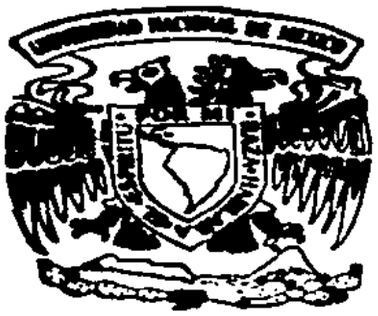
27
2ej



UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

BIRREFRINGENCIA BICOLOR DE ALTA RESOLUCION

T E S I S
Que para obtener el título de
F I S I C A
p r e s e n t a

MONICA HOCHSTEIN GLAZMAN

Director de Tesis: Dr. Enrique Geffroy Aguilar

México, D. F. 1997

UNIVERSIDAD NACIONAL
AVENIDA DE
MEXICO

M. en C. Virginia Abrín Batule
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
P r e s e n t e

Comunicamos a usted que hemos revisado el trabajo de Tesis: Birrefringencia Bicolor de Alta Resolución

realizado por Mónica Hochstein Glazman

con número de cuenta 9251764-2 **, pasante de la carrera de** Física

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis	
Propietario	DR. ENRIQUE GEFFROY AGUILAR
Propietario	DR. MAYO VILLAGRAN MUÑIZ
Propietario	FIS. ANDRES VALENTIN PORTA CONTRERAS
Suplente	DR. MARCIAL BONILLA MARIN
Suplente	M. EN C. FRANCISCO MORALES LEAL

Consejo Departamental de Física
DR. ROBERTO ALEJANDRO RUELAS MAYORGA

Birrefringencia Bicolor de Alta Resolución.

Mónica Hochstein Glazman

הָרִים נָזְלוּ מִפְּנֵי יְהוָה זֶה
סִינַי מִפְּנֵי יְהוָה אֱלֹהֵי יִשְׂרָאֵל׃

A mis padres por haberme apoyado en todo.

A mis hermanos aunque se hayan opuesto.

A Jaime por no dejarme caer aún cuando yo misma me tiraba.

A Juan Anonio por su paciencia y cariño.

A todos los que hicieron posible que este sueño se hiciera realidad.

Agradecimientos

En primer lugar quisiera agradecer al Dr. Enrique Geffroy Aguilar por haber compartido conmigo una visión distinta de la física, además de guiarme por el camino de la paciencia.

También deseo agradecer a mis sinodales Dr. Marcial Bonilla, M. en C. Francisco Morales, Fis. Andrés Porta y Dr. Mayo Villagrán, por haber revisado este trabajo.

Por último, agradezco a todos los que me han acompañado en este tiempo y han compartido conmigo la pasión del saber.

Índice

Sumario

En la industria, con frecuencia se requiere conocer y mejorar las propiedades reológicas de los materiales plásticos, con el propósito de alcanzar un uso de éstos más competitivo. Desde el punto de vista en ciencia de polímeros, la necesidad industrial planteada implica conocer y entender los mecanismos mediante los cuales los materiales poliméricos relajan sus esfuerzos después de haber sido deformados. Los esfuerzos son el resultado de los cambios en la microestructura del material, siendo estos últimos resultado de las deformaciones debidas al procesado del material.

Este proyecto se centra en el desarrollo de metodologías computacionales para técnicas ópticas capaces de caracterizar los cambios en la microestructura del material, que macroscópicamente pueden medirse como una anisotropía óptica que es inducida por las deformaciones. La técnica óptica desarrollada en el Laboratorio de Reología-Óptica se conoce como "birrefringencia bicolor inducida por flujos" y consiste en atravesar el líquido polimérico bajo estudio con dos haces de láser, de distinto color. Dadas las propiedades de polarización de los rayos de entrada y salida, es posible conocer la anisotropía del material. Esta técnica tiene grandes ventajas debido a que: (a) permite hacer mediciones rápidas comparadas con los tiempos de relajación del material o con las mediciones obtenidas mediante otras técnicas de tipo mecánico, (b) permite obtener la anisotropía y su ángulo de orientación casi simultáneamente, y (c) permite estudiar flujos no homogéneos bidimensionales, pues es una técnica capaz de hacer medidas locales.

La caracterización de materiales poliméricos requiere operar esta técnica con gran precisión, rapidez y sincronizando una diversidad de instrumentos, pues sólo así se pueden evaluar cuidadosamente los procesos de relajación. Es por ello que resulta indispensable realizar estos experimentos bajo el control de un procesador. Así, *el objetivo principal*

de este trabajo ha sido el desarrollo de los códigos computacionales para la ejecución de experimentos de relajación en fluidos poliméricos.

En particular, se han desarrollado, en forma detallada, los códigos que permiten llevar a cabo el control del experimento, la calibración de sensores luminosos, así como realizar la adquisición de datos. Los procedimientos de calibración y adquisición se hacen ahora con velocidades hasta 100 veces más rápidas y con precisiones en las medidas hasta 100 veces mejores que lo antes reportado. Para alcanzar estos logros, especial cuidado se ha tenido en reconocer las diferentes fuentes de errores que afectan los resultados experimentales, y el programa de ejecución de experimentos incluye los algoritmos que evaluán las contribuciones de errores y sistemáticamente maximizan la relación señal/ruido.

El programa desarrollado utiliza las técnicas de programación en tiempo real, así como estructuras de bases de datos que hacen uso extensivo de apuntadores (pointers) y colas (queues), optimizando los recursos computacionales y de los instrumentos periféricos disponibles.

Introducción

El uso de computadoras para la ejecución de experimentos o el control de instrumentos es cosa común hoy en día, en particular enfatizando dos tipos de aplicaciones. Por una parte, para experimentos que son muy complejos —como los de anisotropía óptica inducida por flujos— no sólo la necesidad es real sino que el uso de computadores es la única alternativa viable de trabajo productivo. Por otra parte, el uso de procesadores es ventajoso para reconocer estados anómalos que se pueden presentar de manera inesperada y que requieren la toma de decisiones con criterios posiblemente nunca antes considerados. En general, no se tienen estas ventajas en el control de experimentos con técnicas e instrumentos no programables. Sin embargo, para resolver los dos puntos anteriores, además del procesador, se requiere de un tipo de programación que no es común y que se denomina *programación en tiempo real.*

En este trabajo se presenta un desarrollo de la técnica de birrefringencia bicolor inducida por flujos mediante un procesador que realiza automáticamente las operaciones de control, supervisión y adquisición de datos. Dichas operaciones las realiza un "despachador" basado en técnicas de programación en tiempo real y cuyo objetivo es la medición, con gran precisión, de las anisotropías inducidas por flujos.

Para lo anterior, en el Capítulo 1 se plantean cuatro ideas básicas: la anisotropía que se produce en un material viscoelástico, las propiedades de la interacción luz-materia que permiten caracterizar la anisotropía, la técnica experimental que se utiliza para tales mediciones y las condiciones necesarias para la realización de los experimentos. El Capítulo comienza con una pequeña reseña histórica del desarrollo del conocimiento de la luz polarizada, además de explicar brevemente la física de materiales anisotrópicos. Luego, se da una descripción de la anisotropía óptica en materiales poliméricos en

términos del cálculo de Müller y los vectores de Stokes que permite representar la interacción de luz y materia. El siguiente paso tiene que ver con el arreglo experimental propuesto; se analiza matemática y técnicamente cuál es la complejidad del problema a resolver. Finalmente se plantean y se describen las necesidades experimentales para la generación de los flujos, las necesidades de muestreo de lecturas que se pueden calibrar rápidamente y de manera conveniente, además de que garanticen lecturas libres de errores sistemáticos.

El Capítulo 2 tiene como objetivo describir detalladamente los elementos de programación necesarios para la comunicación con los instrumentos, el control de los mismos y la ejecución de una variedad de flujos de interés. En particular se describe el manejo de las mediciones de las señales luminosas, así como el por qué de la precisión requerida y del tiempo de muestreo necesario de lograr. Luego se describe el código diseñado para ejecutar el experimento de birrefringencia, y finalmente se presentan y analizan las ideas básicas de los códigos que involucran programación en tiempo real y la utilización de despachadores.

En los Apéndices A y B se presentan los códigos para la adquisición, el control y la supervisión de instrumentos en forma detallada, pues existen diferentes formas de estructurar las ideas básicas de las técnicas de programación en tiempo real. Sin embargo, son pocos los sistemas operativos y los lenguajes de programación que tienen desarrollados los implementos necesarios para estas tareas. Por tanto, el desarrollo del despachador de eventos en tiempo real aquí dado es una alternativa específica al problema planteado y a los recursos computacionales disponibles; existe poca información publicada hasta ahora sobre códigos complejos para despachadores en el laboratorio. Aún así, las metodologías desarrolladas son suficientemente generales y con características adecuadas para una gran diversidad de experimentos o actividades de control en tiempo real,

utilizando para ello recursos que con frecuencia existen en un laboratorio de investigación.

Por último, el Capítulo 3 presenta las características que se lograron en este trabajo y plantea también algunas actividades por realizar que requieren un grado de complejidad aún mayor y que sin embargo son perfectamente factibles de lograrse.

Capítulo 1.

Anisotropía Óptica Inducida por Flujos.

Para el estudio de la dinámica no lineal de polímeros desde una base experimental es deseable que los dispositivos utilizados cumplan con varios requisitos. En particular, se busca una técnica experimental que permita inducir los cambios en la microestructura de las macromoléculas, pues el estudio de la dinámica de cualquier sistema se puede hacer cuando se perturba su estado de equilibrio y se observan los mecanismos de relajamiento, así como sus tiempos característicos.

Dentro de los métodos experimentales comúnmente utilizados para evaluar los cambios microestructurales de los polímeros se encuentran: (a) las técnicas mecánicas, mediante la medición de esfuerzos resultado de aplicar deformaciones a la muestra bajo estudio, donde se utilizan los denominados reómetros; o, (b) aunque menos utilizadas, las técnicas que miden la anisotropía óptica del medio de interés cuando éste es sometido a una deformación. Las técnicas mecánicas son actualmente las más utilizadas en la industria ya que dan información de interés directo en el procesado de plásticos, etc. En general, los métodos ópticos resultan más ventajosos tanto técnica como científicamente, aunque son más complejos de establecer en el laboratorio. La ventaja principal del uso de técnicas ópticas para las actividades del Laboratorio de Reología Óptica, donde se

realiza este trabajo, es su gran sensibilidad para detectar cambios de la microestructura que se estudia, así como la rapidez con que es posible evaluar tales cambios. Por ello, este Capítulo analiza la técnica que se utiliza en dicho Laboratorio para la medición de anisotropías ópticas cuando éstas son generadas por flujos en líquidos poliméricos.

Sin embargo, no es suficiente para los estudios de *dinámica no lineal* poder medir las anisotropías inducidas en el polímero. Resulta crítico que el fluido se pueda someter a cambios estructurales que sean característicos de estados fuera del equilibrio; en particular, que no sean simplemente resultado de *perturbaciones* (por deformaciones infinitesimales) del estado de reposo.

Es entonces indispensable acoplar las mediciones de anisotropía óptica con un dispositivo capaz de generar *deformaciones finitas* en líquidos poliméricos. En este Laboratorio se ha propuesto un molino de dos rodillos como aquel capaz de generar un *flujo fuerte* cuyas deformaciones finitas son apropiadas para los estudios de la dinámica no lineal[1]. Este molino consiste de dos rodillos iguales, cuyos ejes son paralelos y giran en la misma dirección y con la misma velocidad. En estos flujos, la región entre los rodillos tiene dos características muy ventajosas para estos estudios. Primero, existe un punto de estancamiento en el cual un elemento de volumen permanece estacionario por tiempos muy largos, de modo que observando las macromoléculas de esta región se puede estudiar su dinámica. Segundo, en este punto de estancamiento, si bien la velocidad es cero y las partículas permanecen ahí, el gradiente del campo de velocidad en las cercanías al punto de estancamiento es del tipo de *flujo fuerte*. Para un flujo fuerte, dos elementos del fluido, cercanos entre sí, tienden a alejarse exponencialmente como función del tiempo. En los *flujos débiles*, tales como el flujo cortante simple, la separación de estos elementos del

[1] Para mayor información sobre este tema referirse a la tesis de Ingeniero Mecánico de M. A. Reyes Huesca: *"Estudio y Solución Analítica de un Flujo Fuerte Generado por un Molino de Dos Rodillos"*, E.N.E.P. Aragón, U.N.A.M. (1997), y en la tesis doctoral de E. Geffroy: *"Birefringence of Polymer Solutions in Time Dependent Flows"*, CALTECH (1990).

fluido crece con el tiempo de manera lineal únicamente. Los flujos fuertes son capaces de inducir en un líquido polimérico grandes deformaciones, que conllevan grandes cambios en su microestructura. Por lo anterior, en este trabajo se toma como premisa la bondad de acoplar un flujo fuerte con las mediciones de anisotropía óptica.

Por tanto, en este Capítulo brevemente se presenta (a) el desarrollo histórico de los conceptos básicos de la física de la interacción entre la radiación electromagnética y la materia (Sección 1.1); (b) se presentan las ideas de cómo los esfuerzos macroscópicos pueden generar anisotropías ópticas en un líquido polimérico (Sección 1.2), y cómo estas anisotropías están relacionadas con los cambios estructurales microscópicos; (c) en la Sección 1.3 se presenta detalladamente la técnica óptica a utilizar, denominada birrefringencia bicolor inducida por flujos (conocida como "Two-Color Flow Birefringence" en la literatura científica) [1], enfatizando especialmente las necesidades experimentales, tanto en la adquisición de la información, como en el análisis de los datos; finalmente (d) en la Sección 1.4, se plantean otras necesidades básicas para el montaje del experimento que, aunado a las dificultades de la adquisición de datos experimentales y su procesamiento, definen *el ambiente computacional que este trabajo de tesis resuelve para el estudio de la dinámica no lineal de polímeros.*

Sección 1.1. El Desarrollo Histórico de la luz Polarizada y de la Física de Materiales Birrefringentes.

Las investigaciones sobre las propiedades de la luz polarizada comenzaron en 1669 con el descubrimiento, por Erasmus Bartholinus, del fenómeno de la doble refracción en cristales de calcita [2, 3, 4]. Este fenómeno fue interpretado poco después, por Christian Huygens en 1690, quien propuso que en los cristales de calcita se propaga, además de una onda esférica primaria, una onda elipsoidal secundaria. A lo largo de sus investigaciones, Huygens también hizo un descubrimiento fundamental en el campo de la polarización:

uno de los dos rayos que sale de la refracción de un cristal de calcita se extingue si pasa por un segundo cristal de este mismo material que está orientado perpendicularmente a la dirección del rayo y al eje principal del primer cristal [2].

La polarización de la luz por reflexión fue descubierta por Etienne Louis Malus [2, 3, 4] un día que se encontraba en el Palacio de Luxemburgo de París, y observó que la reflexión del sol sobre un cristal de calcita, de uno de los vitrales del lugar, formaba dos imágenes, las cuales desaparecían alternadamente mientras el cristal giraba. Malus nunca intentó dar una interpretación a este fenómeno ya que creía que las teorías eran incapaces de proveer una explicación verdadera de las cosas.

El progreso en la ciencia durante el siglo XIX encausó a las matemáticas en el análisis y representación de los fenómenos físicos. En particular, en 1818 Agustín Jean Fresnel [3] demostró que el trabajo de las envolventes hecho por Huygens y de la interferencia de haces desarrollado por Young eran suficientes para explicar la propagación rectilínea de la luz y el fenómeno de difracción, todo ello como consecuencia de un campo oscilatorio. Fresnel calculó la difracción generada por pequeñas rendijas y pantallas, que fue corroborada experimentalmente por Dominique François Aragó. Junto con Aragó, Fresnel investigó la interferencia de la luz polarizada y ambos encontraron que dos rayos polarizados en ángulo recto, uno con respecto al otro, nunca interfieren. Fue Fresnel quien dio la primera explicación sobre la dispersión de la luz, tomando en cuenta la estructura molecular de la materia. Estas observaciones fueron, sin duda, las que permitieron establecer el carácter ondulatorio de la luz y ayudaron a Fresnel a deducir las leyes que hoy en día llevan su nombre y que hablan de la intensidad y de la polarización de los rayos de luz generados por reflexión y refracción.

Así y por muchos años más, quedó establecido que la luz es un fenómeno ondulatorio, en el que su intensidad, color y polarización son algunos de los parámetros necesarios

para caracterizarla.

Las investigaciones de la electricidad y el magnetismo se desarrollaron de manera independiente a la óptica. James Clerk Maxwell [2] agrupó todas las experiencias anteriores en un sistema de ecuaciones que describen la electricidad y el magnetismo. De estas ecuaciones dedujo la existencia de *ondas electromagnéticas* que se propagan a cierta velocidad, que posteriormente se demostró éstas viajan a igual velocidad que la luz. Esto permitió a Maxwell concluir que las ondas luminosas son, en realidad, ondas electromagnéticas [3].

Como resultado de la teoría electromagnética de la luz, la búsqueda del Éter (el medio elástico requerido para soportar la propagación de un fenómeno ondulatorio) terminó. También facilitó el estudio de las interacciones de luz-materia, pues sentó las bases para la descripción de la radiación como un fenómeno que además de color, intensidad y polarización, requiere ahora describir también la coherencia del haz para estar completamente definida.

Especialmente de relevancia para el análisis presentado a continuación en la teoría electromagnética, la polarización es el resultado de la selección de un campo electromagnético en el que tanto el campo eléctrico como el magnético, y la dirección de propagación son ortogonales entre sí. Asimismo, las amplitudes de los campos electromagnéticos están asociadas sólo indirectamente a las observables físicas, pues la intensidad de un haz de luz es simplemente proporcional al cuadrado de la intensidad promedio de campo electromagnético.

George Gabriel Stokes publicó, en 1852, dos artículos en óptica. El primero de ellos *"On the Composition and Resolution of Streams of Polarized Light from Different Sources"* [5], ahora considerado como uno de los mejores artículos de la óptica clásica, provee la formulación matemática para describir cualquier estado de luz, *polarizada y no*

polarizada, en términos de *intensidades* (observables físicas) en lugar de las amplitudes de los campos eléctrico y magnético. Cerca del final de su artículo, Stokes menciona que con cuatro parámetros, ahora conocidos como parámetros de polarización de Stokes, se puede caracterizar cualquier estado de luz polarizada. Este artículo sobre polarización estuvo prácticamente olvidado por cerca de un siglo hasta que su importancia fue reconocida por Subrahmanya Chandrasekhar cuando en 1942 lo descubrió, y utilizó los parámetros de Stokes para describir los efectos de la luz polarizada en las ecuaciones de transferencia radiativa [2]. Son estos parámetros la base para la descripción de la interacción luz-materia que a continuación se presenta.

Las propiedades de polarización de la luz se pueden expresar por los cuatro parámetros de Stokes, y que son

1. I: Intensidad total del haz,
2. $Q = I_0 - I_{\pi/2}$: Diferencia entre las intensidades de las componentes horizontal (OX) y vertical (OY) de luz polarizada linealmente,
3. $U = I_{\pi/4} - I_{-\pi/4}$: Diferencia entre las intensidades de las componentes a $\pi/4$ y $-\pi/4$ con respecto al eje OX de luz polarizada linealmente.
4. $V = I_{lcd} - I_{lci}$: Diferencia entre las intensidades de luz polarizada circular derecha e izquierda.

Además y con base en la teoría electromagnética de Maxwell, la anisotropía óptica de un medio se puede expresar mediante un índice de refracción tensorial, con elementos complejos, el cual tiene diferentes valores principales. La parte real de los elementos tensoriales corresponde a una medida de la velocidad de propagación de la onda electromagnética en el medio a lo largo de los ejes principales, y cuando los valores principales son diferentes, se dice que el material es ópticamente birrefringente. La parte imaginaria de cada elemento es una medida de la atenuación de estas ondas por efectos

de dispersión y absorción mientras viajan por el medio, y si sus valores principales son diferentes, se dice que el material es dicroico. Utilizando luz polarizada, es posible medir *la diferencia* de los valores principales de las partes real o imaginaria del tensor índice de refracción de un medio anistrópico, en el plano del vector eléctrico de la onda que se propaga.

El descubrimiento de la *birrefringencia inducida por esfuerzos* lo hizo Brewster en 1816. Brewster encontró que un plato de vidrio sujeto a una tensión simple adquiere las propiedades de un cristal uniaxial tal como la calcita. Las primeras observaciones publicadas acerca de la birrefringencia inducida por flujos las hizo Mach en 1873. El primer intento real de tomar medidas cuantitativas y de desarrollar una teoría en este campo las hizo Kundt en 1874, quien asumió que los esfuerzos producen birrefringencia en los líquidos de la misma manera que lo hacen en los sólidos, *i.e.*, los ejes ópticos deben coincidir con los ejes de los esfuerzos [6].

Cuando un líquido con estructura (como un fluido macromolecular) fluye y a la vez se ve a través de un sistema de polarizadores cruzados, la luz viaja perpendicularmente tanto al gradiente de velocidad como a las líneas de campo del flujo (ver Figura 1.1). Si los polarizadores están orientados en direcciones diferentes a la del vector de polarización de la luz que viaja en el fluido y a 90° entre si, entonces se observa que la luz pasa el arreglo óptico. Cuando el líquido está en reposo (sin flujo), el medio aparece obscuro. Es decir, bajo la influencia de fuerzas hidrodinámicas el fluido se vuelve ópticamente biaxial, *i.e.*, tiene tres ejes principales con tres índices de refracción principales. Uno de estos ejes coincide con la dirección del haz de luz mientras que los otros dos quedan en el plano de observación. Así, el medio aparece obscuro, o luminoso, dependiendo de si la orientación de los polarizadores cruzados coincide, o no, con la orientación de estos dos ejes principales.

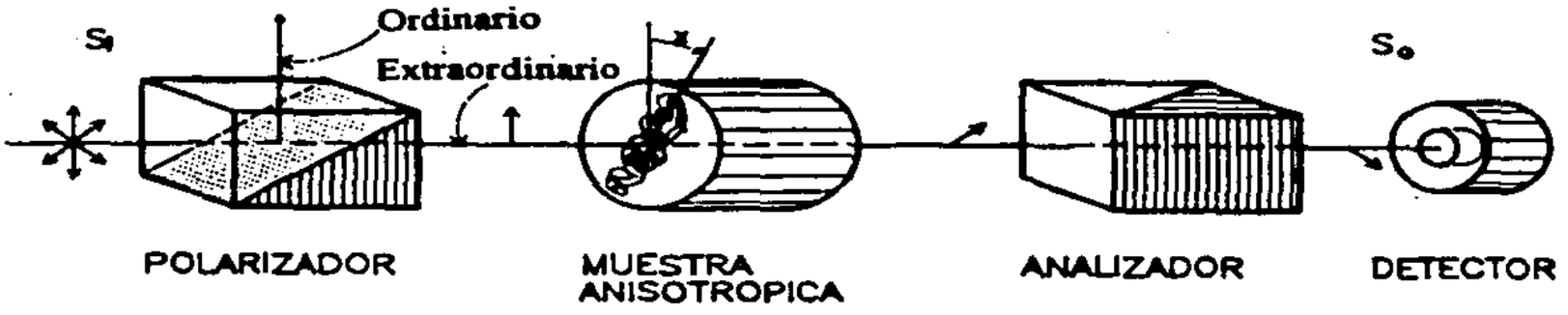


Figura 1.1. Esquema de un sistema de polarizadores cruzados. La flechas que aparecen sobre el eje óptico muestran la dirección de polarización del haz de luz, mientras que χ representa el ángulo de orientación de la birrefringencia.

El cálculo matricial propuesto por Müller [7] en 1948 permite representar la interacción de la luz polarizada con un medio anisotrópico. De manera equivalente al cálculo matricial de Jones [8, 9], el cálculo de Müller expresa dicha interacción como la transformación lineal que la anisotropía del medio M produce sobre el vector de Stokes que representa la luz incidente S_o. Así, el vector de Stokes para el haz de luz resultante queda como

$$S_{Salida} = M \; S_o \tag{1.1}$$

La diferencia principal con el cálculo de Jones es que en (1.1) todos los elementos representan observables físicas, *i.e.*, intensidades luminosas. El cálculo de Müller tiene, sin embargo, una ventaja sobre el de Jones, ya que permite la descripción de luz *parcialmente* polarizada.

Sección 1.2. La Anisotropía Óptica en Medios Poliméricos.

Ahora, aun cuando es fácil proponer una relación entre las deformaciones y las anisotropías ópticas en el caso de un sistema sólido deformado, lo que aquí interesa es poder relacionar las anisotropías ópticas de un material *líquido* —constituido de moléculas lineales de gran longitud (alto peso molecular)— con su dinámica microscópica. Así, dos son los puntos importantes de considerar: (1) ¿cómo se generan anisotropías macroscópicas en un líquido que en principio es homogéneo? y (2) ¿cómo estas anisotropías están relacionadas con cambios en su microestructura?

Es bien sabido que la mayoría de las moléculas son ópticamente anisotrópicas, es decir, su estructura es tal que el índice de refracción, a través del eje principal de la molécula, es diferente al índice de refracción a lo largo del eje perpendicular a éste. Cuando los monómeros que constituyen las diferentes moléculas están distribuidos completamente al azar, entonces el material se comporta como un medio isotrópico. Lo anterior es como resultado del estado de equilibrio final, que se caracteriza por un valor máximo de la entropía y que corresponde a un arreglo microestructural de máximo desorden. Sin embargo, cuando los monómeros que conforman el polímero están alineados, uno tras otro, entonces la molécula está perfectamente estirada y su entropía es mínima. Por lo contrario, cuando el sistema tiene máxima entropía, la conformación de cada cadena macromolecular es consistente con el mayor número de configuraciones posibles para todos los monómeros, lo que corresponde a visualizar cada polímero como una bola de estambre, en el que cada segmento del hilo tiene igual probabilidad de estar orientado en cualquier dirección.

Así, como consecuencia de una deformación en las macromoléculas, se logra una orientación preferencial de los elementos de la cadena, reduciéndose el número de

configuraciones posibles y reduciéndose, a la vez, la entropía de la cadena. Es por ello, que las deformaciones macroscópicas pueden inducir cambios microscópicos en el arreglo molecular y que corresponden a una anisotropía conformacional, dependiente de la deformación, y ésta se desvanece si los esfuerzos que causan la deformación cesan de aplicarse.

Ahora, dadas las propiedades de anisotropía óptica de los elementos de la cadena polimérica, un grado de orientación preferencial de los segmentos igualmente implica un grado de polarizabilidad neto de toda la cadena, y que macroscópicamente se observa como una diferencia del índice de refracción en dos direcciones perpendiculares. Estas direcciones coinciden con las direcciones principales de la orientación preferencial de los segmentos. Esta orientación se puede inducir por la aplicación de un campo eléctrico, de un campo magnético, de ondas acústicas o de flujos [1, 10]

En principio, se debe esperar que cualquier material muestre tanto el fenómeno de dicroísmo como el de birrefringencia; esto es, debe suponerse que el índice de refracción complejo tenga partes real e imaginaria diferentes de cero. Sin embargo en la práctica, las soluciones poliméricas son altamente birrefringentes y con sólo una contribución pequeña de dicroísmo. Experimentalmente hay que tener cuidado, ya que el dicroísmo es un efecto de absorción y en consecuencia debe existir una región, o regiones, del espectro electromagnético en donde la onda propagada se atenúa, y entonces el dicroísmo es igual de importante que la contribución de la birrefringencia. Por lo tanto, el análisis de datos experimentales debe tomar en cuenta la posibilidad de que ambos fenómenos físicos se presenten simultáneamente.

Los efectos microscópicos que generan la anisotropía inducida por flujos tienen dos contribuciones. Una de éstas, llamada birrefringencia de forma, está relacionada con la orientación de los segmentos poliméricos; la otra, llamada birrefringencia intrínseca, se

relaciona con la anisotropía de la función de correlación de la densidad de los segmentos de las cadenas[11, 12]. Para los estudios aquí propuestos, el origen de la birrefringencia que se observa es esencialmente birrefringencia intrínseca, que se presenta en soluciones concentradas de polímeros y domina la anisotropía observable.

Sección 1.3. La Técnica de Birrefringencia Bicolor.

Existe una gran variedad de técnicas experimentales que se pueden desarrollar para evaluar la dependencia de los cambios de la conformación polimérica con las características del flujo aplicado a dicho polímero. Como se mencionó anteriormente, las técnicas utilizadas más frecuentemente involucran mediciones de esfuerzos en instrumentos mecánicos, y cuyo objetivo es conocer las propiedades reológicas del material. Sin embargo, los parámetros experimentales de medidas reológicas que usualmente se reportan corresponden a promedios macroscópicos evaluados en la *frontera* del flujo (esto es, sobre las paredes que definen la celda de flujos utilizada) [1, 6, 13]. Son medidas adecuadas para deformaciones infinitesimales, pues para deformaciones finitas, no se puede considerar siempre válido que el valor de un parámetro dentro del flujo sea el mismo que su valor en la frontera.

Los métodos ópticos son una alternativa a las medidas mecánicas que, además resultan de gran utilidad cuando es necesario evaluar una respuesta *rápida* del sistema, a la vez de que proveen una indicación del comportamiento del material que es *local* en la escala del flujo [10]. Por ello, la técnica utilizada se conoce como *birrefringencia inducida por flujos*. Las técnicas ópticas son muy ventajosas, en particular, cuando los cambios conformacionales en la microestructura polimérica son inducidos por un flujo no-homogéneo, como es el caso de aquellos generados por el molino de dos rodillos. Además, las técnicas ópticas no son invasivas, pues permiten evaluar la anisotropía inducida sin que el campo del flujo sea perturbado sustancialmente.

Para los estudios de dinámica no lineal que se plantean para el Laboratorio de Reología-Óptica, se requiere una técnica capaz de: (a) medir tanto el grado de anisotropía como la orientación de los ejes principales del índice de refracción *en forma simultánea*, además de (b) poder realizar ambas medidas con gran rapidez y para un elemento de volumen pequeño dentro del flujo. Los requerimientos anteriores se deben a que la anisotropía varía en el tiempo tanto en su intensidad como en la orientación de ésta.

Hay dos técnicas, accesibles en la literatura, que cumplen con las exigencias anteriores [14, 15]. Éstas son la birrefringencia bicolor inducida por flujos (TCFB por sus siglas en inglés) [1, 16, 17] y las diferentes variantes de un elipsómetro de nulos automatizado [1, 6, 10]. Ambas técnicas son igualmente rápidas y accesibles. Sin embargo, la de birrefringencia bicolor, aunque necesita un arreglo óptico más complejo, requiere de elementos ópticos más sencillos. El inconveniente principal de los elipsómetros de nulos automatizados es que requieren de mover la óptica, ya sea el analizador o el retardador de 1/4 de onda, lo cual conlleva a alcanzar una menor precisión en la definición del camino óptico, característica que es crítica para los experimentos aquí propuestos.

La idea básica de la técnica de birrefringencia bicolor inducida por flujos consiste en evaluar "instantáneamente" la birrefringencia y su ángulo de orientación, tomando dos medidas simultáneas, utilizando para ello dos rayos de luz de diferente color, en este caso azul y verde, y un sistema doble de elipsómetros de nulos. El uso de dos haces de distinto color permite realizar las dos medidas indispensables para conocer tanto la orientación como la birrefringencia.

Además y puesto que el flujo es no-homogéneo, el experimento requiere que las medidas sean sobre el mismo elemento de volumen todo el tiempo. Por lo tanto, las características del arreglo óptico utilizado en el experimento que definen las propiedades ópticas de los dos rayos de luz mientras pasan por la muestra, deben ser, en la

medida de lo posible, idénticas. Además, se necesita que la sección transversal del elemento de fluido perpendicular al eje óptico sea pequeña, comparada con la escala de variaciones de las propiedades hidrodinámicas del campo de flujo, para que las medidas sean representativas de las propiedades del fluido cuando éste se somete a una historia de flujo bien determinada.

En el arreglo experimental que aquí se analiza, cada elipsómetro de nulos está compuesto por un polarizador, un medio birrefringente y un analizador, por lo que el vector de Stokes a la salida del arreglo, para ese color, queda como

$$S_o = P(\alpha_A)B(\beta,\delta)P(\alpha_P)S_i, \tag{1.2}$$

donde S_o es el vector de Stokes del haz de salida, S_i es el vector de Stokes del haz incidente, y $P(\alpha_i)$ y $B(\beta,\delta)$ son las matrices de Müller que representan a los polarizadores y al medio birrefringente, respectivamente. La matriz que representa a los polarizadores $P(\alpha)$ y que depende de la orientación azimutal α con respecto al eje horizontal está dada por

$$P(\alpha) = \begin{pmatrix} 1 & cos2\alpha & sin2\alpha & 0 \\ cos2\alpha & cos^2 2\alpha & sin2\alpha cos2\alpha & 0 \\ sin2\alpha & sin2\alpha cos2\alpha & sin^2 2\alpha & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{1.3}$$

mientras que la matriz que representa al medio birrefringente $B(\beta,\delta)$ para una retardancia δ y una orientación β relativa al eje OX (línea horizontal) es

$$B(\beta,\delta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos^2 2\beta + sin^2 2\beta cos\delta & cos2\beta sin2\beta(1-cos\delta) & sin2\beta sin\delta \\ 0 & cos2\beta sin2\beta(1-cos\delta) & sin^2 2\beta + cos^2 2\beta cos\delta & -cos2\beta sin\delta \\ 0 & -sin2\beta sin\delta & cos2\beta sin\delta & cos\delta \end{pmatrix}. \tag{1.4}$$

Así, considerando que el haz de entrada tiene polarización vertical, los polarizadores se alinean a $\pm\pi/8$ de la vertical y sus respectivos analizadores están rotados $\pi/2$. La diferencia de $\pi/4$ en las direcciones de polarización de ambos colores simplifica la forma algebraica de las expresiones matemáticas resultante como se ve más adelante.

Para el elipsómetro del haz azul los parámetros de Stokes de entrada son $S_i = (E_{y,A}{}^2, -E_{y,A}{}^2, 0, 0)$ y el cálculo de matrices de Müller queda como $S_o = P(-\pi/8)B(\beta,\delta)P(3\pi/8)S_i$. Así, la intensidad de la luz azul que alcanza el detector colocado después del analizador queda como

$$\hat{I_A} = \frac{E_{y,A}^2}{2sin^2\alpha_P}[1 - 2cos2\beta sin2\beta(1 - cos\delta)] \tag{1.5}$$

Experimentalmente, es conveniente medir los ángulos del medio anisotrópico con respecto a las orientaciones principales de los polarizadores mismos, por lo que es mejor redefinir a $\beta = \alpha_P - \chi$. Así se tiene que la Ecuación 1.5 queda como

$$\hat{I_A} = E_{y,A}^2 sin^2(\delta_A/2)sin(2\chi). \tag{1.6}$$

Para el haz verde, la orientación de su polarizador está girada $\pi/4$ con respecto a la dirección del polarizador azul; sea entonces a $5\pi/8$. Si la orientación de referencia es el polarizador azul, entonces la intensidad de luz que recibe el detector verde está dada por

$$\hat{I_V} = E_{y,V}^2 sin^2(\delta_V/2)cos^2(2\chi). \tag{1.7}$$

Las Ecuaciones 1.6 y 1.7 dan la información necesaria para calcular la orientación del índice de refracción del medio χ, y su retardancia δ. Experimentalmente, resulta conveniente realizar dos medidas "complementarias", con una separación de $\pi/4$ para las polarizaciones, pues de esta manera es posible hacer aproximaciones a (1.6) y (1.7) que facilitan el cálculo de χ y δ, como se muesta más adelante.

