetCtrlVal(triggerHandle, TRIGGER_TRIG_SIG_ACTIVE, 1);
                        InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Trigger signal acti
ve low: Enabled");
                    }
            }

            GetPropertyFloat(port, "Strobe.Delay", &StrobeDelay);
            SetCtrlVal (triggerHandle, TRIGGER_STROBE_DELAY, StrobeDelay);
            sprintf(message,"Strobe Delay: %f", StrobeDelay);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            GetPropertyFloat(port, "Strobe.PulseWidth", &StrobePulseWidth);
            SetCtrlVal(triggerHandle, TRIGGER_STROBE_PULSE_WIDTH, StrobePulseWidth);
            sprintf(message,"Strobe Pulse Width: %f", StrobePulseWidth);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            GetPropertyString(port, "Strobe.Invert", data);
            sts=atoi(data);
            if (sts<1){
                    SetCtrlVal(triggerHandle, TRIGGER_STROBE_SIG_ACTIVE, 0);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Strobe signal active lo
w: Disabled");
                }
                else {
                    SetCtrlVal(triggerHandle, TRIGGER_STROBE_SIG_ACTIVE, 1);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Strobe signal active lo
w: Enabled");
                }

            //Current Data Resolution
            GetPropertyString(port, "DataResolution", data);
            GetPropertyInt(port, "DataResolution", &Res);
            if(Res==0){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Resolution: 12 bit");
            SetCtrlAttribute(dataoutputHandle, DATAOUTPUT_RESOLUTION, ATTR_CTRL_INDEX, 0);
            }
            else if(Res==1){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Resolution: 10 bit");
            SetCtrlAttribute(dataoutputHandle, DATAOUTPUT_RESOLUTION, ATTR_CTRL_INDEX, 1);
            }
            else {
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Resolution: 8 bit");
            SetCtrlAttribute(dataoutputHandle, DATAOUTPUT_RESOLUTION, ATTR_CTRL_INDEX, 2);
            }

            //Get Data Output Mode
            GetPropertyInt(port, "DataOutMode", &OutputMode);
            if(OutputMode==0){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Output Mode: Normal");
            SetCtrlAttribute(dataoutputHandle, DATAOUTPUT_OUTPUT_MODE, ATTR_CTRL_INDEX, 0);
            }
            else if(OutputMode==1){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Output Mode: LFSR");
            SetCtrlAttribute(dataoutputHandle, DATAOUTPUT_OUTPUT_MODE, ATTR_CTRL_INDEX, 1);
            }
            else if(OutputMode==2){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Output Mode: Ramp");
            SetCtrlAttribute(dataoutputHandle, DATAOUTPUT_OUTPUT_MODE, ATTR_CTRL_INDEX, 2);
            }
            else {
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Output Mode: LUT");
            SetCtrlAttribute(dataoutputHandle, DATAOUTPUT_OUTPUT_MODE, ATTR_CTRL_INDEX, 3);
            }

            //Get Digital Gain
            GetPropertyInt(port, "DigitalGain", &Gain);
            if(Gain==0){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Digital Gain : 1x");
            SetCtrlAttribute(dataoutputHandle, DATAOUTPUT_DIGITAL_GAIN, ATTR_CTRL_INDEX, 0);
            }
            else if(Gain==1){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Digital Gain : 2x");
            SetCtrlAttribute(dataoutputHandle, DATAOUTPUT_DIGITAL_GAIN, ATTR_CTRL_INDEX, 1);
            }
            else {
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Digital Gain : 4x");
            SetCtrlAttribute(dataoutputHandle, DATAOUTPUT_DIGITAL_GAIN, ATTR_CTRL_INDEX, 2);
            }

            //LinLog Mode, Skiming and Simultaneous Readout Status
            GetPropertyString (port, "LinLog.Mode", data);
            GetPropertyInt(port, "LinLog.Mode", &LinLog);
            if(LinLog==0){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "LinLog Mode: Off");

            GetPropertyString(port, "Skim", data);
```

```
            GetPropertyInt(port, "Skim", &Skim);
            sprintf(message,"Skimming : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
            sts=atoi(data);
                if (sts>0){
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 0);
                    SetCtrlVal (characteristicsHandle, CHARAC_SKIMMING, Skim);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Enabled");
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR, V
AL_RED);
                    SetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR, V
AL_DK_GRAY);
                }
                else {
                    GetPropertyString(port, "Trigger.Interleave", data);
                    sts=atoi(data);
                    if (sts<1){
                        SetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, 0);
                        if (Trig==0) {
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
VAL_DK_GRAY);
                        }
                        else {
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
VAL_RED);
                        }
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 0);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Simultaneous readout: D
isabled");
                    }
                    else {
                        SetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, 1);
                        SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                        SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                        SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3,  ATTR_FRAME_COLOR, VAL
_DK_GRAY);
                        SetCtrlVal (characteristicsHandle, CHARAC_SKIMMING, 0);
                        InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Disabled")
;
                        SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 1);
                        SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLO
R, VAL_DK_GRAY);
                        InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Simultaneous readou
t: Enabled");
                    }
                }

            SetCtrlAttribute(characteristicsHandle, CHARAC_LINLOG, ATTR_CTRL_INDEX, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 1);
            }
            else if(LinLog==1){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "LinLog Mode: Low Compression");
            SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 0);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Enabled");
            SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR, VAL_RED);

            GetPropertyString(port, "Skim", data);
            GetPropertyInt(port, "Skim", &Skim);
            sprintf(message,"Skimming : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
            sts=atoi(data);
                if (sts>0){
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 0);
                    SetCtrlVal (characteristicsHandle, CHARAC_SKIMMING, Skim);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Enabled");
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR, V
AL_RED);
                    SetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR, V
AL_DK_GRAY);
                }
                else {
                    GetPropertyString(port, "Trigger.Interleave", data);
                    sts=atoi(data);
                    if (sts<1){
                        SetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, 0);
                        if (Trig==0) {
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
VAL_DK_GRAY);
                        }
                        else {
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
VAL_RED);
                        }
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 0);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Simultaneous readout: D
isabled");
                    }
                    else {
                        SetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, 1);
                        SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                        SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                        SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3,  ATTR_FRAME_COLOR, VAL
_DK_GRAY);
                        SetCtrlVal (characteristicsHandle, CHARAC_SKIMMING, 0);
                        InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Disabled")
;
                        SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 1);
                        SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLO
R, VAL_DK_GRAY);
                        InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Simultaneous readou
t: Enabled");
                    }
                }

            SetCtrlAttribute(characteristicsHandle, CHARAC_LINLOG, ATTR_CTRL_INDEX, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 1);
            }
            else if(LinLog==2){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "LinLog Mode: Normal Compression
");
            SetCtrlVal (characteristicsHandle, CHARAC_SKIMMING, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 1);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Disabled");
            SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR, VAL_DK_GR
AY);
            GetPropertyString(port, "Trigger.Interleave", data);
            sts=atoi(data);
                if (sts<1){
                    SetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, 0);
                    if (Trig==0) {
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
VAL_DK_GRAY);
                        }
                        else {
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
VAL_RED);
```

```
                        }
                        InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Simultaneous readou
t: Disabled");
                }
                else {
                    SetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3,  ATTR_FRAME_COLOR, VAL_DK_
GRAY);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Simultaneous readout: E
nabled");
                }

            SetCtrlAttribute(characteristicsHandle, CHARAC_LINLOG, ATTR_CTRL_INDEX, 2);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 1);
            }
            else if(LinLog==3){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "LinLog Mode: High Compression")
;
            SetCtrlVal (characteristicsHandle, CHARAC_SKIMMING, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 1);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Disabled");
            SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR, VAL_DK_GR
AY);
            GetPropertyString(port, "Trigger.Interleave", data);
            sts=atoi(data);
                if (sts<1){
                    SetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, 0);
                    if (Trig==0) {
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
VAL_DK_GRAY);
                        }
                        else {
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
VAL_RED);
                        }
                        InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Simultaneous readou
t: Disabled");
                }
                else {
                    SetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3,  ATTR_FRAME_COLOR, VAL_DK_
GRAY);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Simultaneous readout: E
nabled");
                }

            SetCtrlAttribute(characteristicsHandle, CHARAC_LINLOG, ATTR_CTRL_INDEX, 3);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 1);
            }
            else {
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "LinLog Mode: User Defined");
            SetCtrlVal (characteristicsHandle, CHARAC_SKIMMING, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 1);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Disabled");
            SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR, VAL_DK_GR
AY);
            SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR, VAL_YELLO
W);
            GetPropertyString(port, "Trigger.Interleave", data);
            sts=atoi(data);
                if (sts<1){
                    SetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, 0);
                    if (Trig==0) {
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
VAL_DK_GRAY);
                        }
                        else {
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
VAL_RED);
                        }
                        InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Simultaneous readou
t: Disabled");
                }
                else {
                    SetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3,  ATTR_FRAME_COLOR, VAL_DK_
GRAY);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Simultaneous readout: E
nabled");
                }

