SANDIA REPORT
SAND2001-0963
Unlimited Release
Printed April 2001

Current Filament Semiconductor Lasers (CFSL)

Fred J. Zutavern, Albert G. Baca, Weng W. Chow, Michael J. Hafich, Harold P. Hjalmarson, Guillermo M. Loubriel, Alan Mar, Martin W. O'Malley, and G. Allen Vawter

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/ordering.htm

SAND2001-0963
Unlimited Release
Printed April 2001

Current Filament Semiconductor Lasers (CFSL)

Fred J Zutavern, Guillermo M. Loubriel, Alan Mar, and Martin W. O'Malley
Directed Energy Special Applications Department

Albert G. Baca, Michael J. Hafich, and G. Allen Vawter
RF Microsystems Technologies Department

Weng W. Chow
Semiconductor Materials and Device Sciences Department

Harold P. Hjalmarson
Computational Biology and Materials Technologies Department

Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185

contact: Fred Zutavern 505-845-9128, fjzutav@sandia.gov

A Final Report for SNL LDRD Project 10696

Abstract

A new type of semiconductor laser (patent pending) is demonstrated from optically-initiated current filaments in high gain photoconductive semiconductor switches (PCSS). The electron-hole plasma at the center of the current filaments in semi-insulating GaAs structures is used as the active region in optical cavities built around the filaments. Four device configurations were fabricated and results from two configurations have shown five properties which demonstrate optical amplification from stimulated emission: high intensity, temporal pulse compression, spectral narrowing, light versus current thresholds, and low beam divergence. These miniature electron-hole plasma based lasers are not limited in volume by the diffusion length that limits one dimension of conventional, carrier-injected semiconductor lasers. Their comparatively larger diameters imply higher energies and reduced divergence in diffraction-limited devices which are consistent with our "exploratory" results. Potential applications include active optical sensing and imaging, micro-machining, short pulse optical communication, and permanent optical memory.

Acknowledgements

The authors acknowledge Darwin J. Brown, Gary J. Denison, Wesley D. Helgeson,, and Luis L. Molina for their assistance in developing optical triggers, electrical pulsers, and other test equipment for PCSS and filament lasers. Their laboratory expertise, suggestions, and hard work are deeply appreciated.

Contents

Figures

(This page intentionally left blank)

Current Filament Semiconductor Lasers (CFSL)

The CFSL Concept

High gain GaAs photoconductive semiconductor switches (PCSS) were reported in 1982 as optically-activated high power electrical switches with characteristics similar to high power plasma breakdown gas switches (spark gaps).[1] Compared to conventional linear photoconductive switches, these devices required significantly lower optical trigger energy.[2,3] They also exhibited an interesting property called "lock-on." At low fields and after the optical trigger ended, carriers would recombine with a few nanosecond exponential time constant and the switch would return to its high resistance state. However, when triggered at fields above 10 kV/cm, the switches would remain in the low-resistance state as long as a minimum field (4-6 kV/cm) was maintained. Similar to a very high voltage Zener diode, these switches would conduct whatever current the circuit could supply while maintaining the constant "lock-on field" across the switch. Several groups studied these switches in the 1980s and 1990s to develop very large devices (with insulating gaps up to 3 cm long) for extremely high power, fast risetime, electrical applications (100 kV, 1 kA, sub-nanosecond).[4,5] Images of current filaments in high-gain PCSS (figure 1) were first shown in 1991 at the 8th International Pulse Power Conference.[6,7] Further reductions in the optical trigger energy requirement and the precise control of filament locations were demonstrated as higher total currents and longer-lived switches were developed. Eventually, collective avalanche carrier generation was proposed as the source of additional carriers and the explanation for high gain.[8] The ability to produce optically-initiated electron-hole plasmas with precise control of their initiation timing and location presented new opportunities to explore and

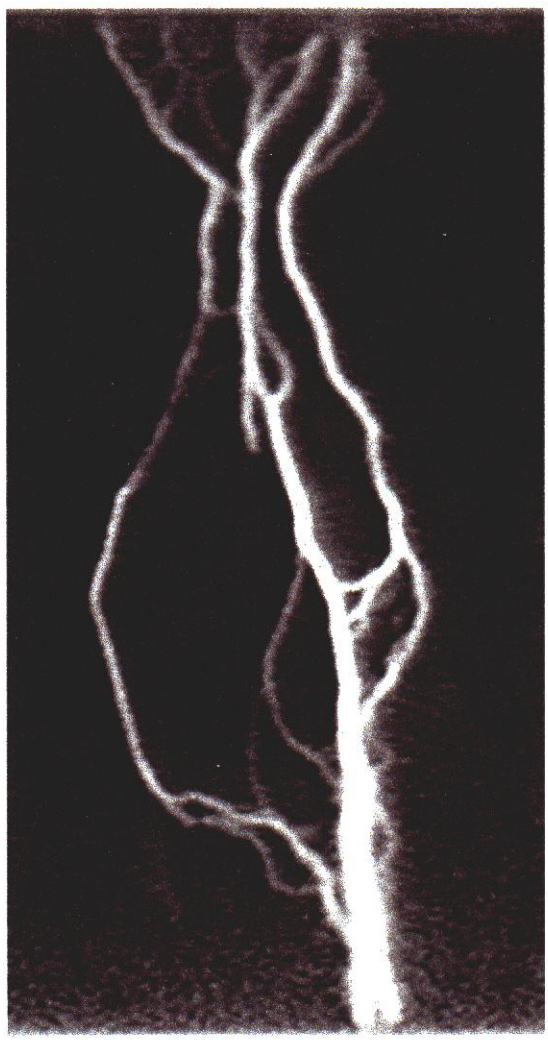


Figure 1. The picture on the right shows a current filament in a 1.5 cm long, high gain PCSS recorded with an open-shutter camera. The light that produced this image came from electron-hole recombination emitted at 880 nm during a 6 ns long current pulse. This switch was charged to 40 kV and conducted 350 kA when triggered. The filament was initiated with a 1 mm diameter optical pulse applied at the bottom of the photograph.

manipulate relatively high density plasmas embedded in a semiconductor lattice. The production of extremely straight line filaments for high current density switches (figure 2) led to a concept for a new type of semiconductor laser. Spontaneous emission from carrier recombination was producing images of the filaments and also being considered as an explanation for their rapid formation (growth velocities approaching the speed of light in GaAs)[9]. The new type of semiconductor laser (patent pending) was proposed when optical feedback was suggested by encompassing the filaments in a lasing cavity.

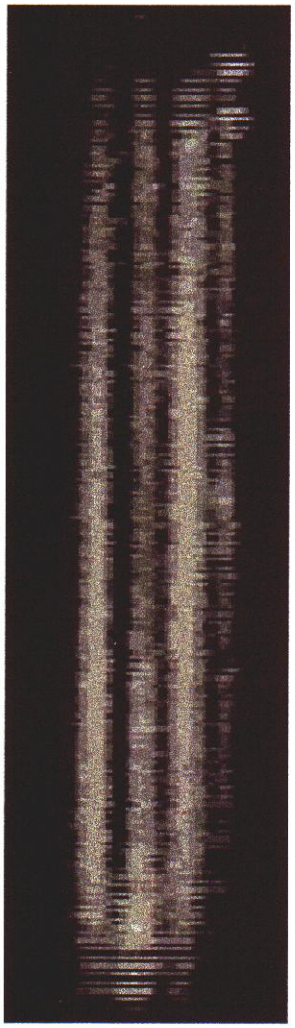


Figure 2. The image (left) contains four filaments in a 1.5 cm long GaAs PCSS showing precise control of the filament location and shape when triggering with optical lines. The filaments are less than 1 mm apart and each carry ~125 A.