Igualmente conveniente resulta la normalización de las medidas de intensidad que se utilizan en las Ecuaciones 1.6 y 1.7, principalmente por dos razones. La primera busca que ambas ecuaciones muestren la simetría del problema, y no deben depender de la irradiancia absoluta que llega a los detectores. Esto es, dada una intensidad I_o en S_o, lo relevante para la física del problema es la fracción de esta intensidad que alcanza al

detector. Por ello las intensidades de luz, después de los analizadores, se normalizan respecto a la intensidad del rayo correspondiente, como se muestra enseguida. La segunda razón, por la que la normalización de intensidades es también importante, tiene que ver con las fluctuaciones en la intensidad del haz, así como aquellas fluctuaciones parásitas debidas a partículas de polvo que se encuentran en la solución polimérica como se discute más adelante.

Una manera sencilla de normalizar las intensidades que aparecen en las Ecuaciones 1.6 y 1.7 se logra colocando dos detectores más que muestrean los rayos *antes* de su paso por la muestra. De esta manera, los cuatro valores determinados experimentalmente se utilizan para definir dos intensidades válidas para el análisis de la anisotropía dadas por

$$\begin{aligned} i_A &= \frac{\hat{I}_A}{E^2_{y,A}} = \frac{\hat{I}}{I_{max,A}} = sin^2(\delta_{A/2})sin^2(2\chi), \\ i_V &= \frac{\hat{I}_V}{E^2_{y,V}} = \frac{\hat{I}}{I_{max,V}} = sin^2(\delta_{V/2})cos^2(2\chi). \end{aligned} \tag{1.8}$$

Con las ecuaciones anteriores, es posible obtener una solución aproximada para la birrefringencia y su orientación utilizando el método de perturbaciones. En este caso, tomando nuevamente al rayo azul como el color de referencia, la retardancia se puede expresar en términos de pequeñas perturbaciones, ϵ, considerando

$$\epsilon = \frac{\lambda_V - \lambda_A}{\lambda_V} < 0.05, \tag{1.9}$$

$$\delta_A = \frac{2\pi d}{\lambda_A}\Delta n' = \frac{2\pi d}{\lambda}\Delta n' = \delta, \tag{1.10}$$

$$\delta_V = \frac{2\pi d}{\lambda_V}\Delta n' = \frac{2\pi d}{\lambda}\Delta n'(1-\epsilon) = (1-\epsilon)\delta, \tag{1.11}$$

$$\delta(\epsilon) = \delta_0 + \delta_1\epsilon + \delta_2\epsilon^2 + \mathcal{O}(\epsilon^3), \tag{1.12}$$

donde $\Delta n'$ es la birrefringencia de la muestra, d es su grosor y λ_l es la longitud de onda de cada color. Ahora, resolviendo se tiene que

$$i_{TOT} = i_A + i_V =$$
$$sin^2(\delta_0 + \delta_1\epsilon + \delta_2\epsilon^2 + \mathcal{O}(\epsilon^3))sin^2(2\chi) + \tag{1.13}$$
$$sin^2[(1-\epsilon)(\delta_0 + \delta_1\epsilon + \delta_2\epsilon^2 + \mathcal{O}(\epsilon^3))]cos^2(2\chi).$$

La Ecuación 1.13 se puede reducir suponiendo $\beta << 1$ y

$$sin(\alpha \pm \beta) = sin\alpha\, cos\beta \pm cos\alpha\, sin\beta;$$

entonces,

$$sin^2(\alpha \pm \beta) \simeq sin^2\alpha \pm \beta sin(2\alpha) + \beta^2(1 - 2sin^2\alpha). \tag{1.14}$$

Ahora, agrupando términos semejantes se tiene que

$$\delta_1 = \delta_0\left(1 + \frac{\epsilon i_V}{i_{TOT}}\right), \tag{1.15}$$

y

$$\delta_2 = \delta_0\left(1 + \frac{\epsilon i_V}{i_{TOT}}\right) + \frac{1}{2i_{TOT}}\left[\frac{\delta_0^2(2i_{TOT}-1)i_V i_A}{i_{TOT}^{5/2}(1-i_{TOT})^{1/2}} + 2\frac{\delta_0 i_V}{i_{TOT}} - 1\right]\epsilon^2. \tag{1.16}$$

Para aproximaciones de primer orden, el sistema de Ecuaciones 1.8 se puede resolver quedando la birrefringencia $\Delta n'$ y el ángulo de orientación χ como

$$\Delta n' = \frac{\lambda_A}{\pi d}\left[1 + \frac{i_V\epsilon}{i_{TOT}}\right]sin^{-1}(i_{tot})^{1/2}, \tag{1.17}$$

$$\chi = tan^{-1}\left(\frac{i_A}{i_V}\frac{sin^2(\pi d\Delta n'(1-\epsilon)/\lambda_A)}{sin^2(\pi d\Delta n'/\lambda_A)}\right)^{1/2}. \tag{1.18}$$

Con la ecuación $\Delta n' = f(i_A + i_V)$, y la ecuación $\chi = f(i_A/i_V)$ es posible caracterizar la anisotropía de un fluido polimérico. Son estas ecuaciones principales que definen las bondades y defectos del arreglo óptico a utilizar.

El dispositivo experimental utilizado en el Laboratorio se muestra en la Figura 1.2. La mejor opción para este arreglo experimental es un láser de Argón ionizado, emitiendo en

las longitudes de onda de 4880 Å y 5145 Å, las cuales sufren atenuaciones despreciables cuando atraviesan polímeros orgánicos tales como poliestireno, polietileno, etc.[2] El láser utilizado tiene grandes ventajas[3], entre ellas el emitir ambos haces simultáneamente lo que permite asegurar que las condiciones ópticas iniciales de éstos son idénticas. El interferómetro de Fabry-Perot permite ajustar las intensidades relativas de los dos colores y a la vez revisar la estabilidad del láser. Los caminos ópticos que recorren los dos haces son tales que permiten observar, para ambos colores, elementos de fluido idénticos, en forma y posición y permiten ajustar, de manera muy precisa, las orientaciones de polarización de cada haz. Esta orientación se obtiene separando los haces mediante filtros para cada una de las longitudes de onda deseadas, luego haciéndolos pasar por un polarizador de calcita y finalmente haciéndolos nuevamente recorrer el camino óptico principal. Ambos polarizadores, P_A y P_V, tienen una orientación de la polarización con una resolución aproximada de 1.0 minuto de arco. Para el haz verde es importante, cuando incide sobre el filtro azul, hacer un ángulo lo más pequeño posible con respecto al eje óptico principal (*i.e.*, el eje del haz azul) para así minimizar los cambios en la orientación de su polarización. Una vez juntos, los rayos se hacen incidir sobre una celda de flujos que contiene al sistema de dos rodillos, y donde se encuentra la muestra polimérica. Luego se tiene la sección de detección de intensidades luminosas, la cual requiere de una nueva separación de los rayos de distinto color, para pasar enseguida por el correspondiente

---

[2] Estas son las líneas de emisión de mayor intensidad y estabilidad. El láser de Argón ionizado es de la marca Spectra-Physics, modelo BeamLok 2060,. El experimento necesita de dos haces de distinta longitud de onda, pero la diferencia entre estas longitudes debe ser pequeña. En este arreglo se pueden utilizar también láseres de Nd:YAG y Nd:YLF que emiten en longitudes de 1064 Å y 1072 Å respectivamente. El problema que se presenta en el estudio de muestras poliméricas es la alta absorción en el infrarrojo de estos materiales que conlleva la presencia del fenómeno de dicroísmo.

[3] Este láser mantiene su potencia con fluctuaciones menoreas a ±0.3% y una estabilidad en la posición del haz menor que 0.5 $\mu m/^{\circ}C$[18].

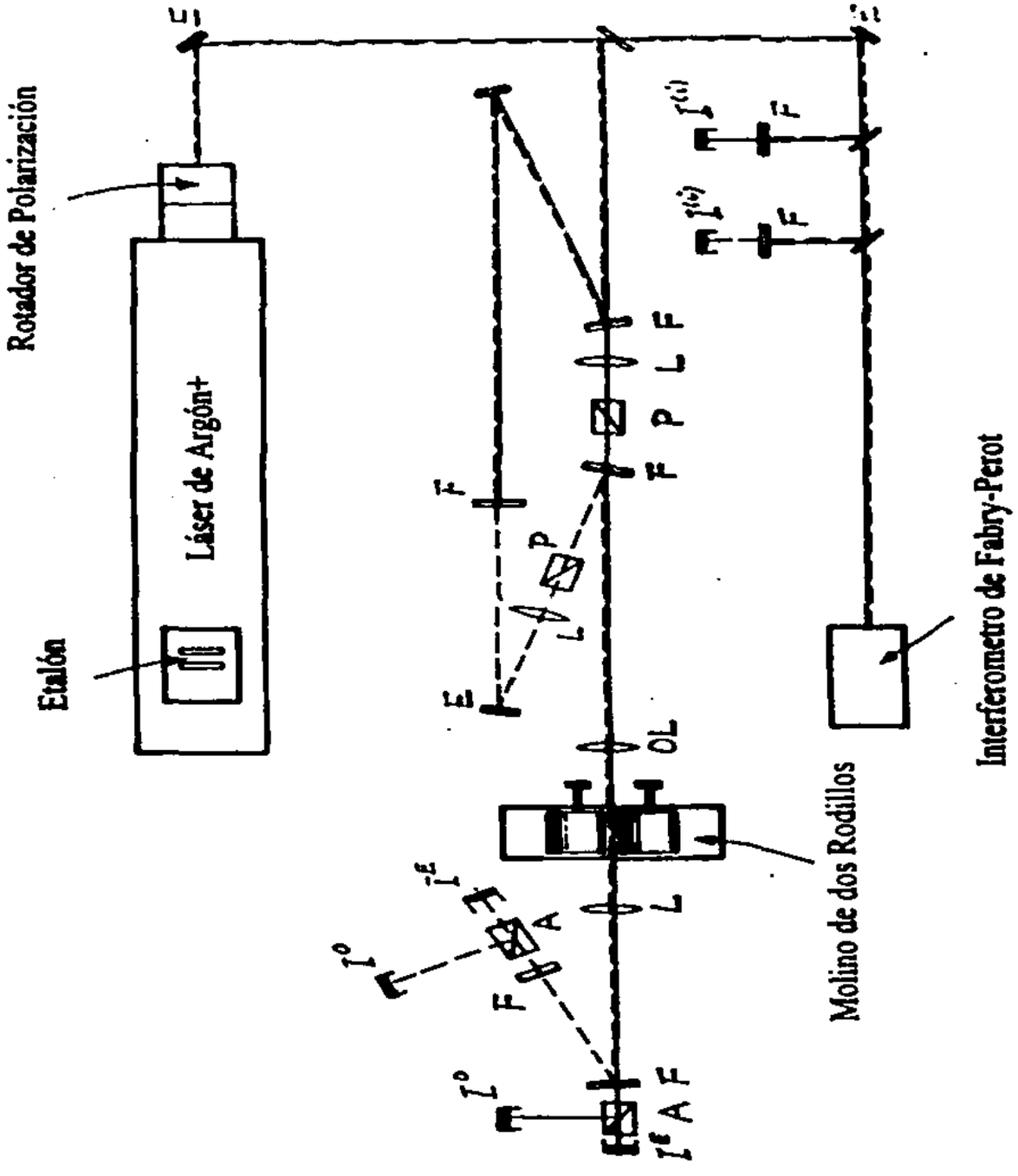


Figura 1.2. Arreglo experimental para el experimento de birrefringencia bicolor inducida por flujos. La línea continua representa al haz azul mientras que la línea punteada representa al haz verde. Aquí E representa espejos, F filtros, L lentes P polarizadores y A analizadores.

analizador que divide el haz en sus componentes ordinario (polarización normal al plano de dispersión) y extraordinario (polarización en el plano de dispersión). Siguen los cuatro fotodetectores que miden las intensidades del rayo ordinario y extraordinario para cada uno de los colores. En este arreglo óptico, es importante minimizar el número de elementos ópticos entre los polarizadores y los analizadores para así tener razones de extinción[4] máximas y por consiguiente la mayor sensibilidad posible. Esto es resultado de que la mayoría de los componentes ópticos son residualmente anisotrópicos debido a su proceso de fabricación.

Finalmente, se tiene la sección de amplificación electrónica de las señales luminosas y de medición de éstas mediante un adquisitor de datos HP3852A que se conecta con la computadora HP382. Dicha computadora controla la operación de los amplificadores programables, realiza la eliminación de ruido computacional y de las anisotropías parásitas, hace todo el análisis estadístico de los datos, y lleva además el mando de los diferentes instrumentos. A continuación se define el ambiente experimental que aunado a las características de medición y procesamiento de las señales luminosas conforman el problema experimental a resolver mediante técnicas computacionales.

Sección 1.4. Las Necesidades para el Experimento de TCFB.

Uno de los métodos más simples para la caracterización de líquidos poliméricos es la determinación de su espectro de tiempos de relajación cuando el fluido está sujeto a una deformación macroscópica impuesta por un campo de flujo. Las escalas de tiempo observadas dependen, en gran medida, del *tipo de deformación*, así como de la *historia de deformación*. Dado que las historias de deformación dependen del flujo aplicado, el

---

[4] La razón de extinción se define como la intensidad luminosa que pasa el analizador, normalizada por la luz incidente. Las razones de extinción menores corresponden a arreglos de polarizadores-analizadores más perfectos. En nuestro caso es deseable que la calidad del material de los polarizadores permita alcanzar razones de extinción hasta de 10^{-7}.

cálculo de las escalas de tiempos de relajación, cuando la microestructura polimérica ha estado sujeta a grandes deformaciones, es, en la mayoría de los casos, difícil. Sin embargo los tiempos característicos medidos se ven afectados por una complejidad resultado de la naturaleza del estudio, además de las complicaciones inherentes del arreglo experimental. En la Sección 2.4 se amplía las ideas relativas a las complicaciones inherentes del arreglo. La principal complejidad en determinar los espectros de relajamiento del material se deben al acoplamiento de los efectos de la respuesta del material y los efectos del flujo. Este acoplamiento es observable y hasta ahora ha sido imposible de comprender y de modelar analíticamente. Aún más, este acoplamiento impone requisitos sobre el arreglo experimental pues de otra manera resulta difícil determinar con precisión los espectros de relajación del material polimérico sujeto a deformaciones fuertes.

El acoplamiento se debe a que los grandes cambios conformacionales del líquido con estructura generalmente *conllevan efectos sobre el flujo* que alteran sus características. Esto es, en el estudio de la dinámica no-lineal se debe suponer que el flujo modifica sustancialmente la estructura del líquido y éste, a su vez, altera las características más relevantes del flujo. Por ello, para generar datos útiles para la caracterización de materiales poliméricos es necesario medir, simultáneamente y de manera independiente, las propiedades del campo de flujo y de la conformación polimérica.

En el laboratorio es posible el uso simultáneo de la técnica de birrefringencia bicolor inducida por flujos y la dispersión homodinea de luz. La primera permite conocer los cambios de la microestructura, mientras que la segunda evalúa los parámetros relevantes del flujo en forma independiente. Lo anterior es posible para flujos bidimensionales como los que se generan en un molino de dos rodillos. En la región localizada entre los rodillos se tiene un punto de estancamiento y un flujo de tipo elongacional. En este punto, la velocidad del campo es nula y una molécula permanece ahí indefinidamente,

sujeta a una rapidez de deformación que depende de la velocidad de los rodillos. Es, esta región, donde se puede estudiar la dinámica no-lineal de los líquidos macromoleculares, además de que se cuenta con una solución analítica para el flujo en dicha región en el caso de flujos lentos [19].

Para los experimentos considerados en el Laboratorio de Reología Óptica del Instituto de Investigaciones en Materiales, el molino de dos rodillos debe de cumplir con tres requisitos directamente relacionados con este trabajo de tesis y que se relacionan con las complicaciones inherentes del arreglo experimental antes mencionado: (a) debe permitir el control y monitoreo de la temperatura del fluido, ya que, en la mayoría de las soluciones poliméricas, pequeñas variaciones en su temperatura alteran la viscosidad y modifican las características del espectro de relajación; (b) debe permitir mediciones ópticas a lo largo de los rodillos, en la región central entre los mismos, para realizar las medidas de los cambios de conformación; y (c) debe permitir una amplia variedad de posibles historias de deformación de modo de facilitar el estudio de diversos aspectos de la respuesta no-lineal de líquidos poliméricos.

El control de la temperatura de la muestra debe ser rápido, versátil e inteligente, pues para una gran diversidad de flujos las fuentes de energía, tales como la disipación viscosa, varían sustancialmente en potencia (además de otros parámetros) y son función del tiempo. La respuesta del sistema de control de temperatura depende asimismo de los coeficientes de transporte y de las propiedades termodinámicas del líquido. Por tanto, este control conviene desarrollarlo suponiendo que se cuenta con una computadora que realiza los ajustes correspondientes en un ambiente de control flexible y con una respuesta a los estímulos casi instantánea.

La región central del molino de dos rodillos es accesible a las técnicas ópticas colocando un par de ventanas sobre las tapas de la celda de flujos, como se muestra

en la Figura 1.3. Sin embargo, el procedimiento para fijar tales ventanas conlleva una desventaja para las metodologías de medición de las anisotropías ópticas. Esta desventaja aparece como resultado de los esfuerzos parásitos que existen en tales ventanas o que aparecen como resultado de sujetarlas a la celda de flujos. Es por ello que, para medidas de precisión de las anisotropías ópticas de los líquidos poliméricos, se requiere aplicar un procedimiento de calibración, previamente a la ejecución del experimento, y cuyo *único próposito* es tener la capacidad para corregir la existencia de anisotropías parásitas del arreglo experimental o corregir los datos por otras fuentes de error.

Como se menciona en la Sección 1.3, para el experimento de birrefringencia bicolor de flujos es conveniente la normalización de las intensidades que alcanzan los detectores con base en las intensidades de los haces *antes* de que incidan sobre el líquido polimérico. El objetivo es normalizar las intensidades de acuerdo a (1.8), de manera que la expresión (1.17) esté correctamente definida. En la práctica, la normalización propuesta ayuda a resolver varios problemas relativos a variaciones por cambios ligeros en la alineación del arreglo óptico, o las fluctuaciones de la intensidad luminosa debidas a las no idealidades de la cavidad resonante del láser. Estas variaciones se pueden compensar fácilmente con cuatro detectores, dos detectores midiendo las intensidades de los dos colores *antes* de la celda y dos mas *después* del molino de dos rodillos.

Sin embargo, en la solución polimérica existen generalmente partículas pequeñas de polvos que producen un efecto dispersivo difícil de discriminar. La manera propuesta por Geffroy [1] para resolver el problema anterior requiere utilizar seis detectores en total (3 para cada color) y cuyo propósito es conocer con precisión la intensidad total *antes y después* de la muestra. De esta manera, lo deseable para el experimento de birrefringencia de flujos implica un experimento que cuenta con 6 detectores cuyas características son semejantes entre sí y, de acuerdo con el arreglo de la Figura 1.2.

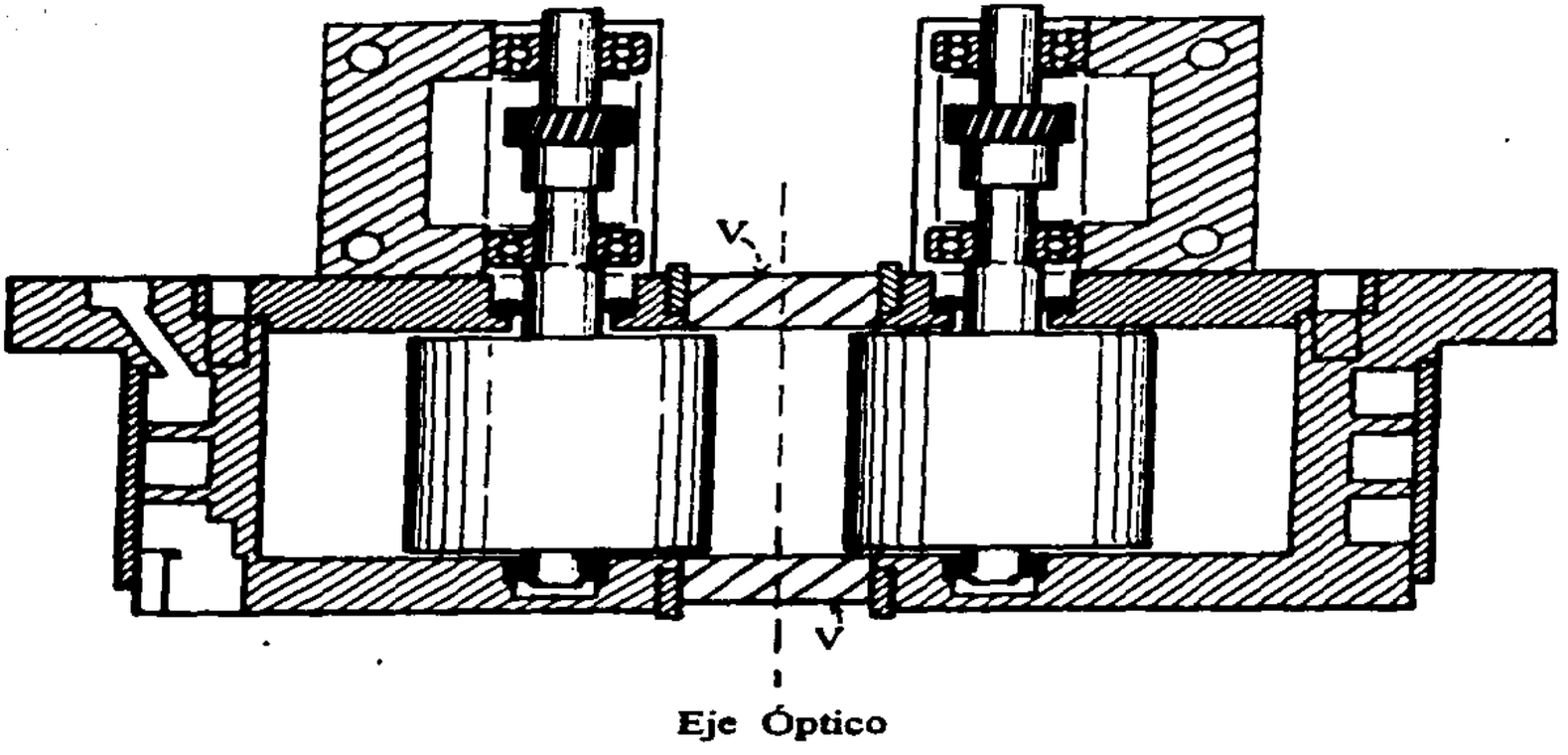


Figura 1.3. Esquema del molino de dos rodillos. En esta figura se muestra la disposición de las ventanas (V) en la celda de flujos, y la dirección de propagación del haz: eje óptico.

Esta configuración, si bien fue propuesta en 1990 [1], a la fecha, éste es el trabajo que desarrolla operativamente el arreglo. Aún más, los experimentos reportados hasta 1996 [20] sólo permiten mediciones de una de las partes del índice de refracción complejo: ya sea la parte real—birrefringencia— o la parte imaginaria —dicrofsmo.

Asimismo, tales experimentos muestran velocidades máximas de muestreo de datos de aproximadamente 0.05 segundos. Ahora, con el nuevo experimento se busca aumentar las velocidades de muestreo hasta 20 veces más, considerando a la vez los seis detectores. Con los seis detectores y el procedimiento de calibración adecuado, es posible hacer mediciones en forma simultánea de las dos partes del índice de refracción.

En el siguiente Capítulo se propone un procedimiento de calibración adecuado a estas nuevas necesidades. Por consiguiente, el desarrollo de estas metodologías experimentales incluyendo el procedimiento de calibración requiere de una computadora poderosa para que realice estas actividades de manera cuasi-automática. Los amplificadores de las señales luminosas correspondientes a los seis detectores ya se tienen —su desarrollo corresponde a los trabajos de tesis de García Muñoz [21] y Corona Pastrana [22], y forman parte del adquisitor HP3852A.

Para la generación de las diversas historias de deformación que se aplican al líquido polimérico, la manera más eficiente de efectuar los desplazamientos angulares de los rodillos es mediante un controlador inteligente para un motor de pasos. El requerimiento anterior se debe esencialmente a la infinidad de posibles historias de deformación que son de interés y por la necesidad de efectuar en el laboratorio varias de ellas. Esto se debe a que la respuesta del material es fundamentalmente no lineal y las diferentes historias de deformación pueden dar información complementaria e indispensable para estos estudios. Sin embargo, la adquisición de datos para una historia de deformación es por si misma compleja, y hacerlo para una serie numerosa de diferentes experimentos implica que éstos se deben realizar bajo el control de una computadora pues, es tal vez, la única manera de garantizar su éxito.

Así, para ejecutar el experimento propuesto anteriormente es necesario: (a) poder tener un control preciso de los motores de paso que generan el movimiento de los rodillos del molino; (b) tener el mando del láser; (c) manejar y controlar la temperatura del líquido estudiado; y (d) controlar el sistema de detección de las señales del experimento de birrefringencia bicolor realizando la adquisición de datos a gran velocidad, para hacer posteriormente un cálculo de la birrefringencia, mientras, el motor de pasos genera en sincronía el flujo deseado. La manera más eficiente de hacer todo lo anterior es con el

uso de una computadora que centraliza el mando de toda la instrumentación y analiza los datos generados. Para hacer esto, es necesario que el mando de la computadora se efectúe en *tiempo real*. La parte de *la programación relacionada con la adquisición y manejo de los datos es la parte primordial de este trabajo* y se describe con mayor detalle en el siguiente Capítulo.

Capítulo 2.

Diseño Experimental.

El uso de computadoras en la adquisición de datos experimentales se considera hoy como una de las actividades en las ciencias con la que se puede lograr un avance significativo [23], tanto en la precisión de las mediciones, como en el grado de complejidad de los experimentos posibles de realizar. Además de las ventajas anteriores, el investigador puede también aumentar considerablemente su productividad, pues es posible automatizar un sinnúmero de acciones que permiten generar bases de datos experimentales muy completas, y que de otra manera resultan prácticamente imposibles de recabar.

Para ello, dos técnicas deben conjuntarse. La primera involucra los dispositivos (hardware[5]) que permiten desarrollar con un procesador muchas de las acciones de supervisión, control y sincronización requeridas para la ejecución del experimento deseado y que, de otra manera, el experimentalista está obligado a realizar. Entre éstos se deben considerar relojes de eventos, marcadores de tiempo, controladores de ambientes gráficos y numéricos, interfaces de comunicación, semáforos para señales y variables, manejadores de servicios, etc. Muchos de estos dispositivos existen en

[5] Tanto las palabras "hardware" como "software" vienen del inglés y se refieren a la parte de dispositivos y a la de programación respectivamente.

cualquier computadora, sin embargo, el acceso o uso de ellos está limitado para las operaciones *internas* del procesador, buscando así que sus beneficios sean transparentes para el usuario de la máquina.

La segunda técnica involucra los programas (software) mediante los cuales, y con la ayuda de procesadores, es posible realizar operaciones complejas que de otra manera los dispositivos (o entre los dispositivos) no son capaces de realizar. En particular, las técnicas de *programación en tiempo real* se deben considerar como indispensables para la ejecución de experimentos, pues permiten la interacción de los diferentes instrumentos en forma ordenada y con un propósito único, el cual está definido por el experimento a realizar.

La técnica de programación en tiempo real tiene poco en común con las técnicas de programación científica, las cuales se utilizan primordialmente para la simulación de fenómenos físicos. Sin embargo, la programación en tiempo real es muy útil y poderosa en el ambiente del laboratorio, y con certeza es indispensable cuando la complejidad del experimento rebasa la capacidad física del operario para poder realizar su ejecución. Por ello, en las próximas secciones se detalla el uso de los recursos tanto de hardware como de software que se han desarrollado e integrado para la ejecución del experimento. La Sección 2.1 detalla los instrumentos requeridos para las funciones importantes del experimento, para posteriormente describir el control de los mismos, utilizando para ello los dispositivos de control y el programa en tiempo real.

Sección 2.1. El Arreglo Experimental para la Caracterización de Polímeros.

Para el desarrollo del experimento para estudios de la dinámica no lineal de polímeros, en el Capítulo anterior se menciona la necesidad de que: (a) se controle la temperatura del líquido bajo estudio con precisión, (b) se realice un conjunto de diferentes historias

de deformación con una celda de flujos del tipo de molino de dos rodillos cuyo control del movimiento se lleve a cabo mediante un controlador de motores de pasos que sea programable; (c) se cuente con la electrónica para la conversión y amplificación de las señales luminosas provenientes del arreglo para la determinación de la birrefringencia bicolor inducida por flujos; y (d) se tenga bajo el control de la computadora la fuente de luz azul y verde. La Figura 2.1 muestra como bloques los instrumentos y dispositivos que el experimento requiere.

Todas las acciones de control de la instrumentación y la adquisición de datos se hacen a través de una Unidad de Control y Adquisición de Datos HP 3852A y una estación de trabajo HP 382, comunicadas entre sí vía una interface[6] IEEE-488 (HP-IB). El control y corrección de la temperatura de la muestra se hace utilizando un vóltmetro de 5 1/2 dígitos HP 44701A (5 1/2 Digits Voltmeter) y un convertidor digital analógico HP 44727 (4 Channel Voltage / Current DAC). El manejo de los rodillos que generan el flujo en la muestra polimérica se hace con un controlador de motores de paso HP 44714A (3 Channel Stepper Motor Controller). Finalmente, la sección de adquisición de datos está compuesta por (a) una tarjeta de amplificación HP 44736A (Breadboard), donde se instaló todo el sistema de preamplificación y amplificación de las señales luminosas; (b) un multiplexor HP 44713 (24 Channel High Speed FET Multiplexer) que sirve para alimentar las señales provenientes de los amplificadores y que corresponden a los distintos detectores; y por último, (c) un vóltmetro de alta velocidad HP 44704A (16 bit High Speed Voltmeter), el cual digitaliza las señales (voltajes) de los seis detectores en forma secuencial. Además, de la intrface de comunicación entre el Adquisitor y

---

[6] La palabra "interface" se toma del inglés y se refiere al lugar donde sistemas independientes se encuentran e interactúan entre sí. En nuestro caso particular, una interface es la sección de los circuitos que permite la comunicación y coordinación entre los distintos instrumentos utilizando protocolos mecánicos, eléctricos y de comunicación estandar.

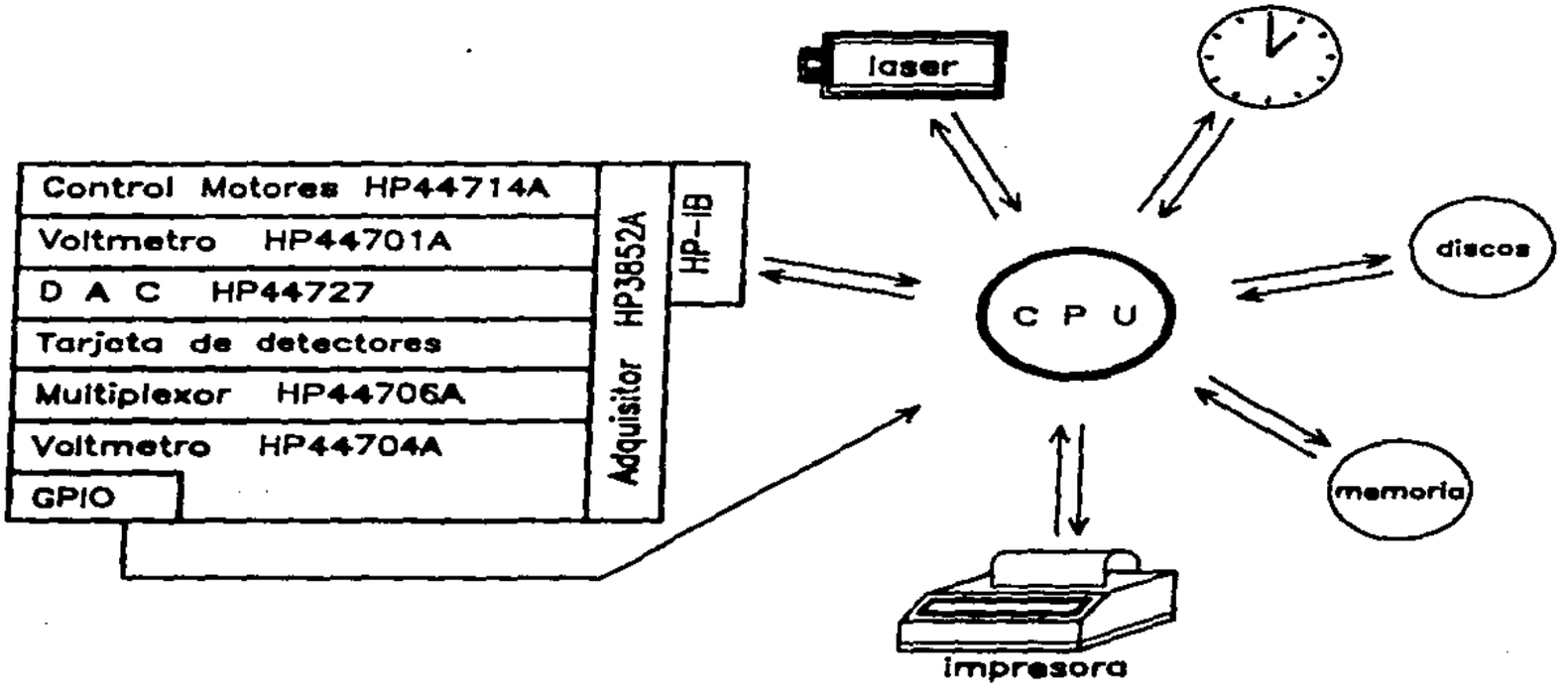


Figura 2.1. Diagrama de bloques de los instrumentos y dispositivos que el experimento de birrefringencia bicolor necesita. Las flechas indican la dirección del flujo de información entre las partes.

el controlador usando el protocolo HP-IB, existe una conexión entre el vóltmetro y el controlador vía una interface GPIO[7], a través de la cual se hace la transferencia de datos en forma expedita, para hacer posteriormente el análisis de los datos en la computadora.