            SetCtrlAttribute(characteristicsHandle, CHARAC_LINLOG, ATTR_CTRL_INDEX, 4);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 0);

            //Get User Defined LinLog Value 1
            GetPropertyString(port, "LinLog.Value1", data);
            GetPropertyInt(port, "LinLog.Value1", &Value1);
            sprintf(message,"LinLog Mode User Defined Value 1 : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
            SetCtrlVal(characteristicsHandle, CHARAC_VALUE_1, Value1);

            //Get User Defined LinLog Time 1
            GetPropertyString(port, "LinLog.Time1", data);
            GetPropertyInt(port, "LinLog.Time1", &Time1);
            sprintf(message,"LinLog Mode User Defined Time 1 : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
            SetCtrlVal(characteristicsHandle, CHARAC_TIME_1, Time1);

            //Get User Defined LinLog Value 2
            GetPropertyString(port, "LinLog.Value2", data);
            GetPropertyInt(port, "LinLog.Value2", &Value2);
            sprintf(message,"LinLog Mode User Defined Value 2 : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
            SetCtrlVal(characteristicsHandle, CHARAC_VALUE_2, Value2);

            //Get User Defined LinLog Time 2
            GetPropertyString(port, "LinLog.Time2", data);
            GetPropertyInt(port, "LinLog.Time2", &Time2);
            sprintf(message,"LinLog Mode User Defined Time 2 : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
            SetCtrlVal(characteristicsHandle, CHARAC_TIME_2, Time2);
            }

            GetPropertyString(port, "Voltages.BlackLevelOffset", data);
            BlkOffset=atoi(data);
            SetCtrlVal (characteristicsHandle, CHARAC_BLK_LVL_OFFSET, BlkOffset);
            sprintf(message,"Black level offset : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            GetPropertyInt(port,  "Correction.Mode", &OptionIndex);
            Radio_SetMarkedOption(correctionHandle, CORRECTION_CORRECTION_MODE, OptionIndex);
            if(OptionIndex==0) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode : Off");
            }
```

```
            else if(OptionIndex==1) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                    Offset");
            }
            else if(OptionIndex==2) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1,"Correction Mode :
                                    Offset + Hotpixel");
            }
            else if(OptionIndex==3) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                    Hotpixel");
            }
            else if(OptionIndex==4) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                    Offset + Gain");
            }
            else if(OptionIndex==5) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                    Offset + Gain + Hotpixel");
            }
            else if(OptionIndex==6) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                    Black Reference Image");
            }
            else {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                    Grey Reference Image");
            }

            SetCtrlAttribute(panelHandle, PANEL_PORT_INFO, ATTR_DIMMED, 1);
            break;
    }
    return 0;
}

int CVICALLBACK Reset (int panel, int control, int event, void *callbackData, int eventData1,
                       int eventData2)
{
    switch (event) {
        case EVENT_COMMIT:
            ResetTextBox (panelHandle, PANEL_DISPLAY_PORTS, "Reseting and Initializing");
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Got reset property notice");
            SetCtrlVal (triggerHandle, TRIGGER_FREE_RUNNING, 0);
            SetCtrlVal (triggerHandle, TRIGGER_INTERFACE_TRIGGER, 0);
            SetCtrlVal (triggerHandle, TRIGGER_IO_TRIGGER, 0);
            SetCtrlVal(triggerHandle, TRIGGER_BY_CAMERA, 0);
            SetCtrlVal(triggerHandle, TRIGGER_TRIG_PULSE, 0);
            SetCtrlVal(triggerHandle, TRIGGER_TRIGGER_DELAY, 0.00000);
            SetCtrlVal(triggerHandle,TRIGGER_TRIG_SIG_ACTIVE, 0);
            SetPropertyString(port, "Reset", "1");
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1,
                                   "Resetting camera attributes");
            Port_Info(99,99,EVENT_COMMIT,0,99,99);

            break;

        case EVENT_RIGHT_CLICK:
             MessagePopup ("Reset Button Info", "Reset the camera and load the default
                           configuration.");

            break;
    }
    return 0;
}

int CVICALLBACK Store_Default (int panel, int control, int event, void *callbackData,
                               int eventData1, int eventData2)
{
   switch (event) {
        case EVENT_COMMIT:
            if (ConfirmPopup ("Store Defaults", "Are you sure to store the current settings as
                                                 defaults?\nThis will set the current settings to
                                                 the power-on defaults?")>0) {
                SetPropertyString(port, "StoreDefaults", "1");
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1,
                                   "Settings stored as default");
            }

            break;
    }
    return 0;
}

int CVICALLBACK Factory_Reset (int panel, int control, int event, void *callbackData,
                               int eventData1, int eventData2)
{
   switch (event) {
        case EVENT_COMMIT:
            if (ConfirmPopup("Factory Reset?","Are you sure you want to restore the factory
                              defaults?\nThis will set the camera to delivery status!")>0) {

                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Got reset property
                                   notice");
                ResetTextBox (panelHandle, PANEL_DISPLAY_PORTS, "Resetting camera to factory
                                   defaults.........");
                SetCtrlVal (triggerHandle, TRIGGER_FREE_RUNNING, 0);
                SetCtrlVal (triggerHandle, TRIGGER_INTERFACE_TRIGGER, 0);
                SetCtrlVal (triggerHandle, TRIGGER_IO_TRIGGER, 0);
                SetCtrlVal(triggerHandle, TRIGGER_BY_CAMERA, 0);
                SetCtrlVal(triggerHandle, TRIGGER_TRIG_PULSE, 0);
                SetCtrlVal(triggerHandle,TRIGGER_TRIG_SIG_ACTIVE, 0);
                SetPropertyString(port, "FactoryReset", "1");
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Factory Reset Done");
                Port_Info(99,99,EVENT_COMMIT,0,99,99);

            }
            break;
    }
    return 0;
}

int CVICALLBACK Frame_Rate_Update (int panel, int control, int event, void *callbackData,
                                   int eventData1, int eventData2)
{
    char    message[100], data[64];

    switch (event) {
        case EVENT_COMMIT:
            GetPropertyString(port,"FrameRate", data);
            FrameRateUpdate=atof(data);
            SetCtrlVal(cameraHandle, CAM_PANEL_FRAME_UPDATE, FrameRateUpdate);
            sprintf(message, "Frame Rate: %s fps", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            break;
   }
    return 0;
}

int CVICALLBACK Average_Value_Update (int panel, int control, int event, void *callbackData,
                                      int eventData1, int eventData2)
{
    char message[100];

    switch (event) {
        case EVENT_COMMIT:
            GetPropertyInt(port,"Average.Value", &AverageValue);
            SetCtrlVal(cameraHandle, CAM_PANEL_AVERAGE_VALUE, AverageValue);
            sprintf(message, "Average Value: %d", AverageValue);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            break;
   }
    return 0;
}

/*********************************************************************************
*----------------------------------CAMERA---------------------------------------*
```

```
*                              Exposure Settings                                  *
**********************************************************************************/

int CVICALLBACK Exposure_Time (int panel, int control, int event, void *callbackData,
                               int eventData1, int eventData2)
{
    char    message[100];
    int     sts;

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (exposureHandle, EXPOSURE_EXPOSURETIME, &ExpTime);
            GetCtrlVal(exposureHandle, EXPOSURE_CONST_FRAME_RATE, &sts);
            SetPropertyFloat(port, "ExposureTime", ExpTime);

            if (sts<1){
                if (ExpTime<412){
                    FramTime = ExpTime + 6.664;
                }
                else {
                    FramTime = 419.000;
                }
                SetPropertyFloat(port, "FrameTime", FramTime);
                SetCtrlVal(exposureHandle, EXPOSURE_FRAMETIME, FramTime);
            }
            else {
                SetCtrlAttribute(exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 0);
            }

            GetPropertyFloat(port, "ExposureTime", &ExpTime);
            sprintf(message,"Exposure Time: %f", ExpTime);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            if (sts<1){
                GetPropertyFloat(port, "FrameTime", &FramTime);
                sprintf(message,"Frame Time: %f", FramTime);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
            }

            break;
   }
    return 0;
}

int CVICALLBACK Frame_Time (int panel, int control, int event, void *callbackData,
                            int eventData1, int eventData2)
{
    char message[100];