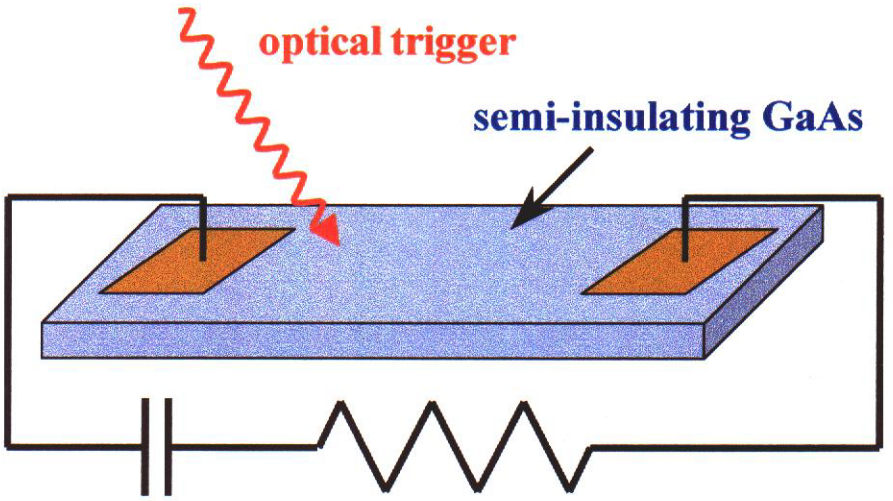


Figure 3. A typical lateral PCSS is shown above. The electric field and direction of current flow are parallel to the surface through which the carrier-generating light is absorbed. Energy from a storage element (capacitor or transmission line) is normally discharged through the switch into a resistive load.

At issue with the new concept for a CFSL was the problem of a carrier population inversion. Previous experiments on PCSS had set limits on the carrier density in the filaments at 10^{16}-10^{18} cm^{-3}.[10,11] However, a high density electron-hole plasma will not lase if the probability for absorption is greater than that for emission[12]. The carrier generation process during filament formation and "lock-on" is not well understood, primarily because it happens at extremely low fields (4-6 kV/cm) compared to the nominal bulk breakdown strength of GaAs (200-300 kV/cm) and other semiconductors. If carriers are ionized into the conduction band well above the band minimum where direct-gap (photo-emitting)

carrier recombination occurs, the filaments may not be capable of stimulated emission, and may actually have an increased probability of photo-absorption. On the other hand, if they cool rapidly and reach the bottom of the conduction band, the conditions may be ideal for lasing. Related research into electron beam pumped semiconductor lasers[13] was encouraging because in these devices, lasing had been demonstrated in many semiconductors using relatively high energy electron beams, an even higher energy excitation process than avalanche carrier generation which was believed to account for filament formation in PCSS.

The challenge to testing the CSFL concept was the fabrication of a lasing cavity around a current filament. High resistivity, semi-insulating GaAs must be used to hold-off the electric fields required to initiate (10 kV/cm) and sustain (5 kV/cm) the current filaments. Electrical contacts (figure 3) must be deposited on the device that can produce these fields and the supply the current in the filament once it has formed. Experience with high-gain PCSS has shown that contacts are critical to the stability and longevity of the device. The predominate location for device degradation is at the interface between the metallic contact and the semiconductor where an electric potential barrier will dissipate significant energy[14]. In addition to electrical contacts, the ends of the filaments must also be near surfaces which can provide optical feedback. High-gain PCSS are typically fabricated in a lateral configuration, where the current runs parallel to the surface of the substrate between two large area surface contacts at the ends of the device (figure 3). This configuration provides easy access to the surface through which filaments are optically initiated and spontaneously emit. The large area contacts at the ends of the filament, unfortunately, do not allow easy access to or observation of the optical emission along the axis of the filament. Long narrow contacts were proposed to give some access to the ends of the filament for observation and feedback, while still providing reliable electrical contacts for the filament. Our first approach was a lateral PCSS/edge-emitting-device hybrid where optical feedback was obtained by reflecting light from the cleaved device edges. After submitting a patent for current filament based light sources, results from this project were presented at several conferences and institutions[15, 16, 17, 18, 19, 20. 21, 22].

(This page intentionally left blank)

Device Fabrication

Over the course of this project, four different device configurations were developed. In general, lateral devices are the easiest to manufacture because processing is required only on one surface. However, CFSL require optical access through at least two orthogonal surfaces. A line of light incident on one surface is used to trigger a straight line filament. Optical emission along the axis of the filament is observed and controlled through a second surface, that may be coated with reflective or anti-reflective coatings to modify the intensity of the emission. Electrical contacts at the ends of the filament must be positioned so that they do not interfere with either the optical line trigger or the axial cavity emission.

First, the standard lateral PCSS configuration (figure 3) was modified to initiate filaments that would span the insulating region and be enclosed in the cavity formed by the cleaved edges of the device (figure 4). The contacts on these devices are very narrow so that the filaments must extend nearly to the cleaved edges. The substrate must be thinned

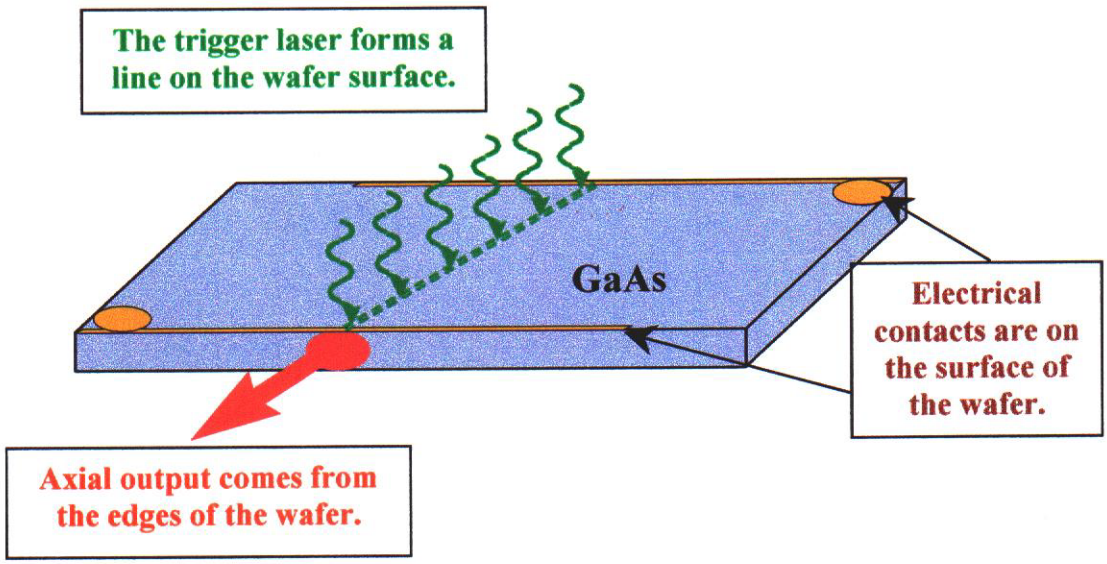


Figure 4. The modified lateral PCSS configuration for a CFSL is shown above. In this device a straight line filament is initiated parallel to its surface between metallic contacts with an optical line trigger. The cavity is formed by the cleaved edges of the wafer that have 30% reflectivity without coating. Axial emission is observed through the cleaved edges. The narrow contacts on the surface of the wafer determine the location of the ends of the filaments, which ideally should extend to the cleaved edges. The region below the contacts is shadowed from the optical trigger and causes current pinching as the filaments approach the edge of the contact. Since any carrier-free region strongly absorbs optical emission from the filament, a passive Q-switch may be formed under the contacts that must be bleached by strong optical emission along the axis of the filament.

to less than 150 microns by lapping to achieve a tight (50 micron) tolerance and crystallographically smooth mirror cleaves.