La unidad del láser también es parte de los instrumentos bajo control de la computadora, pues resulta indispensable revisar la estabilidad de la fuente de luz azul y verde y también es necesario monitorear los problemas que en ella puedan presentarse.

---

[7] GPIO (General Purpose Input Output) se refiere a la interface HP 98622A, que permite un intercambio bidireccional de datos de 16 bits entre la computadora y el instrumento periférico y con una velocidad de 750 KBytes/segundo.

Esta unidad es accesible mediante la interface HP-IB. El láser es de Argón ionizado de emisión continua cuyas líneas más intensas son: 488 nm (Azul) y 514.5 nm (Verde). La potencia nominal del sistema es de 9 W, siendo algunas de sus características más importantes: (a) la estabilidad de alineamiento, pues cuenta con un sistema de retroalimentación para ajustar dinámicamente la posición del haz y, (b) su estabilidad en la potencia de emisión, también controlada por un circuito de retroalimentación.

Se requieren además varios instrumentos que si bien son igualmente necesarios para el experimento, la mayoría no son programables y son difíciles de supervisar mediante la computadora; por lo tanto no se muestran en la Figura 2.1. Entre ellos pueden mencionarse un baño de temperatura constante, bombas de aire filtrado, *etc.* y asisten en el control de la temperatura.

Sección 2.2. La Programación Necesaria para el Arreglo Experimental.

Los experimentos propuestos se pueden realizar sólo si se cuenta con los dispositivos encargados de la sincronización de los diferentes eventos que se realizan con los instrumentos antes mencionados. Esto es, resulta imprecindible que el experimento cuente, por una parte, con los programas complementarios para el acceso a los diferentes instrumentos, así como la ejecución del experimento —la programación se compone de 16 módulos mas el programa principal MHGMAIN. Por otra parte, se requiere (a) un procesador para el manejo y cómputo de grandes cantidades de información, (b) un reloj programable con el que se hacen las operaciones de sincronización de los eventos, así como (c) el software que permita construir el despachador de eventos y cuya única función es otorgar la capacidad del procesador a aquellos eventos u operaciones que por razones de prioridad se deben ejecutar inmediatamente.

La Programación para la Instrumentación.

Los 16 módulos atienden las necesidades específicas de las interfaces, los instrumentos y los posibles experimentos. Los trece módulos están compuestos cada uno de múltiples subrutinas. Estos módulos son:

IODECLARATIONS, GENERAL_1, GENERAL_2, GENERAL_3, DGL_LIB, AsmMod, UserProc, FileOper, Monitor, Laser, AD_Converter, Correlator, Gral_Exp, Bir_library, Dich_library y DataEvaluation.

Los primeros 5 módulos son parte del sistema operativo Pascal HP. Para mayor información referente a éstos es conveniente consultar a los manuales del lenguaje [24].

El módulo AsmMod está relacionado con los algoritmos necesarios para manejar los pixeles de la pantalla gráfica y está escrito en lenguaje ensamblador para así agilizar su ejecución. UserProc es una colección de utilerías cuyo propósito es facilitar la programación en general; así sólo se programa una vez las secciones de código que se utilizan repetitivamente en otras subrutinas. FileOper contiene toda las estructuras de datos relacionadas con las variables que son específicas de este experimento, además de contener las subrutinas que permiten un uso eficiente y compacto de las variables y de las secciones de memoria. Estas estructuras requieren de un buen manejo de apuntadores (pointers) y de colas (queues). Monitor es el módulo que permite manejar la graficación de los datos recabados en tiempo real, o del análisis posterior de información. Laser, SMI, AD_Converter y Correlator contienen las subrutinas que permiten el manejo de tales instrumentos; SMI contiene las subrutinas para el controlador del motor de pasos que genera el flujo. Gral_Exp contiene los procedimientos necesarios para el reporte del *status quo* y los resultados de los distintos experimentos. Bir_library y Dich_library contienen los procedimientos necesarios para la ejecución del experimento de birrefringencia de flujos. Finalmente, DataEvaluation permite la evaluación y el análisis de los datos recabados, tanto de forma gráfica, como numérica.

El Programa Principal: MHGMAIN.

MHGMAIN se apoya en los 16 módulos anteriores para realizar todas las tareas asociadas con el dispositivo experimental y los posibles experimentos a realizar. En este trabajo se han desarrollado los códigos necesarios para realizar un experimento de birrefringencia, con los instrumentos propuestos en la Sección 2.1, y están escritos en lenguaje Pascal (Hewlett-Packard Pascal, Version 3.22). Este programa se ha diseñado pensando facilitar al usuario: (a) el acceso y control desde la computadora de todos los instrumentos utilizados en el experimento; (b) la ejecución de experimentos de birrefringencia para determinar la conformación de moléculas poliméricas cuando están sujetas a flujos generados por el molino de dos rodillos, y (c) la posibilidad de leer y manipular los datos experimentales obtenidos tiempo atrás, con fines de su análisis posterior. Específicamente, el trabajo ahora reportado se centra alrededor de un experimento de birrefringencia. En particular, en la integración de los instrumentos necesarios: el láser, el controlador de motores de pasos, el sistema de amplificación y lectura de señales luminosas provenientes de los detectores, y la ejecución de un flujo fuerte estacionario, todos ellos en sincronía y supervisados mediante un despachador.

La subrutina para la ejecución de experimentos que miden la birrefringencia de medios poliméricos, está incluida en el programa principal, MHGMAIN. Esta subrutina es esencialmente una interface de usuario, esto es, permite el intercambio de información entre el usuario y la computadora, y viceversa, quedando la ejecución del experimento mismo contenido en subrutinas que son parte de ésta y que se describen detalladamente en la Sección 2.5. A continuación se muestra la estructura de la subrutina principal, escrita en pseudocódigo, y enfatizando las acciones en bloque a realizar, tanto para el montado del experimento como su ejecución y resguardo de la información generada. La estructura de la subrutina para el experimento de birrefringencia es a grandes rasgos la siguiente:

```
EXECUTION OF BIREFRINGENCE EXPERIMENTS;
(1)     Initialization of variables and pointers;
        Turn equipment on;
        Read tag of last experiment (old_label);
        Modify parameters?   (y/n);
(5)          IF yes
                Display & Modify (old_label,new_label);
             ELSE new_label is old_label;
       REPEAT                          (..  Until all data has been feed ..)
             Read type of experiment to execute;
(10)         Read all necessary parameters for experiment;
        UNTIL no_more experiments to do;
        CASE Exptype OF
             Mapping of Birefringence;
             Degradation of Birefringence;
(15)         Automatic operation;
             Manual  execution;
               REPEAT                (..  until all experiments are done ..)
                   Initialize memory allocation necesary for current
                    experiment;
(20)               Reset error trapping mechanisms;
                    TRY
                     IF not done previously THEN BEGIN
                          Initialize instrumentation;
                          Initialize calibration of light detectors;
(25)                   END;
                      Execute experiment;
                     Update experiment tag (new_label) with current
```

```
                    experimental conditions;
                    Print hardcopy of experiment report;
(30)                Save new_label + new_data if so requested;
                  RECOVER if an error ocurs
                    Clean_up instruments, computer devices, etc.;
                    Print status_quo report;
                  END;                    (..  of error recover block ..)
(35)          UNTIL No_more experiments in exp_queue;
              END;                        (..  of manual execution ..)
          END;                            (..  of Case ..)
      END;    (..  end of execution of birefringence experiments ..)
```

Los números entre paréntesis que aparecen a la izquierda del listado de pseudocódigo corresponden simplemente al número de línea del código, con el único propósito de poder referir, con facilidad, a la operación que se realiza. Entonces, la Línea 1 se refiere a la inicialización de todas las variables involucradas en el experimento. En este bloque se inicializan una gran diversidad de variables, algunas asociadas a los diferentes estados posibles para los instrumentos que se utilizan, otras asociadas a variables del manejo de información que requiere el programa mismo entre otras.

La Línea 2 incluye todas las subrutinas para el encendido correcto de los instrumentos. En ellas el experimento solicita la ayuda del operario cuando es necesaria (*i.e.* que encienda los equipos, etc.) y le asiste en detectar posibles errores de configuración de los instrumentos, o en detectar anomalías en ellos. Una vez que los equipos son operativos, entonces, el programa (Líneas 3–11) solicita la información sobre los parámetros del experimento a realizar, para posteriormente proceder a la ejecución de éste, si así lo requiere el operario.

Los parámetros que contiene la base de datos son de dos tipos: (a) los relacionados con las propiedades del fluido bajo estudio, así como algunos parámetros experimentales que no varían frecuentemente, tales como la dirección de giro de los rodillos, la posición en el flujo en donde se realizan las medidas, etc., y (b) los parámetros que definen los procedimientos de medida, la operación del motor de pasos, etc; que difieren de una corrida experimental a otra.

Por lo anterior y previamente a la realización de un experimento, las Líneas 3 a la 7 revisan los archivos de datos existentes en la computadora relativos a las características del último fluido estudiado, a los parámetros utilizados para la calibración de los equipos, o a las condiciones bajo las cuales se efectuó el último experimento, entre otras. El usuario tiene la opción de actualizar estos datos y entonces se genera una nueva estructura de datos denominada `new_label`; si la actualización de la información no procede, entonces `new_label` es igual a `old_label`.

Las Líneas 9 y 10 dentro del bloque REPEAT-UNTIL tienen como propósito que el usuario dé la información necesaria para la ejecución del experimento. En particular, el usuario debe dar información de las características del flujo a generar, los procedimientos de adquisición de datos a seguir y, en suma, debe dar todos los valores necesarios para las variables del experimento. Estos datos se guardan temporalmente en una cola de experimentos, `exp_queue`, con tantos elementos como experimentos se han de realizar.

El programa considera tres clases de experimentos posibles de realizar, cuyos objetivos son: mapear la anisotropía para regiones globales del flujo, evaluar los cambios del peso molecular promedio del fluido resultado de sus estiramientos —que comúnmente se conoce como degradación— y, por último, realizar diferentes historias de deformación. A partir de la Línea 12 comienza la ejecución de la serie de experimentos `exp_queue`. Así, en la Línea 13 se plantea la ejecución de experimentos de "mapeo" cuyo interés

principal es evaluar la birrefringencia en función de la posición dentro de la celda de flujos. Lo que aquí interesa es conocer la anisotropía inducida por el flujo que muestre el efecto de las no-homogeneidades de las historias de deformación.

El mapeo repite el mismo experimento para diferentes posiciones en el campo de flujo y caracteriza así la anisotropía en función de la posición dentro del flujo. Este bloque de código resulta crítico cuando se estudia la mecánica de fluidos viscoelásticos [19].

Como se mencionó anteriormente, si los cambios conformacionales para las moléculas del líquido son importantes, entonces con frecuencia, las cadenas poliméricas se rompen debido a los esfuerzos aplicados por el flujo alternando así el peso molecular promedio. Por ello, la Línea 14 contiene un bloque de instrucciones que permiten conocer el grado de degradación que ha sufrido el líquido polimérico bajo estudio [25]. El experimento de degradación se refiere a una serie de experimentos realizados de manera secuencial, sin modificar ningún otro parámetro durante la ejecución de esta serie y cuyo objetivo es exponer al líquido a deformaciones severas. La degradación del líquido se puede conocer fácilmente mediante experimentos de relajación de la anisotropía óptica [1] y comparando la evolución de los tiempos característicos del material en función de la historia de deformación aplicada.

Cuando la secuencia de experimentos a realizar no busca hacer un mapeo de la birrefringencia en las cercanías del punto de estancamiento, o no desea conocer si la solución polimérica se degrada, entonces la secuencia completa de experimentos en `exp_queue` se puede realizar *automáticamente*, sin que requiera la intervención del experimentalista, o se puede ejecutar en forma *manual*, en la que el programa da la oportunidad al experimentalista de tomar decisiones mientras éstos se llevan a cabo. De esta manera, la Línea 15 es la opción para realizar automáticamente la secuencia de experimentos y es totalmente equivalente a las líneas restantes del código. Por tanto, a

continuación únicamente se analiza detalladamente la opción *manual* de los experimentos.

Finalmente, para dar comienzo al bloque de la ejecución manual del experimento, se requiere la inicialización de las porciones de memoria que se van a utilizar, Línea 18, las cuales varían considerablemente dependiendo del tipo de experimento, así como todas las variables relacionadas con los posibles errores generados por el experimento. Esto es, dependiendo del tipo de experimento, el número de Bytes de memoria que se reservan para la adquisición de datos, así como su manejo en forma gráfica, varía fuertemente. Es por ello, que las estructuras de datos se generan en forma *dinámica*, justo al iniciar la ejecución del experimento, con el propósito de optimizar los recursos de cómputo disponibles; estas estructuras consumen varios MBytes de memoria, y por lo tanto dichos recursos se deben utilizar de manera juiciosa.

Una vez que los recursos para la ejecución del experimento se han asignado, lo que procede es calibrar los diferentes instrumentos que intervienen, así como "cargar" aquellos programables —como por ejemplo, el controlador de los motores de pasos que requiere los códigos con que se ejecuta el flujo y que se detalla en la siguiente Sección. El tipo de experimento, así como los parámetros con que se cargan los instrumentos se toman de exp_queue que se han dado previamente a la computadora en las Líneas 9–10. Ahora, si bien los procesos de calibración requieren la asistencia del experimentalista, este procedimiento se ejecuta sólo una vez antes de dar inicio a la serie de experimentos. Sin embargo, la necesidad de alimentar a los diferentes instrumentos con los códigos de ejecución, es tarea que siempre se realiza antes de cada experimento, pues esta información es específica de éste e indispensable para su ejecución. Estas acciones se realizan en el bloque definido por las Líneas 22 y 23 y *son una parte esencial del desarrollo de estos programas*. El procedimiento de calibración se describe, en detalle, en la Sección 2.4.

Una vez calibrado el sistema de adquisición de datos y cargados los demás instrumentos que intervienen en el experimento, ahora sigue el bloque de instrucciones con las que se ejecuta el experimento: Línea 26. Puesto que la ejecución del experimento requiere de la sincronización de diferentes eventos que se realiza en, o con la asistencia de diferentes instrumentos, entonces es aquí indispensable utilizar técnicas de *programación en tiempo real* [26] para desarrollar estas operaciones. Este bloque representa el desarrollo de las necesidades del experimento, considerando la idiosincracia del procesador con que se cuenta, el sistema operativo que tiene y los diferentes servicios que pueden utilizarse con este propósito; se describe detalladamente en la Sección 2.5.

Las Líneas 27, 28 y 29 están relacionadas con las opciones para almacenar los datos generados una vez que ha concluido el experimento. Estos datos aparecen también en pantalla y se pueden guardar tanto en los discos de la computadora como conservar de manera impresa. Finalmente, en las Líneas 31 a la 34, el código realiza las operaciones necesarias para "limpiar" todos los instrumentos y variables involucradas en el experimento, con el fin de no corromper los datos de posteriores experimentos y recuperar los recursos de memoria asignados dinámicamente en la Línea 18.

A continuación se presentan las ideas necesarias y los procesos que se realizan en relación a: (a) las características sobresalientes del experimento a realizar, tales como el flujo estacionario y como la historia de deformación que interesa, entre otras que se proponen (Sección 2.3); (b) los instrumentos con los cuales existe una comunicación constante y bidireccional, esto es, con los cuales se requiere enviar y recibir información constantemente, como es el caso de los amplificadores de señales luminosas, el multiplexor y el vóltmetro de alta velocidad (Sección 2.4); (c) el reloj programable; así como (d) el despachador de eventos (Sección 2.5). Si bien, las funciones de los demás instrumentos ya se han integrado al programa de control principal, los códigos

detallados para la operación precisa de algunos de éstos todavía no existen, tales las subrutinas específicas para la operación del controlador del motor de pasos, o de control de la temperatura del fluido. Los implementos para la realización de estas actividades se han recibido recientemente en el Laboratorio y corresponde a otra persona la realización de tales actividades. Se ha desarrollado la totalidad de los códigos de adquisición de datos con el vóltmetro de alta velocidad; asimismo, a continuación se dan los códigos para la ejecución de un tipo de flujo (flujo estacionario), definiendo las características que deben instrumentarse en el controlador de motores para su ejecución.

Sección 2.3. Las Características del Flujo de Estado Estacionario que Induce la Anisotropía Óptica.

El conocimiento de la respuesta del líquido viscoelástico cuando a éste se aplica a una rapidez de deformación constante resulta de interés tanto teórico (porque partiendo de un modelo simple e independiente del tiempo, con frecuencia se pueden predecir las propiedades de un material), como tecnológico porque los procesos industriales se realizan generalmente a una velocidad de procesado constante. Es por ello que en este trabajo se han desarrollado los códigos necesarios para la medición de las anisotropías inducidas por flujos con una rapidez de deformación constante, y que aquí se denomina *flujo estacionario*. Se han considerado también otros tipos de posibles flujos que son útiles para estudiar los tiempos de relajación del material y de los que se amplia la información más adelante.

En el molino de dos rodillos corrotacionales, un flujo estacionario se genera cuando los rodillos giran con una velocidad angular constante por un tiempo suficientemente largo de manera que el fluido alcance una condición de equilibrio con las fuerzas generadas con el flujo. Para los líquidos viscoelásticos, con tiempos característicos hasta

de varios segundos, entonces se requiere mantener el flujo por al menos durante decenas de segundo.

Sin embargo, la respuesta del material a las fuerzas macroscópicas depende de la relación entre las escalas de tiempo de relajación interna del líquido macromolecular y los tiempos característicos del flujo. El número (adimensional) de Deborah, De, expresa la relación entre el tienpo caracteristico[8] del material, τ, y de la deformación (en este caso de el flujo), γ^{-1}. Para el material, en general, se tiene un *espectro* de tiempos de relajación mientras que para el flujo su escala de tiempo depende inversamente de la rapidez de deformación. Cuando De< 1, la respuesta del material es escencialmente viscosa y $\tau << \gamma^{-1}$. Si $\tau >> \gamma^{-1}$ entonces el flujo deforma fuertemente al material y se observan efectos elásticos en el fluido.

Es por lo anterior que resulta útil conocer la anisotropía inducida en el fluido para toda una *serie* de velocidades de deformación y consecuentemente, los experimentos de este tipo de flujo consisten en realidad de una sucesión de mediciones sobre estados estacionarios. Para cada flujo estacionario se aplica una velocidad angular *constante* de los rodillos. La serie de flujos comienza con una velocidad lenta, e incrementa la velocidad en una cantidad predeterminada después de un periodo de tiempo largo (con el propósito de garantizar que la anisotropía del fluido polimérico esté en equilibrio con el flujo). El periodo de tiempo durante el cual la velocidad permanece constante debe ser mayor que el tiempo necesario para que el líquido alcance el equilibrio con las fuerzas externas. El procedimiento se repite hasta alcanzar la máxima velocidad. El número de "saltos" en la velocidad que conforman la serie, el valor inicial de la velocidad, el incremento en cada paso y el tiempo de permanencia en una velocidad dada

---

[8] El tiempo característico del material también se conoce como "tiempo de relajación" (o relajamiento) o como la escala de tiempo.

son parámetros que el usuario aporta a la computadora, previamente a la ejecución del experimento y son parte de los elementos de *exp_queue*.

La Figura 2.2 muestra la ejecución del experimento de flujo estacionario. Los tiempos inmediatamente después del cambio de la velocidad de deformación, Δt, representan los periodos en los cuales la respuesta del material difícilmente alcanza el equilibrio con las fuerzas del flujo. De esta manera, si se desea una medida de la anisotropía óptica del fluido polimérico que sea representativa del flujo estacionario, entonces para cada velocidad, la medición de birrefringencia se debe hacer justo antes de iniciar el cambio de velocidad. Así, los algoritmos propuestos ejecutan la lectura después de transcurridos $0.9\times\Delta T$ e incrementan la velocidad en ΔT. Es obvio que ΔT debe ser más largo que el tiempo característico del material (a la velocidad de deformación válida).

Además del flujo estacionario, existe una gran diversidad de otros flujos útiles para conocer los tiempos característicos del material. Entre ellos, y que ya se han considerado en MHGMAIN se encuentran (a) el arranque de flujo, (b) el cese de flujo constante, (c) el flujo oscilatorio, (d) el flujo en dos pasos, y (e) las deformaciones de dos pasos. El arranque de flujo consiste en hacer girar los rodillos súbitamente y observar el aumento de la birrefringencia hasta que ésta alcanza a un estado estacionario. El cese de flujo es esencialmente el flujo opuesto al arranque y para una birrefringencia en estado estacionario, los rodillos se detienen abruptamente, midiendo el decaimiento posterior de la birrefringencia hasta que ésta alcanza un valor cercano a cero, como se observa en la Figura 2.3. El flujo oscilatorio consiste en medir la birrefringencia mientras los rodillos giran alternadamente en el sentido levógiro y dextrógiro. El flujo en dos pasos consiste en una combinación de *dos* arranques de flujo, el segundo a partir de la velocidad de giro de la primera etapa. Por último, la deformación de dos pasos consiste en la aplicación de dos deformaciones finitas separadas por un intervalo de tiempo.

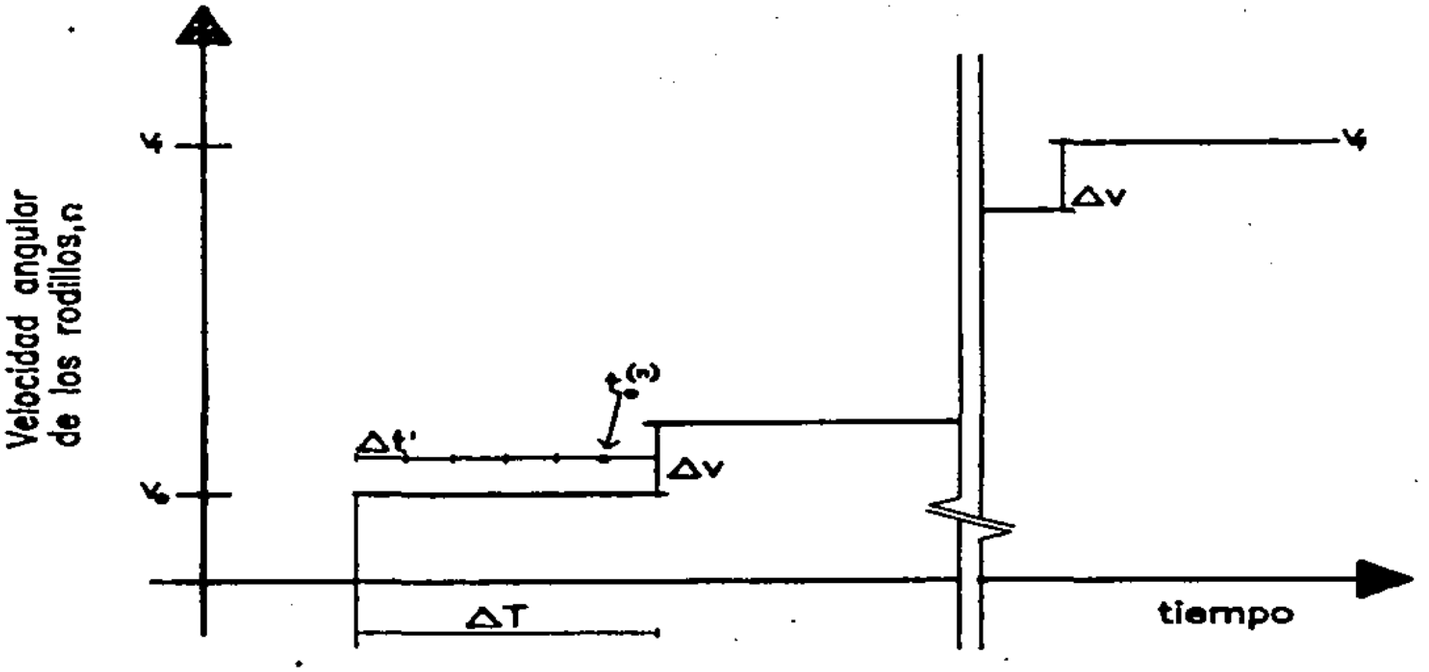


Figura 2.2. Patrón de velocidades angulares para un experimento de flujo estacionario. ΔT es el intervalo de tiempo entre incrementos de velocidad, $\Delta t'$ es el tiempo que dura cada pulso del reloj y Δv es el incremento de velocidad.

La totalidad de estos flujos (el estacionario más los anteriores) aportan información útil para el estudio de la dinámica no lineal del material. Las partes básicas de programación para estos flujos ya se han considerado en el programa principal **MHGMAIN** y la subrutina para la birrefringencia. Sin embargo, todavía no se codifican las subrutinas específicas para generar los flujos.

Sección 2.4. El Procesamiento de las Señales Luminosas.

Esencialmente el arreglo experimental permite la medición de las señales luminosas,

primero llevándolas mediante una fibra óptica a un fotodetector de silicio tipo PIN[9], para posteriormente amplificarlas electrónicamente. El experimento requiere que las señales se lean: (a) con rapidez, (b) para un intervalo amplio de irradiancias, y (c) con buena precisión en las lecturas, tanto en términos de la razón de señal/ruido, como en la sensibilidad con que se pueden medir las intensidades más pequeñas. El sistema de adquisición realiza físicamente la detección de las intensidades mediante la preamplificación —en donde convierte la señal luminosa en una señal de voltaje proporcional— y posteriormente amplificando la señal —con parámetros de ganancia y offset[10] ajustables a través de la computadora y el adquisitor— y cuyo propósito es mantener la relación señal/ruido en su máximo valor.

Además y cuando se cuenta con un procesador que controla el experimento, es posible establecer procedimientos para "calibrar" tanto las medidas específicas como las relaciones entre las medidas, y lograr así los requerimientos planteados. Con el propósito de fijar los puntos anteriores, a continuación se describe con detalle el proceso de medición de anisotropía inducida por flujos para un experimento típico de considerar.

Las Características de las Señales Experimentales.

En la Figura 2.3 se muestra una gráfica típica generada por un experimento de *cese de flujo* [1, 10]. En estos experimentos se desea conocer la respuesta del material cuando el flujo que genera la anisotropía se detiene abruptamente, y entonces se espera

---

[9] Dispositivo semiconductor de dos capas tipo: P y N. Un voltaje de polarización genera entre éstas una región intrínseca (tipo I).

[10] La palabra offset viene del inglés y se refiere a un desplazamiento o corrimiento de la señal, no deseable. En este caso, se refiere al corrimiento que hay entre los valores medidos y los datos reales o válidos.

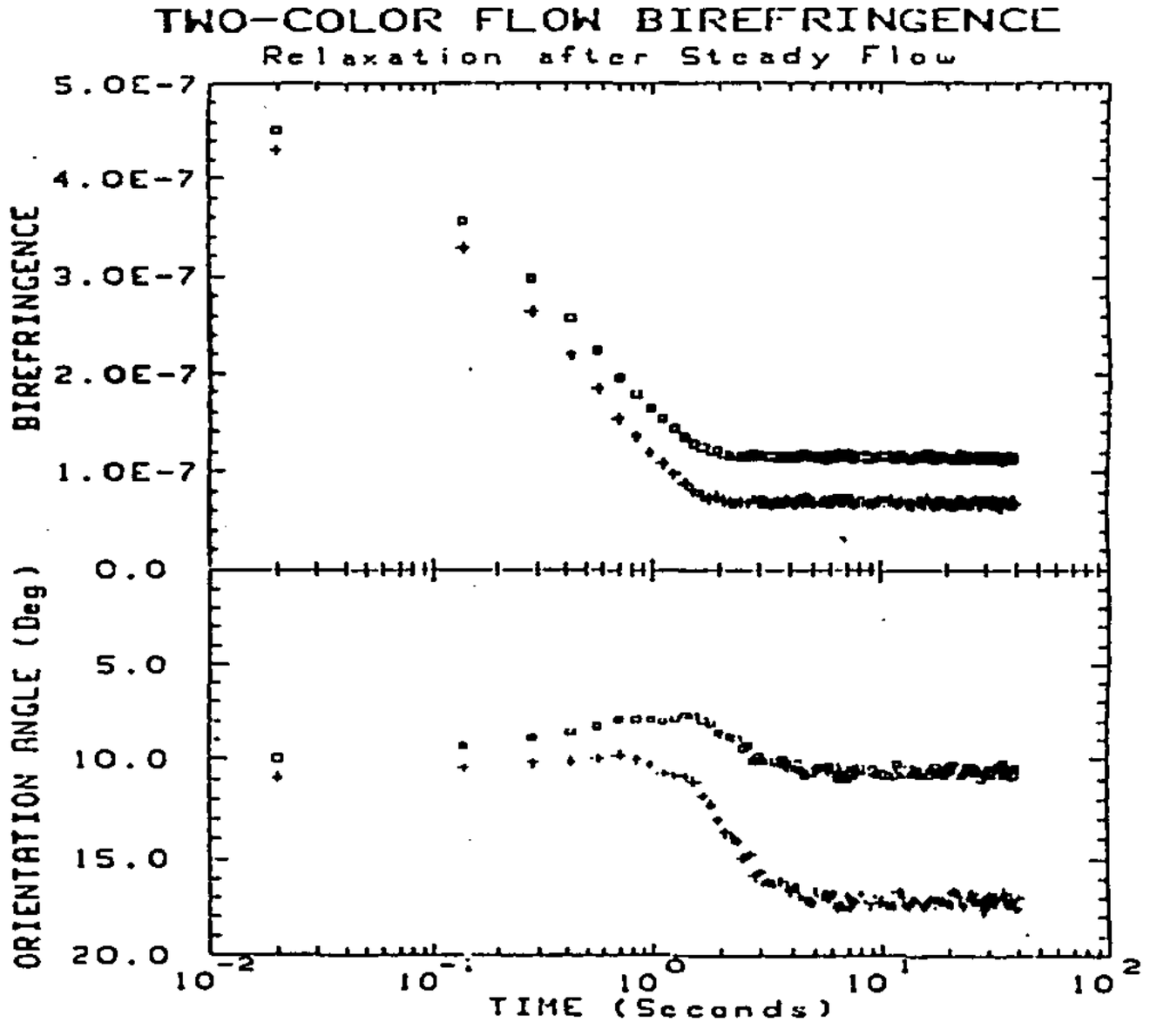


Figura 2.3. Gráfica típica generada por el experimento de cese de flujo en donde podemos ver la dependencia tanto de la birrefringencia como de su ángulo de orientación en función del tiempo[1, 10]. Los datos con cuadros corresponden a medidas con contribuciones de birrefringencia espuria debidas al tren óptico y al sistema de detección, y los datos representados por cruces muestran aquéllos donde se minimizaron las contribuciones espurias de tipo óptico.

que la configuración polimérica se relaje, con las señales luminosas paulatinamente disminuyendo hasta valores fuera de la capacidad de detección experimental. La anisotropía óptica correspondiente debe disminuir en magnitud y se espera también que exista un alineamiento del fluido polimérico que no varía con el tiempo.

Para el relajamiento de la anisotropía (desde un estado fuertemente deformado hasta el relajamiento completo) que se muestra en la Figura 2.3, el intervalo de anisotropías requiere al menos de 3 decadas de variación para algunas de las intensidades de los haces. Por ello, es deseable que el sistema de adquisición de datos tenga también la capacidad de ajustar la ganancia de los amplificadores casi instantáneamente y lo haga antes de cada secuencia de medidas para un "punto". Como resultado de la necesidad de ajustar tal ganancia, también es necesario hacer un ajuste de offset pues los corrimientos no deseados dependen en general del valor de la ganancia del circuito[21, 22]. Por ejemplo, en la Figura 2.3, el proceso de detección debe seguir la señal cambiando la ganancia del amplificador de acuerdo a la magnitud de la intensidad, sin que con ello se pierda precisión en las lecturas.

Los valores de la anisotropía que interesan medir corresponden a intensidades luminosas que van desde 500 pW hasta 2 mW (*i.e.*, cubren intervalos de irradiancia del orden de 7 decadas) [22]. Para el sistema de detección utilizado, las potencias luminosas anteriores corresponden a voltajes de entrada a la etapa de amplificación de 250 μV (para las intensidades más débiles y operando con ganancias cercanas a 10,000), mientras que las intensidades del orden de 1 mW generan aproximadamente 10 V a la salida del amplificador (con una ganancia de 1). La cota mínima de 250 μV se debe fundamentalmente a la existencia de no-idealidades en los elementos electrónicos del sistema de amplificación que presentan un corrimiento de voltaje indeseable (offset de carácter electrónico) [21].

En el intervalo entre $2\times10^{-2}s$ y $8\times10^{-2}s$ (Figura 2.3) es claro que no se tiene información sobre la birrefringencia ni sobre el ángulo de orientación[11]. Si se desea mejorar estos resultados, es necesario realizar del orden de 100 lecturas en una décima de segundo. Esto requiere de un sistema de detección y amplificación de señales que sea capaz de ajustar sus parámetros y realizar una medición en, por lo menos, una milésima de segundo. Ahora, el vóltmetro HP44704 tiene capacidad para realizar hasta cien mil lecturas por segundo, siendo entonces posible hacer múltiples lecturas sobre cada canal del multiplexor (que corresponden a las diferentes intensidades luminosas), sin que la relajación de la anisotropía cambie sustancialmente. Así, los algoritmos propuestos realizan múltiples lecturas para cada "punto" de la gráfica de anisotropía. La desventaja principal de este esquema consiste en requerir mayores cantidades de memoria pero ésta se compensa con una mejora en la precisión de los datos del "punto" [21].

Las Correcciones por Anisotropías Espurias.

En la Figura 2.3 los datos señalados por cuadros representan las medidas realizadas incluyendo contribuciones de birrefringencia parásita debida a las imperfecciones de los diferentes elementos ópticos que conforman el arreglo, de modo que se mide una birrefringencia total: la muestra polimérica —de interés—, mas la del sistema óptico —la espuria. Los datos marcados con cruces son las mismas medidas pero sustrayendo las contribuciones espurias de origen óptico. En general, la anisotropía residual del tren óptico incluye una diversidad de causas, algunas de ellas varían con periodos de meses, otras con periodos de varias horas. Como puede observarse en dicha figura, los errores sistemáticos más fuertes ocurren con relación a la precisión para determinar la orientación de la birrefringencia, que con frecuencia son de decenas de grados. Es por ello

---

[11] La máxima rapidez de muestreo de la anisotropía que está reportada en la literatura es de 12 "puntos" (lecturas) por segundo, que se debe al mecanismo de sincronización y adquisición utilizado.

que ahora se propone, previamente a la ejecución de un experimento, un procedimiento de calibración, entre cuyos objetivos está la evaluación de la anisotropía residual que no altere las medidas intrínsecas del fluido.