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (exposureHandle, EXPOSURE_FRAMETIME, &FramTime);
            SetPropertyFloat(port, "FrameTime", FramTime);
            GetPropertyFloat(port, "FrameTime", &FramTime);
            sprintf(message,"Frame Time: %f", FramTime);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            break;
    }
    return 0;
}

int CVICALLBACK Constant_Frame_Rate (int panel, int control, int event, void *callbackData,
                                     int eventData1, int eventData2)
{
    int sts;

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (exposureHandle, EXPOSURE_CONST_FRAME_RATE, &sts);

            if (sts>0){
                SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 0);
                SetPropertyString(port,"Trigger.CFR", "1");
            }
            else {
                SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 1);
                SetPropertyString(port,"Trigger.CFR", "0");
            }

            GetPropertyInt(port, "Trigger.CFR", &sts);

            if (sts<1){
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Constant frame rate:
                                   Disabled");
            }
            else {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Constant frame rate:
                                   Enabled");
            }

            break;
    }
    return 0;
}

int CVICALLBACK Counter_Image_Update (int panel, int control, int event, void *callbackData,
                                      int eventData1, int eventData2)
{
   switch (event) {
        case EVENT_COMMIT:
            GetPropertyInt(port,"Counter.Image", &CounterImageUpdate);
            SetCtrlVal(exposureHandle, EXPOSURE_IMG_COUNTER, CounterImageUpdate);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Image Counter Updated");

            break;
    }
    return 0;
}

int CVICALLBACK Image_Counter_Reset (int panel, int control, int event, void *callbackData,
                                     int eventData1, int eventData2)
{
   switch (event) {
        case EVENT_COMMIT:
            SetPropertyString(port, "Counter.ImageReset", "1");
            GetPropertyInt(port,"Counter.Image", &CounterImageUpdate);
            SetCtrlVal(exposureHandle, EXPOSURE_IMG_COUNTER, CounterImageUpdate);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Image Counter Reseted");

            break;
    }
    return 0;
}

int CVICALLBACK Missed_Trigger_Counter (int panel, int control, int event, void *callbackData,
                                        int eventData1, int eventData2)
{
   switch (event) {
        case EVENT_COMMIT:
            GetPropertyInt(port,"Counter.MissedTrigger", &MissedTriggerCounter);
            SetCtrlVal(exposureHandle, EXPOSURE_MISS_TRIG_COUNTER, MissedTriggerCounter);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Missed Trigger Counter
                               Updated");

            break;
    }
    return 0;
}


int CVICALLBACK Missed_Trigger_Reset (int panel, int control, int event, void *callbackData,
                                      int eventData1, int eventData2)
{
   switch (event) {
        case EVENT_COMMIT:

            SetPropertyString(port, "Counter.MissedTriggerReset", "1");
```

```
            GetPropertyInt(port,"Counter.MissedTrigger", &MissedTriggerCounter);
            SetCtrlVal(exposureHandle, EXPOSURE_MISS_TRIG_COUNTER, MissedTriggerCounter);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Missed Trigger Counter
                               Reseted");

            break;
    }
    return 0;
}


int CVICALLBACK Status_Line (int panel, int control, int event, void *callbackData,
                             int eventData1, int eventData2)
{
    int sts;
    char data[64];

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (exposureHandle, EXPOSURE_STATUS_LINE, &sts);

            if (sts>0){
                SetPropertyString(port,"EnStatusLine", "1");
            }
            else {
                SetPropertyString(port,"EnStatusLine", "0");
            }

            GetPropertyString(port, "EnStatusLine", data);
            sts=atoi(data);
            if (sts<1){
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Status Line: Disabled");
            }
            else {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Status Line: Enabled");
            }

            break;
    }
    return 0;
}

/*********************************************************************************
*----------------------------------CAMERA----------------------------------------*
*                              Trigger Settings                                  *
**********************************************************************************/

int CVICALLBACK Trigger (int panel, int control, int event, void *callbackData, int eventData1,
                         int eventData2)
{
    int     i, sts;

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (triggerHandle, control, &sts);
            if (!sts) {
                SetCtrlVal (triggerHandle, control, 1);

            return 0;
            }

            if (control==TRIGGER_FREE_RUNNING) {
                    SetPropertyInt (port, "Trigger.Source", 0);
                    SetPropertyInt (port, "Trigger.LevelControlled", 0);
                    SetCtrlVal(triggerHandle, TRIGGER_BY_CAMERA, 1);
                    SetCtrlVal(triggerHandle, TRIGGER_TRIG_PULSE, 0);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_EXPOSURETIME, ATTR_DIMMED, 0);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_CONST_FRAME_RATE, ATTR_DIMMED, 0);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_DECORATION_1, ATTR_FRAME_COLOR,
                                      VAL_DK_BLUE);
                    GetCtrlVal (exposureHandle, EXPOSURE_CONST_FRAME_RATE, &sts);
                    if (sts>0){
                        SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 0);
                    }
                    else {
                        SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 1);
                    }
                    SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_TRIGGER_DELAY, ATTR_DIMMED, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
                                      VAL_DK_GRAY);
                    SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_SIG_ACTIVE, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_LINLOG, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_1, ATTR_FRAME_COLOR,
                                      VAL_GREEN);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR,
                                      VAL_YELLOW);
            }
            else if (control == TRIGGER_INTERFACE_TRIGGER) {
                    SetPropertyInt (port, "Trigger.Source", 1);
                    GetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, &sts);
                        if (sts>0){
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
                                              VAL_DK_GRAY);
                        }
                        else {
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
                                              VAL_RED);
                        }
                    SetCtrlAttribute (triggerHandle, TRIGGER_TRIGGER_DELAY, ATTR_DIMMED, 0);
                    SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_SIG_ACTIVE, ATTR_DIMMED, 0);
            }
            else  {
                    SetPropertyInt (port, "Trigger.Source", 2);
                    GetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, &sts);
                        if (sts>0){
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
                                              VAL_DK_GRAY);
                    }
                        else {
                            SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 0);
                            SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
                                              VAL_RED);
                        }
                    SetCtrlAttribute (triggerHandle, TRIGGER_TRIGGER_DELAY, ATTR_DIMMED, 0);
                    SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_SIG_ACTIVE, ATTR_DIMMED, 0);
            }

            for (i = 0; i < 3; i++) {
                if (control != ctl[i]) {
                    SetCtrlVal (panel, ctl[i], 0);
                }
            }

            GetPropertyInt (port, "Trigger.Source", &Trig);

            if (Trig==0) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Trigger Source:
                                   Free Running");
            }
            else if (Trig==1) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Trigger Source:
                                   Interface Trigger");
            }
            else {
```

```
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Trigger Source:
                                   I/O Trigger");
            }


            break;
            }
    return 0;
}

int CVICALLBACK Exposure_Defined (int panel, int control, int event, void *callbackData,
                                  int eventData1, int eventData2)
{
    int     i, sts;

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (triggerHandle, control, &sts);

            if (!sts) {
                SetCtrlVal (triggerHandle, control, 1);
                return 0;
            }

            if (control==TRIGGER_BY_CAMERA) {
                    SetPropertyInt (port, "Trigger.LevelControlled", 0);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_EXPOSURETIME, ATTR_DIMMED, 0);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_CONST_FRAME_RATE, ATTR_DIMMED, 0);
                    GetCtrlVal (exposureHandle, EXPOSURE_CONST_FRAME_RATE, &sts);
                    if (sts>0){
                        SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 0);
                    }
                    else {
                        SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 1);
                    }
                    SetCtrlAttribute (exposureHandle, EXPOSURE_DECORATION_1, ATTR_FRAME_COLOR,
                                      VAL_DK_BLUE);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_LINLOG, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_1, ATTR_FRAME_COLOR,
                                      VAL_GREEN);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR,
                                      VAL_YELLOW);
            }
            else  {
                    SetPropertyInt (port, "Trigger.LevelControlled", 1);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_EXPOSURETIME, ATTR_DIMMED, 1);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_FRAMETIME, ATTR_DIMMED, 1);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_CONST_FRAME_RATE, ATTR_DIMMED, 1);
                    SetCtrlAttribute (exposureHandle, EXPOSURE_DECORATION_1, ATTR_FRAME_COLOR,
                                      VAL_DK_GRAY);
                    SetCtrlIndex (characteristicsHandle, CHARAC_LINLOG, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_LINLOG, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_1, ATTR_FRAME_COLOR,
                                      VAL_DK_GRAY);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 0);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR,
                                      VAL_RED);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 1);
                    SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR,
                                      VAL_DK_GRAY);
            }

            for (i = 0; i < 2; i++) {
                if (control != expdef[i]) {
                    SetCtrlVal (panel, expdef[i], 0);
                }
            }