In the second configuration (figure 5), edge contacts were proposed to eliminate the regions which are "shadowed" by the contacts on the lateral devices and to make the filament reach both edges of the device. To make these devices, the wafers must be cleaved into bars with widths that are the desired cavity lengths. Then the bars are stacked on their edges and processed with ion implantation and evaporation to produce edge contacts. Because of the thinness of these devices, sample handling was impractical and successful implantation was never achieved.

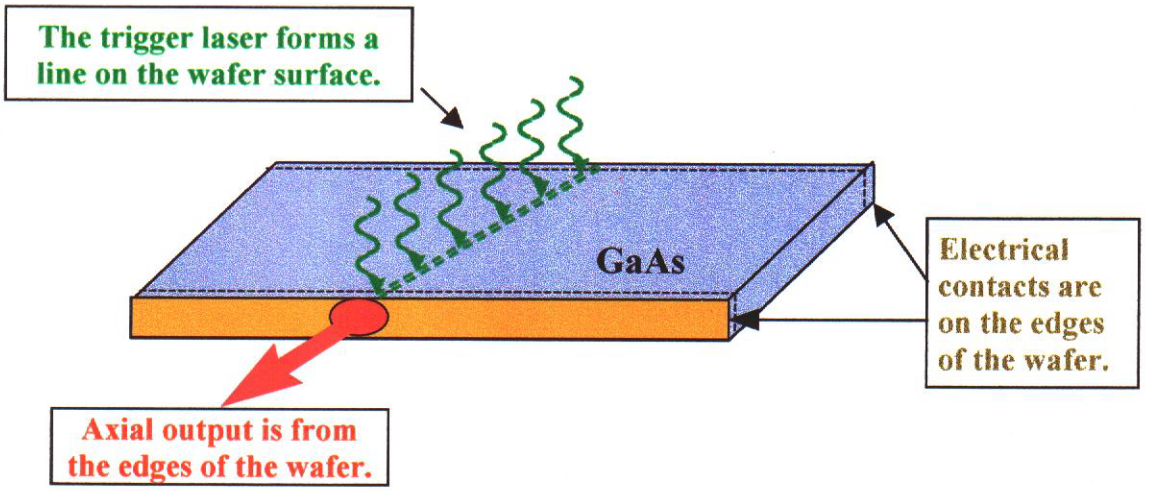


Figure 5. Edge contacts in the above configuration eliminate the "shadowed" regions under the contacts of the surface contacted lateral device (figure 4). Ion implanted layers at the edges with n and p dopants and metallized layers with holes for axial emission provide semi-transparent contacts for these devices. This type of contact avoids the potential current pinching and passive Q-switching that was characteristic of the surface contacted devices. The tedious handling of these thinned devices after cleaving and before ion implantation and evaporating made this fabrication impractical.

The difficulties involved in processing the edges of cleaved devices led to the third configuration, the quasi-vertical device (figure 6). In this device, p and n doped contacts were epitaxially grown on the opposing surfaces of a wafer. Metallic layers with small elliptical holes for axial light emission were evaporated onto the surfaces. At this point these would be surface emitting, vertical devices, with no orthogonal surface through which the filament could be triggered. Although axial triggering with wavelength tuned absorption was considered, issues were raised about the problems caused by exponential absorption in controlling the location of the filament of the entire thickness of the device. To allow optical line triggering from an orthogonal surface, the devices were cleaved through the centers of the holes and triggered on a cleaved edge under a hole, producing the "quasi-vertical" device.

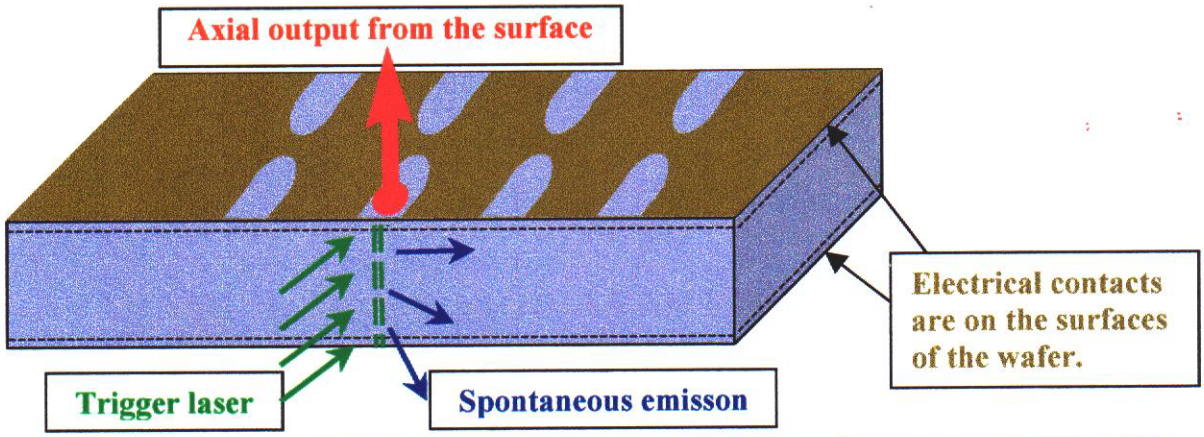


Figure 6. The quasi-vertical configuration is shown above. Epitaxially grown and metallized contacts are made on the top and bottom surfaces. One contact is patterned with elliptical holes to allow axial emission. A straight line filament is formed on the cleaved edge of the device with an optical line trigger. Like the previous configuration, this one eliminates current pinching and Q-switching near the contacts. Heavily doped n and p regions makes this a very long lived device. Although the processed surfaces are not as perfect as the cleaved edges of the previous configurations, the nature of processing makes this approach much more practical to manufacture than the others.

Finally, to achieve a simple form of optical and electrical confinement and improve charge density and beam uniformity, ribbed waveguide devices were developed (figure 7). In this configuration, we returned to the original lateral configuration and pits were etched into the surface of the wafer to form "ridges" or "ribs." If a filament is initiated down the center of a rib, the charge carriers and photons would be physically confined by the semiconductor boundary on three sides of the rib. Some of these devices have been fabricated, but testing was not possible before this project ended.