Aún después de substraer los errores sistemáticos de origen óptico, las contribuciones remanentes de birrefringencia son del orden de $5*10^{-8}$, las cuales se deben esencialmente a ruido electrónico, pues existe un voltaje remanente (estrictamente positivo) que en el experimento se interpreta como una pequeña intensidad de luz asociada a una anisotropía parásita resultado de a las no-idealidades de los circuitos electrónicos.

Es por ello, que la sensibilidad con que se puede determinar la birrefringencia está definida por la sensibilidad con que se puede medir la intensidad de luz. Esto es, una detección precisa y de alta resolución de las intensidades de luz es primordial en la determinación de la anisotropía, en especial para valores pequeños de irradiancia que corresponden con la mínima intensidad detectable por los sensores de luz del sistema.

Las Características de los Voltajes Medidos.

Además de las imperfecciones en los voltajes medidos ya mencionadas, tales como la necesidad de ajustar la ganancia y offset de los amplificadores, existen dos fuentes de errores sistemáticos que se pueden conocer y corregir. Primero, el convertidor analógico-digital HP44704A tiene una resolución aproximada de 14 bits, pero presenta un corrimiento en su calibración. Segundo, la conversión de los datos digitales acarrea un error en el bit menos significativo, resultado del error de redondeo en la conversión del valor analógico a digital que realiza la computadora. Así, con el propósito de corregir las no idealidades en los preamplificadores y los amplificadores es necesario calibrar periódicamente los instrumentos de medida electrónica, además de hacer un muestreo de los voltajes espurios de esta etapa.

La solución propuesta aquí es la medición de voltajes cuando la luz incidente en los detectores es nula. Es decir, se desea conocer los voltajes generados en el sistema de preamplificación y amplificación, para cada canal y para todas las ganancias a utilizar, cuando no llega luz a los fotodiodos y éstos están operando. Esta calibración permite corregir las imperfecciones de los componentes electrónicos de las etapas de detección, así como de la contribución debida al bit menos significativo del convertidor HP44704A.

Las Correcciones de los Voltajes Medidos.

Además de la calibración con cero irradiancia, dado que se cuenta con un vóltmetro que maneja grandes velocidades de toma de datos es de esperarse que la precisión de lectura no sea lo deseable. Sin embargo, es fácil corregir este defecto haciendo un ajuste lineal de los datos. Los factores de corrección que se utilizan en esta subrutina fueron calculados al hacer 1,000 medidas con el vóltmetro de alta velocidad (HP 44704A) y tomando simultáneamente medidas con un vóltmetro de precisión (HP 34420A). Las fuentes de voltaje son una colección de baterías de 0.150, 1.5 y 9 V que se mantienen a temperatura constante. Los diferentes voltajes corresponden al uso de cada batería conectada en paralelo con los vóltmetros. Las medidas de voltaje se hicieron sobre cada uno de los canales del multiplexor, en cada una de las diferentes ganancias y para las opciones de lectura de "single channel" y "multiple channel". Un ejemplo de estas medidas se puede apreciar en la Figura 2.4. Como se observa, es factible alcanzar precisiones cercanas a una parte en 10,000 con voltajes absolutos. Las curvas de calibración para todos los canales se presentan en el Apéndice C.

Por último, es necesaria una transferencia de información, desde el adquisitor hacia la computadora a la mayor velocidad. Para lograr lo anterior es necesario que los datos

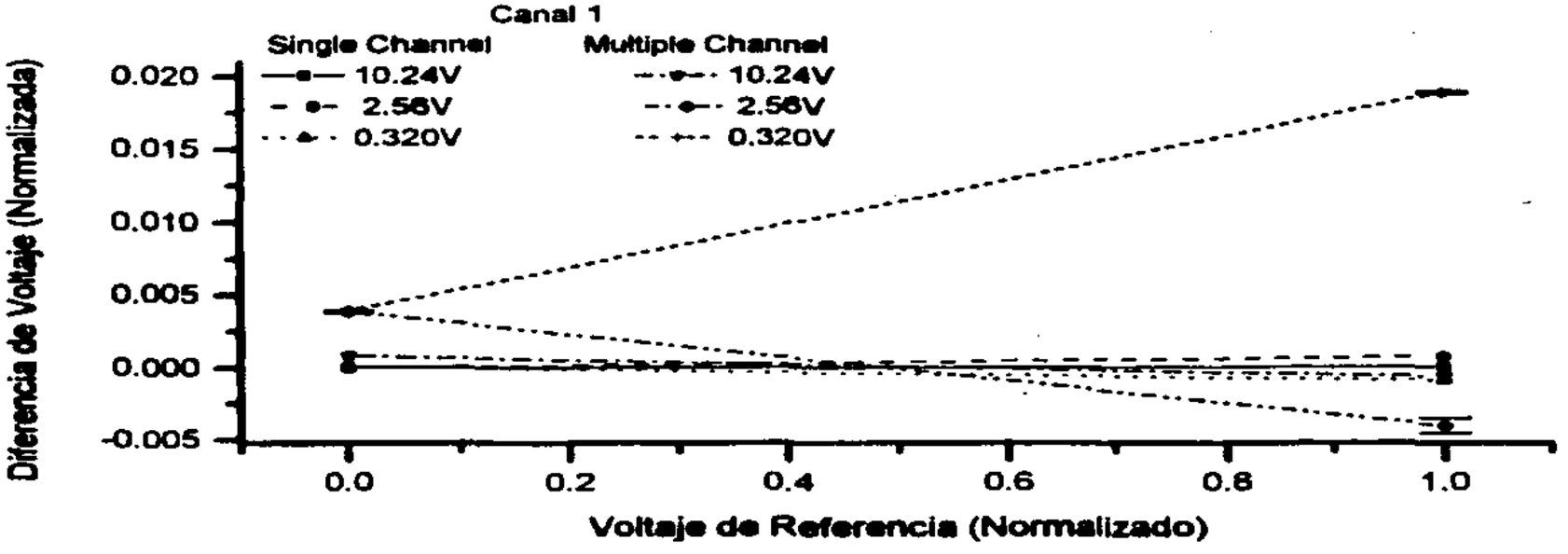


Figura 2.4. Esta gráfica muestra las diferencias de voltajes entre el voltaje medido y el voltaje absoluto (medido con el vóltmetro de precisión HP 34420A), en función del voltaje de referencia. La información de estas gráficas se usa para determinar los factores de corrección de las medidas hechas con el vóltmetro de alta velocidad.

generados por el vóltmetro no sean decodificados sino hasta que llegan a la computadora. Es decir, es necesario que la transferencia de lecturas de voltajes (dos Bytes) se haga en números *2–complementarios*[12]. Una vez que se tienen los Bytes en la computadora, se ejecuta la subrutina de decodificación para tener lecturas en valores reales. Lo anterior se desarrolla en el Procedure `MHGBintoReal`. Esta subrutina realiza la conversión de los

[12] Del idioma inglés "2–complement" para describir un número binario complemento a dos.

Bytes según el signo de la lectura, el tipo de lectura (el vóltmetro puede ejecutar medidas de voltaje, corriente y resistencia) y de la ganancia a la cual se realizó la medida. La ecuación necesaria para convertir los números binarios a números reales es

$$Real = (2Complement)/(800), \tag{2.1}$$

si se trata de una cantidad positiva, y para una cantidad negativa la ecuación correspondiente es

$$Real = -MaxRange + (2Complement)/(800). \tag{2.2}$$

Aquí, *Real* se refiere al número real que se desea obtener, *2Complement* es el número binario 2–complementario y *MaxRange* es el valor máximo válido para la ganancia elegida. Para la mínima ganancia el voltaje máximo es 10.24 V. Por ello, si para representar un intervalo completo de ±10.24 V se consideran 14 bits, entonces el valor de los 2 Bytes se debe dividir entre 800 para obtener el voltaje correspondiente.

Sección 2.5. La Ejecución del Experimento de Birrefringencia.

Son cuatro las actividades principales a realizar durante la ejecución del experimento: el control de los instrumentos, la adquisición de datos, el procesado de la información y la graficación de los parámetros de la anisotropía óptica para supervisar la evolución del experimento. La adquisición de datos efectúa la medición de voltajes en intervalos de tiempo muy precisos para así determinar las intensidades luminosas como función del tiempo. El control de la instrumentación está relacionado con la petición por parte de los instrumentos de realizar acciones pre-determinadas, tales como cambios en la dirección de giro de los rodillos, el aumento de velocidad para el flujo estacionario, entre muchos otros. El procesamiento de datos tiene que ver con el manejo de las medidas de voltaje para convertirlas en medidas de intensidad luminosa. Por último, el graficado

en pantalla de los datos requiere del cálculo de la anisotropía de la muestra, tal y como se describe en el Capítulo 1.

Estas actividades requieren de recursos de cómputo que deben compartirse entre las actividades. Sin embargo, algunas de estas actividades tienen una necesidad extrema de ejecutarse en tiempos muy estrictos, tal como la adquisición de datos que requiere "etiquetar" las lecturas con tiempos del orden de milésimas de segundo. En contraparte, la graficación de la información se puede hacer en "tiempo real" del orden de segundos, sin que ello implique una seria desventaja para el experimentalista.

La manera de compartir los recursos computacionales disponibles para las actividades principales es utilizando un despachador. El despachador es un elemento de sofware cuya función es asignar los recursos de cómputo de acuerdo a las necesidades instantáneas del experimento. Este elemento se activa cada vez que existe una solicitud de recursos computacionales y los otorga al código que los solicita y que tiene la más alta prioridad. Por ejemplo, las necesidades de adquisición de datos siempre se ejecutan a expensas de la graficación, pues para esta última, retrasos de centésimas de segundo son irrelevantes para el usuario.

La Figura 2.5 muestra las relaciones que existen entre los dispositivos (hardware) y las actividades (software). En esta Figura se muestra el desarrollo del *despachador en tiempo real* que se activa para la ejecución del experimento. La relación entre los elementos toma en cuenta los instrumentos externos con que cuenta el experimento así como aquellos internos que se pueden accesar y poner bajo el control del experimento como es el caso del reloj. El despachador en tiempo real se ha estructurado tomando en cuenta las características del sistema operativo, que si bien tiene un kernel (núcleo) poderoso para aplicaciones en el laboratorio, el desarrollo del despachador es aún primitivo, en particular cuando se compara con alternativas factibles de programar utilizando lenguajes

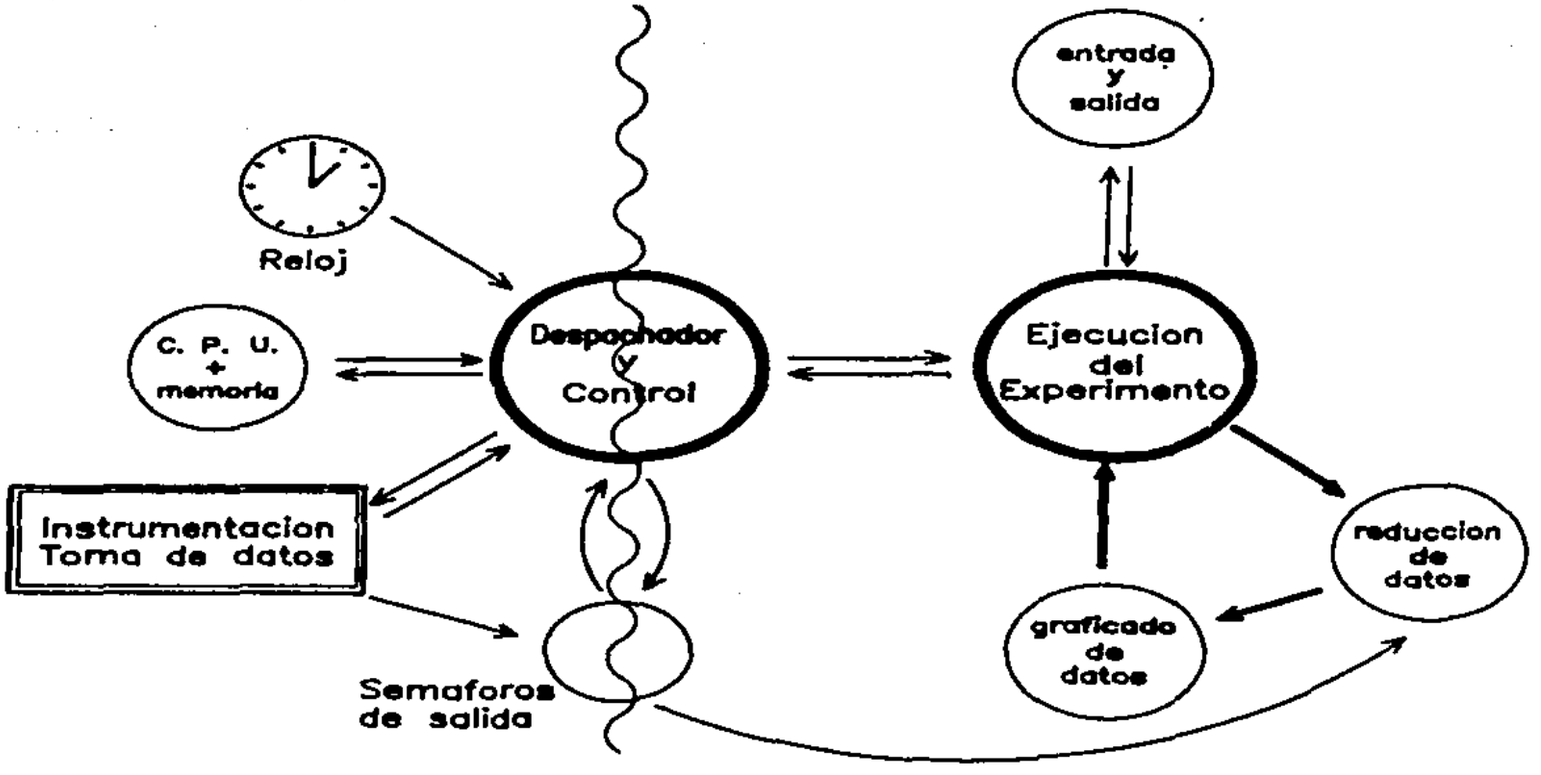


Figura 2.5. Relaciones entre los diferentes componentes del software —lado derecho de la línea vertical— y el hardware —lado izquierdo. Los elementos principales corresponden al despachador, encargado del control en tiempo real y el programa de ejecución del experimento. Las flechas gruesas indican el flujo del programa principal que se interrumpe en cualquier instante si las actividades de control del despachador así lo requieren.

más modernos como MODULA, Oberon, Ada, etc.

Existen además otras limitaciones del despachador debidas al hardware disponible. Para instrumentos como los aquí utilizados, una reacción ante un estímulo suficientemente rápida es del orden de centésimas de segundo, mientras que hoy en día las computadoras tienen tiempos de respuesta mucho más rápidos. Los tiempos lentos de la instrumentación

periférica se deben fundamentalmente a tiempos lentos en las interfaces de comunicación entre el CPU y el instrumento. En estos casos, difícilmente se logra tener velocidades de sincronización mayores que una centésima de segundo.

El sistema de control también debe ser *completo* y *correcto*. Es decir, el sistema debe ser capaz de contemplar todas las posibles situaciones generadas en un sistema experimental incluyendo aquellas anómalas. En realidad, es realmente difícil lograr lo anterior ya que por cada evento debe corresponder una acción bien definida, de la cual no necesariamente se conoce *a priori* su causa o efecto. Como resultado de esto, el código debe tener una programación "defensiva" con la finalidad de *identificar y manejar* posibles errores o situaciones excepcionales en la ejecución de experimentos, que no fueron previstos inicialmente por el diseñador del software, y que se activan en los instrumentos ante condiciones anómalas. Por ello, los elementos de la Figura 2.5 muestran flechas en ambas direcciones, y representan la necesidad de interacción bidireccional con los instrumentos y de operación en "modo protegido", capaz de detectar errores.

La adquisición, el procesamiento y la graficación de los datos son procesos que están relacionados de tal forma que la ejecución de uno desactiva a los otros dos. Además, su relación también requiere del uso de variables o parámetros en común que sean manejados de manera segura. Esto se puede resolver utilizando copias de las variables comunes, las cuales se muestran como semáforos en la Figura 2.5. Afortunadamente, estos tres procesos tienen prioridades de ejecución muy diferentes, teniendo la adquisición de datos la prioridad más alta. Sin embargo, la adquisición de datos y el control de la instrumentación son operaciones complementarias y por lo tanto su sincronización es esencial. Actualmente estos dos eventos deben correr siempre en sincronía, y bajo el control de tiempos marcado por el reloj programable.

La Figura 2.6 muestra del lado izquierdo el código correspondiente al despachador

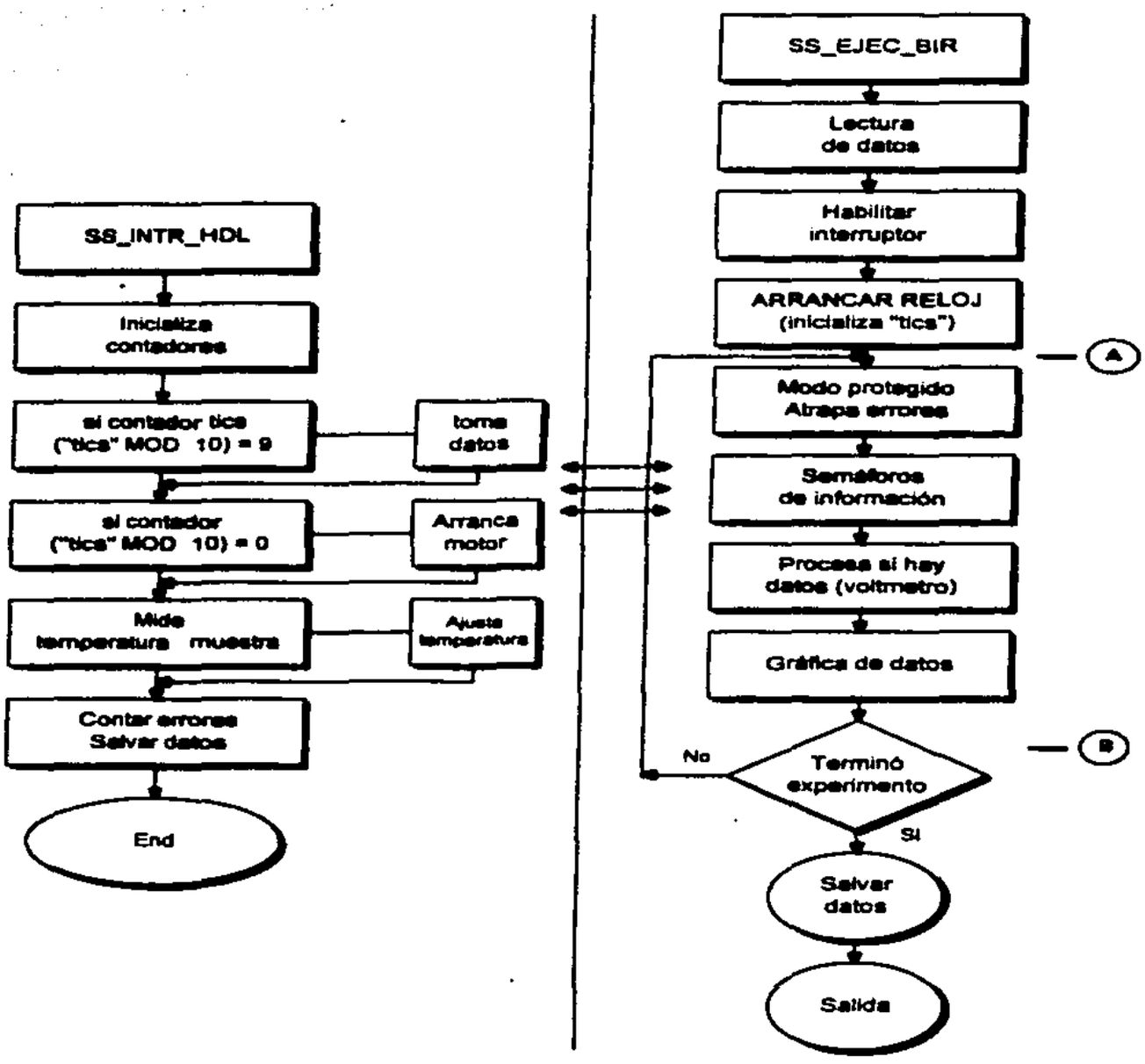


Figura 2.6. Los bloques de código para la ejecución del experimento de birrefringencia de flujos estacionarios. El diagrama derecho corresponde a la ejecución del experimento y corre con baja prioridad. El diagrama izquierdo en el despachador y corre con la más alta prioridad. Este código interrumpe al código de la derecha en cualquier punto del loop AB.

y que se ejecuta para manejar las interrupciones del reloj, y a su vez para realizar la comunicación entre los instrumentos y la adquisición de datos. El reloj se arranca justo antes del punto A de la Figura 2.6. Una vez que el experimento inicia y entre los puntos A y B de la ejecución de la subrutina SS_Exec_Bir, el reloj causa interrupciones del código de la derecha, independientemente del punto en el que se encuentra la ejecución. El lado derecho es el código principal lleva a cabo el procesamiento y la graficación de los datos con baja prioridad.

Para el experimento de estado estacionario, descrito en la Sección 2.3, se requiere del siguiente manejador de interrupciones que corresponde al Procedure SS_Intr_HDL, dentro de la subrutina SS_Exec_Bir y listado en el Apéndice A.

La función del despachador, SS_Intr_HDL, es escoger la tarea con prioridad más alta y secuencialmente realizar las otras tareas que también se deben ejecutar. Para SS_Intr_HDL la variable CYCLICCOUNT representa el tiempo real medido como el número de pasos (pulsos o "tics") dados por el reloj cíclico con una rapidez hasta de centésima de segundo[13]. El intervalo se asigna de acuerdo con el intervalo de tiempo más largo común a todos los procesos, y el manejador de interrupciones decide qué proceso es el que cuenta, en ese momento, con los recursos del procesador. CYCLICCOUNT se inicializa (CYCLICCOUNT = 0) antes de que el reloj cíclico comience a funcionar. Luego de la primera interrupción (CYCLICCOUNT = 1), el manejador manda al controlador de motores de paso la instrucción de arranque de flujo manteniendo la velocidad constante hasta el décimo paso del reloj, para luego aumentar la velocidad y nuevamente mantenerla a ésta durante 10 pasos más. Esto se repite hasta que el barrido de velocidades indicadas

---

[13] La precisión del reloj es del orden de 5 μs pues tiene una fuente de 1 MHz. Sin embargo, se puede programar únicamente intervalos de 0.01 s o más largos.

```
      PROCEDURE SSINTR_HDL(VAR STATBYTE, DATABYTE: BYTE;
                           VAR DOIT: BOOLEAN);
      {- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
       Esta subrutina secuestra al CPU cuando ocurre una interrupción
       y sólo le devolverá el mando cuando esta interrupción haya sido
       atendida. Este procedimiento se encarga de hacer la adquisición
       de datos a través del ADC y de hacer el disparo del SMI para
       generar la escalera de pasos.
       - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }

      VAR   i : NATURAL;

(1)   BEGIN
        IF (ODD(STATBYTE DIV 32) AND ODD(DATABYTE DIV 32)) THEN BEGIN
          IF Incr_Cyc THEN BEGIN
            CYCLICCOUNT := CYCLICCOUNT + 1;
(5)         CYCLICFLAG:=TRUE;              {contador para la validación
                                            de la ejecución del experimento}
            END;
          IF (CYCLICCOUNT MOD 10 = 1) THEN
            TRIGGER(SMI_isc);     { disparar el SMI. Velocidad aumenta! }
          IF (CYCLICCOUNT MOD 10 = 0) THEN BEGIN
(10)                          {.. Comienzo de la adquisición de datos.. }
            TRY
              Incr_Cyc := FALSE;
              MHGMeasure_Scan(ADC_Dptr^);
              Incr_Cyc := TRUE;
(15)          RECOVER BEGIN
                IF (ESCAPECODE = IOESCAPECODE) AND (IOE_ISC = 18) THEN
                  Add_Error(ErrorLT, ADC3)
                ELSE
                  IF (ESCAPECODE = -1) THEN
(20)                Add_Error(ErrorLT, ADC4)
                  ELSE ESCAPE(ESCAPECODE);
                Incr_Cyc := TRUE
              END;
            Save_dpoint(ADC_Dptr, GainRECIn, datqueue);
(25)                        {Las lecturas del ADC son puestas en una cola.}
            STEPCOUNT := STEPCOUNT + 1;
            STEPFLAG := TRUE;              {contador para la validación
                                            de la ejecución del experimento}
          END
        END
(30)  ELSE CALL(SAVEISRHOOK,STATBYTE, DATABYTE, DOIT);
      END;          {- - - - - - - Fin del Procedure SSINTR_HDL.}
```

por el usuario antes de la ejecución del experimento se complete. La sincronización de las actividades de generación del flujo y de la adquisición de datos se da en la Línea 9 del código cuando CYCLICCOUNT es un múltiplo de 10. En este instante se realizan las lecturas de datos sobre todas las intensidades de luz azul y verde.

La adquisición de datos se lleva a cabo en un bloque de captura de errores demarcado por TRY-RECOVER. De esta manera, cualquier falla en el convertidor digital-analógico se reporta al sistema y los datos son descartados. MHGMeasure_Scan toma hasta 5000 lecturas por detector de luz, con el propósito de aumentar la precisión de las medidas, siendo éstas salvadas en el apuntador ADC_Dptr que es parte de la cola datqueue. Inmediatamente después, el contador de pasos, STEPCOUNT, aumenta. STEPCOUNT es la variable semáforo que permite la lectura de variables comunes al bloque de ejecución principal y al despachador. STEPCOUNT activa el procesamiento de datos sólo después de que éstos han sido validados. Puesto que el procesamiento de datos en el CPU es más veloz que la toma de datos, la validación es indispensable para el buen éxito del experimento, ya que de otra manera, la "cola" de datos puede agotarse quedando el CPU en vacío o procesando operaciones con ceros.

Cuando se termina la ejecución del despachador todos los registros y contadores del CPU se reestablecen con los valores del proceso interrumpido para así continuar ejecutando los procesos de baja prioridad.

La subrutina de SS_Exec_Bir es la encargada de la ejecución del experimento de anisotropías inducidas por flujos estacionarios. A continuación se muestra un pseudocódigo que describe las acciones de esta subrutina. El código completo se puede consultar en el Apéndice A de esta Tesis.

```
(1) Initialization of variables and pointers;
    Set_Voltmeter;
    Initialization of data buffers;
    Save default interrupt handlers;
(5) Mount cyclic interrupt handler(SS_INT_HDL); (Habilita el Reloj)
    TRY                                          (Inicia modo protegido)
      Initialize Variables;
```

```
        Load cyclic timer(time_lapse);
        Setup execution of SMI program;
(10)    Initialize tag for datafile;
        Start cyclic timer;
        WHILE Experiment not completed DO BEGIN   (Punto A Fig.  2.6)
          IF data readings finished THEN BEGIN
            Disarm timer;
(15)        Reset interrupt handler;
            Denergize SMI windings;
           END;
         IF rawdata available THEN BEGIN
            Update variables for backround processes;
(20)        Convert data from binary to real;
            Do Statistical analysis of data;
            Adjust Voltage;
            Normalize data;
          END;
(25)     IF plotting birefringence THEN BEGIN
           Compute orientation angle and birefringence;
           Plot data;
         END;                                          (Punto B Fig.  2.6)
       END;
(30)   RECOVER                              (Excepción del modo protegido)
        Disarm and replace default interrupt handler;
        Clean dynamic variables;
        Reset motor to initial state;
        Print hardcopy of bir.plot;
(35)    Reset monitor (graphics to alphanumerics);
```

```
        END;
    END;
```

La inicialización del sistema es siempre la primera operación que se debe efectuar. Previo a la ejecución del experimento, *Set_Voltmeter* se encarga de cargar en el adquisitor todos los datos necesarios para la medición de irradiancias, tales como número de lecturas, velocidad de muestreo, etc. En la Línea 5 se habilitan las variables necesarias para el manejador de interrupciones. En la Línea 10 se hace una inicialización de la etiqueta (record de información) del experimento en donde se guarda la información referente a la corrida a efectuarse. Ésta es una medida de seguridad mediante la cual es posible desdoblar la información contenida en los discos en el caso de que algo anómalo le pase al experimento. En la Línea 11 se inicia la operación del manejador de interrupciones que ya se describió con detalle.

Si la toma de datos ha terminado de acuerdo al protocolo de ejecución del experimento a continuación se procede a poner los instrumentos en un estado desactivado, Líneas 13–17. Si aún existen datos sin procesar, entonces se continúa con el análisis de éstos. En primer lugar, los datos en forma binaria se convierten a números reales, luego se procede a la reducción estadística de los datos. Como se trata del vóltmetro de alta velocidad hay que hacer una calibración de los voltajes. Acto seguido, se calculan las intensidades relativas. En las Líneas 25–28 se procede, si así fue especificado por el usuario, a graficar en pantalla los datos obtenidos. Por último, en las Líneas 29–35, el código realiza las subrutinas necesarias para dejar "limpios" todos los intrumentos y variables involucradas en el experimento, con el fin de que no se corrompan los datos de experimentos posteriores.

Capítulo 3.

Conclusiones.

En este trabajo se presentó una descripción completa de la programación desarrollada en el experimento de birrefringencia bicolor inducida por flujos, con el cual se pueden caracterizar los cambios microestructurales de fluidos poliméricos, sujetos a condiciones de flujo. Este experimento se ha desarrollado en el Laboratorio de Reología-Óptica del Instituto de Investigaciones en Materiales de la U.N.A.M., y este trabajo se ubica en el desarrollo de los códigos para procesadores digitales que llevan a cabo las mediciones con dicha técnica. En especial, el trabajo instrumenta el despachador de eventos que es la parte medular de la programación en tiempo real.

El despachador representa la manera más eficiente de compartir los recursos computacionales disponibles entre las diferentes actividades a realizar. Este elemento se activa cada vez que existe una solicitud de recursos computacionales y los otorga al código que los solicita y que tiene la más alta prioridad.

En la Figura 2.5 se muestra la interdependencia de los instrumentos, los recursos de cómputo y el despachador. Aunque el despachador se ha estructurado de manera

poderosa y flexible la principal limitante que existe ahora en el experimento es el reloj programable, pues sus intervalos de tiempo mínimo son de una centésima de segundo. La sincronización entre módulos es sin embargo menor a 1 milisegundo, y este valor depende de la velocidad de respuesta del instrumento más lento del conjunto. Este despachador puede realizar la caracterización de la anisotropía de polímeros con tiempos de respuesta del orden de una centésima de segundo y representa una mejora de 8 veces sobre lo reportado hasta ahora.

Para los experimentos propuestos, la adquisición de datos debe ser muy precisa. Para ello, los códigos realizan la calibración de los instrumentos y el experimento completo, buscando: (a) eliminar las contribuciones debidas a imperfecciones en la electrónica utilizada, esta contribución se calcula al hacer las medidas de los datos cuando se bloquea la fuente de luz; (b) eliminar las contribuciones debidas a las imperfecciones de la óptica del sistema, estas contribuciones se calculan sin considerar un medio birrefringente en el tren óptico; (c) hacer la correción de las medidas generadas en el vóltmetro de alta velocidad ya que acarrea errores sistemáticos en los datos debidos a su rapidez, los factores de correción se definen comparando las lecturas con un vóltmetro de alta resolución; y por último, (d) hacer un análisis estadístico de los datos. Para poder mejorar la precisión de las medidas es utilizando la opción, en el vóltmetro de alta velocidad, de resolución de 16 bits, actualmente se usan 14 bits. El problema que surgió alrededor de este punto es que el "driver" para 16 bits no es compatible con la computadora utilizada; además la documentación al respecto no está actualizada a la versión de software-hardware disponibles.

Por último, la adquisición de datos también debe ser muy rápida lo que obliga a la utilización eficiente y extensiva de apuntadores y colas de datos, para no saturar la memoria RAM disponible. También se hace una transferencia binaria de datos entre el

adquisitor y la computadora, siendo esta última la encargada de decodificar los datos. Un ejemplo de la bondad de transferir los datos en binario es que si se toman 10,000 datos y se envían en forma de reales, la transferencia toma alrededor de 2 minutos mientras que si se hace la misma operación de manera binaria el tiempo requerido es menor a 7 segundos.

Es importante mencionar que la instrumentación utilizada junto con los códigos computacionales desarrollados en este trabajo mejoran las medidas reportadas con anterioridad en hasta dos ordenes de magnitud en la precisión y en 100 veces en la velocidad de muestreo. Los datos reportados [1, 20, 25] tienen incertidumbres relativas del orden del 2% sin que se haya evaluado la exactitud de los voltajes obtenidos. Con este trabajo se espera reducir las incertidumbres relativas a menos del 0.5% sobre voltajes absolutos. Las velocidades de muestreo reportadas [1, 20, 25] son de 12 "puntos" por segundo siendo ahora posible más de 1,000; sin embargo, los tiempos de respuesta del motor de pasos distan mucho de ser los adecuados para una velocidad de muestreo como la posible actualmente.

Para el desarrollo del experimento con todas las opciones de flujo mencionadas en la Sección 2.3 se requiere el desarrollo de los códigos para el control del motor de pasos, lo que involucra tal vez el desarrollo de mas de 1,000 líneas de programación. Otro paso en la mejora de este trabajo plantea el uso de la interface GPIO, en lugar del HP-IB, la cual permite una velocidad en la transferencia de información del vóltmetro a la computadora de 750 KBytes/segundo a diferencia del HP-IB que logra únicamente 330 KBytes/segundo.