            GetPropertyInt (port, "Trigger.LevelControlled", &TrigLevel);
            if (TrigLevel==0) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Exposure Time defined by:
                                   Camera");
            }
            else {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Exposure Time defined by:
                                   Trigger Pulse Width");
            }

            break;
            }
    return 0;
}

int CVICALLBACK Trigger_Delay (int panel, int control, int event, void *callbackData,
                               int eventData1, int eventData2)
{
    char    message[100], data[64];

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (triggerHandle, TRIGGER_TRIGGER_DELAY, &TrigDelay);
            SetPropertyFloat (port,  "Trigger.Delay", TrigDelay);
            GetPropertyFloat(port, "Trigger.Delay", &TrigDelay);
            sprintf(message,"Trigger Delay: %f", TrigDelay);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            break;
    }
    return 0;
}

int CVICALLBACK Trigger_Signal_Active_Low (int panel, int control, int event, void *callbackData,
                                           int eventData1, int eventData2)
{
    int sts;

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (triggerHandle, TRIGGER_TRIG_SIG_ACTIVE, &sts);

            if (sts>0){
                SetPropertyString(port,"Trigger.Invert", "1");
            }
            else {
                SetPropertyString(port,"Trigger.Invert", "0");
            }

            GetPropertyInt(port, "Trigger.Invert", &sts);
            if (sts<1){
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Trigger signal active low:
Disabled");
            }
            else {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Trigger signal active low:
Enabled");
            }

            break;
    }
    return 0;
}

int CVICALLBACK Strobe_Delay (int panel, int control, int event, void *callbackData,
                              int eventData1, int eventData2)
{
    char    message[100], data[64];

    switch (event) {
        case EVENT_COMMIT:
```

```

            GetCtrlVal (triggerHandle, TRIGGER_STROBE_DELAY, &StrobeDelay);
            SetPropertyFloat (port,  "Strobe.Delay", StrobeDelay);
            GetPropertyFloat(port, "Strobe.Delay", &StrobeDelay);
            sprintf(message,"Strobe Delay: %f", StrobeDelay);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            break;
    }
    return 0;
}

int CVICALLBACK Strobe_Pulse_Width (int panel, int control, int event, void *callbackData,
                                    int eventData1, int eventData2)
{
    char    message[100], data[64];

    switch (event) {
        case EVENT_COMMIT:

            GetCtrlVal (triggerHandle, TRIGGER_STROBE_PULSE_WIDTH, &StrobePulseWidth);
            SetPropertyFloat (port,  "Strobe.PulseWidth", StrobePulseWidth);
            GetPropertyFloat(port, "Strobe.PulseWidth", &StrobePulseWidth);
            sprintf(message,"Strobe Pulse Width: %f", StrobePulseWidth);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            break;
    }
    return 0;
}

int CVICALLBACK Strobe_Signal_Active_Low (int panel, int control, int event, void *callbackData,
                                          int eventData1, int eventData2)
{
    int sts;
    char data[64];

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (triggerHandle, TRIGGER_STROBE_SIG_ACTIVE, &sts);

            if (sts>0){
                SetPropertyString(port,"Strobe.Invert", "1");
            }
            else {
                SetPropertyString(port,"Strobe.Invert", "0");
            }

            GetPropertyString(port, "Strobe.Invert", data);
            sts=atoi(data);
            if (sts<1){
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Strobe signal active low:
                                   Disabled");
            }
            else {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Strobe signal active low:
                                   Enabled");
            }

            break;
    }
    return 0;
}

/*********************************************************************************
*----------------------------------CAMERA----------------------------------------*
*                                Data Output                                     *
**********************************************************************************/

int CVICALLBACK Output_Mode (int panel, int control, int event, void *callbackData, int eventData1,
                             int eventData2)
{
   switch (event) {
        case EVENT_COMMIT:
            GetCtrlAttribute(dataoutputHandle, DATAOUTPUT_OUTPUT_MODE, ATTR_CTRL_INDEX,
                             &OutputMode);
            SetPropertyInt(port, "DataOutMode", OutputMode);
            GetPropertyInt(port, "DataOutMode", &OutputMode);
            if(OutputMode==0){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Output Mode: Normal");
            }
            else if(OutputMode==1){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Output Mode: LFSR");
            }
            else if(OutputMode==2){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Output Mode: Ramp");
            }
            else {
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Output Mode: LUT");
            }

            break;
    }
    return 0;
}

int CVICALLBACK Resolution (int panel, int control, int event, void *callbackData, int eventData1,
                            int eventData2)
{
   switch (event) {
        case EVENT_COMMIT:
            GetCtrlAttribute(dataoutputHandle, DATAOUTPUT_RESOLUTION, ATTR_CTRL_INDEX, &Res);
            SetPropertyInt(port, "DataResolution", Res);
            GetPropertyInt(port, "DataResolution", &Res);
            if(Res==0){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Resolution: 12 bit");
            }
            else if(Res==1){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Resolution: 10 bit");
            }
            else {
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Data Resolution: 8 bit");
            }

            break;
    }
    return 0;
}

int CVICALLBACK Digital_Gain (int panel, int control, int event, void *callbackData,
                              int eventData1, int eventData2)
{
   switch (event) {
        case EVENT_COMMIT:
            GetCtrlAttribute(dataoutputHandle, DATAOUTPUT_DIGITAL_GAIN, ATTR_CTRL_INDEX, &Gain);
            SetPropertyInt(port, "DigitalGain", Gain);
            GetPropertyInt(port, "DigitalGain", &Gain);
            if(Gain==0){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Digital Gain : 1x");
            }
            else if(Gain==1){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Digital Gain : 2x");
            }
            else {
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Digital Gain : 4x");
            }

            break;
    }
    return 0;
}

/*********************************************************************************
*----------------------------------CAMERA----------------------------------------*
*                           Characteristic Settings                              *
*********************************************************************************/

int CVICALLBACK LinLog_Mode (int panel, int control, int event, void *callbackData,
```

```
                             int eventData1, int eventData2)
{
    int     sts;
    char    message[100], data[64];

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlAttribute(characteristicsHandle, CHARAC_LINLOG, ATTR_CTRL_INDEX, &LinLog);
            GetCtrlVal (characteristicsHandle, CHARAC_SKIMMING, &Skim);
            GetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, &sts);
            SetPropertyInt(port, "LinLog.Mode", LinLog);
            GetPropertyInt(port, "LinLog.Mode", &LinLog);
            if(LinLog==0){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "LinLog Mode: Off");

            if (sts>0) {
                SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 1);
                SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR,
                                  VAL_DK_GRAY);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Disabled");
            }
            else {
                SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 0);
                SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR,
                                  VAL_RED);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Enabled");

                if (Skim>0) {
                    SetPropertyInt(port, "Skim", Skim);
                    GetPropertyString(port, "Skim", data);
                    sprintf(message,"Skimming : %s", data);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
                }
            }

            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 1);
            }
            else if(LinLog==1){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "LinLog Mode: Low
                               Compression");
            if (sts>0) {
                SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 1);
                SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR,
                                  VAL_DK_GRAY);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Disabled");
            }
            else {
                SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 0);
                SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR,
                                  VAL_RED);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Enabled");

                if (Skim>2){
                    Skim=2;
                    SetPropertyInt(port, "Skim", Skim);
                    SetCtrlVal(characteristicsHandle, CHARAC_SKIMMING, Skim);
                    GetPropertyString(port, "Skim", data);
                    sprintf(message,"Skimming : %s", data);
                    InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);
                }
            }
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 1);
            }
            else if(LinLog==2){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "LinLog Mode: Normal
                               Compression");

            if (Skim>0) {
                Skim=0;
                SetPropertyInt(port, "Skim", Skim);
                SetCtrlVal(characteristicsHandle, CHARAC_SKIMMING, Skim);
                SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR,
                                  VAL_DK_GRAY);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Disabled");
                SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 0);
                SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR, VAL_RED);
            }

            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 1);
            }
            else if(LinLog==3){
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "LinLog Mode: High
                               Compression");

            if (Skim>0) {
                Skim=0;
                SetPropertyInt(port, "Skim", Skim);
                SetCtrlVal(characteristicsHandle, CHARAC_SKIMMING, Skim);
                SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR,
                                  VAL_DK_GRAY);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Disabled");
                SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 0);
                SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR, VAL_RED);
            }