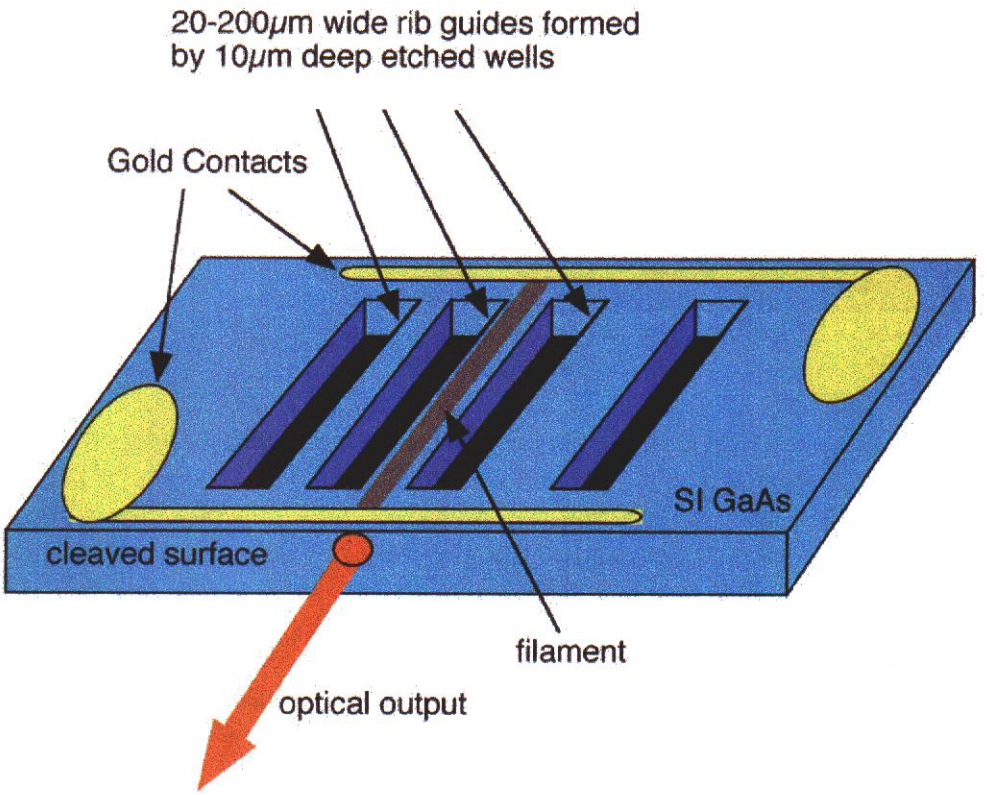
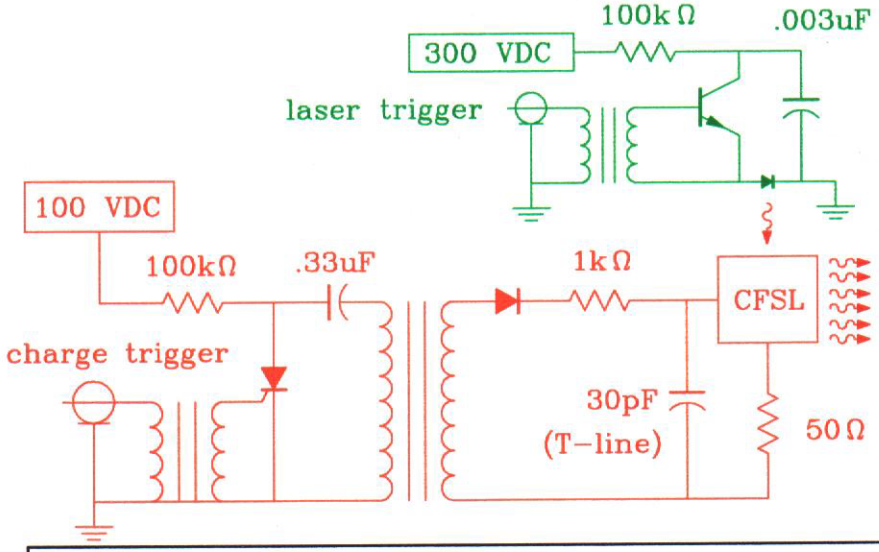


Figure 7. The ribbed configuration shown above is an approach to electrical and optical confinement. This design is similar to the original lateral PCSS, but with etched ribs. Etched rib waveguides provide physical confinement to improve carrier density and beam uniformity.

Device Testing

CFSL testing is similar to high gain PCSS testing except that the axial radiation is not accessible on a typical lateral PCSS. The CFSL is pulse charged to 5-30 kV/cm, so that filaments can be initiated. To initiate a straight line filament, an optical line trigger produced by a multi-mode, wide-stripe, edge emitting semiconductor laser. is applied to the surface of the device. Schematics of the electrical circuits are shown in figure 8. Solid state (frequency doubled Nd:YAG and Ti:Al_2O_3) and fiber coupled semiconductor lasers have also been used to trigger PCSS with straight line filaments. The narrowest (approximately 40 microns) and most convenient optical triggers were achieved by imaging a wide-stripe, edge-emitting semiconductor laser directly onto the surface of the CFSL. A small aspherical lens was positioned between the trigger laser and the CFSL to obtain enough magnification or reduction so that the optical line completely crossed the active region of the CFSL and slightly overlapped the contacts. A narrow filament that was initiate in this fashion on a PCSS is shown in figure 9.

Figure 8. Typical pulse charging and optical trigger circuits are shown below. These circuits are identical to those used for PCSS triggering. Current filament semiconductor lasers (CFSL) were tested to several kV depending upon their length (20-30 kV/cm) and up to 110 A. Contact and facet damage limited most of the testing to 30 A.



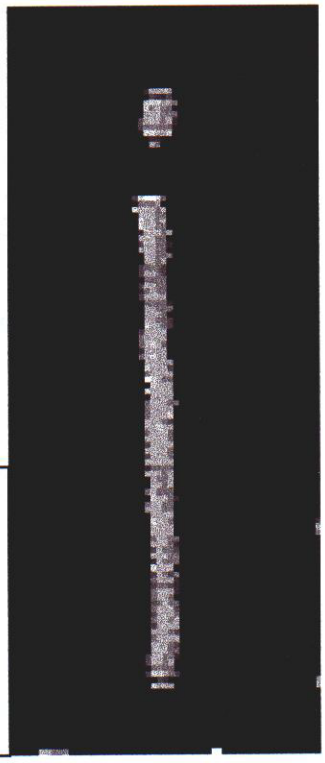


Figure 9. The filament shown on the right was triggered by imaging a wide-stripe multi-mode diode laser directly onto a small PCSS. The filament is 1 mm tall and ~40 μm wide. The spot at the top was the reflection from the solder. Switch voltage was 3.2 kV. Switch current was 22 Amps. Direct imaging with a small aspherical lens produced a much narrower trigger line than any other approach.