Apéndice A. Código del Módulo Dich_library

```
$HEAP_DISPOSE ON$
$SYSPROG ON$
$REF 100$
MODULE Dich_Lib;
{ ---------------- MODULE Dich, Supplement to Module Bir_Lib.----------------}

$SEARCH '#11:/PASCAL_F/MHG/MHGLIB.'$

IMPORT SYSGLOBALS, IODECLARATIONS, SYSDEVS, A804XDVR, DGL_LIB,GENERAL_0,
       GENERAL_1, GENERAL_2, GENERAL_3, GENERAL_4,
       hpib_0, hpib_1, hpib_2, hpib_3,                    { .. system modules.}
       asmmod, MATH_functions, Thinkjet, UserProc, FileOper,
       Monitor, AD_Converter, Laser, SMI, General_exp,
       Bir_Library;                          { .. this experiment set-up modules.}

EXPORT

   PROCEDURE Set_DBirRept(ExpCond: BExpCondREC; lbnumber: NATURAL;
                     VAR Bparinfo: Parinforec; VAR Newlabel: EXPDATREC;
                        VAR InstrFLAG: INSTRFLAGREC; VAR Errorlist: ErrLIST);

   PROCEDURE DB_ProcB(Newtag: EXPDATREC; InstrFLAG: INSTRFLAGREC; VAR
                             BExpCond: BExpCondREC; exp_IDnum: NATURAL;
                               VAR DatLIST: EXPLIST; VAR Errors: ErrLIST);

IMPLEMENT

  CONST    TransDsize     = 40;
           totalsamples   = 600;

  TYPE     ID_Attr        = (xaxtick, splnum, cyclenum);{ ... data structure ... }
           IDtag          = ARRAY[ID_Attr] OF NATURAL;  { .. for normalization ..}
           TransDatARRAY  = ARRAY[1..TransDsize] OF REAL;{ \                      }
           TransArrayPTR  = ^TransDatARRAY;              {  \                     }
           TransDatptr    = ^TransDatTYPE;               {   \                    }
           TransDatTYPE   = RECORD                       {    \                   }
               IDnums     : IDtag;                       {     \  .......       }
               TransData  : TransDatARRAY;               {      > .......       }
               next       : TransDatptr;                 {     /  .......       }
             END;                                        {    /                   }
                                                         {   /                    }
           TransDatQUEUE  = RECORD                       {  /                     }
               first,                                    {  /                     }
               last : TransDatptr;                       { ..  and temporary  .. }
             END;                                       {.. storage of rawdata.. }
```

```
        color_stat   = (bluemean, greenmean, sqrblue, sqrgreen, crossbg);
        IDspec       = (numpts, xaxcount);          {  .. data structure for .. }
        IDlabel      = ARRAY[IDspec] OF NATURAL;    {  .. statistics of data .. }
        STATarray    = ARRAY[color_stat] OF REAL;   {  .. readings from ADC card.}
        STAT_totals  = RECORD
            IDinfo     : IDlabel;
            subtotals  : STATarray;
          END;

        GainREC = RECORD
          VoltG_Bref,
          VoltG_Gref,
          VoltG_Bord,
          VoltG_Bext,
          VoltG_Gord,
          VoltG_Gext             :INTEGER;
          END;

        TotalsARRAY = ARRAY[1..totalsamples] OF STAT_totals;
        means       = ARRAY[1..totalsamples] OF DATAarray;
        means_ptr   = ^means;

        str255      = STRING[255];
        iarraytype  = ARRAY[0..maxint] OF shortint;
        rarraytype  = ARRAY[0..maxint] OF REAL;
        rarraypt    = ^rarraytype;
        iarraypt    = ^iarraytype;


CONST   zeroID      = IDlabel[0, 0];
        zerototals  = STATarray[0, 0, 0, 0, 0];
        Invalid15   = DStat_Array[15 OF 0];



PROCEDURE Voltage_Adjust(Gain, numpoints :INTEGER;
                         Rawvoltage:ARRAY[xmin..xmax:NATURAL] OF REAL;
                         VAR Realdata:ARRAY[ymin..ymax:NATURAL] OF REAL);
{-------------------------------------------------------------------------
 This PROCEDURE converts the voltage redings from the 16 bit voltmeter into
 real lectures. The correction factors were calibrated with the HP 34420A
 nanovoltmeter.                                                    MHG 6/96.
-------------------------------------------------------------------------}
TYPE  FactorARRAY     = ARRAY[1..4,1..6] OF REAL;    {Array that contains the
                                                      coorection factors for
                                                      voltage readings trough
                                                      the 3852A}

VAR k :INTEGER;
    Factor, Offset: FactorARRAY;
```

```
BEGIN
  Factor[2,1]:=0.999604569;                    Offset[2,1]:=1.250644E-3;
  Factor[2,2]:=0.999695469;                    Offset[2,2]:=1.561138E-3;
  Factor[2,3]:=0.999650400;                    Offset[2,3]:=1.480594E-3;
  Factor[2,4]:=0.999701144;                    Offset[2,4]:=1.566516E-3;
  Factor[2,5]:=0.999757632;                    Offset[2,5]:=1.566722E-3;
  Factor[2,6]:=0.999697165;                    Offset[2,6]:=1.410780E-3;
  Factor[3,1]:=0.997478207;                    Offset[3,1]:=9.960994E-3;
  Factor[3,2]:=0.997470725;                    Offset[3,2]:=1.000005E-2;
  Factor[3,3]:=0.997479436;                    Offset[3,3]:=1.000035E-2;
  Factor[3,4]:=0.997625173;                    Offset[3,4]:=1.000428E-2;
  Factor[3,5]:=0.997433906;                    Offset[3,5]:=1.000526E-2;
  Factor[3,6]:=0.997495539;                    Offset[3,6]:=9.965936E-3;
  Factor[4,1]:=0.999419171;                    Offset[4,1]:=8.336048E-3;
  Factor[4,2]:=0.999553941;                    Offset[4,2]:=9.064708E-3;
  Factor[4,3]:=0.999564734;                    Offset[4,3]:=8.965992E-3;
  Factor[4,4]:=0.999595729;                    Offset[4,4]:=9.33544E-3;
  Factor[4,5]:=0.999505199;                    Offset[4,5]:=9.068326E-3;
  Factor[4,6]:=0.999486638;                    Offset[4,6]:=8.256216E-3;

  FOR k:=1 TO numpoints DO BEGIN
    Realdata[6*k-5]:=Factor[Gain,1]*(Rawvoltage[6*k-5]-Offset[Gain,1]);
    Realdata[6*k-4]:=Factor[Gain,2]*(Rawvoltage[6*k-4]-Offset[Gain,2]);
    Realdata[6*k-3]:=Factor[Gain,3]*(Rawvoltage[6*k-3]-Offset[Gain,3]);
    Realdata[6*k-2]:=Factor[Gain,4]*(Rawvoltage[6*k-2]-Offset[Gain,4]);
    Realdata[6*k-1]:=Factor[Gain,5]*(Rawvoltage[6*k-1]-Offset[Gain,5]);
    Realdata[6*k]:=Factor[Gain,6]*(Rawvoltage[6*k]-Offset[Gain,6]);
   END;

  END;


PROCEDURE DB_DetectorCal(AutoFLAG: BOOLEAN; VAR ADCpar: ADCparREC;
          VAR Newtag: EXPDATREC; VAR Errors: ErrLIST; VAR Instr: INSTRFLAGREC);
{-------------------------------------------------------------------------
 This procedure is part of the calibration necessary for the measurement of
 relative values of light intensity, where the temporal fluctuations of the
 incident light intensity and anisotropic scattering (or absorption) by dust
 particles has been taken into account. The incident, and light past the flow
 cell are measured using six detectors. Two of these detectors are located
 before the flow cell and are used to monitor the incident laser beam inten-
 sities, for both colors. The maximum expected light intensity through the
 fluid sample is measured (as the sum of light intensities for the ordinary
 and extraordinary light detectors after the sample.) The extraordinary de-
 tector is located along the incident beam path. The ordinary detector is lo-
 cated 45 degrees off the incident beam. The calibration is done using a half
 wave retarder inserted before the analizer with angular orientation such
 that the complementary detector also evaluates Imax. The procedure is repe-
 ated for both colors. The ADC card takes a set of measurements for both
 colors and evaluates the calibraton constants necessary to correlate the
 measured values at the detectors for the ordinary and extraordinary beams.
 These correction factors are saved as variables ADCpar.ReltI_Color.
 as ADCpar.NormI_Color.
```

```
The ratio of light intensities before and after the flow cell are stored
as ADCpar.NormI_Color.
Enrique Geffroy.                                  Rev. 1.0   22/March/90.
All the parameters related with de aquisition were update.     MHG 10/96
-------------------------------------------------------------------------}

LABEL   100, 150, 200;

CONST   Blue_AtnFactor = 0.987;                { .. last measured 11/Jan/89.  }
        Green_AtnFactor= 0.997;
        resolution     = 4095;
        p_size = 1;             p3_size  = 3;
        START = 1;              STOP = 6;
        p_size = 1;             p3_size  = 3;
        g_size  = 3;
        c_size   = 3;           d_size   = 30000;
        REPT    = 1000;

TYPE    i_array         = ARRAY[1..3] OF SHORTINT;
        r_array         = ARRAY[1..3] OF REAL;
        ia_ptr          = ^i_array;
        ra_ptr          = ^CALCarray;
        r_ptr           = ^REAL;
        r3_ptr          = ^r_array;

        SSd_array       = ARRAY[1..d_size] OF REAL;
        SSd_ptr         = ^SSd_array;
        BigArray        = ARRAY[1..REPT] OF REAL;
        BigArray_ptr    = ^BigArray;
        BufTYPE         = BUF_INFO_TYPE;
        BufD_ptr        = ^BufTYPE;


        BufD_nodeptr    = ^BufD_nodetype;

        BufD_nodetype   = RECORD
              Bufdata   : BufTYPE;
              Gaindata  : GainREC;
              Next      : BufD_nodeptr;
            END;

        BufD_queue      = RECORD
                front   : BufD_nodeptr;
                rear    : BufD_nodeptr;
            END;


VAR     key                                 : CHAR;
        channel, gain3                      : ia_ptr;
        pace                                : r_ptr;
        pace3                               : r3_ptr;
        Rawdata1, Rawdata2                  : SSd_ptr;
        ADC_Dptr, Char_Dptr                 : BufD_ptr;
```

```
        BufRawdata1, BufRawdata2,
        Realdata, Realdata2                 : SSd_ptr;
        datqueue                            : BufD_queue;
        kaputFLAG, OutFLAG, BlueFLAG,
        GreenFLAG                           : BOOLEAN;
        sigmean, sigma,
        BEmean, BERmean, BOmean, BORmean,       { ..  given in mWatts * 5.0  }
        GEmean, GERmean, GOmean, GORmean,       { ..  given in mWatts * 5.0  }
        BTmean, BTRmean, GTmean, GTRmean,       { ..  given in Volts (10 max)}
        BlueExtExp, GreenExtExp,
        BlueOrdExp, GreenOrdExp,
        BCal_Const, GCal_Const,                 {ratio of Int before/after FC}
        IO_BlueRatio, IO_GreenRatio,            {Calib Const Extra/Ord detec.}
        IO_Ratio, BExtRatio, GExtRatio : REAL;
        k, IOError                          : NATURAL;
        VG_BExt, VG_BOrd, VG_BRef,
        VG_GExt, VG_GOrd, VG_GRef           : 0..6;
        BrefMean, BrefDelta,
        GrefMean, GrefDelta,
        BintE, BintED, BintO, BintOD,
        GintE, GintED, GintO, GintOD        : BigArray_ptr;
        GainRECIn                           : GainREC;
        Bufindx, Conv2, Byte1               : INTEGER;
        Unconverted1, Unconverted2          : CHAR;

BEGIN
 IF Instr.LASERsetFLAG AND Instr.ADCsetFLAG THEN BEGIN
   NEW(channel);                                                { .. Sundries for .. }
   NEW(pace);              pace^ := 0.0001;                     {    .  .  .  .  .  }
   NEW(gain3);             gain3^[1] := 1;                      {       .  .  .  .  }
   gain3^[2] :=1;          gain3^[3] := 1;                      {       .  .  .  .  }
   NEW(pace3);             pace3^[1] := 0.0001;                 {       .  .  .  .  }
   pace3^[2] := 0.0001;    pace3^[3] := 0.0001;                 {       .  .  .  .  }
   NEW(Rawdata1);          NEW(Rawdata2);                       { .. data reading .. }
   NEW(ADC_Dptr);          NEW(BufRawdata1);       NEW(BufRawdata2);
   NEW(Realdata);          NEW(Realdata2);

   WITH GainRECIn DO BEGIN
     VoltG_Bref := 4;                                   {\.............................}
     VoltG_Gref := 4;                                   {.\............................}
     VoltG_Bord := 4;                                   {..\...........................}
     VoltG_Bext := 1;                                   {...\..........................}
     VoltG_Gord := 4;                                   {....\.........................}
     VoltG_Gext := 1;                                   {.....\........................}
    END;                                                {TEMPORARY values.}
                                                        {They have to be removed}
   VG_BExt := 1;                                        {...../........................}
   VG_BOrd := 4;                                        {..../.........................}
   VG_BRef := 4;                                        {.../..........................}
   VG_GExt := 1;                                        {../...........................}
   VG_GOrd := 4;                                        {./............................}
   VG_GRef := 4;                                        {/.............................}
```

```
NEW(BintE);          NEW(BintED);                    { . . . . . . . }
NEW(BintO);          NEW(BintOD);                    { . . Data arrays .. }
NEW(GintE);          NEW(GintED);                    {      . . for . .   }
NEW(GintO);          NEW(GintOD);                    {      . . data . .  }
NEW(BrefMean);       NEW(BrefDelta);                 { ..  calculations.  }
NEW(GrefMean);       NEW(GrefDelta);                 {  .  .  .  .  .  .  }
REPEAT
  OutFLAG := FALSE;
  WITH ADCpar DO BEGIN     { .. initialize variables with known values .. }
      ReltI_Blue  := Blue_AtnFactor;
      ReltI_Green := Green_AtnFactor;
      NormI_Blue  := Blue_AtnFactor;
      NormI_Green := Green_AtnFactor
    END;

  WRITELN;
  WRITE('    CALIBRATION OF LIGHT DETECTORS FOR TWO-COLOR ');
  WRITELN('DICHROISM & BIREFRINGENCE.');
  WRITELN;
  WRITELN('In order to proceed, The minimum (Imin) and maximum ',
                                        'light intensity (Imax)');
  WRITELN('for the BLUE & GREEN extraordinary and ordinary beams, ',
                                        'needs to be evalu-');
  WRITELN('ated. FIRST, ajust the blue-beam optics for maximum ',
                                        'extinction through the');
  WRITELN('EXTRAORDINARY beam; the ordinary blue beam should ',
                                        'read Imax. Now, ajust the');
  WRITELN('green-beam optics for maximum extinction through the ',
                                        'EXTRAORDINARY green');
  WRITELN('beam while the ordinary beam reads Imax.   It is also ',
                                        'desirable that the');
  WRITELN('voltage output of the light detectors have values of ',
                                        'about 8 Volts, assur-');
  WRITELN('ing the highest resolution of the Analog to Digital ',
                                        'conversion. Therefore,');
  WRITELN('adjust the calibration  potentiometers so that  all ',
                                        'light intensity meters');
  WRITELN('read approximately 75% of full scale for the maximum ',
                          'value. ', #132'Do not change'#128);
  WRITELN('the calibration setting beyond this point.');
  WRITELN;
  WRITE('                    ');
  WRITE('                If Calibrate hit RETURN ....  Q = <new values> ');

  TRY
    Read_Returnkey(kaputFLAG);
    WRITE('                .    ');
    WRITE('      .    .    .    .    .    . DO NOT DISTURB!!  Stop = <esc>');
    WRITESTRINGLN(Mainframe, 'RST');
    Set_Voltmeter(REPT,10,0.00001);
    IOBUFFER(ADC_Dptr^,60000);
    BUFFER_RESET(ADC_Dptr^);
```

```
      TRY
        Bufindx:=1;
        MHGMeasure_Scan(ADC_Dptr^);          {Triggering the ADC}
        WHILE BUFFER_DATA(ADC_Dptr^) >0 DO BEGIN
          READBUFFER(ADC_Dptr^, Unconverted2);
          Conv2:=ORD(Unconverted2);
          READBUFFER(ADC_Dptr^, Unconverted1);
          Byte1:=ORD(Unconverted1);
          MHG_BintoReal(Conv2, Byte1, 4,BufRawdata1^[Bufindx]);
          Bufindx:=Bufindx+1;
         END;
        Voltage_Adjust(4, REPT, BufRawdata1^, Realdata^);
        WRITELN;
      RECOVER BEGIN
        IF (ESCAPECODE = IOESCAPECODE) AND (IOE_ISC = 18) THEN BEGIN
           IOError := IOE_RESULT;
           IORESET(18);
           TwoBEEPS;
           WRITELN;
           WRITELN('ADC I/O library Error in READING voltages ',
                                              'for Calibration..');
           WRITE(IOERROR_MESSAGE(IOError),
                                          '   RTN to Restart. Q=<esc> ');
           Read_Returnkey(kaputFLAG);
           IF kaputFLAG THEN BEGIN
             OutFLAG := TRUE;
             Add_Error(Errors, DetCal1);
            END
           ELSE OutFLAG := FALSE;
           GOTO 200
          END
        ELSE ESCAPE(ESCAPECODE)
       END;
      FOR k := 1 TO REPT DO BEGIN     { .. means for reference detectors .. }
          BrefMean^[k]   := Realdata^[6*k-3];
          BrefDelta^[k]  := 10.0/resolution;
          Grefmean^[k]   := Realdata^[6*k-2];
          GrefDelta^[k]  := 10.0/resolution;
        END;
      fit_LSQR(BrefMean^, BrefDelta^, REPT, BTmean, sigmean, sigma);
      fit_LSQR(GrefMean^, GrefDelta^, REPT, GTmean, sigmean, sigma);

      BlueExtExp := 1/poweri(10.0, VG_BExt);
      BlueOrdExp := 1/poweri(10.0, VG_BOrd);
      GreenExtExp := 1/poweri(10.0, VG_GExt);
      GreenOrdExp := 1/poweri(10.0, VG_GOrd);

  { .... readings normalized to discriminate for temporal fluctuation  .... }
      FOR k := 1 TO REPT DO BEGIN    { .. voltage to light int. conversion. }
          BintE^[k]   := Realdata^[6*k-5]*BlueExtExp           { .. blues .. }
                                 *BTmean/(BrefMean^[k]);
          BintED^[k] := BlueExtExp/ Resolution;
```

```
      BintO^[k]   := Realdata^[6*k-3]*BlueOrdExp
                          *BTmean/(BrefMean^[k]);
      BintOD^[k]  := BlueOrdExp/ Resolution;
      GintE^[k]   := Realdata^[6*k-4]*GreenExtExp          { .. greens .. }
                          *GTmean/(GrefMean^[k]);
      GintED^[k]  := GreenExtExp/ Resolution;
      GintO^[k]   := Realdata^[6*k-2]*GreenOrdExp
                          *GTmean/(GrefMean^[k]);
      GintOD^[k]  := GreenOrdExp/ Resolution
    END;
  fit_LSQR(BintE^, BintED^, REPT, BEmean, sigmean, sigma);
  fit_LSQR(BintO^, BintOD^, REPT, BOmean, sigmean, sigma);
  fit_LSQR(GintE^, GintED^, REPT, GEmean, sigmean, sigma);
  fit_LSQR(GintO^, GintOD^, REPT, GOmean, sigmean, sigma);

  IF (BTmean <0)OR(GTmean < 0)OR
     (BEmean < 0)OR(BOmean < 0)OR
     (GEmean < 0)OR(GOmean < 0)   THEN BEGIN
    WRITELN('CAUTION: one or more of the voltage readings for light ',
                                                    'intensities have');
    WRITELN('mean NEGATIVE values! Please, check the calibration of ',
                                                    'light detectors ... ');
    WRITE('                  ');
    WRITE('       To re-start calibration  hit RETURN ...  Q = <esc> ');
    Read_Returnkey(kaputFLAG);
    IF kaputFLAG THEN BEGIN
      Add_Error(Errors, DetCal2);
      OutFLAG := TRUE
     END
    ELSE OutFLAG := FALSE;
    GOTO 200
   END;
  REPEAT
    BlueFLAG := FALSE;
    WRITELN;
    WRITELN('                    CALIBRATION OF BLUE BEAM DETECTORS');
    WRITELN('Now the calibration of the ext. beam detector is need',
                                                    'ed with respect to ');
    WRITELN('the ord. detector past the analyzers. The procedure re',
                                                    'quires the measure-');
    WRITELN('ment of Imax for the BLUE laser beam at both detectors ',
                                                    'in an independent ');
    WRITELN('manner. First, SET the blue and green detectors with ',
                                                  'gains that do not sat-');
    WRITELN('urate the power meters. Now for the BLUE beam place the',
                                                    ' polychomatic half-');
    WRITELN('wave retarder after the flow device and before the blue',
                                                    ' detectors. Ajust');
    WRITELN('the orientation of the retarder (by rotating it) until ',
                                                  'the maximum intensity');
    WRITELN('is observed at the blue extraordinary detector (minimum',
                                    ' at the ordinary de-');
    WRITELN('tector.)');
    WRITELN;
```

```
WRITE('                    ');
WRITE('                              When ready hit RETURN ...  Q = <esc> ');
Read_Returnkey(kaputFLAG);
IF kaputFLAG THEN escape(-20);
WRITELN;
WRITE('              .     ');
WRITE('     .    .    .    .    .    ... DO NOT DISTURB!!  Stop = <esc> ');
channel^[1] := 1;  channel^[2] := 3;  channel^[3] := 5;
TRY
  Bufindx:=1;
  MHGMeasure_Scan(ADC_Dptr^);          {Triggering the ADC}
  WHILE BUFFER_DATA(ADC_Dptr^) >0 DO BEGIN
    READBUFFER(ADC_Dptr^, Unconverted2);
    Conv2:=ORD(Unconverted2);
    READBUFFER(ADC_Dptr^, Unconverted1);
    Byte1:=ORD(Unconverted1);
    MHG_BintoReal(Conv2, Byte1, 4,BufRawdata2^[Bufindx]);
    Bufindx:=Bufindx+1;
   END;
  Voltage_Adjust(4, REPT, BufRawdata2^, Realdata2^);
  WRITELN;
RECOVER BEGIN
  WRITELN;
  IOError := IOE_RESULT;
    IF (ESCAPECODE = IOESCAPECODE) AND (IOE_ISC = 18) THEN BEGIN
     IOError := IOE_RESULT;
     IORESET(18);
     TwoBEEPS;
     WRITE('ADC I/O library Error in READING voltages ');
     WRITELN('for blue Calibration..');
     WRITE(IOERROR_MESSAGE(IOError),
                                        '   RTN to Restart. Q = <esc> ');
     Read_Returnkey(kaputFLAG);
     IF kaputFLAG THEN BEGIN
       OutFLAG := TRUE;
       Add_Error(Errors, DetCal1);
       GOTO 200
      END
     ELSE BEGIN
       BlueFLAG := FALSE;
       GOTO 100
      END;
    END
  ELSE
    IF ESCAPECODE = - 20 THEN BEGIN
        WRITE('Stopped by Operator ...  ');
        WRITE('RTN to re-start blue calibration. Q = <esc> ');
        Read_Returnkey(kaputFLAG);
        IF NOT(kaputFLAG) THEN GOTO 100
        ELSE ESCAPE(-20)
      END
    ELSE BEGIN
      WRITELN('Error trapped reading Voltages ! ', ESCAPECODE:0);
      ESCAPE(ESCAPECODE)
     END
END;
```

```
FOR k := 1 TO REPT DO BEGIN    { .. means of reference detectors .. }
    BrefMean^[k]   := Realdata2^[3*k];
    BrefDelta^[k]  := 10.0/resolution;
  END;
fit_LSQR(BrefMean^, BrefDelta^, REPT, BTRmean, sigmean, sigma);
BlueExtExp := 1/poweri(10.0, VG_BExt);
BlueOrdExp:= 1/poweri(10.0, VG_BOrd);
FOR k := 1 TO REPT DO BEGIN
    BintE^[k]   := Realdata2^[3*k-2]*BlueExtExp
                                   *BTRmean/(BrefMean^[k]);
    BintED^[k]  := BlueExtExp/ Resolution;
    BintO^[k]   := Realdata2^[3*k-1]*BlueOrdExp
                                   *BTRmean/(BrefMean^[k]);
    BintOD^[k]  := BlueOrdExp/ Resolution
  END;
fit_LSQR(BintE^, BintED^, REPT, BERmean, sigmean, sigma);
fit_LSQR(BintO^, BintOD^, REPT, BORmean, sigmean, sigma);

IF (BTRmean < 0)OR(BERmean < 0)OR(BORmean < 0) THEN BEGIN
  WRITELN('CAUTION: one or more of the voltage readings of BLUE ',
                                                  'intensities have');
  WRITELN('mean NEGATIVE values! Please, check the calibration of ',
                                                  'light detectors ... ');
  WRITE('                    ');
  WRITE('To re-start Blue calibration  hit RETURN ...  Q = <esc> ');
  Read_Returnkey(kaputFLAG);
  IF kaputFLAG THEN BEGIN
    Add_Error(Errors, DetCal2);
    OutFLAG := TRUE;
    GOTO 200
   END
  ELSE GOTO 100
 END;


BCal_const := (BERmean / Blue_AtnFactor - BEmean) /
              (BOmean - BORmean / Blue_AtnFactor);
IF (BCal_Const < 0.66)OR(1.5 < BCal_Const) THEN BEGIN
    WRITELN('   .  .  .  .  .  Calibration may NOT be correct !!');
    WRITELN('Blue RELATIVE light intensities between ORD and EXTRA',
                                                  'ordinary detectors');
    WRITELN('are too disimilar. Check (a) gain values given to the',
                                                  ' computer during cal-');
    WRITELN('ibration procedure, or (b) power meter gain settings',
                                                  ' used during calibra-');
    WRITELN('tion,or (c) alignment of blue beam detectors.');
    WRITE('                          ');
    WRITE('                    Is calibration  acceptable ? (Y/N) ');
    Read_input(YNset, key);
    IF (key = 'N')OR(key = 'n') THEN GOTO 100
    ELSE Add_Error(Errors, DetCal3);
  END;
```

```
  { .. relative values of the blue light BEFORE and AFTER the flow cell .. }
      IO_BlueRatio := (BEmean + Bcal_Const*BOmean)*
                                powerI(10, VG_Bref)/BTmean;
                        { .. uncalibrated! REF det. It is assumed mWatts.}
      IO_Ratio := powerI(10, 3 - VG_Bref)* IO_BlueRatio;
      IF (IO_Ratio < 0.66)OR(1.5 < IO_Ratio) THEN BEGIN
          WRITELN('   .   .   .   .   .  Calibration may NOT be correct !!');
          WRITELN('Blue light intensities BEFORE and AFTER flow cell are',
                                                    ' too disimilar. Check');
          WRITELN('(a) the gain values given to the computer during cali',
                                                   'bration procedure, or');
          WRITELN('(b) gain settings used for the power meters, or (c) ',
                                                   'alignment of blue beam');
          WRITELN('detectors.');
          WRITE('                          ');
          WRITE('                      Is calibration  acceptable ? (Y/N) ');
          Read_input(YNset, key);
          IF (key = 'N')OR(key = 'n') THEN GOTO 100
          ELSE Add_Error(Errors, DetCal3);
        END;

      WITH Newtag, Birpar, Execute DO BEGIN
        IminB := BEmean*ORD(BEmean > 0)/5.0;    { .. convert to mWatts .. }
        ImaxB := (BCal_Const * BOmean + BEmean)/5.0;
        BExtRatio := ImaxB / IminB;
        WRITELN('IminB: ', IminB:10, ',    Ext. ratio = ', BExtRatio:6:0,
                           ',     Ratio I/O B_Int: ', IO_BlueRatio:5:3)
       END;
      BlueFLAG := TRUE;
100:  UNTIL BlueFLAG;     { ... calibration of Blue detector successful ... }

    {  .... calibration of ORD, EXTRA, and REF green light detectors  ....  }
      REPEAT
        GreenFLAG := FALSE;
        WRITELN;
        WRITELN('                          GREEN BEAM DETECTORS CALIBRATION.');
        WRITELN('Now the whole procedure needs to be repeated for the ',
                                                  'GREEN beam. Ajust the');
        WRITELN('half wave retarder''s orientation (by rotating it) until ',
                                                  'the maximum inten-');
        WRITELN('sity is observed at the green extraordinary detector. ',
                                                  '(Minimum at the or--');
        WRITELN('dinary detector.)');
        WRITELN;
        WRITE('                  ');
        WRITE('                          When ready hit RETURN ...  Q = <esc> ');
        Read_Returnkey(kaputFLAG);
        IF kaputFLAG THEN ESCAPE(-20);
        WRITELN;
        WRITE('              .   ');
        WRITE('    .   .   .   .   .   .... DO NOT DISTURB!!  Stop = <esc>');
        channel^[1] := 2;  channel^[2] := 4;  channel^[3] := 6;
```

```
  TRY
    Bufindx:=1;
    MHGMeasure_Scan(ADC_Dptr^);              {Triggering the ADC}
    WHILE BUFFER_DATA(ADC_Dptr^) >0 DO BEGIN
      READBUFFER(ADC_Dptr^, Unconverted2);
      Conv2:=ORD(Unconverted2);
      READBUFFER(ADC_Dptr^, Unconverted1);
      Byte1:=ORD(Unconverted1);
      MHG_BintoReal(Conv2, Byte1, 4, BufRawdata2^[Bufindx]);
      Bufindx:=Bufindx+1;
     END;
    Voltage_Adjust(4, REPT, BufRawdata2^, Realdata2^);
    WRITELN;
  RECOVER BEGIN
    WRITELN;
    IF (ESCAPECODE = IOESCAPECODE) AND (IOR_ISC = 18) THEN BEGIN
       IOError := IOR_RESULT;
       IORESET(18);
       TwoBEEPS;
       WRITE('ADC I/O library Error in READING voltages ');
       WRITELN('for Green Calibration..');
       WRITE(IOERROR_MESSAGE(IOError),
                                          '   RTN to Restart. Q = <esc> ');
       Read_Returnkey(kaputFLAG);
       IF kaputFLAG THEN BEGIN
         OutFLAG := TRUE;
         Add_Error(Errors, DetCal1);
         GOTO 200
        END
       ELSE GOTO 150
      END
    ELSE
      IF ESCAPECODE = - 20 THEN BEGIN
          WRITE('Stopped by Operator ...   ');
          WRITE('RTN to re-start green calibration. Q = <esc> ');
          IF NOT(kaputFLAG) THEN GOTO 150
          Read_Returnkey(kaputFLAG);
          IF NOT(kaputFLAG) THEN GOTO 150
          ELSE ESCAPE(-20)
       END
      ELSE BEGIN
        WRITELN('Error trapped reading Voltages ! ', ESCAPECODE:0);
        ESCAPE(ESCAPECODE)
       END
  END;
                                    :
 FOR k := 1 TO REPT DO BEGIN    { .. means of reference detectors .. }
     GrefMean^[k]   := Realdata2^[3*k];
     GrefDelta^[k]  := 10.0/resolution;
   END;

 fit_LSQR(GrefMean^, GrefDelta^, REPT, GTRmean, sigmean, sigma);
 GreenExtExp := 1/poweri(10.0, VG_GExt);
 GreenOrdExp:= 1/poweri(10.0, VG_GOrd);
```

```
      FOR k := 1 TO REPT DO BEGIN
          GintE^[k]   := Realdata2^[3*k-2]*GreenExtExp
                                          *GTRmean/GrefMean^[k];
          GintED^[k]  := GreenExtExp /Resolution;
          GintO^[k]   := Realdata2^[3*k-1]*GreenOrdExp
                                          *GTRmean/GrefMean^[k];
          GintOD^[k]  := GreenOrdExp/ Resolution;



        END;
      fit_LSQR(GintE^, GintED^, REPT, GERmean, sigmean, sigma);
      fit_LSQR(GintO^, GintOD^, REPT, GORmean, sigmean, sigma);
      IF (GTRmean < 0)OR(GERmean < 0)OR(GORmean < 0) THEN BEGIN
        WRITELN('CAUTION: one or more of the voltage readings of GREEN ',
                                                        'intensities have');
        WRITELN('mean NEGATIVE values! Please, check the calibration of ',
                                                    'light detectors ... ');
        WRITE('                    ');
        WRITE('To re-start Green calibration  hit RETURN.    Q = <esc> ');
        Read_Returnkey(kaputFLAG);
        IF kaputFLAG THEN BEGIN
          Add_Error(Errors, DetCal2);
          OutFLAG := TRUE;
          GOTO 200
         END
       ELSE GOTO 150
      END;

     GCal_const  := (GERmean / Green_AtnFactor - GEmean) /
                              (GOmean - GORmean / Green_AtnFactor);
     IF (GCal_Const < 0.5)OR(1.8 < GCal_Const) THEN BEGIN
         WRITELN('    .  .  .  .  .  Calibration may NOT be correct !!');
         WRITELN('Green RELATIVE light intensities between ORD and ',
                                              'EXTRA ordinary detectors');
         WRITELN('are too disimilar. Check (a) gain values given to ',
                                              'the computer during cal-');
         WRITELN('ibration procedure, or (b) power meter gain ',
                                             'settings used during calibra-');
         WRITELN('tion, or (c) alignment of blue beam detectors.');
         WRITE('                        ');
         WRITE('                    Is calibration  acceptable ? (Y/N) ');
         Read_input(YNset, key);
         IF (key = 'N')OR(key = 'n') THEN GOTO 150
         ELSE Add_Error(Errors, DetCal3);
       END;


{ .. relative values of the green light BEFORE and AFTER the flow cell .. }
       IO_GreenRatio := (GEmean + GCal_Const * GOmean)*
                         poweri(10, VG_GRef)/ GTmean;   { .. uncalibrated! }
       IO_Ratio := poweri(10, 3 - VG_Gref)* IO_GreenRatio;
```

```
        IF (IO_Ratio < 0.66)OR(1.5 < IO_Ratio) THEN BEGIN
            WRITELN('   .  .  .  .  .  Calibration may NOT be correct !!');
            WRITELN('Green light intensities BEFORE and AFTER the flow ',
                                                'cell are TOO  disimilar.');
            WRITELN('Check (a) gain values given to the computer during ',
                                                'calibration procedure,');
            WRITELN('or (b) gain settings used for the power meters, or ',
                                                '(c) alignment of blue');
            WRITELN('beam detectors.');
            WRITE('                                ');
            WRITE('                          Is calibration  acceptable ? (Y/N) ');
            Read_input(YNset, key);
            IF (key = 'N')OR(key = 'n') THEN GOTO 150
            ELSE Add_Error(Errors, DetCal3);
          END;

        WITH Newtag, Birpar, Execute DO BEGIN
          IminG := GEmean* ORD(GEmean > 0)/5.0;  { .. convert to mWatts .. }
          ImaxG := (GCal_Const * GOmean + GEmean)/5.0;
          GExtRatio := ImaxG / IminG;
          WRITELN('IminG: ', IminG:10, ',     Ext. ratio = ', GExtRatio:6:0,
                             ',     Ratio  I/O G_Int: ', IO_GreenRatio:5:3)
         END;
        GreenFLAG := TRUE;
150:  UNTIL GreenFLAG;

      WITH ADCpar DO BEGIN
        WRITE('REMEMBER: to remove the (calibration) retarder from ');
        WRITELN('the optical rail.');
        WRITE('                  ');
        WRITE('                              When ready hit RETURN ...  Q = <esc> ');
        Read_Returnkey(kaputFLAG);
        IF kaputFLAG THEN BEGIN
          WRITE('                                ');
          WRITE('            RE-START calibration procedure ..  (Y/N)? ');
          Read_input(YNset, key);
          IF (key = 'Y')OR(key = 'y') THEN GOTO 200
          ELSE BEGIN
            Add_Error(Errors, DetCal4);
            OutFLAG := TRUE
           END
         END
        ELSE BEGIN
          ReltI_Blue := BCal_const;   ReltI_Green := GCal_const;
          NormI_Blue := IO_BlueRatio; NormI_Green := IO_GreenRatio;
          Instr.PhotoDetCALIB := TRUE;
          OutFLAG := TRUE
         END
       END;
    RECOVER BEGIN
        Instr.PhotoDetCALIB := FALSE;
        IF ESCAPECODE = -1 THEN BEGIN
            TwoBEEPS;
            WRITELN;
```

```
              WRITELN('Abnormal exit. Calibr. Ref_photodiode FAILED!');
              WRITE('          RTN to Restart full calibration.  Q = <esc> ');
              Read_Returnkey(kaputFLAG);
              IF kaputFLAG THEN BEGIN
                  OutFLAG := TRUE;
                  Add_Error(Errors, DetCal5)
                END
              ELSE OutFLAG := FALSE
            END
          ELSE
            IF ESCAPECODE = - 20 THEN BEGIN
                WRITE('Stopped by Operator...    ');
                WRITE('RTN to Restart.   Q = <esc> ');
                Read_Returnkey(kaputFLAG);
                IF kaputFLAG THEN OutFLAG := TRUE
              END
            ELSE ESCAPE(ESCAPECODE);
        END;
200: UNTIL OutFLAG;
     DISPOSE(channel);                                    { .. Sundries for .. }
     DISPOSE(pace);          DISPOSE (gain3);             {       . . . . .    }
     DISPOSE(pace3);                                      {       . . . . .    }
     DISPOSE(Rawdata1);      DISPOSE (Rawdata2);          { .. data reading .. }
     DISPOSE(ADC_Dptr);      DISPOSE(BufRawdata1);     DISPOSE(BufRawdata2);
     DISPOSE(Realdata);      DISPOSE(Realdata2);

     DISPOSE(BintE);         DISPOSE(BintED);             { . . . . . . . . }
     DISPOSE(BintO);         DISPOSE(BintOD);             { . . Data arrays .. }
     DISPOSE(GintE);         DISPOSE(GintED);             {       . . for . .  }
     DISPOSE(GintO);         DISPOSE(GintOD);             {       . . data . . }
     DISPOSE(BrefMean);      DISPOSE(BrefDelta);          { ..  calculations.  }
     DISPOSE(GrefMean);      DISPOSE(GrefDelta)           { . . . . . . . .    }
    END
  ELSE Instr.PhotoDetCALIB := FALSE;
END;        { - - - - - - - - - - - - - - - - - - End of PROCEDURE DB_DetectorCal.}


PROCEDURE Photo_Cal(VAR Newtag: EXPDATREC; VAR ADCpar: ADCparREC);
{-------------------------------------------------------------------------
This procedure calibrates the Photo Detectors at zero light intensity.
                                                    MHG          28/June/96
 -------------------------------------------------------------------------}
LABEL  100;

CONST   Mainframe = 709;
        datSize = 30000;                {Max. # of available data in the buffer}
        Size = 8;                       {# of available channels}


TYPE ChanARRAY     = ARRAY[1..Size] OF INTEGER;
     DatARRAY      = ARRAY[1..datSize] OF REAL;
     meanARRAY     = ARRAY[1..Size] OF REAL;
```

```
CONST    ChanStrConst    = ChanStrARRAY[Size OF '25'];
         ConstChanArr    = ChanARRAY[Size OF 25];
         ConstDatARRAY   = DatARRAY[datSize OF 37.0E+00];
         meanCONST       = meanARRAY[Size OF 37.00E+00];

 VAR Numpoints, pos, Byte1, Conv2,
     Dataindx, f_range, Byte2, Sign,
     numdetec, Voltindx, i              : INTEGER;
     MaxRange, mean, sigma, time        : REAL;
     ESCFLAG, ReptFLAG                  : BOOLEAN;
     Unconverted1, Unconverted2         : CHAR;
     buffer                             : BUF_INFO_TYPE;
     ChGain                             : ARRAY[1..8] OF CHAR; {Gain array in
                                                                char from 1-4}
     ChannSTR, RangeSTR                 : ChanStrARRAY;{Arrays that export the
                                                        channels and ranges to
                                                        be used in the read_
                                                        multiple procedure.
                                                        They have to be STRINGS}
     ChannI                             : ChanARRAY;   {Array that keep the
                                                        channels to be used in
                                                        form of INTEGER.}
     Volt, Data                         : DatARRAY;    {Arrays that keeps the
                                                        converted data}
     meanDAT                            : meanARRAY;   {Array that keeps the
                                                        mean for every detector}
     ChNum                              : BooleanARRAY;{Array thah tells wich
                                                        channel is active}


 BEGIN
   ChannSTR:=ChanStrConst;
   Data:=ConstDatARRAY;
   Volt:=ConstDatARRAY;
   meanDAT := meanCONST;
   Numpoints:=1500;
   numdetec:=8;
   time:=0.00001;
   ESCFLAG:=FALSE;
   Voltindx:=1;
   FOR i:=1 TO numdetec DO BEGIN
     ChannI[i]:=500+i-1;
     STRWRITE(ChannSTR[i], 1, pos, ChannI[i]);
     SETSTRLEN(ChannSTR[i], pos-1);
     RangeSTR[i]:='0.04';
     ChGain[i]:='1';
     ChNum[i]:=TRUE;
    END;
   IOBUFFER(buffer,60000);
   BUFFER_RESET(buffer);
   WRITELN;
   WRITE('    CALIBRATION OF LIGHT DETECTORS AT ZERO LIGHT FOR ');
   WRITELN('TWO-COLOR DICHROISM & BIREFRINGENCE.');
```

```
  WRITELN;
  WRITELN('In order to proceed, The laser beam must be blocked ');
  WRITELN;
  WRITELN(' When ready, Hit the RETURN key... Q = <esc> ');
  Read_Returnkey(ESCFLAG);
  IF ESCFLAG THEN BEGIN
    GOTO 100;
   END;
  WRITELN('Please wait!.............');
  TRY
    read_multiple(Numpoints, numdetec, time, ChannSTR, RangeSTR, ChNum);
    MHGMeasure_Scan(buffer);
    WHILE BUFFER_DATA(buffer) >0 DO BEGIN
      READBUFFER(buffer, Unconverted2);
      Conv2:=ORD(Unconverted2);
      READBUFFER(buffer, Unconverted1);
      Byte1:=ORD(Unconverted1);
      MHG_BintoReal(Conv2, Byte1, 4, Volt[Voltindx]);
      Voltindx:=Voltindx+1;
     END;
 RECOVER BEGIN
    IF ESCAPECODE = IOESCAPECODE THEN
    WRITELN (IOERROR_MESSAGE(IOR_RESULT));
    ESCAPE(ESCAPECODE);
  END;
 FOR i:=1 TO numdetec DO BEGIN              {Putting data in small arrays that}
     Dataindx:=1;                           {contain data from every single}
     Voltindx:=i;                           {detector in order to calculate}
     REPEAT                                 {their mean and deviation}
       Data[Dataindx]:=Volt[Voltindx];      {.........................}
       Voltindx:=Voltindx+numdetec;         {.........................}
       Dataindx:=Dataindx+1;                {.........................}
     UNTIL Dataindx = Numpoints + 1;        {.........................}
     LSQR_fit(Data, Numpoints, meanDAT[i], sigma);
     IF (mean>=-100) THEN BEGIN
       TwoBEEPS;    WRITELN;
       WRITELN('Error in you Range Option! Data is not valid');
       WRITELN;
      END;
  END;
 WITH ADCpar DO BEGIN
   NoLightBRef:=meanDAT[1];
   NoLightBOrd:=meanDAT[2];
   NoLightBExt:=meanDAT[3];
   NoLightGRef:=meanDAT[4];
   NoLightGOrd:=meanDAT[5];
   NoLightGExt:=meanDAT[6];
  END;
WRITELN;
WRITE('In order to proceed, The blocking beam ');
WRITELN('mechanism must be removed ');
WRITELN;
WRITELN(' When ready, Hit the RETURN key... Q = <esc> ');
Read_Returnkey(ESCFLAG);
```

```
   IF ESCFLAG THEN BEGIN
     GOTO 100;
    END;

  100:END;        { - - - - - - - - - - - - - - - - - End of PROCEDURE Photo_Cal.}

PROCEDURE Set_DBirRept(ExpCond: BExpCondREC; lbnumber: NATURAL;
                  VAR Bparinfo: Parinforec; VAR Newlabel: EXPDATREC;
                     VAR InstrFLAG: INSTRFLAGREC; VAR Errorlist: ErrLIST);
{-------------------------------------------------------------------------
 This procedure sets up the instruments for repeated execution of birefrin-
 gence. For the first run, the calibration as well as initialization of
 intruments takes place. For all other subsequent runs, only the ADC card,
 laser and stepping motor are reset to the required initial values.
                                                        EGA Rev 1.1, 11/Apr/89.
 This revision allows the operator to perform calibration of detectors by
 all possible configurations by setting the appropriate flags at the begin-
 ning of the experiments.                               EGA Rev 1.2,  4/Jul/90.
 The the Calibration and Double_Strain and Oscillatory Procedures for the
 SMI were added.                                        MHG          25/June/96.
 -------------------------------------------------------------------------}

  VAR   Exprm           : EXPKIND;
        EndFLAG         : BOOLEAN;
        ADCpar          : ADCparREC;
        YNkey           : CHAR;

  BEGIN
    WITH Errorlist DO BEGIN {.. initialize this experiment list of errors.. }
       first := nil;   last := nil
      END;
    Exprm := Bir;
    WITH Bparinfo, ExpCond DO BEGIN
      Laser_PWR(AutoFLAG,Laser_I,Errorlist,InstrFLAG);      { .. laser set ..}
      InstrFLAG.LASERsetFLAG := TRUE;
      CASE Exp OF      { .. load stepping motor's program for experiment ..}
          SB : SMI_Steady_State(AutoFLAG, Exprm,
                                Errorlist, SMIpar, InstrFLAG);
          RB : SMI_Run_halt(AutoFLAG, Exprm,
                                Errorlist, SMIpar, InstrFLAG);
          LB : SMI_Low_high(AutoFLAG, Exprm,
                                Errorlist, SMIpar, InstrFLAG);
          TB : SMI_Strain_Step(AutoFLAG, Exprm,
                                Errorlist, SMIpar, InstrFLAG);
          OB : SMI_Oscillatory(AutoFLAG, Exprm,
                                Errorlist, SMIpar, InstrFLAG);
          otherwise ;
        END;
       IF NOT(InstrFLAG.ConvsetFLAG) THEN BEGIN
           ADC_ON(InstrFLAG, Errorlist);
         END;
```

```
      IF NOT(InstrFLAG.PhotoDetCALIB) THEN BEGIN
        Photo_Cal(Newlabel, ADCpar);
        DB_DetectorCal(AutoFLAG, ADCpar, Newlabel, Errorlist, InstrFLAG);
       END;
                          { .. calibration of light detectors is done !!  ... }
    END;
   Update_label(Bparinfo, lbnumber, Errorlist, Newlabel);
 END;              { - - - - - - - - - - - - - - - - End of PROCEDURE Set_BirRept.}


PROCEDURE DB_ReltInt(Rawdata: ARRAY[ xmin..xmax: NATURALS] OF REAL;
                     InstrGain : GainREC;
                     VAR Normdata: ARRAY[ymin..ymax: NATURALS;
                     zmin..zmax: NATURALS] OF REAL; ADpar: ADCparREC);
{-------------------------------------------------------------------------
 This procedure calculates the RELATIVE intensities of the extraordinary and
 ordinary beams based on the incident intensity falling upon the flow device.
 The normalizing intensity is measured by the reference detector. Thus, di--
 chroism and birefringence can be evaluated simultaneously, since the total
 intensity past the flow cell is calculated as the sum of the extraord. and
 ord. light intensities. This procedure requires three detectors per color
 in order to calculate dichroism and birefringence and discriminate for the
 dust scattering and temporal fluctuation of laser output.
                                          EGA     Rev.  1.0,      23/March/90.
 -------------------------------------------------------------------------}

 VAR     expBE, expGE, expBO, expGO    : REAL;
         k                             : NATURAL;

 BEGIN
   WITH InstrGain, ADpar DO BEGIN
        expBE := poweri(10, VoltG_Bref - VoltG_BExt);
        expGE := poweri(10, VoltG_Gref - VoltG_GExt);
        expBO := poweri(10, VoltG_Bref - VoltG_BOrd);
        expGO := poweri(10, VoltG_Gref - VoltG_GOrd);
        FOR k := 1 TO Numsamples DO BEGIN
            Normdata[1, k] := Rawdata[6*k-5]*expBE
            Normdata[2, k] := (30/4095) * Normdata[1, k];
            Normdata[3, k] := Rawdata[6*k-4]*expGE
                              /(Rawdata[6*k]*NormI_Green);
            Normdata[4, k] := (30/4095) * Normdata[3, k];
            Normdata[5, k] := Rawdata[6*k-3]*ReltI_Blue
                              *expBO/ (Rawdata[6*k-1]*NormI_Blue);
            Normdata[6, k] := (30/4095) * Normdata[5, k];
            Normdata[7, k] := Rawdata[6*k-2]*ReltI_Green
                              *expGO/ (Rawdata[6*k]*NormI_Green);
            Normdata[8, k] := (30/4095) * Normdata[7, k]
          END
     END
 END;          { - - - - - - - - - - - - - - - - - - - - End of PROCEDURE DB_ReltInt.}
```

```
PROCEDURE DB_AvrgRawDat(Normdata: ARRAY[xmin..xmax:
                        NATURALS; ymin..ymax: NATURALS] OF REAL;
                                  VAR DB_DStat: DStat_Array; count: NATURALS);
{-------------------------------------------------------------------------
 This procedure averages the rawdata values as obtained by the ADC card for
 the same experimental conditions during each run of the steady state bire--
 fringence experiment. The generated data values are subsequently used for
 calculating the birefringence and dichroism of the polymeric fluid. These
 averaged values are also stored permanently in the hard disk. Rawdata is a
 conformant array. LSQR_fit and Covariance also require conformant arrays.
 Dxi contains the resolution of the ADC card.    EGA    Rev. 1.0    28/March/90.
 -------------------------------------------------------------------------}
 VAR      mean, sigma, sigmean     : REAL;

 BEGIN
   fit_LSQR(Normdata[1], Normdata[2], count, mean, sigmean, sigma);
   DB_DStat[2] := mean; DB_DStat[6] := sigmean;      { mean and S dev for BE.}
   fit_LSQR(Normdata[3], Normdata[4], count, mean, sigmean, sigma);
   DB_DStat[3] := mean; DB_DStat[7] := sigmean;      { mean and S dev for GE.}
   fit_LSQR(Normdata[5], Normdata[6], count, mean, sigmean, sigma);
   DB_DStat[4] := mean; DB_DStat[8] := sigmean;      { mean and S dev for BO.}
   fit_LSQR(Normdata[7], Normdata[8], count, mean, sigmean, sigma);
   DB_DStat[5] := mean; DB_DStat[9] := sigmean;      { mean and S dev for GO.}
   DB_DStat[10] := Covar(Normdata[1], Normdata[3], count);     { covariances }
   DB_DStat[11] := Covar(Normdata[1], Normdata[5], count);     {     for     }
   DB_DStat[12] := Covar(Normdata[1], Normdata[7], count);     {     all     }
   DB_DStat[13] := Covar(Normdata[3], Normdata[5], count);     {    light    }
   DB_DStat[14] := Covar(Normdata[3], Normdata[7], count);     {  detectors  }
   DB_DStat[15] := Covar(Normdata[5], Normdata[7], count)
 END;       { - - - - - - - - - - - - - - - - - - End of Procedure DB_AvrgRawDat.}


PROCEDURE DB_P_SS(par1: SMIparREC; par2: ADCparREC; SMItrigger: BOOLEAN;
                        VAR BExpCond: BExpCondREC; exp_IDnum: NATURAL;
                       VAR Expdat_LIST: EXPLIST; VAR ErrorLT: ErrLIST);
{-------------------------------------------------------------------------
 This subroutine corresponds to the DB_ProcB  portion associated to the ex-
 periments of steady state flow induced optical anisotropy. This procedure
 uses the cyclic timer of SYSDEVS. The interrupt handler SSINTR_HDL, works
 as a dispatcher which decides whether to increment the SMI speed or take
 the ADC measurement. For real-time computers see the book by Allworth,
 "Introduction to Real-Time Software Design."          EGA Rev. 2.0, 28/Jan/87.
 All the procedures related with the aquisition were updated.
                                                       MHG        11/96.
 -------------------------------------------------------------------------}

    CONST    d_size          = 6000;
             pt_size         = 1000;

    TYPE     r_ptr           = ^CALCarray;
             BIGarray        = ARRAY[1..8, 1..pt_size] OF REAL;
```

```
          BIG_ptr          = ^BIGarray;
          SSd_array        = ARRAY[1..d_size] OF REAL;
          SSd_ptr          = ^SSd_array;
          BufTYPE          = BUF_INFO_TYPE;
          BufD_ptr         = ^BufTYPE;

          BufD_nodeptr     = ^BufD_nodetype;
          BufD_nodetype    = RECORD
                Bufdata    : BufTYPE;
                Gaindata   : GainREC;
                Next       : BufD_nodeptr;
              END;

          BufD_queue       = RECORD
                 front     : BufD_nodeptr;
                 rear      : BufD_nodeptr;
              END;

CONST     readintrmask     = 4;       { .. sundries associated to interrupts .. }
          START      = 1;     p_size = 6;          { .. and these sundries...  }
          STOP       = 6;     g_size = 1;
          c_size     = 6;
          Tpace      = 0.00005;

TYPE      i_array          = ARRAY[1..6] OF SHORTINT;
          i_ptr            = ^i_array;

VAR       ADC_Dptr                                    : BufD_ptr;
          Char_Dptr                                   : BufD_ptr;

          Norm_Dptr                                   : BIG_ptr;
          RAW_Vptr, Real_Vptr                         : SSd_ptr;
          datqueue                                    : BufD_queue;
          Stor_values                                 : DATAarray;
          meanvalues                                  : DStat_Array;
          Calcpoint                                   : CALCarray;
          Success, ExpDONE, FLAG1,
          TrigFLAG, Incr_Cyc, CYCLICFLAG, STEPFLAG    : BOOLEAN;
          CYCLICCOUNT, STEPCOUNT, STEPEX, COUNT,
          points, i, j, REPT, centisec                : NATURALS;
          SScount, pos, Error, RAWindx                : INTEGER;
          SAVEOLDMASK                                 : BYTE;
          TDATA                                       : TIMERDATA;
          SAVEISRHOOK                                 : KBDHOOKTYPE;
          Sampletime, pace                            : REAL;
          Plotlimits                                  : BirExp_Range;
          PlottingLT                                  : PBirLIST;
          PlotCoord                                   : PBir_data;
          GWorld, BWorld                              : WorldCoord;
          GreenData, BlueData                         : Plot_dataREC;
          GLInt_LT, BLInt_LT                          : PlotDataLIST;
          CopyRaster                                  : mcopyptr;
          dummy                                       : ARRAY[1..2] OF INTEGER;
```

```
            strng                                   : STRING[255];
            buffer                                  : BUF_INFO_TYPE;
            GainRECIn, GainRECOut                   : GainREC;
            Bufindx, Conv2, Byte1                   : INTEGER;
            Unconverted1, Unconverted2              : CHAR;


 FUNCTION Dat_Empqueue(q: BufD_queue): BOOLEAN;
   BEGIN
     IF q.front = nil THEN
       Dat_Empqueue := TRUE
     ELSE Dat_Empqueue := FALSE
   END;          { - - - - - - - - - - - - - End of Function Dat_Empqueue.BirP_BSS}

 PROCEDURE Read_dpoint(VAR convdata: BufD_ptr; VAR InstrGain: GainREC;
                       VAR q: BufD_queue);
 {-------------------------------------------------------------------------
  This sub-procedure uses a queue to temporarily store the ADC card values
  before they are processed.
  -------------------------------------------------------------------------}

   VAR  p: BufD_nodeptr;
        LastChar : CHAR;


   BEGIN
     IF Dat_Empqueue(q) THEN WRITEBUFFER(convdata^, CHR(255))
     ELSE BEGIN
          p := q.front;
          WITH p^ DO BEGIN
              convdata^ := Bufdata;
              InstrGain := Gaindata;
              q.front := Next;
              IF q.front = nil THEN
                q.rear := nil
            END
       END;
     DISPOSE(p)
   END;          { - - - - - - - - - - - - - End of PROCEDURE BirP_BSS.Read_dpoint.}


PROCEDURE Save_dpoint(convdata: BufD_ptr; InstrGain: GainREC;
                      VAR q:BufD_queue );
{-------------------------------------------------------------------------
 This sub-procedure retrieves for processing the ADC card readings store
 in the nonpermanent queue.
 -------------------------------------------------------------------------}

  VAR    p: BufD_nodeptr;

  BEGIN
    NEW(p);
    WITH p^ DO BEGIN
        p^.Next := nil;
```

```
        p^.Bufdata := convdata^;
        p^.Gaindata := InstrGain;
      END;
    WITH q DO BEGIN
        IF rear = nil THEN
          front := p
        ELSE rear^.Next := p;
        rear := p
      END
  END;          { - - - - - - - - - - - - - End of PROCEDURE BirP_BSS.Save_dpoint.}


PROCEDURE SSINTR_HDL(VAR STATBYTE, DATABYTE: BYTE; VAR DOIT: BOOLEAN);
{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 This sub-procedure of BirP_BSS takes over the CPU when an interrup occurs.
 It is in charge of doing the data acquisition thru the ADC and triggering
 of the SMI to generate the step ramp.
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }

 VAR    i : NATURAL;

 BEGIN
   IF (ODD(STATBYTE DIV 32) AND ODD(DATABYTE DIV 32)) THEN BEGIN
       IF Incr_Cyc THEN BEGIN
         CYCLICCOUNT := CYCLICCOUNT + 1;
         CYCLICFLAG:=TRUE;
        END;
       IF (CYCLICCOUNT MOD 10 = 1) THEN BEGIN
         TRIGGER(SMI_isc);                { trigger SMI. Speed incremented! }
       END;
       IF (CYCLICCOUNT MOD 10 = 0) THEN BEGIN{.. Data acquisition begins.. }
         TRY
           Incr_Cyc := FALSE;
           MHGMeasure_Scan(ADC_Dptr^);
           Incr_Cyc := TRUE;
         RECOVER BEGIN
             IF (ESCAPECODE = IOESCAPECODE) AND (IOE_ISC = 18) THEN
               Add_Error(ErrorLT, ADC3)
             ELSE IF (ESCAPECODE = -1) THEN
                    Add_Error(ErrorLT, ADC4)
                  ELSE BEGIN
                  ESCAPE(ESCAPECODE);
                  END;
             Incr_Cyc := TRUE
           END;
         Save_dpoint(ADC_Dptr, GainRECIn, datqueue);
                                              {ADC readings stored in queue.}
         STEPCOUNT := STEPCOUNT + 1;
         STEPFLAG := TRUE;
        END
    END
  ELSE CALL(SAVEISRHOOK,STATBYTE, DATABYTE, DOIT);
END;          {- - - - - - - - - - - - - End of Procedure BirP_BSS.SSINTR_HDL.}
```

```
BEGIN          {--------------------------- Procedure BirP_BSS begins ....  }
  WITH BExpCond.GContrlFLAGS DO BEGIN {.. Cntlflags required by graphics ..}
     FrameDrawn := FALSE;     GphOutRange := FALSE;
     Scale1 := 0;  Scale2 := 0;
    END;
  IF BExpCond.PlotBir THEN BEGIN  {parameters needed for Bir data plotting.}
    WITH Plotlimits, Par1 DO BEGIN
        PXmin := minspeed;  PXmax := maxspeed; {.. These are the initial.. }
        PBirmin := 0.0;    PBirmax := 0.0;     {..     plotting limits...  }
        PChimin := 0.0;    PChimax := 45.0     {.. for the birefringence.  }
      END;
    WITH PlottingLT DO BEGIN          {.. Plot list of bir and Chi points .. }
        first := nil;     last := nil
      END
   END
  ELSE BEGIN          {.. parameters needed for B & G light data plotting.. }
    WITH BWorld, Par1 DO BEGIN
      WX1 := minspeed;  WX2 := maxspeed;
      WY1 := 0.0;  WY2 := 0.0
     END;
    WITH GWorld, Par1 DO BEGIN
      WX1 := minspeed;  WX2 := maxspeed;
      WY1 := 0.0;  WY2 := 0.0
     END;
    WITH BLInt_LT DO BEGIN           {.. list of BL points to be plotted .. }
        first := nil;     last := nil
      END;
    WITH GLInt_LT DO BEGIN           {.. list of GL points to be plotted .. }
        first := nil;     last := nil
      END
  END;
 WITH datqueue DO BEGIN
     front := nil;     rear := nil
   END;

 WITH GainRECIn DO BEGIN                        {.. TEMPORAY values ..}
   VoltG_Bref := 4;                             {.. They have to be removed ..}
   VoltG_Gref := 4;                             {..............................}
   VoltG_Bord := 4;                             {..............................}
   VoltG_Bext := 1;                             {..............................}
   VoltG_Gord := 4;                             {..............................}
   VoltG_Gext := 1;                             {..............................}
 END;

CYCLICFLAG:=FALSE;
STEPFLAG:=FALSE;
pace := par2.ADCscan_time/6;
REPT := par2.Numsamples;
NEW(ADC_Dptr);     NEW(RAW_Vptr);  NEW(Norm_Dptr);    NEW(Char_Dptr);
NEW(Real_Vptr);
```

```
Set_Voltmeter(REPT, 10.24, pace);
IOBUFFER(ADC_Dptr^,60000);
BUFFER_RESET(ADC_Dptr^);
IOBUFFER(Char_Dptr^,60000);
BUFFER_RESET(Char_Dptr^);
SAVEISRHOOK := TIMERISRHOOK;                    {save timer's hook of system}
TIMERISRHOOK := SSINTR_HDL;                      {timer's interrupt handler}
centisec := ROUND((par1.ttop/10)*100.0);
                        {... The cyclic timer requires time in centisec.. }
cmd_read_1(readintrmask, SAVEOLDMASK);    {.. recalls interrupt status... }
TRY
  FOR i := 1 TO par1.runs DO BEGIN
    IF i >= par1.runs THEN ExpDONE := TRUE
    ELSE ExpDONE := FALSE;
    BExpCond.DrawnPoints := 0;
    BExpCond.Numpoints := par1.numsteps;
    TDATA.count := centisec;
    call(timeriohook, CYCLICT, SETT, TDATA);        { .. set cyclic timer.}
    SMIGATO1;                          {.. Start Execution of SMI program .. }
    Incr_Cyc := TRUE;
    CYCLICCOUNT := 0;  STEPCOUNT := 0;  points := 0;
    COUNT:=0;
    Stor_values[1] := -3;              Stor_values[2] := exp_IDnum;
    Stor_values[3] := par1.numsteps;   Stor_values[4] := i;
    Stor_values[5] := par1.runs;              { .. little tag in data list. }
    Store_dataLT(Expdat_LIST, Stor_values);
    IF ODD(SAVEOLDMASK DIV 4) THEN
      call(MASKOPSHOOK, TIMERMASK, 0);       { .. Cyclic timer enable!... }
    REPEAT
      IF STEPFLAG THEN BEGIN
        WRITELN('This is STEPEX: ', STEPEX);
        STEPFLAG:=FALSE;
       END;
      IF CYCLICFLAG THEN BEGIN
        WRITELN('This is CYCLICCOUNT: ', CYCLICCOUNT);
        CYCLICFLAG:=FALSE;
       END;
      STEPEX := STEPCOUNT;
      IF STEPEX >= par1.numsteps THEN BEGIN       { .. SMI ramping done.. }
          IF ODD(SAVEOLDMASK DIV 4) THEN
              call(MASKOPSHOOK, 0, TIMERMASK);       { .. disarm timer... }
          TDATA.count := 0;
          call(timeriohook, CYCLICT, SETT, TDATA);       { .. reset timer.}
          SMI_SPOLL(FLAG1, TrigFLAG);
          IF TrigFLAG THEN TRIGGER(SMI_isc);
          SMIWE;                                 { .. and Denergize SMI windings.}
        END;
      IF (STEPEX <= points) THEN BEGIN
          FOR j :=1 TO 1000 DO ; { .. kill some CPU-time while waiting.. }
        END
      ELSE BEGIN                          { ...  otherwise process rawdata..   }
        RAWindx:=1;
        Read_dpoint(Char_Dptr, GainRECOut, datqueue);
```