            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 1);
            SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 1);
            }
            else {
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "LinLog Mode: User Defined");

            if (Skim>0) {
                Skim=0;
                SetPropertyInt(port, "Skim", Skim);
                SetCtrlVal(characteristicsHandle, CHARAC_SKIMMING, Skim);
                SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_2, ATTR_FRAME_COLOR,
                                  VAL_DK_GRAY);
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Skimming Disabled");
                SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 0);
                SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR, VAL_RED);
            }

            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_1, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_VALUE_2, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_1, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_TIME_2, ATTR_DIMMED, 0);
            SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 1);
            }
            break;
    }
    return 0;
}

int CVICALLBACK Value_1 (int panel, int control, int event, void *callbackData, int eventData1,
                         int eventData2)
{
    char message[100], data[64];

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal(characteristicsHandle, CHARAC_VALUE_1, &Value1);
            GetCtrlVal(characteristicsHandle, CHARAC_VALUE_2, &Value2);

            if (Value1<Value2){
                SetPropertyInt(port, "LinLog.Value1", Value1);
```

```
                SetCtrlVal(characteristicsHandle, CHARAC_VALUE_2, Value1);
            }
            else {
                SetPropertyInt(port, "LinLog.Value1", Value1);
            }
            GetPropertyString(port, "LinLog.Value1", data);
            sprintf(message,"LinLog Mode User Defined Value 1 : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            break;
    }
    return 0;
}

int CVICALLBACK Time_1 (int panel, int control, int event, void *callbackData, int eventData1,
                        int eventData2)
{
    char message[100], data[64];

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal(characteristicsHandle, CHARAC_TIME_1, &Time1);
            GetCtrlVal(characteristicsHandle, CHARAC_TIME_2, &Time2);

            if (Time1>Time2){
                SetPropertyInt(port, "LinLog.Time1", Time2);
                SetCtrlVal(characteristicsHandle, CHARAC_TIME_1, Time2);
                }
            else {
                SetPropertyInt(port, "LinLog.Time1", Time2);
            }
            GetPropertyString(port, "LinLog.Time1", data);
            sprintf(message,"LinLog Mode User Defined Time 1 : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            break;
    }
    return 0;
}

int CVICALLBACK Value_2 (int panel, int control, int event, void *callbackData, int eventData1,
                         int eventData2)
{
    char message[100], data[64];

   switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal(characteristicsHandle, CHARAC_VALUE_2, &Value2);
            GetCtrlVal(characteristicsHandle, CHARAC_VALUE_1, &Value1);

            if (Value2>Value1){
                SetPropertyInt(port, "LinLog.Value2", Value1);
                SetCtrlVal(characteristicsHandle, CHARAC_VALUE_2, Value1);
                }
            else {
                SetPropertyInt(port, "LinLog.Value2", Value2);
            }
            GetPropertyString(port, "LinLog.Value2", data);
            sprintf(message,"LinLog Mode User Defined Value 2 : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            break;
    }
    return 0;
}

int CVICALLBACK Time_2 (int panel, int control, int event, void *callbackData, int eventData1,
                        int eventData2)
{
    char message[100], data[64];

   switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal(characteristicsHandle, CHARAC_TIME_2, &Time2);
```

```c
            GetCtrlVal(characteristicsHandle, CHARAC_TIME_1, &Time1);

            if (Time2<Time1){
                SetPropertyInt(port, "LinLog.Time2", Time2);
                SetCtrlVal(characteristicsHandle, CHARAC_TIME_1, Time2);
            }
            else {
                SetPropertyInt(port, "LinLog.Value1", Value1);
            }
            GetPropertyString(port, "LinLog.Time2", data);
            sprintf(message,"LinLog Mode User Defined Time 2 : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            break;
    }
    return 0;
}


int CVICALLBACK Skimming (int panel, int control, int event, void *callbackData, int eventData1,
                          int eventData2)
{
    int sts;
    char message[100], data[64];

    switch (event) {
        case EVENT_COMMIT:

            GetCtrlVal(characteristicsHandle, CHARAC_SKIMMING, &Skim);
            GetPropertyInt(port, "LinLog.Mode", &LinLog);

            if(LinLog==0){
                SetPropertyInt (port, "Skim", Skim);
            }
            else if(LinLog==1){
                if (Skim>2){
                    Skim=2;
                    SetCtrlVal (characteristicsHandle, CHARAC_SKIMMING, Skim);
                    SetPropertyInt (port, "Skim", Skim);
                }
                else {
                    SetPropertyInt (port, "Skim", Skim);
                }
            }
            else {
                Skim=0;
                SetCtrlVal (characteristicsHandle, CHARAC_SKIMMING, Skim);
                SetPropertyInt (port, "Skim", Skim);
            }
            GetPropertyString(port, "Skim", data);
            sts=atoi(data);
            if (sts>0){
                SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 1);
                SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR,
                                  VAL_DK_GRAY);
            }
            else {
                SetCtrlAttribute (characteristicsHandle, CHARAC_SIM_READ_OUT, ATTR_DIMMED, 0);
                SetCtrlAttribute (characteristicsHandle, CHARAC_DECORATION_4, ATTR_FRAME_COLOR,
                                  VAL_YELLOW);
            }
            sprintf(message,"Skimming : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            break;
    }
    return 0;
}


int CVICALLBACK Sim_Read_Out (int panel, int control, int event, void *callbackData,
                              int eventData1, int eventData2)
{
    int sts;
```

```c
    char data[64];

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (characteristicsHandle, CHARAC_SIM_READ_OUT, &sts);
            GetPropertyInt (port, "Trigger.Source", &Trig);

            if (sts>0){

                SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3,  ATTR_FRAME_COLOR,
                                  VAL_DK_GRAY);
                SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 1);
                SetPropertyString(port,"Trigger.Interleave", "1");
            }
            else {
                if (Trig==0) {
                    SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 1);
                    SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
                                      VAL_DK_GRAY);
                }
                else {
                    SetCtrlAttribute (triggerHandle, TRIGGER_BY_CAMERA, ATTR_DIMMED, 0);
                    SetCtrlAttribute (triggerHandle, TRIGGER_TRIG_PULSE, ATTR_DIMMED, 0);
                    SetCtrlAttribute (triggerHandle, TRIGGER_DECORATION_3, ATTR_FRAME_COLOR,
                                      VAL_RED);
                }
                SetPropertyString(port,"Trigger.Interleave", "0");
                GetPropertyInt(port, "LinLog.Mode", &LinLog);
                if(LinLog<=1){
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 0);
                }
                else {
                    SetCtrlAttribute (characteristicsHandle, CHARAC_SKIMMING, ATTR_DIMMED, 1);
                }
            }
            GetPropertyString(port, "Trigger.Interleave", data);
            sts=atoi(data);
            if (sts<1){
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Simultaneous readout:
                                   Disabled");
            }
            else {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Simultaneous readout:
                                   Enabled");
            }

            break;

        case EVENT_RIGHT_CLICK:
             MessagePopup ("Simultaneous Readout (Interleave) Info", "Enables/Disables the
                            simultaneous readout mode.\nThe simultaneous readout mode allows
                            higher frame rate.\n\n Note:\n\n Combination of property Trigger.
                            Interleave and property Skim is not available!\n Combination of
                            property Trigger.Interleave and property Trigger.LevelControlled is
                            not available!");

            break;
    }
    return 0;
}

/*********************************************************************************
*----------------------------------CAMERA----------------------------------------*
*                              Correction Settings                               *
*********************************************************************************/

int CVICALLBACK Correction_Mode (int panel, int control, int event, void *callbackData,
                                 int eventData1, int eventData2)
{
   switch (event) {
        int OptionIndex;
        case EVENT_COMMIT:
            Radio_GetMarkedOption(correctionHandle, CORRECTION_CORRECTION_MODE, &OptionIndex);
            SetPropertyInt(port, "Correction.Mode", OptionIndex);
            GetPropertyInt(port,  "Correction.Mode", &OptionIndex);
            if(OptionIndex==0) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode : Off");
            }
            else if(OptionIndex==1) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                   Offset");
            }
            else if(OptionIndex==2) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                   Offset + Hotpixel");
            }
            else if(OptionIndex==3) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                   Hotpixel");
            }
            else if(OptionIndex==4) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                   Offset + Gain");
            }
            else if(OptionIndex==5) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                   Offset + Gain + Hotpixel");
            }
            else if(OptionIndex==6) {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                   Black Reference Image");
            }
            else {
                InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, "Correction Mode :
                                   Grey Reference Image");
            }

            break;
    }
    return 0;
}

int CVICALLBACK Black_Level_Offset (int panel, int control, int event, void *callbackData,
                                    int eventData1, int eventData2)
{
    char message[100], data[64];