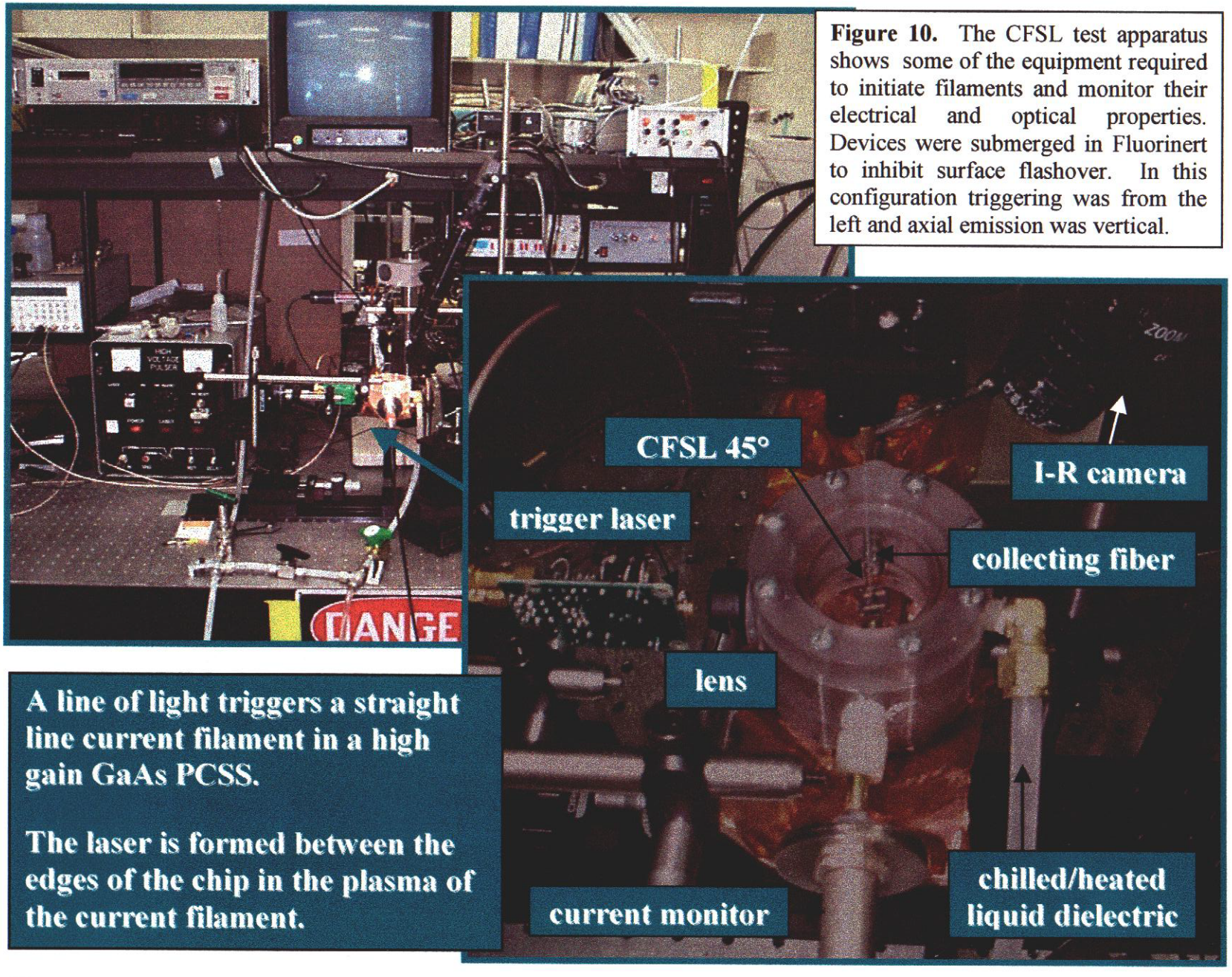


Figure 10. The CFSL test apparatus shows some of the equipment required to initiate filaments and monitor their electrical and optical properties. Devices were submerged in Fluorinert to inhibit surface flashover. In this configuration triggering was from the left and axial emission was vertical.

Photographs of a CFSL and some of the test equipment required for pulse charging, triggering, and diagnostics are displayed in figure 10. A high voltage pulse charger is connected to the CFSL with a low impedance transmission line and copper foil to reduce the inductance and produce fast risetimes when the CFSL is switched. The electrical current is monitored with a high bandwidth current viewing resistor (CVR) located near the CFSL. To inhibit surface flashover, the CFSL is submerged in a liquid dielectric (Fluorinert) that can be circulated and or chilled. The liquid is contained in a plastic enclosure with liquid fittings, electrical feed-thrus, and optical windows. The optical line trigger and a camera have line of sight paths to the orthogonal surfaces of the device to initiate the filament and image its surface and axial emission. A fiber optic is used to transport optical emission from the CFSL to a high speed photodiode or spectrometer.

Initial tests with modified lateral PCSS devices immediately revealed a bright spot of light being emitted from the ends of the filament. Axial emission from two such structures is imaged in figure 11. The first device was cleaved from a wafer was not thinned (600 microns thick). Both the substrate (dark vertical strip) and the spot can be observed in the image of the first device. Increased intensity and a larger diameter spot was observed from the second device which was thinned to 100 microns for improved cleaving. These photographs represent the first observation of the shape of cross-sectional

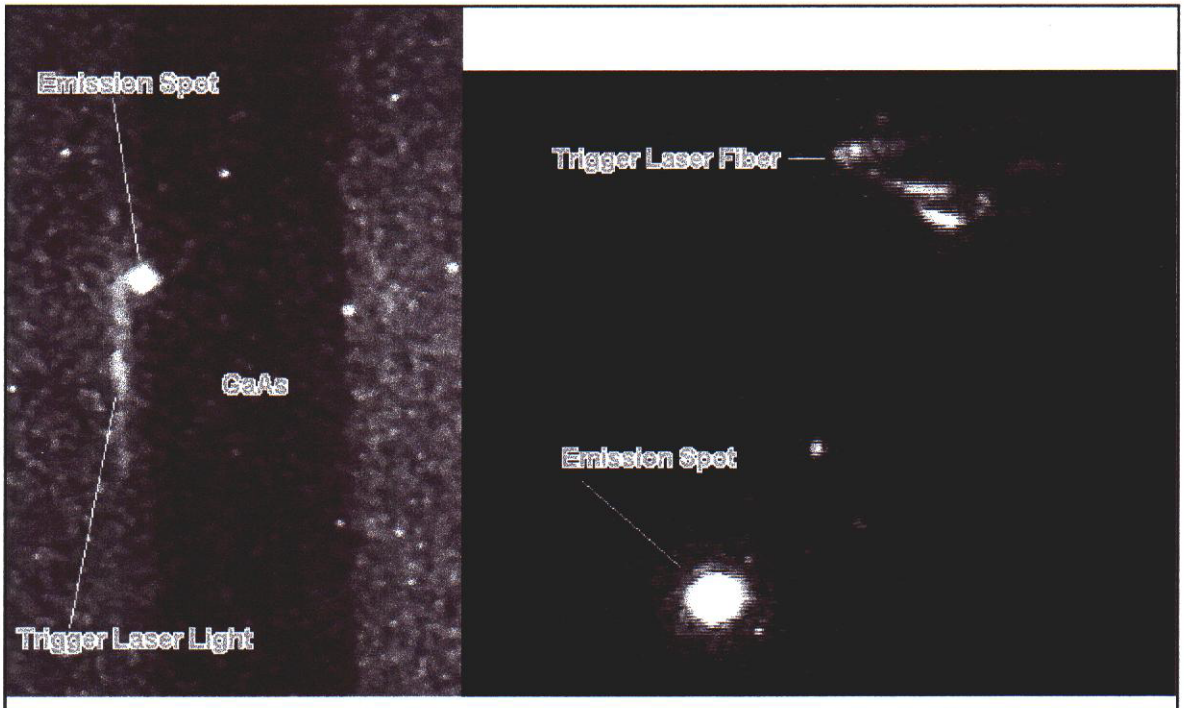


Figure 11. These photos show edge emission from the end of filaments in the modified lateral PCSS configuration. On the left is a 75 micron diameter filament in a 600 μm thick GaAs substrate. The device on the right, was thinned to 100 μm for improved cleaving. The spot is significantly brighter and nearly fills the substrate which is not visible. In both cases, the current was approximately 20 A.