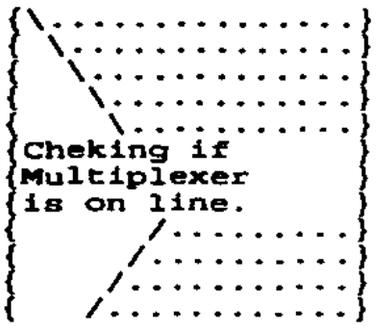
```
        points := points + 1;
        WHILE BUFFER_DATA(Char_Dptr^) >0 DO BEGIN
          READBUFFER(Char_Dptr^, Unconverted2);
          Conv2:=ORD(Unconverted2);
          READBUFFER(Char_Dptr^, Unconverted1);
          Byte1:=ORD(Unconverted1);
          MHG_BintoReal(Conv2, Byte1, 4,RAW_Vptr^[RAWindx]);
          RAWindx:=RAWindx+1;
         END;
        Voltage_Adjust(4, REPT, RAW_Vptr^, Real_Vptr^);
        IF (ABS(13.0 + RAW_Vptr^[1]) < mintol) THEN BEGIN
            meanvalues := invalid15;                    { .. invalid data..}
            Calcpoint := invalid4;                      { .. set to -17.0 .}
            Add_Error(ErrorLT, RawDat1);
          END
        ELSE BEGIN
          DB_ReltInt(Real_Vptr^, GainRECOut, Norm_Dptr^, par2);
          DB_AvrgRawDat(Norm_Dptr^, meanvalues, REPT);
         END;
       END;
    UNTIL points >= par1.numsteps;
    IF BExpCond.GContr1FLAGS.GphOutRange THEN Add_Error(ErrorLT, Graph4);
   END;
  ESCAPE(0);                  { .. Warrants timer reseting to system values.}
RECOVER BEGIN
  IF ODD(SAVEOLDMASK DIV 4) THEN
      call(MASKOPSHOOK, 0, TIMERMASK);                  {..  disarm timer... }
  TDATA.count := 0;                                     {..  and reset it... }
  call(timeriohook, CYCLICT, SETT, TDATA);              {.. to default state.}
  TIMERISRHOOK:=SAVEISRHOOK; {.. Set hook to known state (system's hook.)}

  DISPOSE(ADC_Dptr);                  { .. clean dynamic variables' heap ... }
  DISPOSE(RAW_Vptr);  DISPOSE(Norm_Dptr);   DISPOSE(Char_Dptr);
  DISPOSE(Real_Vptr);

  REPEAT                              { .. & Clean stepping motor program !! }
    Success := TRUE;
    SMI_SPOLL(FLAG1, TrigFLAG);
    IF TrigFLAG THEN BEGIN
        TRIGGER(SMI_isc);    { .. finish with the step motor ramp .. }
        Success := FALSE
      END
  UNTIL Success;
  SMIWE;                                    { .. and denergize SMI windings.}

  IF ESCAPECODE = 0 THEN              { .. NORMAL termination of procedure! }
    IF (ExpDONE)AND(STEPEX >= par1.numsteps) THEN BEGIN
      SMIDA500B404200;{start cooling of fluid .. }
      IF par1.maxspeed > 2000 THEN          { ....    by running the    ... }
        SMIKR       { .. rollers at high speed, .. }
      ELSE SMILR;   { ...   or mininum speed    ... }

      IF BExpCond.SavInf THEN BEGIN {.. dump the plot into the printer ..}
         WRITESTRINGLN(701, #10 + #10 + #10);
```

```
                OUTPUT_ESC(52, 0, 0, dummy, dummy, Error);
                WRITESTRINGLN(701, #12)
              END;
            IF BExpCond.GraphicsOn THEN BEGIN { & reset monitor to alpha mode. }
              DISPLAY_TERM;
              GRAPHICS_TERM;
              BExpCond.GraphicsOn := FALSE
            END
          END
        ELSE BEGIN
          IF BExpCond.GraphicsOn THEN BEGIN  { & reset monitor to alpha mode.. }
              DISPLAY_TERM;
              GRAPHICS_TERM;
              BExpCond.GraphicsOn := FALSE
            END;
          WRITELN('Error position Count = ');
          IF ESCAPECODE = IOESCAPECODE THEN
            WRITELN('I/O Error.... BirP_BSS aborted! ')
          ELSE
            WRITELN('Error(',ESCAPECODE:0,') occured.. BirP_BSS aborted! ');
          ESCAPE(ESCAPECODE)
        END
      END
    END;                { - - - - - - - - - - - - - - - - - - End of PROCEDURE DB_P_SS.}

  PROCEDURE DB_ProcB(Newtag: EXPDATREC; InstrFLAG: INSTRFLAGREC; VAR
                             BExpCond: BExpCondREC; exp_IDnum: NATURAL;
                             VAR DatLIST: EXPLIST; VAR Errors: ErrLIST);
  {-------------------------------------------------------------------------
   This procedure is in charge of syncronization and data acquisition for
   the experiment of flow-birefringence. Expdata contains the raw data once
   the averaging has taken place.
   -------------------------------------------------------------------------}

   BEGIN
     WITH Newtag DO
         CASE Exp OF
             SB: DB_P_SS(Birpar.Motor, Birpar.ADC,
                 InstrFLAG.SMItrigFLAG, BExpCond, exp_IDnum, DatLIST, Errors);
             RB: DB_P_RH(Birpar.Motor, Birpar.ADC,
                 InstrFLAG.SMItrigFLAG, BExpCond, exp_IDnum, DatLIST, Errors);
             LB: DB_P_LH(Birpar.Motor, Birpar.ADC,
                 InstrFLAG.SMItrigFLAG, BExpCond, exp_IDnum, DatLIST, Errors);
             TB: DB_P_Strain;
          END

   END;              { - - - - - - - - - - - - - - - - - - End of PROCEDURE BirProB.}

END;              { - - - - - - - - - - End of Bir_library MODULE implementation. }

  END.                    {------------------------------ End of EXPLIB library. }
```