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal(characteristicsHandle, CHARAC_BLK_LVL_OFFSET, &BlkOffset);
            SetPropertyInt(port, "Voltages.BlackLevelOffset", BlkOffset); GetPropertyString(port,
                                 "Voltages.BlackLevelOffset", data);
            sprintf(message,"Black level offset : %s", data);
            InsertTextBoxLine (panelHandle, PANEL_DISPLAY_PORTS, -1, message);

            break;
    }
    return 0;
}

/*********************************************************************************
*----------------------------------ANALOG OUTPUT MODULE--------------------------*
*                                     Voltage Setting                            *
**********************************************************************************/

int CVICALLBACK RangeCallback(int panel, int control, int event, void *callbackData,
                              int eventData1, int eventData2)
{
    if( event==EVENT_COMMIT ) {
        double  min,max;

        GetCtrlVal(trigHandle,TRIG_PANEL_MINVAL,&min);
        GetCtrlVal(trigHandle,TRIG_PANEL_MAXVAL,&max);
        if( min<max )
```

```
            SetAxisScalingMode(trigHandle,TRIG_PANEL_GRAPH,VAL_LEFT_YAXIS,VAL_MANUAL,min,max);
        return 1;
    }
    return 0;
}

int CVICALLBACK StartCallback(int panel, int control, int event, void *callbackData,
                              int eventData1, int eventData2)
{

    int         outwfm[NUMCHANS];
    int         outcycles[NUMCHANS];
    double      outamp[NUMCHANS];
    double      outfreq[NUMCHANS];
    double      dutycycle[NUMCHANS];
    double      amplitude, amplitude2, amplitude3, amplitude4, Dutycycle, Dutycycle2, Dutycycle3,
                Dutycycle4;
    int         typeofwaveform, typeofwaveform2, typeofwaveform3, typeofwaveform4;
    int         frequency, frequency2, frequency3, frequency4, delay;
    int         error=0;
    TaskHandle  taskHandle=0;
    char        chan[256];
    double      min,max,rate, XCorrect, YCorrect;
    uInt32      sampsPerCycle, numChannels, TotalsampsPerCycle=0, arrayspot = 0;
    float64     *data=NULL,*temp=NULL;
    char        errBuff[2048]={'\0'};
    bool32      done=0;
    double      phase=0.0;
    int         log,j,k,i;
    int32       written;

    if( event==EVENT_COMMIT ) {
        GetCtrlVal (trigHandle, TRIG_PANEL_WFMTYPE, &typeofwaveform);
        GetCtrlVal (trigHandle, TRIG_PANEL_WFMTYPE_2, &typeofwaveform2);
        GetCtrlVal (trigHandle, TRIG_PANEL_WFMTYPE_3, &typeofwaveform3);
        GetCtrlVal (trigHandle, TRIG_PANEL_WFMTYPE_4, &typeofwaveform4);
        GetCtrlVal (trigHandle, TRIG_PANEL_AMPLITUDE, &amplitude);
        GetCtrlVal (trigHandle, TRIG_PANEL_AMPLITUDE_2, &amplitude2);
        GetCtrlVal (trigHandle, TRIG_PANEL_AMPLITUDE_3, &amplitude3);
        GetCtrlVal (trigHandle, TRIG_PANEL_AMPLITUDE_4, &amplitude4);
        GetCtrlVal (trigHandle, TRIG_PANEL_FREQUENCY, &frequency);
        GetCtrlVal (trigHandle, TRIG_PANEL_FREQUENCY_2, &frequency2);
        GetCtrlVal (trigHandle, TRIG_PANEL_FREQUENCY_3, &frequency3);
        GetCtrlVal (trigHandle, TRIG_PANEL_FREQUENCY_4, &frequency4);
        GetCtrlVal (trigHandle, TRIG_PANEL_CHANNEL,chan);
        GetCtrlVal (trigHandle, TRIG_PANEL_MINVAL,&min);
        GetCtrlVal (trigHandle, TRIG_PANEL_MAXVAL,&max);
        GetCtrlVal (trigHandle, TRIG_PANEL_RATE,&rate);
        GetCtrlVal (trigHandle, TRIG_PANEL_DUTYCYCLE,&Dutycycle);
        GetCtrlVal (trigHandle, TRIG_PANEL_DUTYCYCLE_2,&Dutycycle2);
        GetCtrlVal (trigHandle, TRIG_PANEL_DUTYCYCLE_3,&Dutycycle3);
        GetCtrlVal (trigHandle, TRIG_PANEL_DUTYCYCLE_4,&Dutycycle4);
        GetCtrlVal (trigHandle, TRIG_PANEL_X_CORRECT, &XCorrect);
        GetCtrlVal (trigHandle, TRIG_PANEL_Y_CORRECT, &YCorrect);
        GetCtrlVal (trigHandle, TRIG_PANEL_DELAY, &delay);
        sampsPerCycle = rate;
        outwfm[0] = typeofwaveform;
        outwfm[1] = typeofwaveform2;
        outwfm[2] = typeofwaveform3;
        outwfm[3] = typeofwaveform4;
        outcycles[0] = sampsPerCycle;
        outcycles[1] = sampsPerCycle;
        outcycles[2] = sampsPerCycle;
        outcycles[3] = sampsPerCycle;
        outamp[0] = amplitude;
        outamp[1] = amplitude2;
        outamp[2] = amplitude3;
        outamp[3] = amplitude4;
        outfreq[0] = frequency;
        outfreq[1] = frequency2;
        outfreq[2] = frequency3;
        outfreq[3] = frequency4;
        dutycycle[0] = Dutycycle;
        dutycycle[1] = Dutycycle2;
        dutycycle[2] = Dutycycle3;
        dutycycle[3] = Dutycycle4;

        SetCtrlAttribute(trigHandle, TRIG_PANEL_GRAPH,ATTR_XAXIS_GAIN,1.0/rate);
        log = log10(rate);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_GRAPH,ATTR_XPRECISION,log);
        DAQmxErrChk (DAQmxCreateTask("",&taskHandle));
        DAQmxErrChk (DAQmxCreateAOVoltageChan(taskHandle,chan,"",min,max,DAQmx_Val_Volts,NULL));
        DAQmxErrChk (DAQmxCfgSampClkTiming(taskHandle,"",rate,DAQmx_Val_Rising,DAQmx_Val_ContSamps,
                     1000));
        DAQmxErrChk (DAQmxGetTaskAttribute(taskHandle,DAQmx_Task_NumChans,&numChannels));
        TotalsampsPerCycle = numChannels*sampsPerCycle;

        if( (data=malloc(TotalsampsPerCycle*sizeof(float64)))==NULL ) {
            MessagePopup("Error","Not enough memory");
            goto Error;
        }

        if( (temp=malloc(TotalsampsPerCycle*sizeof(float64)))==NULL ) {
            MessagePopup("Error","Not enough memory");
            goto Error;
        }

        for(j=0;j<numChannels;j++){
            switch( outwfm[j] ) {
                case 1:     TriangleWave(sampsPerCycle,outamp[j],outfreq[j]/sampsPerCycle,&phase,
                                         temp);  break;
                case 2:     SquareWave(sampsPerCycle,outamp[j],outfreq[j]/sampsPerCycle,&phase,
                                       dutycycle[j],temp); break;
                case 3:     SawtoothWave(sampsPerCycle,outamp[j],outfreq[j]/sampsPerCycle,&phase,
                                         temp); break;
                case 4:     GenStepWave(sampsPerCycle,outamp[j],outfreq[j]/sampsPerCycle,&phase,
                                        temp); break;
                case 5:     LoadDataFile(sampsPerCycle,sampsPerCycle,temp); break;
                default:    SineWave(sampsPerCycle,outamp[j],outfreq[j]/sampsPerCycle,&phase,temp); b
reak;
            }

            for(k=0;k<sampsPerCycle;k++){
                data[arrayspot+k] = temp[k];
            }
            arrayspot = arrayspot + sampsPerCycle;

        }

        for(k=0;k<sampsPerCycle;k++){
            data[k] = data[k]+XCorrect;
        }

        if(numChannels>1) {
            for(k=0;k<sampsPerCycle;k++){
                data[sampsPerCycle+k] = data[sampsPerCycle+k]+YCorrect;
            }
        }

        if (numChannels>2){
            if(delay>0){
                for(k=0;k<delay;k++){
                    temp[k]=data[(3*sampsPerCycle)-delay+k];
                    }
                    if (delay<sampsPerCycle){
                        for(k=delay;k<sampsPerCycle;k++){
                            temp[k]=data[(2*sampsPerCycle+k)-delay];
                        }
                    }
                for(k=0;k<sampsPerCycle;k++){
                    data[2*sampsPerCycle+k]=temp[k];
                }
            }
        }