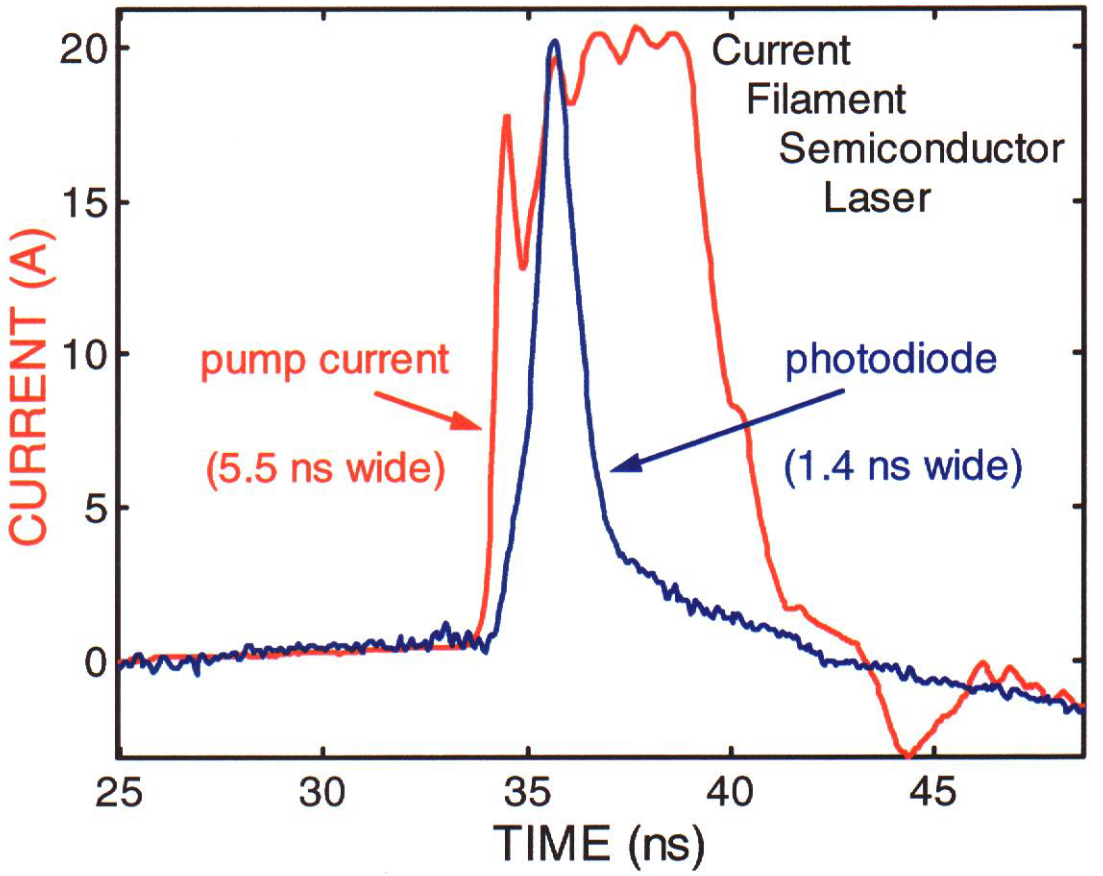


Figure 12. Electrical and optical (axial) pulse shapes from a CFSL are compared. Optical pulse widths ranged from 1 to 2 ns with an occasional second pulse independent of the current pulse width, which was 5.5 ns in this case. The oscillations in the current pulse are caused by the changing impedance of the CFSL which decreases as the filament grows and increases as carriers are lost to spontaneous or stimulated emission. Relative time delays are accurate to 0.5 ns. The small bump in the photodiode signal preceding the pulses is probably the trigger pulse.

shape of a filament in a lateral device. Prior to these observations, filaments had been assumed to be cylindrical. Although there was some speculation that their cross-sectional shape might be distorted by the surface of the device or exponential absorption of the trigger light.

The temporal shape of the axial emission was recorded with a high bandwidth (5 GHz) photodiode and transient digitizer (4 GHz). Figure 12 contains a comparison of the filament current recorded with the CVR and the axial emission. Although tests were performed with current pulses ranging from 5-15 ns, the optical pulse width ranged from 1 to 2 ns. Occasionally a second optical pulse was observed immediately following the first one. The oscillations in the current pulse were found to depend upon the action of the CFSL. As the trigger position was moved along the edge of a quasi-vertical device, the oscillations would change. The oscillations are apparently caused by the changing

impedance of the CFSL which would change the amount of the pulse being reflected back down the charging transmission line. As the filament grows, the resistance of the CFSL decreases, as carriers are emitted, the resistance increases. Similar behavior has been observed when a PCSS is driving a conventional semiconductor diode laser. The small bump in the optical pulse can be enhanced by changing the location of the collecting fiber. It is probably a reflection of the optical trigger pulse which initiates the filament.

The temporal shape of the surface emission from filaments in high gain PCSS has also been recorded. In contrast to axial emission, the pulse width of surface emission is always somewhat wider than the current pulse and is consistent with an exponential carrier recombination time of 10-15 ns. With linear photoconductivity, nearly every absorbed photon creates a carrier, so the light intensity can be considered a carrier generation rate and the surface emission is described by:

$$S(t) \propto \int_{-\infty}^{t} R(s)\, e^{-(t-s)/\tau} ds \qquad [1]$$

where $S(t)$ is the surface emission intensity
$R(t)$ is a carriers generation rate
τ is the carrier recombination time

This model for spontaneous emission follows from the assumption that the carrier recombination rate is proportional to the carrier density. The concentration of electrons and holes is assumed equal since the filament is charge neutral as both photo-absoption and avalanche carrier generation create equal numbers of electrons and holes.

When avalanche carrier generation (collective or not) or photo-absorption at high fields (that saturate the carrier velocity) are more important than charge injection from the contacts, the total current, $I(t)$, is a spatial average of the instantaneous carrier density throughout the active region of the device rather than a carrier generation rate. In this case, the surface emission from carrier recombination is proportional to the carrier density, until the pulse ends, i.e. the field is turned off, and carriers diffuse and recombine. This situation is described by:

$$S(t) \propto I(t) \quad \textit{during the pulse}$$
$$S(t) \propto I(s)e^{-(t-s)} \quad \textit{after the pulse} \qquad [2]$$

where $S(t)$ is the surface emission intensity
$I(t)$ is the total current
s is the time that the pulse ends

This model is consistent with the temporal shape of the observed for surface emission, although spikes have been observed which may be caused by weakly stimulated emission perpendicular to the filament or other unexplained phenomena.

Figure 13 shows some of the time-resolved signals using a fiber-coupled photodiode to measure optical surface emission from a lateral PCSS. The width of the plateau in the pulses is equal to the current pulse width, the spikes are normally at the onset of the pulse but have also been observed in the middle of the pulse. Potential explanations for the spikes may be single pass stimulated emission (no cavity), higher carrier density on the leading edge of the filament during formation due to a large field enhancement, depletion of trapped charge, a reduction in the carrier recombination rate due to increasing lattice temperature. The time evolution of surface emission is also being pursued with time-resolved spectroscopy, which may help explain filament formation and the existence of these spikes.