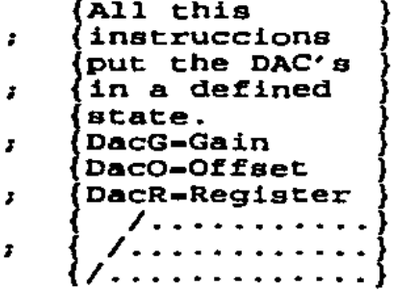
Apéndice B. Código del Módulo AD_Converter

```
$SYSPROG ON$
$HEAP_DISPOSE ON$
$REF 50$
MODULE AD_Converter;
{---------------------------------------- ADC MODULE --------------------
 This is the module for the utilization of the analog to digital converter.

 ------------------------------------------------------------------------}
$SEARCH '@11:/PASCAL_F/MHG/MHGLIB.' $

IMPORT IODECLARATIONS, SYSDEVS, General_0, General_1, General_2,
       General_3, General_4, HPIB_0, HPIB_1, HPIB_2, HPIB_3, Math_functions,
       UserProc, FileOper;

EXPORT CONST Mainframe = 709;                          {HPIB address for 3852A}
             GPIO      = 12;                            {I/O address for GPIO}
             Size      = 8;              {Up to this number of light detectors}
             PRINTER   = 13;                    {I/O address for Parallel Port}

       TYPE ChanStrARRAY = ARRAY[1..Size] OF STRING[16];
            BooleanARRAY = ARRAY[1..Size] OF BOOLEAN;

       PROCEDURE ADC_ON(VAR Instrum: INSTRFLAGREC; VAR Errorlist: ErrLIST);
       PROCEDURE OFF_ADConv(VAR Instrum: INSTRFLAGREC);
       PROCEDURE Set_Voltmeter(Numpoints : NATURAL; Gain, time : REAL);
       PROCEDURE Set_Voltmeter16(Numpoints : NATURAL; Gain, time : REAL);
       PROCEDURE MHGMeasure_Scan(VAR buffer:BUF_INFO_TYPE);
       PROCEDURE MHG_BintoReal(Conv2, Byte1:INTEGER; Gain :INTEGER;
                               VAR Rawdata:REAL);
       PROCEDURE read_volt(Numpoints, Chann : INTEGER; time, Gain : REAL);
       PROCEDURE read_multiple(Numpoints, m : INTEGER; time : REAL;
                         Chann, RangeSTR : ChanStrARRAY; ChNum : BooleanARRAY);
       PROCEDURE AD_Conv;

IMPLEMENT

PROCEDURE Check_MPX(VAR Instrum: INSTRFLAGREC; VAR Errorlist: ErrLIST);
{-------------------------------------------------------------------------
 This Procedure checks if the multiplexer is on slot 5 and generates an
 error if it is not there.                                      MHG 8/96.
-------------------------------------------------------------------------}
VAR MPXiden    :REAL;


BEGIN
  MPXiden:=0;
```

```
   WRITESTRINGLN(Mainframe, 'ID? 500');
   READNUMBER(Mainframe, MPXiden);
   IF MPXiden=44711 THEN
     Instrum.MPXsetFLAG :=TRUE
   ELSE BEGIN
     Instrum.MPXsetFLAG := FALSE;
     WRITE('Multiplexer is NOT located');
     WRITELN(' next to the 16 bits DAC!');
     WRITE('Multiplexer set-up');
     WRITELN(' procedure has FAILED!');
     WRITELN;
     Add_Error(Errorlist, MPXoff);
    END;
 END;
```




```
PROCEDURE Check_OpAmp(VAR Instrum: INSTRFLAGREC; VAR Errorlist: ErrLIST);
{-------------------------------------------------------------------------
 This Procedure checks if the detectors card is on slot 3 and generates an
 error if it is not there. If the card there this Procedure gives initial
 values to the amplificators DAC's.                          MHG 8/96.
-------------------------------------------------------------------------}
VAR OpAmpiden    :REAL;
    DacR, DacG1, DacG2, DacG3, DacG4, DacG5, DacG6,
    DacO1, DacO2, DacO3, DacO4, DacO5, DacO6                : STRING[10];

BEGIN
  OpAmpiden:=0;
  DacR:='2'; DacG1:='3'; DacG2:='4'; DacG3:='5'; DacG4:='6'; DacG5:='7';
  DacG6:='8'; DacO1:='9'; DacO2:='10'; DacO3:='11'; DacO4:='12';
  DacO5:='13'; DacO6:='14';
  WRITESTRINGLN (Mainframe, 'ID? 300');                    {Cheking if       }
  READNUMBER (Mainframe, OpAmpiden);                       {Detectors module}
  IF OpAmpiden=44736 THEN BEGIN                            {is on line.      }
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacR+',0');
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacG1+',13330');
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacO1+',-25');
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacG2+',13330');
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacO2+',-25');
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacG3+',13330');
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacO3+',-25');
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacG4+',13330');
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacO4+',-25');
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacG5+',13330');
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacO5+',-25');
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacG6+',13330');
    WRITESTRINGLN(Mainframe,'SWRITE 300,'+DacO6+',-25');
    Instrum.OpAmpsetFLAG := TRUE;
   END
  ELSE BEGIN
    Instrum.OpAmpsetFLAG := FALSE;
    WRITELN('Detectors module is NOT on line');
    WRITE('Detectors module set_up');
```



```
    WRITELN(' procedure has FAILED!');
    WRITELN;
    Add_Error(Errorlist, OpAmpoff);
   END;
 END;


PROCEDURE Check_Voltmeter(VAR Instrum: INSTRFLAGREC; VAR Errorlist: ErrLIST);
{-------------------------------------------------------------------------
 This Procedure checks if the voltmeter is on slot 6 and generates an
 error if it is not there. If the voltmeter is there, this Procedure makes
 a self test of the voltmeter.                                  MHG 8/96.
-------------------------------------------------------------------------}
CONST         NumRead = 20;                    {Number of readings to take
                                                in the calibration process.}

TYPE          VoltARRAY = ARRAY[1..NumRead] OF REAL;


CONST         ConstArray = VoltARRAY[NumRead OF 37.0E+00];


VAR ADCiden, errornum, mean, sigma    : REAL;
    shortcircuit                      : STRING[3];
    pos, i                            : INTEGER;
    Voltage                           : VoltARRAY;
    Numpoints                         : STRING[16];



BEGIN
  ADCiden:=0;    errornum:=32;    mean:=0;         sigma:=0;
  shortcircuit:='507';
  Voltage:=ConstArray;
  WRITESTRINGLN (Mainframe, 'ID? 600');                    {Cheking if      }
  READNUMBER (Mainframe, ADCiden);                         {Voltmeter       }
  IF ADCiden=44704 THEN BEGIN                              {is on line.     }
    WRITESTRINGLN(Mainframe, 'TEST 600');                {Reseting and testing  }
    WRITESTRINGLN(Mainframe, 'ERRSTR?');                 {the Voltmeter         }
    READNUMBER (Mainframe, errornum);                    {/.....................}
    IF errornum<>0 THEN BEGIN
      WRITELN('Error in Voltmeter Self Test!!!!');
      WRITELN('Check your instruments');
      TwoBEEPS;
      IF errornum=27 THEN
        WRITELN('Self Test Failed. Check Voltmeter Calibration.');
      IF errornum=32 THEN
        WRITELN('No Accesory Present');
      Add_Error(Errorlist, ADCoff);
       Instrum.ADCsetFLAG := FALSE;
     END
```

```
    ELSE BEGIN
      WRITESTRINGLN(Mainframe, 'USE 600');              {\.....................}
      WRITESTRINGLN(Mainframe, 'RANGE 10');             {Measures over the     }
      WRITESTRINGLN(Mainframe, 'RESOL 16');             {shortcircuited channel}
      WRITESTRINGLN(Mainframe, 'SCANMODE ON');          {/.....................}
      WRITESTRINGLN(Mainframe, 'CONFMEAS DCV,'+ shortcircuit + ',NSCAN 20');
      FOR i:=1 TO NumRead DO
        READNUMBER(Mainframe, Voltage[i]);
      LSQR_fit(Voltage, NumRead, mean, sigma);
      IF (-0.0001 <= mean) AND (mean <= 0.0001) THEN
        Instrum.ADCsetFLAG := TRUE
      ELSE BEGIN
        TwoBEEPS;
        WRITELN('Problems in your short circuited channel');
        Add_Error(Errorlist, ADCoff);
        Instrum.ADCsetFLAG := FALSE;
       END;
     END;
   END;
 END;


PROCEDURE ADC_ON(VAR Instrum: INSTRFLAGREC; VAR Errorlist: ErrLIST);
{-------------------------------------------------------------------------
This PROCEDURE tests if the ADC is turned ON, if it is calibrated & left
the ADC in a known state.                                    MHG     10/Sept/96
-------------------------------------------------------------------------}
LABEL 100;

VAR ExitFLAG, ExitFLAG2    :BOOLEAN;
    count                  :INTEGER;


BEGIN
  ExitFLAG:=FALSE;
  ExitFLAG:=FALSE;
  count:=0;
  WRITELN;
  WRITELN('       POWER SUPPLIES: LIGHT DETECTORS');
  WRITELN('Please, turn ON: ');
  WRITELN('Blue & Green light Power Meter: ');
  WRITELN('Blue & Green''s referrence Power Meter: ');
  WRITELN('3852A Data Aquisition and Control Unit: ');
  WRITE('When ready, Hit the RETURN key... Q = <esc> ');
  Read_Returnkey(ExitFLAG);
  IF NOT(ExitFLAG) THEN

    TRY
      REPEAT
      WRITESTRINGLN(Mainframe, 'RST');
      WRITESTRINGLN(Mainframe, 'DISP OFF');
      Check_OpAmp(Instrum, Errorlist);
      Check_MPX(Instrum, Errorlist);
```

```
      Check_Voltmeter(Instrum, Errorlist);
      IORESET(12);
      IF (Instrum.ADCsetFLAG) AND (Instrum.MPXsetFLAG)
        AND (Instrum.OpAmpsetFLAG) THEN BEGIN
        WRITELN;
        WRITELN('WARNING: 3852A Display Off!');
        Instrum.ConvsetFLAG:=TRUE;
       END
      ELSE

        IF count>3 THEN BEGIN
          WRITE ('.  .  .  ...Instruments OFF LINE!!!,');
          Add_Error(Errorlist, Convoff);
          GOTO 100;
         END
        ELSE BEGIN
          count:= count+1;
          WRITELN('Error in your instruments');
          WRITE ('.  .  .  ...Instruments OFF LINE!!!,');
          WRITELN('     RETURN continues Q = <esc>.');
          Read_Returnkey(ExitFLAG2);
          IF ExitFLAG2 THEN GOTO 100;
         END;

      UNTIL Instrum.ConvsetFLAG;

    RECOVER BEGIN

     END;
100: END;        { - - - - - - - - - - - - - - - - - End of ADC_ON PROCEDURE.}


PROCEDURE OFF_ADConv(VAR Instrum: INSTRFLAGREC);
{-------------------------------------------------------------------------
 Message to be display as a reminder of which ADC instruments source to turn
 OFF.                                                          MHG 9/96.
 -------------------------------------------------------------------------}

BEGIN
  WRITESTRINGLN(Mainframe, 'DISP ON');
  WRITELN;
  WRITELN('       POWER SUPPLIES: LIGHT DETECTORS');
  WRITELN('Please, turn OFF: ');
  WRITELN('Blue & Green light Power Meter: ');
  WRITELN('Blue & Green''s referrence Power Meter: ');
  WITH Instrum DO BEGIN
  WRITELN('3852A Data Acquisition and Control Unit: ');
  WITH Instrum DO BEGIN
    Instrum.OpAmpsetFLAG := FALSE;
    Instrum.MPXsetFLAG := FALSE;
    Instrum.ADCsetFLAG := FALSE;
    Instrum.ConvsetFLAG := FALSE;
   END;
```

```
  END;


 PROCEDURE Set_Voltmeter(Numpoints : NATURAL; Gain, time : REAL);
 {-------------------------------------------------------------------------
  This procedure adjusts the parameters to set the voltmeter when the
  resolution is 14 bits.                                          MHG   7/Dic/96
  -------------------------------------------------------------------------}
 VAR delay, NumRead, Range   : STRING[16];
     pos, i, count    : INTEGER;

 BEGIN
   NumRead:=''; delay:=''; Range:='';
   WRITESTRINGLN (Mainframe, 'OUTBUF ON');
   WRITESTRINGLN (Mainframe, 'END ON');
   WRITESTRINGLN (Mainframe, 'USE 600');
   WRITESTRINGLN (Mainframe, 'RANGE 10');          {Always RANGE before RESOL 16}
   WRITESTRINGLN (Mainframe, 'RESOL 14');
   WRITESTRINGLN (Mainframe, 'SCANMODE ON');
   WRITESTRINGLN (Mainframe, 'RESOL 14');
   WRITESTRINGLN (Mainframe, 'FUNC DCV');
   WRITESTRINGLN (Mainframe, 'TERM RIBBON');
   WRITESTRINGLN (Mainframe, 'RDGSMODE DAV');
   WRITESTRINGLN (Mainframe, 'NRDGS 1');
   STRWRITE(delay, 1, pos, time:8:5);
   SETSTRLEN(delay, pos -1);
   WRITESTRINGLN (Mainframe, 'SPER ' + delay);
   STRWRITE(NumRead, 1, pos, Numpoints);
   SETSTRLEN(NumRead, pos -1);
   WRITESTRINGLN (Mainframe, 'PRESCAN '+ NumRead);
   WRITESTRINGLN (Mainframe, 'ASCAN ON');
   STRWRITE(Range, 1, pos, Gain);
   SETSTRLEN(Range, pos -1);
   WRITESTRINGLN (Mainframe, 'CLWRITE 500-505, RANGE ' + Range);
 END;              {- - - - - - - - - - - - - - - - End of PROCEDURE read_mult.}


PROCEDURE Set_Voltmeter16(Numpoints : NATURAL; Gain, time : REAL);
{-------------------------------------------------------------------------
 This procedure adjusts the parameters to set the voltmeter when the
 resolution is 16 bits.                                          MHG   7/Dic/96
 -------------------------------------------------------------------------}
VAR delay, NumRead, Range   : STRING[16];
    pos, i, count    : INTEGER;

BEGIN
  NumRead:=''; delay:=''; Range:='';
  WRITESTRINGLN (Mainframe, 'OUTBUF ON');
  WRITESTRINGLN (Mainframe, 'END ON');
  WRITESTRINGLN (Mainframe, 'USE 600');
  WRITESTRINGLN (Mainframe, 'RANGE 10');          {Always RANGE before RESOL 16}
  WRITESTRINGLN (Mainframe, 'RESOL 16');
  WRITESTRINGLN (Mainframe, 'SCANMODE ON');
```

```
  WRITESTRINGLN (Mainframe, 'RESOL 16');
  WRITESTRINGLN (Mainframe, 'FUNC DCV');
  WRITESTRINGLN (Mainframe, 'TERM RIBBON');
  WRITESTRINGLN (Mainframe, 'RDGSMODE DAV');
  WRITESTRINGLN (Mainframe, 'NRDGS 1');
  STRWRITE(delay, 1, pos, time:8:5);
  SETSTRLEN(delay, pos -1);
  WRITESTRINGLN (Mainframe, 'SPER ' + delay);
  STRWRITE(NumRead, 1, pos, Numpoints);
  SETSTRLEN(NumRead, pos -1);
  WRITESTRINGLN (Mainframe, 'PRESCAN '+ NumRead);
  WRITESTRINGLN (Mainframe, 'ASCAN ON');
  STRWRITE(Range, 1, pos, Gain);
  SETSTRLEN(Range, pos -1);
  WRITESTRINGLN (Mainframe, 'CLWRITE 500-505, RANGE ' + Range);
END;              {- - - - - - - - - - - - - - - - End of PROCEDURE read_mult.}


 PROCEDURE MHGMeasure_Scan(VAR buffer:BUF_INFO_TYPE);
{-------------------------------------------------------------------------
 This PROCEDURE triggers the voltmeter to start the measures.
                                                              MHG   5/Jan/97
 -------------------------------------------------------------------------}
CONST Mainframe = 709;

VAR Unconverted      :CHAR;
    REPTFLAG         :BOOLEAN;


BEGIN
  IOBUFFER(buffer,60000);
  BUFFER_RESET(buffer);
  REPTFLAG:=FALSE;
  Unconverted:=CHR(255);
  WRITESTRINGLN (Mainframe, 'SCTRIG INT');    {Triggering the voltmeter.}
  WRITESTRINGLN (Mainframe, 'XRDGS 600, PACK');
  TALK(7,09);                          {Setting the 3852A to be a talker & the}
  LISTEN(7,MY_ADDRESS(7));             {.....................................}
  UNLISTEN(7);                         {computer to be a listener............}
  LISTEN(7,MY_ADDRESS(7));             {.....................................}
  REPEAT
    TRANSFER_END(Mainframe, SERIAL_FASTEST, TO_MEMORY, buffer);
    IF BUFFER_DATA(buffer)<=1 THEN
      READBUFFER(buffer, Unconverted)
    ELSE
      REPTFLAG:=TRUE;
  UNTIL REPTFLAG;
  UNLISTEN(7);                      {Setting the 3852A to be a listener & the}
  LISTEN(7,09);                     {computer to be a talker..................}
  TALK(7,MY_ADDRESS(7));            {.....................................}

 END;            { - - - - - - - - - - - End of PROCEDURE MHGMeasure_Scan.}
```

```
PROCEDURE MHG_BintoReal(Conv2, Byte1:INTEGER; Gain :INTEGER;
                        VAR Rawdata:REAL);
{-------------------------------------------------------------------------
 This PROCEDURE converts two-complement binary data into real voltages when
 the resolution is 14 bits.                                      MHG 10/96.
 -------------------------------------------------------------------------}
VAR Unconverted1, Unconverted2 :CHAR;
    Byte2, i, f_range, Sign        :INTEGER;
    MaxRange : REAL;


BEGIN
    IF Byte1=255 THEN BEGIN            {.. Data out of range ..}
      Rawdata:=1.0E+38;       {.......................}
     END
    ELSE BEGIN
      CASE Gain OF
        1: BEGIN
             MaxRange:=0.04;           {.. 40 mV Gain ..}
             IF Conv2>32 THEN BEGIN
               Byte2:=Conv2-32;
               f_range:=256;
               Sign:=-1;
              END
             ELSE BEGIN
               Byte2:=Conv2;
               f_range:=256;
               Sign:=1
              END;
            END;
       2: BEGIN
             MaxRange:=0.32;           {.. 320 mV Gain ..}
             Conv2:=Conv2-64;
             IF Conv2>32 THEN BEGIN
               Byte2:=Conv2-32;
               f_range:=32;
               Sign:=-1;
              END
             ELSE BEGIN
               Byte2:=Conv2;
               f_range:=32;
               Sign:=1;
              END;
            END;
       3: BEGIN
            MaxRange:=2.56;            {.. 2.56 V Gain ..}
            Conv2:=Conv2-128;
            IF Conv2>32 THEN BEGIN
              Byte2:=Conv2-32;
              f_range:=4;
              Sign:=-1;
             END
```

```
            ELSE BEGIN
              Byte2:=Conv2;
              f_range:=4;
              Sign:=1;
             END;
           END;
       4: BEGIN
            MaxRange:=10.24;              {.. 10.24 V Gain ..}
            Conv2:=Conv2-192;
            IF Conv2>32 THEN BEGIN
              Byte2:=Conv2-32;
              f_range:=-1;
              Sign:=-1;
             END
            ELSE BEGIN
              Byte2:=Conv2;
              f_range:=-1;
              Sign:=1;
             END;
           END;
      END;
     IF Sign=-1 THEN BEGIN
       Rawdata:=-MaxRange-Sign*(Byte2*256+Byte1)/(800*f_range);
      END
     ELSE BEGIN
       Rawdata:=Sign*(Byte2*256+Byte1)/(800*f_range);
      END;
                          {Conveting data from binary to real
                           measures}
    END;
END;


PROCEDURE read_volt(Numpoints, Chann : INTEGER; time, Gain : REAL);
{-------------------------------------------------------------------
 This procedure adjust the parameters to measure over a single channel.
                                                          MHG 10/96
 -------------------------------------------------------------------}

VAR    delay, Channel, NumRead, Range    : STRING[16];
       pos                               : INTEGER;


BEGIN
  NumRead:=''; delay:=''; Channel:=''; Range:='';
  WRITESTRINGLN (Mainframe, 'OUTBUF ON');
  WRITESTRINGLN (Mainframe, 'END ON');
  WRITESTRINGLN (Mainframe, 'USE 600');
  WRITESTRINGLN (Mainframe, 'RANGE 10');    {Always RANGE before RESOL 16}
  WRITESTRINGLN (Mainframe, 'RESOL 14');
  WRITESTRINGLN (Mainframe, 'SCANMODE ON');
  WRITESTRINGLN (Mainframe, 'RESOL 14');
  WRITESTRINGLN (Mainframe, 'FUNC DCV');
  WRITESTRINGLN (Mainframe, 'TERM RIBBON');
```

```
  WRITESTRINGLN (Mainframe, 'RDGSMODE DAV');
  STRWRITE(NumRead, 1, pos, Numpoints);
  SETSTRLEN(NumRead, pos -1);
  WRITESTRINGLN (Mainframe, 'NRDGS' + NumRead);
  STRWRITE(delay, 1, pos, time:8:5);
  SETSTRLEN(delay, pos -1);
  WRITESTRINGLN (Mainframe, 'SPER ' + delay);
  STRWRITE(Channel, 1, pos, Chann);
  SETSTRLEN(Channel, pos -1);
  STRWRITE(Range, 1, pos, Gain:8:5);
  SETSTRLEN(Range, pos -1);
  WRITESTRINGLN (Mainframe, 'CLWRITE ' + Channel + 'RANGE ' + Range);
END;              {- - - - - - - - - - - - - - - - End of PROCEDURE read_volt.}


PROCEDURE read_multiple(Numpoints, m : INTEGER; time : REAL;
                        Chann, RangeSTR : ChanStrARRAY; ChNum : BooleanARRAY);
{-------------------------------------------------------------------------
 This procedure adjust the parameters to measure over multiple channels.
                                                          MHG 10/96.
 -------------------------------------------------------------------------}

VAR    delay, NumRead, indexSTR, entrySTR       : STRING[16];
       pos, i, arrayindex, count                : INTEGER;


BEGIN

  NumRead:=''; delay:=''; indexSTR:=''; entrySTR:='';
  count:=0;
  arrayindex:=m-1;
  STRWRITE(indexSTR, 1, pos, arrayindex);
  SETSTRLEN(indexSTR, pos -1);
  WRITESTRINGLN (Mainframe, 'INTEGER Channlst (' + indexSTR + ')');
  WRITESTRINGLN (Mainframe, 'REAL Rangelst (' + indexSTR + ')');
  FOR i:=1 TO Size DO BEGIN
    IF ChNum[i] THEN BEGIN
    STRWRITE(entrySTR, 1, pos, count);
    SETSTRLEN(entrySTR, pos -1);
    WRITESTRINGLN(Mainframe,'LET Channlst ('+ entrySTR + ')=' + Chann[i]);
    WRITESTRINGLN(Mainframe,'LET Rangelst ('+ entrySTR + ')=' + RangeSTR[i]);
    count:=count+1;
   END;
  END;
  WRITESTRINGLN (Mainframe, 'OUTBUF ON');
  WRITESTRINGLN (Mainframe, 'END ON');
  WRITESTRINGLN (Mainframe, 'USE 600');
  WRITESTRINGLN (Mainframe, 'RANGE 10');          {Always RANGE before RESOL 16}
  WRITESTRINGLN (Mainframe, 'RESOL 14');
  WRITESTRINGLN (Mainframe, 'SCANMODE ON');
  WRITESTRINGLN (Mainframe, 'RESOL 14');
  WRITESTRINGLN (Mainframe, 'FUNC DCV');
```

```
  WRITESTRINGLN (Mainframe, 'TERM RIBBON');
  WRITESTRINGLN (Mainframe, 'RDGSMODE DAV');
  WRITESTRINGLN (Mainframe, 'NRDGS 1');
  STRWRITE(delay, 1, pos, time:8:5);
  SETSTRLEN(delay, pos -1);
  WRITESTRINGLN (Mainframe, 'SPER ' + delay);
  STRWRITE(NumRead, 1, pos, Numpoints);
  SETSTRLEN(NumRead, pos -1);
  WRITESTRINGLN (Mainframe, 'PRESCAN '+ NumRead);
  WRITESTRINGLN (Mainframe, 'ASCAN ON');
  WRITESTRINGLN (Mainframe, 'CLWRITE Channlst, RANGE Rangelst');
END;              {- - - - - - - - - - - - - - - - End of PROCEDURE read_mult.}


PROCEDURE AD_Conv;
{-------------------------------------------------------------------------
 This procedure allows the operator to use the ADC card to to voltage
 measurements on a single voltage source or for several inputs (up to seven).
                                                          MHG 9/96.
 -------------------------------------------------------------------------}
 LABEL    100;
 CONST    tab = 4;
          datSize = 60000;

 TYPE     ChanARRAY      = ARRAY[1..Size] OF INTEGER;
          DatARRAY       = ARRAY[1..datSize] OF REAL;

 CONST    ChanStrConst = ChanStrARRAY[Size OF '25'];
          ChanConst    = ChanARRAY[Size OF 25];
          ConstDatARRAY= DatARRAY[datSize OF 37.0E+00];

 VAR      Success, Quit, AutoFLAG, REPTFLAG, ReadFLAG   : BOOLEAN;
          Sel, key, YNchannel, YNkey,
          Unconverted1, Unconverted2                    : CHAR;
          Channel, Chann, i, j, Voltindx, Numpoints,
          ChoiSet, ChGainset                            : CHARSET;
          Channel, Chann, i, j, Voltindx, Numpoints,
          Dataindx, Numdetect, Gain, pos, Conv1,
          count, Byte1, Conv2, f_range, Byte2, Sign     : INTEGER;
          time, mean, sigma, pacetime, Gainum, MaxRange: REAL;
          lastxpos, lastypos                            : NATURAL;
          ChNum                                         : BooleanARRAY;
          Errorlist                                     : ErrLIST;
          InstrFLAG                                     : INSTRFLAGREC;
          ChannI                                        : ChanARRAY;
          ChanSTR, ChGainum                             : ChanStrARRAY;
          Volt, Data                                    : DatARRAY;
          ChGain                                        : ARRAY[1..8] OF CHAR;
          GainInt                                       : ARRAY[1..8] OF INTEGER;
          ChannelSTR, meanSTR, sigmaSTR                 : STRING[20];
          buffer                                        : BUF_INFO_TYPE;
          strng                                         : STRING[255];
          meanArr, sigmaArr                             : ARRAY[1..Size] OF REAL;
```

```
BEGIN
 IOINITIALIZE;                    { .. I/O initialization always recomended .. }
 ADC_ON(InstrFLAG, Errorlist);
 IF InstrFLAG.ConvsetFLAG THEN BEGIN
   ChoiSet := ['o', 'O', 'm', 'M', 'p', 'P', 'a', 'A', 'q', 'Q'];
   REPEAT
     WRITELN;
     WRITE('Do you want to read data through: One channel, Multiple channels,');
     WRITE(' light Amplifiers, Quit? ');
     Read_input(ChoiSet, Sel);
     CASE Sel OF

       'o', 'O': BEGIN
         ChannelSTR:='';        meanSTR:='';     sigmaSTR:='';
         REPTFLAG:=FALSE;
         Voltindx:=1;
         WRITELN;
         WRITELN('Set parameters for Digital/Analog Card: ');
         ReadFLAG := FALSE;
         REPEAT
           WRITELN('How many readings in how much time? (# readings, sec)');
           TRY
             READLN(Numpoints, pacetime);
             ReadFLAG := TRUE;
           RECOVER BEGIN
             WRITELN;
             ReadErrorRecover
            END;
        UNTIL ReadFLAG;
        ReadFLAG:=FALSE;
        REPEAT
          WRITE('Through which channel you want your measurements(1-24)?');
          TRY
            READLN(Channel);
            ReadFLAG := TRUE;
          RECOVER BEGIN
            WRITELN;
            ReadErrorRecover
           END;
          IF (Channel<1) OR (Channel>24) THEN BEGIN
            WRITELN;
            WRITELN('                Unacceptable Option .... TRY AGAIN!');
            WRITELN;
            ReadFLAG:=FALSE;
           END
          ELSE ReadFLAG:=TRUE;
        UNTIL ReadFLAG;
        ReadFLAG:=FALSE;
        REPEAT
          WRITE('What GAIN value to perform the mesurements (1-4)?');
          TRY
```

```
      READLN(Gain);
      ReadFLAG:=TRUE;
    RECOVER BEGIN
      WRITELN;
      ReadErrorRecover
     END;
     IF (Gain<1) AND (Gain>4) THEN BEGIN
       WRITELN;
       WRITELN('                Unacceptable Option .... TRY AGAIN!');
       WRITELN;
       ReadFLAG:=FALSE;
      END
     ELSE BEGIN
       ReadFLAG:=TRUE;
       CASE Gain OF
          1:Gainum:=0.03;
          2:Gainum:=0.3;
          3:Gainum:=2;
          4:Gainum:=10;
         END;
      END;
UNTIL ReadFLAG;
Chann:=500+Channel-1;
Unconverted1:=CHR(255);
Unconverted2:=CHR(255);
TRY
   read_volt(Numpoints, Chann, pacetime, Gainum);
   MHGMeasure_Scan(buffer);
   WHILE BUFFER_DATA(buffer)>0 DO BEGIN
     READBUFFER(buffer, Unconverted2);
     Conv2:=ORD(Unconverted2);
     READBUFFER(buffer, Unconverted1);
     Byte1:=ORD(Unconverted1);
     MHG_BintoReal(Conv2, Byte1, Gain,Volt[Voltindx]);
     Voltindx:=Voltindx+1;
    END;
RECOVER BEGIN
   IF ESCAPECODE = IOESCAPECODE THEN
   WRITELN (IOERROR_MESSAGE(IOE_RESULT));
   ESCAPE(ESCAPECODE);
 END;
LSQR_fit(Volt, Numpoints, mean, sigma);
WRITE('Voltage through Channel # ', Channel, ' is = ', mean);
WRITELN(' '#254' ', Sigma);
IF (mean>=1E+037) THEN BEGIN
   TwoBEEPS;
   WRITELN('Error in you Range Option! Data is not valid');
 END;
WRITE ('Do you want to PRINT your results? (Y/N)');
Read_input(YNset,YNkey);
IF (YNkey ='y')OR(YNkey='Y') THEN BEGIN
   STRWRITE(ChannelSTR, 1, pos, Channel:0);
   SETSTRLEN(ChannelSTR, pos-1);
   STRWRITE(meanSTR, 1, pos, mean);
```

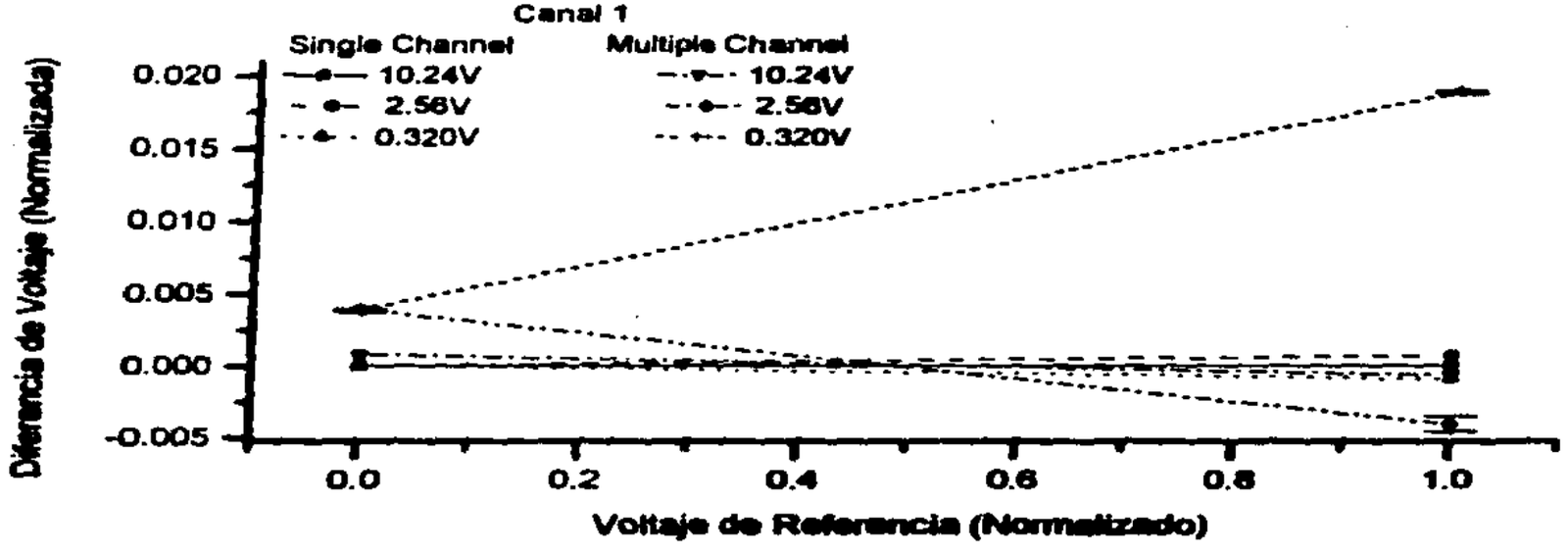
```
    SETSTRLEN(meanSTR, pos-1);
    STRWRITE(sigmaSTR, 1, pos, sigma);
    SETSTRLEN(sigmaSTR, pos -1);
    WRITESTRING   (PRINTER, 'Voltage through Channel # '+ ChannelSTR);
    WRITESTRINGLN (PRINTER, ' is '+ meanSTR+' '#254' '+ sigmaSTR);
   END;
  WRITELN ('Do you want to make a RESET?(Y/N)');
  Read_input(YNset,YNkey);
  IF (YNkey ='y')OR(YNkey='Y') THEN
    WRITESTRINGLN(Mainframe, 'RST');
 END;


'm', 'M': BEGIN
  ChannelSTR:='';          meanSTR:='';     sigmaSTR:='';
  Volt := ConstDatARRAY;          Data:=ConstDatARRAY;
  ChanSTR:=ChanStrConst;     ChGainum:=ChanStrConst;
  ChannI:=ChanConst;
  REPTFLAG:=FALSE;
  Voltindx:=1;
  Unconverted1:=CHR(255);          Unconverted2:=CHR(255);
  WRITELN('SELECT channels to be read by typing (Yes/No)  ! ');
  WRITELN('    Ch1, Ch2, Ch3, Ch4, Ch5, Ch6, Ch7, Ch8. ');
  FOR i := 1 TO Size DO BEGIN
    lastxpos := xpos + tab;
    REPEAT
      xpos := lastxpos;
      call(updatecursorhook);
      TRY
        READ(YNchannel);
        IF YNchannel IN YNset THEN BEGIN
            IF (YNchannel = 'y')OR(YNchannel = 'Y') THEN
                ChNum[i] := TRUE
            ELSE ChNum[i] := FALSE
          END
        ELSE ESCAPE(0);
        Success := TRUE
      RECOVER BEGIN
          Success := FALSE;
          IF ESCAPECODE = 0 THEN BEGIN
              TwoBEEPS;    RESET(INPUT);
              WRITELN;
              WRITELN('Please Yes or No!-Try again ... ');
              WRITELN('    Ch1, Ch2, Ch3, Ch4, Ch5, Ch6, Ch7, Ch8. ')
            END
          ELSE ESCAPE(ESCAPECODE)
        END;
    UNTIL Success;
   END;
  Numdetect:=0;
  FOR i:=1 TO Size DO BEGIN
    IF ChNum[i] THEN BEGIN
      ChannI[i]:=500+i-1;
```