        LinEv1D(data,TotalsampsPerCycle,1.0,(max+min)/2.0,data);
        DeleteGraphPlot(trigHandle, TRIG_PANEL_GRAPH,-1,VAL_DELAYED_DRAW);
        arrayspot = 0;
```

```
        for (i=0; i<numChannels; i++) {
            PlotY (trigHandle, TRIG_PANEL_GRAPH, &data[arrayspot], sampsPerCycle,VAL_DOUBLE,
                   VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,ColorArray[i]);
            arrayspot = arrayspot + sampsPerCycle;
        }

        ArrayToFile("Signal.xls",data,VAL_DOUBLE,TotalsampsPerCycle,TotalsampsPerCycle,
                    VAL_GROUPS_TOGETHER,VAL_GROUPS_AS_ROWS,VAL_CONST_WIDTH,10,VAL_ASCII,
                    VAL_TRUNCATE);
        DAQmxErrChk (DAQmxWriteAnalogF64(taskHandle,sampsPerCycle,0,10.0,DAQmx_Val_GroupByChannel,
                     data,&written,NULL));
        DAQmxErrChk (DAQmxStartTask(taskHandle));
        SetCtrlAttribute(trigHandle, TRIG_PANEL_START,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_MINVAL,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_MAXVAL,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_WFMTYPE,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_WFMTYPE_2,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_WFMTYPE_3,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_WFMTYPE_4,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_FREQUENCY,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_FREQUENCY_2,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_FREQUENCY_3,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_FREQUENCY_4,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_FREQUENCY_5,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_AMPLITUDE,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_AMPLITUDE_2,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_AMPLITUDE_3,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_AMPLITUDE_4,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_AMPLITUDE_5,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_DUTYCYCLE,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_DUTYCYCLE_2,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_DUTYCYCLE_3,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_DUTYCYCLE_4,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_DELAY,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_RATE,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_X_CORRECT,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_Y_CORRECT,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_LASER_PULSE,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_CW,ATTR_DIMMED,1);

        ProcessDrawEvents();
        gRunning = 1;

        while( gRunning ) {
            Sleep(100);
            DAQmxErrChk (DAQmxIsTaskDone(taskHandle,&done));
            if( done )
                gRunning = 0;
            else
                ProcessSystemEvents();
        }
    }

Error:
    if( DAQmxFailed(error) )
        DAQmxGetExtendedErrorInfo(errBuff,2048);
    if( taskHandle!=0 ) {
        DAQmxErrChk (DAQmxStopTask(taskHandle));
        DAQmxErrChk (DAQmxClearTask(taskHandle));
        SetCtrlAttribute(trigHandle, TRIG_PANEL_START,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_MINVAL,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_MAXVAL,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_WFMTYPE,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_WFMTYPE_2,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_WFMTYPE_3,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_WFMTYPE_4,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_FREQUENCY,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_FREQUENCY_2,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_FREQUENCY_3,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_FREQUENCY_4,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_FREQUENCY_5,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_AMPLITUDE,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_AMPLITUDE_2,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_AMPLITUDE_3,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_AMPLITUDE_4,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_AMPLITUDE_5,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_DUTYCYCLE,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_DUTYCYCLE_2,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_DUTYCYCLE_3,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_DUTYCYCLE_4,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_DELAY,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_RATE,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_X_CORRECT,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_Y_CORRECT,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_LASER_PULSE,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_CW,ATTR_DIMMED,0);
        DeleteGraphPlot(trigHandle, TRIG_PANEL_GRAPH,-1,VAL_IMMEDIATE_DRAW);
        return error;
    }
    if( data )
        free(data);
    if( DAQmxFailed(error) )
        MessagePopup("DAQmx Error",errBuff);
    return 0;
}

int CVICALLBACK X_Correct(int panel, int control, int event, void *callbackData, int eventData1,
                          int eventData2)
{
    int         error=0;
    TaskHandle  taskHandle=0;
    char        chan[256]="Dev1/ao0";
    double      min,max;
    float64     data;
    char        errBuff[2048]={'\0'};

    if( event==EVENT_COMMIT ){
        GetCtrlVal(trigHandle,TRIG_PANEL_MINVAL,&min);
        GetCtrlVal(trigHandle,TRIG_PANEL_MAXVAL,&max);
        GetCtrlVal(trigHandle,TRIG_PANEL_X_CORRECT,&data);

        /*********************************************/
        // DAQmx Configure Code
        /*********************************************/
        SetWaitCursor(1);
        DAQmxErrChk (DAQmxCreateTask("",&taskHandle));
        DAQmxErrChk (DAQmxCreateAOVoltageChan(taskHandle,chan,"",min,max,DAQmx_Val_Volts,""));

        /*********************************************/
        // DAQmx Start Code
        /*********************************************/
        DAQmxErrChk (DAQmxStartTask(taskHandle));

        /*********************************************/
        // DAQmx Write Code
        /*********************************************/
        DAQmxErrChk (DAQmxWriteAnalogF64(taskHandle,1,1,10.0,DAQmx_Val_GroupByChannel,&data,NULL,
                     NULL));

    }

Error:
    SetWaitCursor(0);
    if( DAQmxFailed(error) )
        DAQmxGetExtendedErrorInfo(errBuff,2048);
    if( taskHandle!=0 ) {
        /*********************************************/
        // DAQmx Stop Code
        /*********************************************/
        DAQmxStopTask(taskHandle);
        DAQmxClearTask(taskHandle);
    }
    if( DAQmxFailed(error) )
        MessagePopup("DAQmx Error",errBuff);
    return 0;
}

int CVICALLBACK Y_Correct (int panel, int control, int event, void *callbackData, int eventData1,
                           int eventData2)
{
    int         error=0;
    TaskHandle  taskHandle=0;
    char        chan[256]="Dev1/ao1";
    double      min,max;
    float64     data;
    char        errBuff[2048]={'\0'};

    if( event==EVENT_COMMIT ){
        GetCtrlVal(trigHandle,TRIG_PANEL_MINVAL,&min);
        GetCtrlVal(trigHandle,TRIG_PANEL_MAXVAL,&max);
        GetCtrlVal(trigHandle,TRIG_PANEL_Y_CORRECT,&data);

        /*********************************************/
        // DAQmx Configure Code
        /*********************************************/
        SetWaitCursor(1);
        DAQmxErrChk (DAQmxCreateTask("",&taskHandle));
        DAQmxErrChk (DAQmxCreateAOVoltageChan(taskHandle,chan,"",min,max,DAQmx_Val_Volts,""));

        /*********************************************/
        // DAQmx Start Code
        /*********************************************/
        DAQmxErrChk (DAQmxStartTask(taskHandle));

        /*********************************************/
        // DAQmx Write Code
        /*********************************************/
        DAQmxErrChk (DAQmxWriteAnalogF64(taskHandle,1,1,10.0,DAQmx_Val_GroupByChannel,&data,NULL,
                     NULL));

    }

Error:
    SetWaitCursor(0);
    if( DAQmxFailed(error) )
        DAQmxGetExtendedErrorInfo(errBuff,2048);
    if( taskHandle!=0 ) {
        /*********************************************/
        // DAQmx Stop Code
        /*********************************************/
        DAQmxStopTask(taskHandle);
        DAQmxClearTask(taskHandle);
    }
    if( DAQmxFailed(error) )
        MessagePopup("DAQmx Error",errBuff);
    return 0;
}

int CVICALLBACK Laser_Pulse (int panel, int control, int event, void *callbackData, int eventData1,
                             int eventData2)
{

    int         outwfm[1];
    int         outcycles[1];
    double      outamp[1];
    double      outfreq[1];
    double      amplitude;
    int         typeofwaveform;
    int         frequency;
    int         error=0;
    TaskHandle  taskHandle=0;
    char        chan[256]="Dev1/ao3";
    double      min,max,rate;
    uInt32      sampsPerCycle, arrayspot = 0;
    float64     *data=NULL,*temp=NULL;
    char        errBuff[2048]={'\0'};
    bool32      done=0;
    double      phase=0.0;
    int         log,k;
    int32       written;

    if( event==EVENT_COMMIT ) {
        GetCtrlVal (trigHandle, TRIG_PANEL_AMPLITUDE_5, &amplitude);
        GetCtrlVal (trigHandle, TRIG_PANEL_FREQUENCY_5, &frequency);
        GetCtrlVal (trigHandle, TRIG_PANEL_MINVAL,&min);
        GetCtrlVal (trigHandle, TRIG_PANEL_MAXVAL,&max);
        GetCtrlVal (trigHandle, TRIG_PANEL_RATE,&rate);
        sampsPerCycle = rate;
        outcycles[0] = sampsPerCycle;
        outamp[0] = amplitude;
        outfreq[0] = frequency;