Spectra of the axial and surface emission from current filaments are plotted in figure 14. Axial spectra typical contain a 10 nm wide envelope of low level emission containing one or more large peaks. The precise location and intensity of the peaks varies

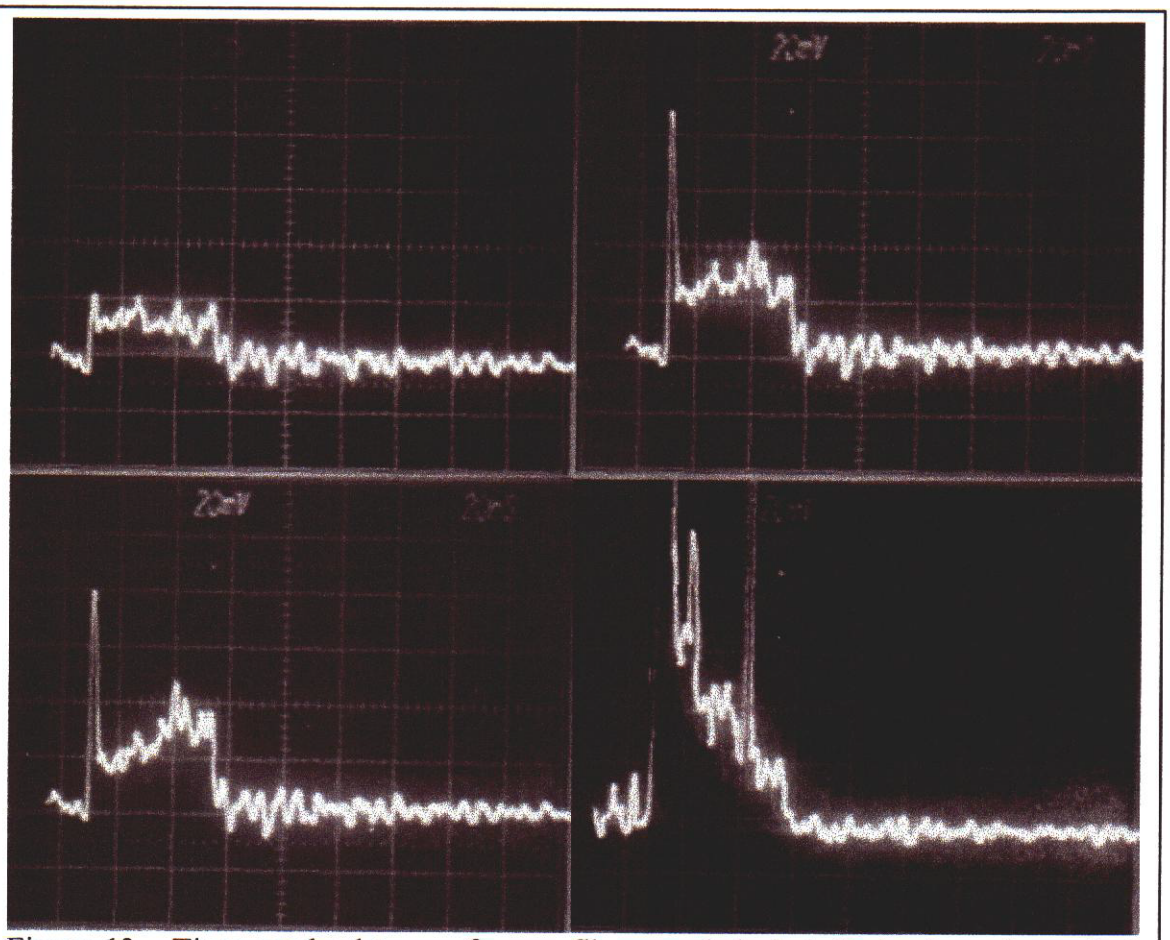


Figure 13. Time resolved traces from a fiber-coupled photodiode show the optical emission from the surface of a PCSS. The waveforms were recorded at 20 ns per division (horizontal) and 20 mV per division (vertical). The current pulse width was 45 ns, the width of the plateau. Two spikes in the lower right photo were off scale.

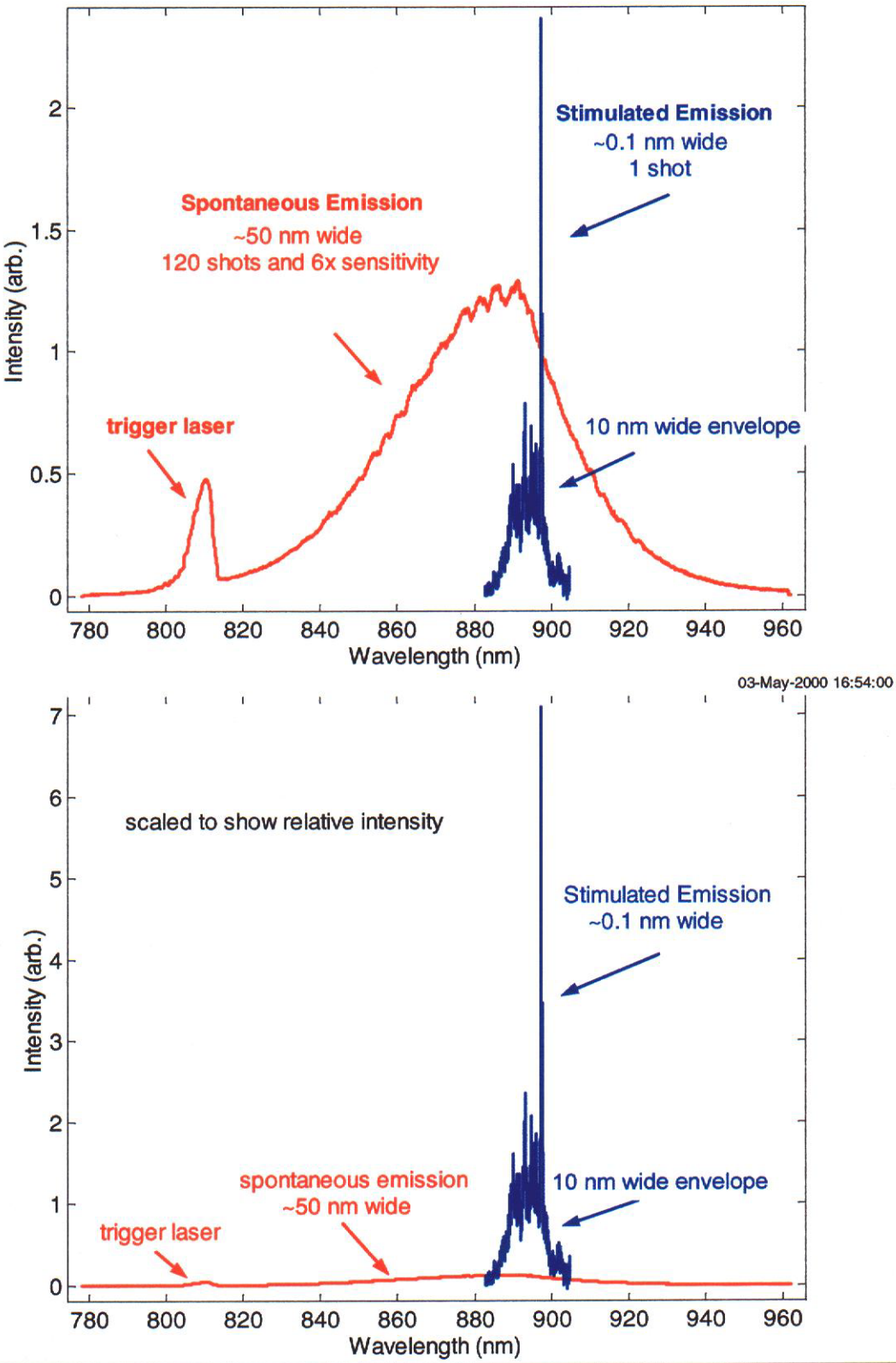


Figure 14. Spectra of axial (stimulated) and surface (spontaneous) emission from a quasi-vertical CFSL are compared above. The top plot is scaled to show relative shapes (broadening). The bottom plot is scaled to show relative intensities.

within the envelope on each shot. One of the narrowest peaks is shown in the figure. The surface emission is much weaker and approximately 50 nm wide. The axial spectrum was obtained with one pulse, while the surface spectrum is the average of 120 shots with a lower resolution grating that gave about 6 times more sensitivity. In the top plot, the spectra are scaled to similar magnitudes to compare their shapes. In the bottom plot, the curves are scaled to show their relative intensities. The surface emission from a filament is much broader than the zero field, 15 nm wide, photo-luminescence excited by the same trigger laser. If the additional width in the filament surface spectrum is due to the thermal broadening of the carrier distribution, then the carriers in a filament are significantly hotter than those in the ambient lattice (approximately 500 C). However, self absorption and a non-uniform carrier distribution complicate the interpretation of the spectra.