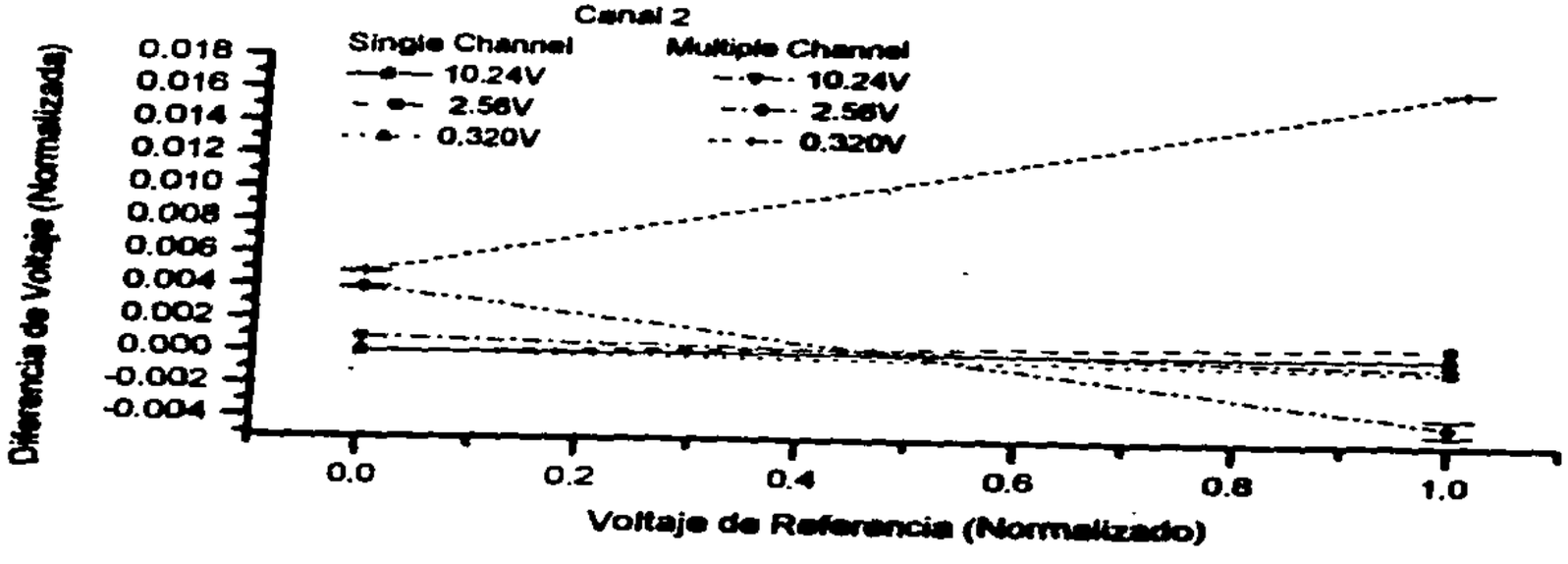
```
      STRWRITE(ChanSTR[i], 1, pos, ChannI[i]);
      SETSTRLEN(ChanSTR[i], pos -1);
      Numdetect:=Numdetect+1;
     END
    ELSE BEGIN
      ChanSTR[i]:='';
     END;
  END;
WRITELN;
ChGainset:=['1', '2', '3', '4'];
WRITELN('SET (1-4) The GAIN for every channel selected! ');
WRITELN('#: Ch1, Ch2, Ch3, Ch4, Ch5, Ch6, Ch7, Ch8. ');
FOR i := 1 TO Size DO BEGIN
  lastxpos := xpos + tab;
  REPEAT
    xpos := lastxpos;
    call(updatecursorhook);
    Success:= FALSE;
    IF ChNum[i] THEN BEGIN
      TRY
        READ(ChGain[i]);
        IF ChGain[i] IN ChGainset THEN BEGIN
          CASE ChGain[i] OF
            '1': BEGIN
                   ChGainum[i]:='0.04';
                   GainInt[i]:=1;
                 END;
            '2': BEGIN
                   ChGainum[i]:='0.320';
                   GainInt[i]:=2;
                 END;
            '3': BEGIN
                   ChGainum[i]:='2.56';
                   GainInt[i]:=3;
                 END;
            '4': BEGIN
                   ChGainum[i]:='10';
                   GainInt[i]:=4;
                 END;
          END;
        END
        ELSE ESCAPE(0);
        Success := TRUE
      RECOVER BEGIN
        Success := FALSE;
        IF ESCAPECODE = 0 THEN BEGIN
          TwoBEEPS;   RESET(INPUT);
          WRITELN;
          WRITELN('Please 1-4! Try again ... ');
          WRITELN('#: Ch1, Ch2, Ch3, Ch4, Ch5, Ch6, Ch7, Ch8. ')
        END
        ELSE ESCAPE(ESCAPECODE)
      END;
    END
```

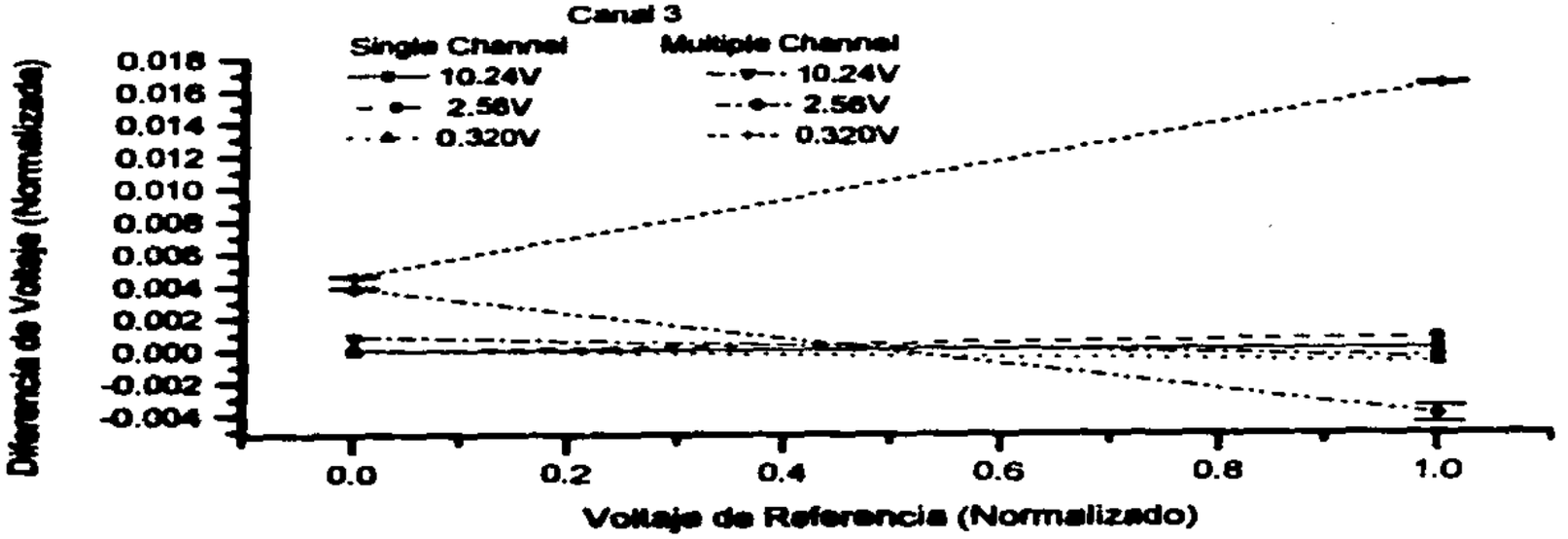
```
      ELSE BEGIN
        READ(ChGain[i]);
        Success := TRUE
       END;
   UNTIL Success;
  END;
WRITELN;
ReadFLAG:=FALSE;
REPEAT
   WRITELN;
   WRITE('How many readings/channel and how much time?');
   WRITELN('  (# readings, time)');
   TRY
     READLN(Numpoints, pacetime);
     ReadFLAG := TRUE;
   RECOVER BEGIN
     WRITELN;
     ReadErrorRecover;
    END;
UNTIL ReadFLAG;
ReadFLAG:=FALSE;
TRY
   read_multiple(Numpoints, Numdetect, pacetime, ChanSTR, ChGainum, ChNum);
   MHGMeasure_Scan(buffer);
   WHILE BUFFER_DATA(buffer) >0 DO BEGIN
     FOR i:=1 TO Size DO BEGIN
       IF ChNum[i] THEN BEGIN
         READBUFFER(buffer, Unconverted2);
         Conv2:=ORD(Unconverted2);
         READBUFFER(buffer, Unconverted1);
         Byte1:=ORD(Unconverted1);
         MHG_BintoReal(Conv2, Byte1, GainInt[i],Volt[Voltindx]);
         Voltindx:=Voltindx+1;
         END;
      END;
    END;
RECOVER BEGIN
   IF ESCAPECODE = IOESCAPECODE THEN
   WRITELN (IOERROR_MESSAGE(IOE_RESULT));
   ESCAPE(ESCAPECODE);
 END;
count:=1;
FOR j:=1 TO Size DO BEGIN
   IF ChNum[j] THEN BEGIN
     Dataindx:=1;
     Voltindx:=count;
     REPEAT
       Data[Dataindx]:=Volt[Voltindx];
       Voltindx:=Voltindx+Numdetect;
       Dataindx:=Dataindx+1;
     UNTIL Dataindx = Numpoints + 1;
     LSQR_fit(Data, Numpoints, meanArr[j], sigmaArr[j]);
     WRITELN('Voltage @ Channel #', j:0,
              'is = ', meanArr[j], ' '#254' ', sigmaArr[j]);
```

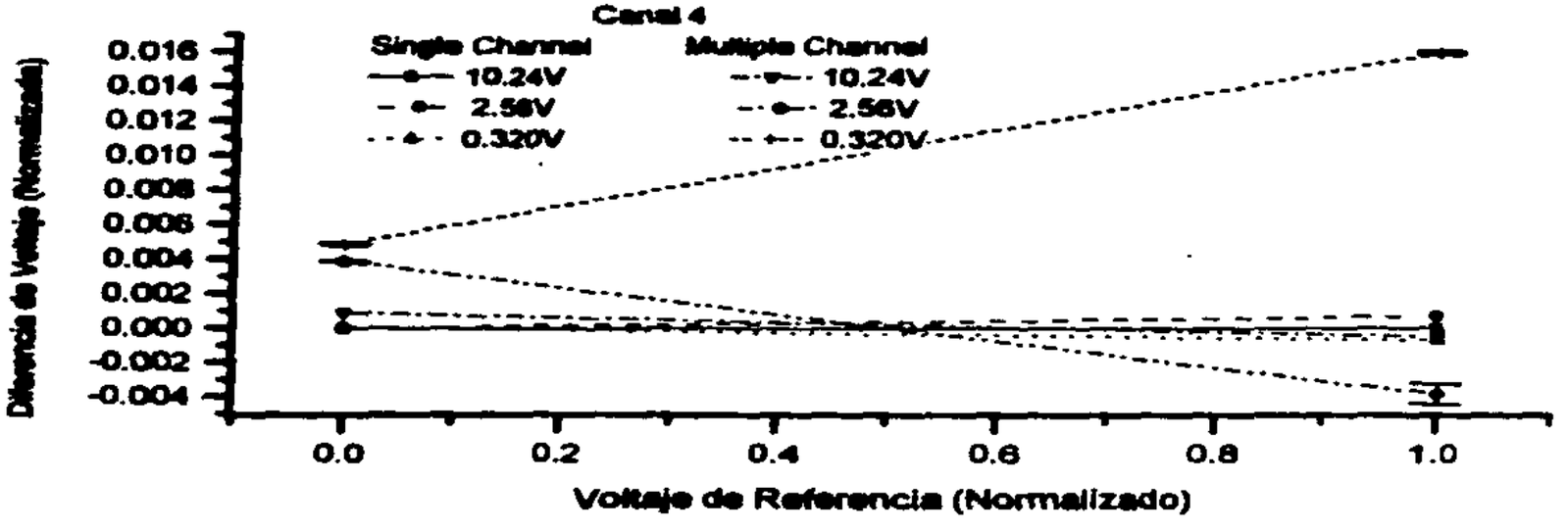
```
            IF (meanArr[j]>=100) THEN BEGIN
              TwoBEEPS;   WRITELN;
              WRITELN('Error in you Range Option! Data is not valid');
              WRITELN;
             END;
            count:=count+1;
           END;
         END;
         WRITE('Do you want to PRINT your results?(Y/N) ');
         Read_input(YNset,YNkey);
         IF (YNkey ='y')OR(YNkey='Y') THEN BEGIN
           FOR j:=1 TO Size DO BEGIN
            IF ChNum[j] THEN BEGIN
              STRWRITE(ChannelSTR, 1, pos, j:0);
              SETSTRLEN(ChannelSTR, pos-1);
              STRWRITE(meanSTR, 1, pos, mean);
              SETSTRLEN(meanSTR, pos-1);
              STRWRITE(sigmaSTR, 1, pos, sigma);
              SETSTRLEN(sigmaSTR, pos -1);
              WRITESTRING   (PRINTER, 'Voltage through Channel # '+ ChannelSTR);
              WRITESTRINGLN (PRINTER, '   is   '+ meanSTR+' '#254' '+ sigmaSTR);
             END;
           END;
         END;
        WRITE('Do you want to RESET 3852A? (Y/N) ');
        Read_input (YNset,YNkey);
        IF (YNkey ='y')OR(YNkey='Y') THEN
       END;


      'a', 'A': BEGIN                          {To be added subsecuently when this}
       END;                                    { instrument is available .........}


      'q', 'Q': BEGIN
        Quit := TRUE;
        GOTO 100
       END
     END;
    WRITE('Do you want to measure another voltage? (Y/N) ');
    Read_input(YNset, key);
    IF (key = 'n')OR(key = 'N') THEN Quit := TRUE
    ELSE Quit := FALSE;
  100: UNTIL Quit;
 END;
 IOUNINITIALIZE
END;       { - - - - - - - - - - - - - - - - - - - - - End OF PROCEDURE ADc_Conv.}

END;               { - - - - - - - - End of A/D converter MODULE implementation.}
```

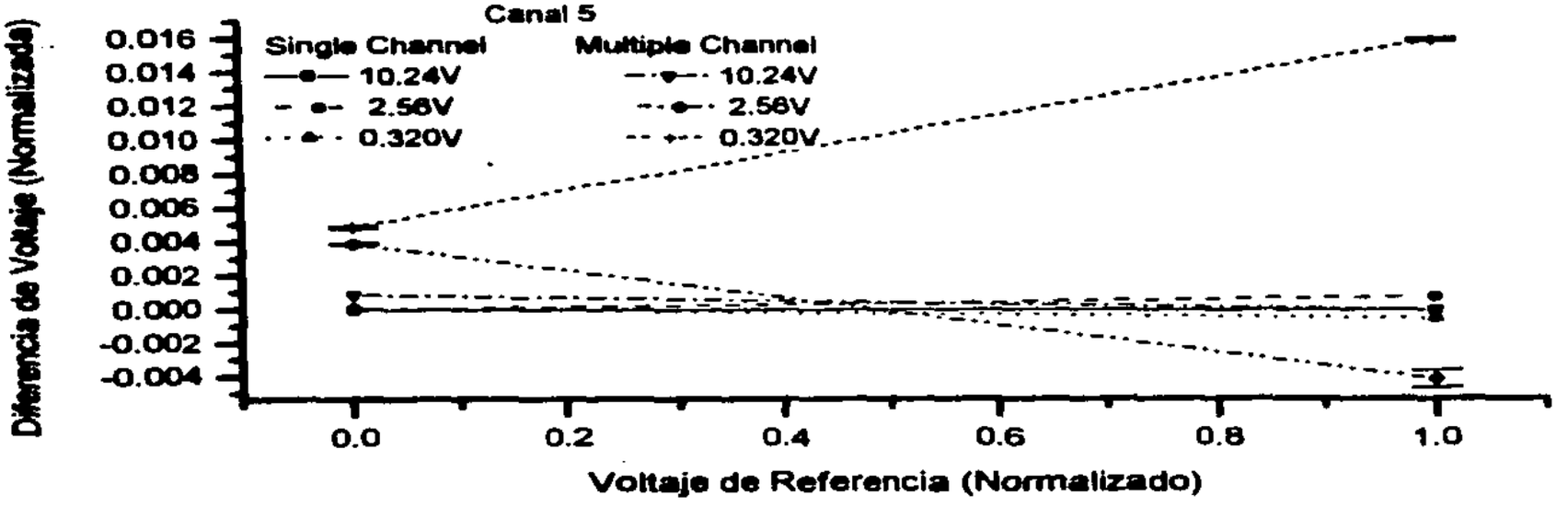
Apéndice C. Gráficas de Calibración del Vóltmetro de Alta Velocidad.

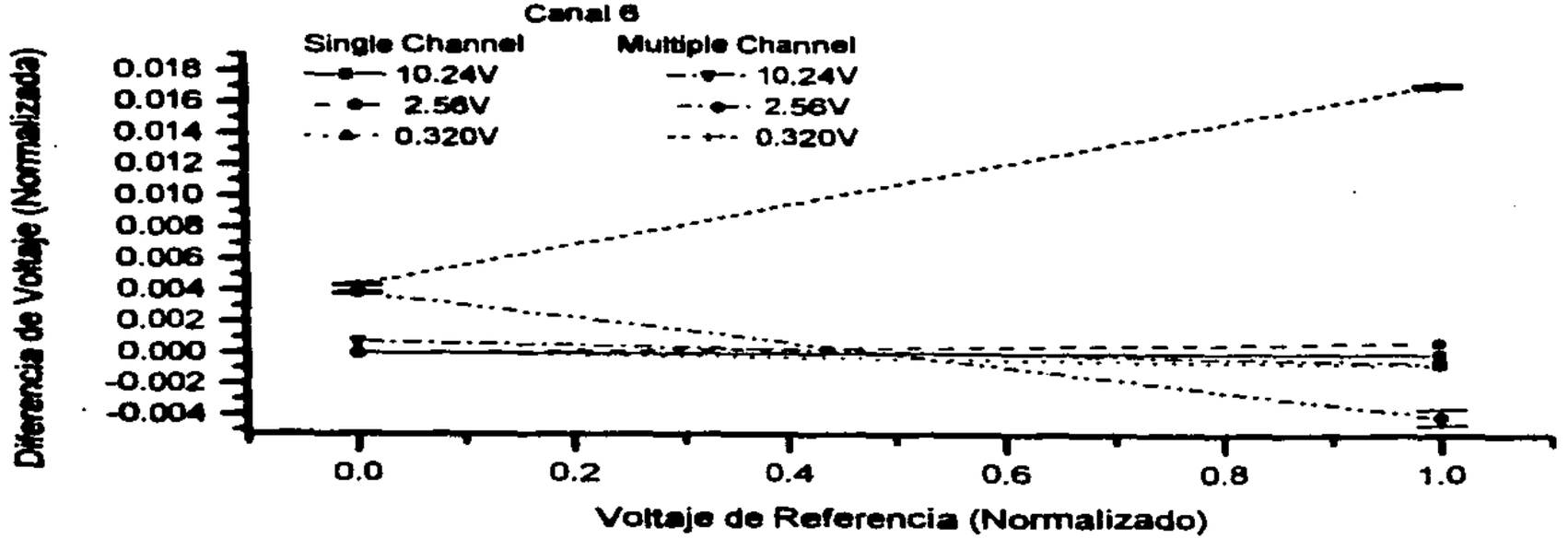


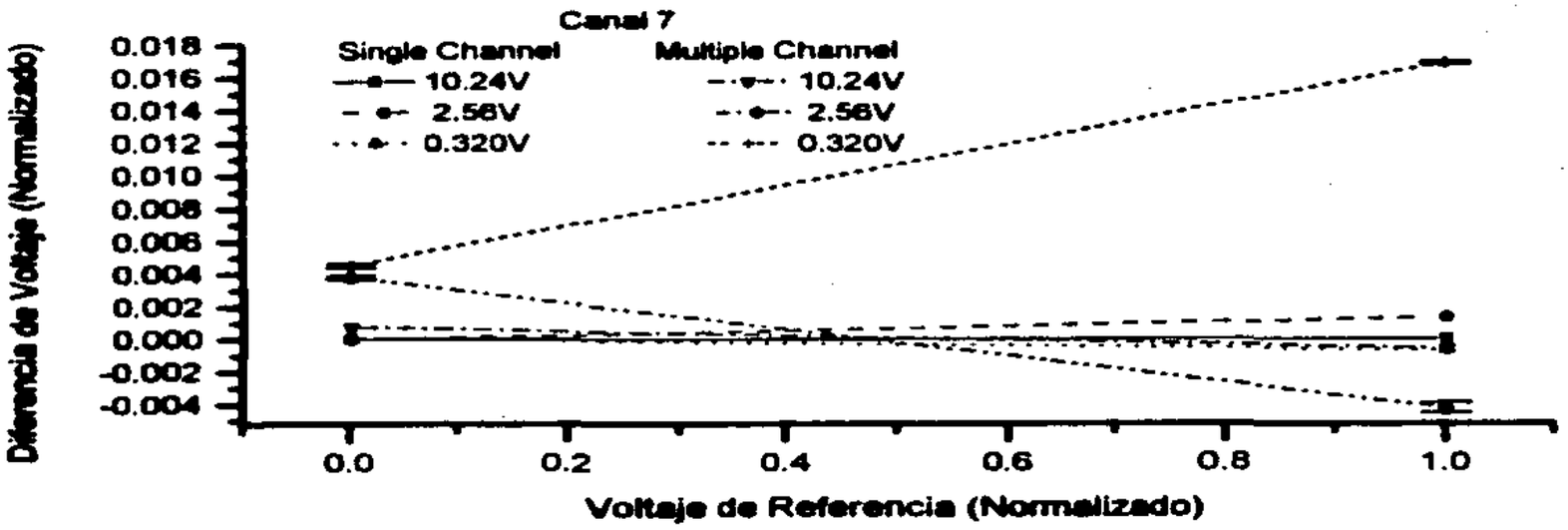


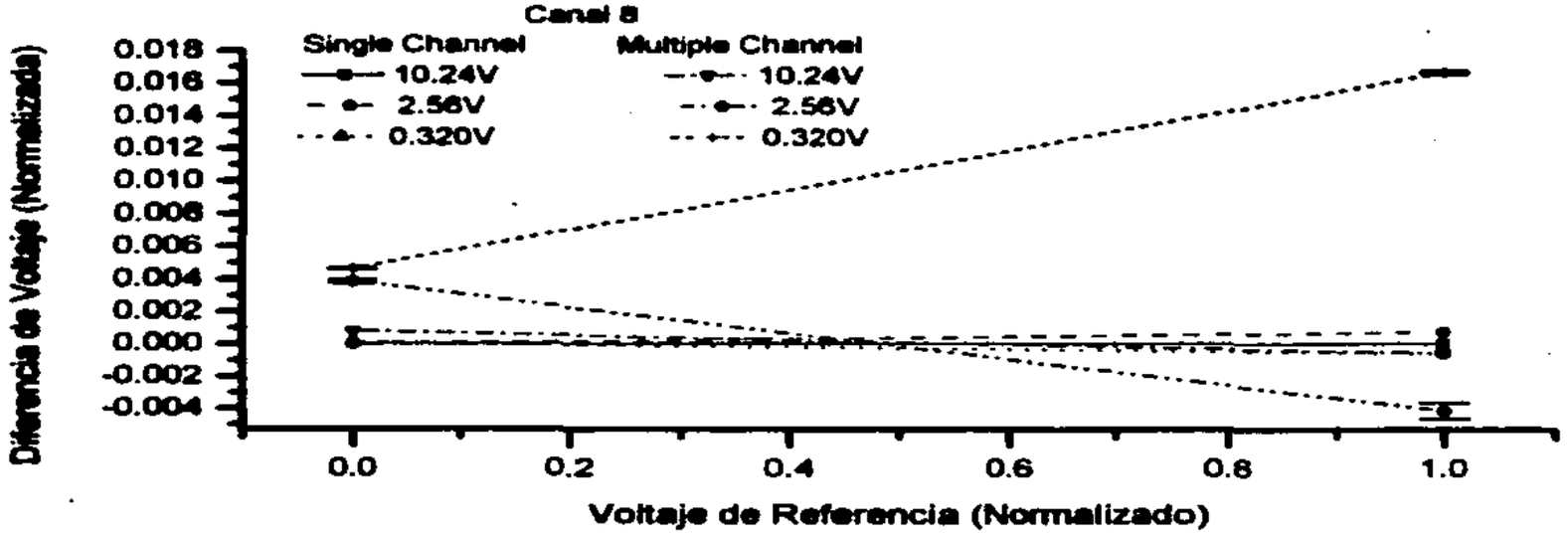

Canal 3
Single Channel
10.24V
2.56V
0.320V
Multiple Channel
10.24V
2.56V
0.320V
Diferencia de Voltaje (Normalizada)
0.018
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
-0.002
-0.004
0.0
0.2
0.4
0.6
0.8
1.0
Voltaje de Referencia (Normalizado)


Canal 4
Single Channel
10.24V
2.56V
0.320V
Multiple Channel
10.24V
2.56V
0.320V
Diferencia de Voltaje (Normalizada)
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
-0.002
-0.004
0.0
0.2
0.4
0.6
0.8
1.0
Voltaje de Referencia (Normalizado)


Canal 5
Single Channel
10.24V
2.56V
0.320V
Multiple Channel
10.24V
2.56V
0.320V
Diferencia de Voltaje (Normalizada)
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
-0.002
-0.004
0.0
0.2
0.4
0.6
0.8
1.0
Voltaje de Referencia (Normalizado)


Canal 6
Single Channel
10.24V
2.56V
0.320V
Multiple Channel
10.24V
2.56V
0.320V
Diferencia de Voltaje (Normalizada)
0.018
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
-0.002
-0.004
0.0
0.2
0.4
0.6
0.8
1.0
Voltaje de Referencia (Normalizado)


Canal 7
Single Channel
10.24V
2.56V
0.320V
Multiple Channel
10.24V
2.56V
0.320V
Diferencia de Voltaje (Normalizado)
0.018
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
-0.002
-0.004
0.0
0.2
0.4
0.6
0.8
1.0
Voltaje de Referencia (Normalizado)


Canal 8
Single Channel
10.24V
2.56V
0.320V
Multiple Channel
10.24V
2.56V
0.320V
Diferencia de Voltaje (Normalizada)
0.018
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
-0.002
-0.004
0.0
0.2
0.4
0.6
0.8
1.0
Voltaje de Referencia (Normalizado)

	Canal 1	Canal 2	Canal 3	Canal 4	Canal 5	Canal 6	Canal 7	Canal 8
Ganancia 10.24 V								
Voltaje de Referencia (V)	9.351879	9.353326	9.353427	9.353791	9.354681	9.355063	9.355294	9.355498
Voltaje Single Channel (V)	9.351784	9.352610	9.352582	9.353348	9.354978	9.355000	9.355000	9.355000
Voltaje Multiple Channels (V)	9.357314	9.357508	9.357500	9.357574	9.359312	9.359868	9.359958	9.359995
Factor de Corrección Single Channel	1.0000102	1.0000766	1.0000903	1.0000474	0.9999683	1.0000067	1.0000314	1.0000532
Factor de Corrección Multiple Channels	0.9994192	0.999554	0.9995647	0.999557	0.9995052	0.9994866	0.9995017	0.9995196
Offset Single Channel (V)	1.991286×10^{-3}	1.808191×10^{-3}	2.965938×10^{-3}	2.709386×10^{-3}	2.098434×10^{-3}	2.352076×10^{-3}	1.959307×10^{-3}	3.215586×10^{-3}
Offset Multiple Channels (V)	8.336048×10^{-3}	9.064708×10^{-3}	8.965992×10^{-3}	9.335440×10^{-3}	9.068326×10^{-3}	8.256216×10^{-3}	7.945810×10^{-3}	8.195300×10^{-3}
Ganancia 2.56 V								
Voltaje de Referencia (V)	1.597599	1.597595	1.597589	1.597577	1.597546	1.597531	1.596357	1.596835
Voltaje Single Channel (V)	1.596880	1.596820	1.596850	1.596836	1.596824	1.596810	1.595016	1.596080
Voltaje Multiple Channels (V)	1.601638	1.601645	1.601626	1.601380	1.601656	1.601542	1.600558	1.600934
Factor de Corrección Single Channel	1.0004503	1.0004797	1.0004628	1.0004515	1.0004521	1.0004515	1.0005607	1.0004730
Factor de Corrección Multiple Channels	0.9974782	0.9974707	0.9974794	0.9976252	0.9974339	0.9974955	0.997373	0.9974396
Offset Single Channel (V)	2.170908×10^{-3}	1.976160×10^{-3}	3.128768×10^{-3}	2.816436×10^{-3}	1.990148×10^{-3}	2.442978×10^{-3}	2.403170×10^{-3}	2.891586×10^{-3}
Offset Multiple Channels (V)	9.960994×10^{-3}	1.000005×10^{-2}	1.000035×10^{-2}	1.000428×10^{-2}	1.000256×10^{-2}	9.965936×10^{-3}	9.831086×10^{-3}	9.895302×10^{-3}

	Canal 1	Canal 2	Canal 3	Canal 4	Canal 5	Canal 6	Canal 7	Canal 8
Ganancia 0.320 V								
Voltaje de Referencia (V)	0.1418145	0.1418142	0.1418268	0.1418322	0.1418984	0.1419486	0.1419767	0.141994
Voltaje Single Channel (V)	0.1418706	0.1418574	0.1418764	0.1418746	0.1419328	0.1419916	0.1420196	0.1420288
Voltaje Multiple Channels (V)	0.1406398	0.1407982	0.1407836	0.1408348	0.1409078	0.1408724	0.1409254	0.1409486
Factor de Corrección Single Channel	0.9996046	0.9996955	0.9996504	0.9997011	0.9997576	0.9996972	0.9996979	0.9997550
Factor de Corrección Multiple Channels	1.0083525	1.0072160	1.0074100	1.0070821	1.0070301	1.0076395	1.0074680	1.0074169
Offset Single Channel (V)	$2.046844\text{x}10^{-3}$	$1.427306\text{x}10^{-3}$	$2.560556\text{x}10^{-3}$	$2.676785\text{x}10^{-3}$	$2.034010\text{x}10^{-3}$	$2.144990\text{x}10^{-3}$	$2.575078\text{x}10^{-3}$	$2.583264\text{x}10^{-3}$
Offset Multiple Channels (V)	1.250644x10	1.561138x10	$1.480594\text{x}10^{-3}$	$1.561138\text{x}10^{-3}$	$1.566722\text{x}10^{-3}$	$1.410780\text{x}10^{-3}$	$1.454150\text{x}10^{-3}$	$1.481736\text{x}10^{-3}$

Bibliografía

[1] E. Geffroy. *Birrefringence of Polymer Solutions in Time Dependent Flow*. PhD thesis, California Institute of Technology, 1990.

[2] E. Collett. *Polarized Light*. Marcel Dekker Inc., USA, 1993.

[3] J. W. Lewis & C. E. Randall D. S. Kliger. *Polarized Light in Optics and Spectroscopy*. Academic Press, USA, 1990.

[4] M. Born & E. Wolf. *Principles of Optics*. Pergamon Press, Great Britain, 4th edition, 1970.

[5] G. G. Stokes. On the composition and resolution of streams of polarized light from different sources. *Trans. Cambridge Phil. Soc.*, 9:399, 1852.

[6] J. L. S. Wales. *The Application of Flow Birefringence to Rheological Studies of Polymer Melts*. Delft University Press, Netherlands, 1976.

[7] H. Müller. The foundations of optics. *J. Opt. Soc. Am.*, 38:661, 1948.

[8] R. C. Jones. A new calculus for the treatment of optical systems. i. description and discussion of calculus. *J. Opt. Soc. Am.*, 31:488–493, 1941.

[9] R. C. Jones. A new calculus for the treatment of optical systems. viii. electromagnetic theory. *J. Opt. Soc. Am.*, 46:126–131, 1956.

[10] H. Janeschitz-Kriegl. *Polymer Melt Rheology and Flow Birefringence*. Springer-Verlag, Germany, 1983.

[11] E. Bautista Thompson. *Evaluación de la técnica de anisotropía bicolor inducida por flujos para muestras con dicroismo, birrefringencia o depolarización residual*. Tesis de Físico, Universidad Nacional Autónoma de México, 1994.

[12] A. Onuki & M. Doi. *J. Chem. Phys.*, 85(2):1190–1997, 1986.

[13] G. H. Meeten. *Optical Properties of Polymers*. Elsevier Applied Science Publishers, Ireland, 1986.

[14] Y. Shindo & H. Hanabusa. *Polym. Comm.*, 24:240, 1983.

[15] Y. Shindo & R. Takigaura. *Polym. Comm.*, 25:378–381, 1984.

[16] A. W. Chow & G. G. Fuller. *J. Rheol*, 28(1):23–43, 1984.

[17] P. L. Frattini & G. G. Fuller. *J. Rheology*, 28(1):61–70, 1984.

[18] Spectra-Physics. *BeamLok™ 2060 and 2080, Ion Laser, Instruction Manual*. Spectra-Physics, 1992.

[19] M. A. Reyes Huesca. *Estudio y solución analítica de un flujo fuerte generado por un molino de dos rodillos*. Tesis de Ingeniero Mecánico, Universidad Nacional Autónoma de México, 1997.

[20] D. Yavich & L. G. Leal. Experimental and theoretical studies of semidilute polyestirene solutions in mixed type of flows. *Rheologica Acta*, 1996.

[21] E. García Muñoz. *Detección automatizada de señales para técnica de birrefringencia bicolor*. Tesis de Físico, Universidad Nacional Autónoma de México, 1996.

[22] C. Corona Pastrana. *Medición de Señales luminosas de alta resolución para estudios de anisotropía de flujos en sistemas poliméricos*. Tesis de Físico, Universidad Nacional Autónoma de México, 1997.

[23] P. A. Psaras & H. D. Langford, editor. *Advancing Materials Research*. National Academy Press, USA, 1987.

[24] Hewlett Packard. *Pascal 3.2 Workstation System. Programming and Configuration Topics*, volume 2. Hewlett Packard, 1991.

[25] E. Geffroy & L. G. Leal. Flow birefringence studies in transient flows of a two-roll mill for the test-fluid m1. *J. Non-Newtonian Fluid Mech.*, 35(3):361–400, 1990.

[26] S. T. Allworth & R. N. Zobel. *Introduction to Real-time Software Designe*. Springer-Verlag New York Inc., second edition, 1987.