        SetCtrlAttribute(trigHandle, TRIG_PANEL_GRAPH,ATTR_XAXIS_GAIN,1.0/rate);
        log = log10(rate);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_GRAPH,ATTR_XPRECISION,log);

        DAQmxErrChk (DAQmxCreateTask("",&taskHandle));
        DAQmxErrChk (DAQmxCreateAOVoltageChan(taskHandle,chan,"",min,max,DAQmx_Val_Volts,NULL));
        DAQmxErrChk (DAQmxCfgSampClkTiming(taskHandle,"",rate,DAQmx_Val_Rising,
                     DAQmx_Val_FiniteSamps,1000));

        if( (data=malloc(sampsPerCycle*sizeof(float64)))==NULL ) {
            MessagePopup("Error","Not enough memory");
            goto Error;
        }

        if( (temp=malloc(sampsPerCycle*sizeof(float64)))==NULL ) {
            MessagePopup("Error","Not enough memory");
            goto Error;
        }

        SquareWave(sampsPerCycle,outamp[0],outfreq[0]/sampsPerCycle,&phase,50.00,temp);

        for(k=0;k<sampsPerCycle;k++){
            data[arrayspot+k] = temp[k];
        }

        arrayspot = arrayspot + sampsPerCycle;

        LinEv1D(data,sampsPerCycle,1.0,(max+min)/2.0,data);
        DeleteGraphPlot(trigHandle, TRIG_PANEL_GRAPH,-1,VAL_DELAYED_DRAW);
        arrayspot = 0;
        PlotY (trigHandle, TRIG_PANEL_GRAPH, &data[arrayspot], sampsPerCycle,VAL_DOUBLE,
               VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,ColorArray[0]);
        arrayspot = arrayspot + sampsPerCycle;

        DAQmxErrChk (DAQmxWriteAnalogF64(taskHandle,sampsPerCycle,0,10.0,DAQmx_Val_GroupByChannel,
                     data,&written,NULL));
        DAQmxErrChk (DAQmxStartTask(taskHandle));
        SetCtrlAttribute(trigHandle, TRIG_PANEL_START,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_LASER_PULSE,ATTR_DIMMED,1);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_CW,ATTR_DIMMED,1);
        ProcessDrawEvents();
        gRunning = 1;

        while( gRunning ) {
            Sleep(100);
            DAQmxErrChk (DAQmxIsTaskDone(taskHandle,&done));
            if( done )
                gRunning = 0;
            else
                ProcessSystemEvents();
        }
    }

Error:
    if( DAQmxFailed(error) )
        DAQmxGetExtendedErrorInfo(errBuff,2048);
    if( taskHandle!=0 ) {
        DAQmxErrChk (DAQmxStopTask(taskHandle));
        DAQmxErrChk (DAQmxClearTask(taskHandle));
        SetCtrlAttribute(trigHandle, TRIG_PANEL_START,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_LASER_PULSE,ATTR_DIMMED,0);
        SetCtrlAttribute(trigHandle, TRIG_PANEL_CW,ATTR_DIMMED,0);
        DeleteGraphPlot(trigHandle, TRIG_PANEL_GRAPH,-1,VAL_IMMEDIATE_DRAW);
```

```
        return error;
    }
    if( data )
        free(data);
    if( DAQmxFailed(error) )
        MessagePopup("DAQmx Error",errBuff);
    return 0;
}


int CVICALLBACK Cont_Wave (int panel, int control, int event, void *callbackData, int eventData1,
                           int eventData2)
{
    int         error=0, test;
    TaskHandle  taskHandle=0;
    char        chan[256]="Dev1/ao3";
    double      min,max;
    float64     data;
    char        errBuff[2048]={'\0'};

    if( event==EVENT_COMMIT ){
        GetCtrlVal(trigHandle,TRIG_PANEL_MINVAL,&min);
        GetCtrlVal(trigHandle,TRIG_PANEL_MAXVAL,&max);

        GetCtrlVal (trigHandle, TRIG_PANEL_CW, &test);

        if(test) {
            data = 5.0;
            SetCtrlAttribute(trigHandle, TRIG_PANEL_LASER_PULSE,ATTR_DIMMED,1);
//          SetCtrlAttribute(trigHandle, TRIG_PANEL_START,ATTR_DIMMED,1);
            /*********************************************/
            // DAQmx Configure Code
            /*********************************************/
            SetWaitCursor(1);
            DAQmxErrChk (DAQmxCreateTask("",&taskHandle));
            DAQmxErrChk (DAQmxCreateAOVoltageChan(taskHandle,chan,"",min,max,DAQmx_Val_Volts,""));

            /*********************************************/
            // DAQmx Start Code
            /*********************************************/
            DAQmxErrChk (DAQmxStartTask(taskHandle));

            /*********************************************/
            // DAQmx Write Code
            /*********************************************/
            DAQmxErrChk (DAQmxWriteAnalogF64(taskHandle,1,1,10.0,DAQmx_Val_GroupByChannel,&data,
                         NULL,NULL));
        }
        else {
            data = 0.0;
            SetCtrlAttribute(trigHandle, TRIG_PANEL_LASER_PULSE,ATTR_DIMMED,0);
//          SetCtrlAttribute(trigHandle, TRIG_PANEL_START,ATTR_DIMMED,0);
            /*********************************************/
            // DAQmx Configure Code
            /*********************************************/
            SetWaitCursor(1);
            DAQmxErrChk (DAQmxCreateTask("",&taskHandle));
            DAQmxErrChk (DAQmxCreateAOVoltageChan(taskHandle,chan,"",min,max,DAQmx_Val_Volts,""));

            /*********************************************/
            // DAQmx Start Code
            /*********************************************/
            DAQmxErrChk (DAQmxStartTask(taskHandle));

            /*********************************************/
            // DAQmx Write Code
            /*********************************************/
            DAQmxErrChk (DAQmxWriteAnalogF64(taskHandle,1,1,10.0,DAQmx_Val_GroupByChannel,&data,
                         NULL,NULL));
        }
    }

Error:
    SetWaitCursor(0);
```

```c
    if( DAQmxFailed(error) )
        DAQmxGetExtendedErrorInfo(errBuff,2048);
    if( taskHandle!=0 ) {
        /*********************************************/
        // DAQmx Stop Code
        /*********************************************/
        DAQmxStopTask(taskHandle);
        DAQmxClearTask(taskHandle);
    }
    if( DAQmxFailed(error) )
        MessagePopup("DAQmx Error",errBuff);
    return 0;
}



static void GenStepWave(int numElements, double amplitude, double frequency, double *phase,
                        double stepWave[])
{
    int     i=0;

    for(;i<numElements;++i) {
        double  phase_i=fmod(*phase+360.0*frequency*i,360.0);
        double  percentPeriod=phase_i/360.0;
        double  dat=amplitude*2.0*percentPeriod;

        dat=TruncateRealNumber(dat);
        stepWave[i] = percentPeriod<=0.5 ? dat : dat-2.0*amplitude;
    }
    *phase = fmod(*phase+frequency*360.0*numElements,360.0);
}

static void LoadDataFile(int numElements, int numGroups, double data[])
{
    int status;

    status = FileSelectPopup(proj_dir,"*.xls","*.xls","Load Datafile",VAL_LOAD_BUTTON,0,1,1,1,
                             file_name);
    FileToArray(file_name,data,VAL_DOUBLE,numElements,numGroups,VAL_GROUPS_TOGETHER,
                VAL_GROUPS_AS_ROWS,VAL_ASCII);
}

int CVICALLBACK StopCallback(int panel, int control, int event, void *callbackData,
                             int eventData1, int eventData2)
{
    if( event==EVENT_COMMIT )
        gRunning = 0;
        return 0;
}


int CVICALLBACK Text_Box (int panel, int control, int event, void *callbackData, int eventData1,
                          int eventData2)
{
    int     status,lineNumber=0, StartVal=0;
    char    message[100], currentCmd[1024];


    {
        GetCtrlAttribute(panelHandle, PANEL_DISPLAY_PORTS, ATTR_TEXT_SELECTION_START, &StartVal);
        sprintf(message,"\n");
    }
    return 0;
}
```