The energy in the optical pulse was most accurately measured by calibrating the high bandwidth photodiode with a short pulse, high repetition-rate, conventional semiconductor laser and a power meter. Once the energy per pulse in the semiconductor laser was determined with the power meter, its single pulse temporal waveform was measured with the photodiode. Calibration of the photodiode was achieved by integrating the single pulse waveform and selecting a scale factor that set the integral equal to the energy obtained from the power meter. Then the single pulse energy from the CFSL was measured by integrating its photodiode response and scaling it with the calibration factor. This procedure gave reliable energy measurements from 0.5 to 500 nJ. Photodiode energy measurements at the upper end of this range were in agreement with measurements made with an energy meter which gives reliable measurements from 0.050 to 1000 μJ. Both the conventional semiconductor laser and the CFSL were coupled to the photodiode with a 400 micron multi-mode fiber optic which was also used with the power meter. The largest uncertainty in this procedure was in collecting all the light from the lasers through the fiber in the photodiode. This was accomplished with a small diameter aspherical lens by maximizing the photodiode signal.

Optical pulse energies and filament currents were recorded with several different CFSL. The highest energy observed was 75 nJ from a modified lateral PCSS structure that was 2.5 mm long with a 40 A, 20 ns current pulse. At this time, the optical energy was measured with a 400 micron multi-mode fiber into a pulse energy meter with a minimum range of 20 nJ. Several readings of 30-75 nJ were observed before the CFSL failed. This device had no coatings on its cleaved edges. Assuming the same energy was being emitted from both ends, the total axial optical energy was 150 nJ. All of the CFSL tested showed large amplitude variations. The filament path initiated with spot triggering was very crooked and undefined. Imaging the wide-stripe trigger laser (5 X 200 μm) directly through an aspherical lens onto the CFSL produced a tightly focused line trigger (figure 9) and the least amplitude variation. The quasi-vertical (QV) structures seemed to show the the most reproducible operation. The maximum energy recorded from one end of these 0.5 mm long QV devices was 40 nJ. Measurements were made on QV CFSL with different reflective coatings to determine how the optical feedback affected their output energy. Figure 15 shows the output energy as a function of filament current for three QV CFSL

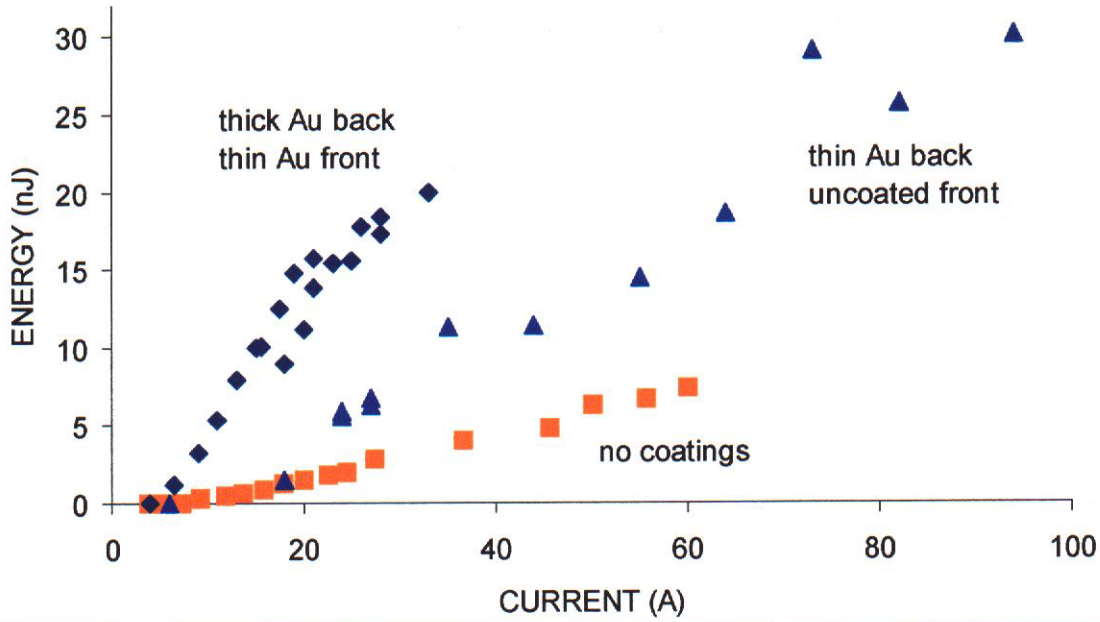


Figure 15. The effect of optical feedback is displayed above where the axial emission energy is plotted as a function of filament current for three different quasi-vertical CFSL with different facet reflectivities.

with different types of coatings. These light-current curves leave the current axis at approximately 6 A and have slopes that depend on the reflectivities indicated in the figure. In all cases surface emission indicated the formation of filaments at slightly lower currents (4-5 A). These tests were selected from many because they had the least amplitude variation. The maximum optical energy was plotted for each current value.

Initial measurements of axial emission divergence were made on 2.5 mm long modified lateral PCSS devices. Measurements were uncertain for several reasons. There was significant shot to shot variation in the axial output energy. Beam diameters were estimated by measuring the spot size of a small number of images recorded directly with a an infrared sensitive camera. To collect enough signal in each camera pixel, the camera was very close to the CFSL which generates strong, high-bandwidth electro-magnetic interference (EMI). Nevertheless, a measurement of 2-5 degrees was determined with estimated uncertainty. Later, when performing tests on a 0.5mm long CFSL, the divergence measuring techniques were improved. A 25.4 mm diameter, 50.8 mm focal length, achromatic lens was positioned between the CFSL and the camera a twice the focal length (100 mm) from each. This 1-to-1 imaging configuration (shown in figure 16) moved the camera farther away from the CFSL, reduced the EMI, and allowed a greater range in the distance over which the beam size could be recorded. For one measurement of the divergence, 10 images were recorded at 14 locations to verify the location of the image and get accurate measurements of the beam diameter as it diverged. A typical image of the axial emission recorded in the 1-to-1 imaging configuration is shown in figure 17. For each image, the diameter of the circle with the same area as the 37 % (1/e), 50 %, and 90%

contour is plotted versus camera position in figure 18. The divergence is defined as the full conical angle formed by the diverging beam:

$$divergence = 2\tan^{-1}(\frac{slope}{2})$$
$$slope = \frac{change\ in\ diameter}{change\ in\ position} \qquad [3]$$

The average divergence (using 1/e contours) was 7 degrees with an estimated uncertainty of ±1 degree. The smaller divergence obtained previously, while less accurate, may still be

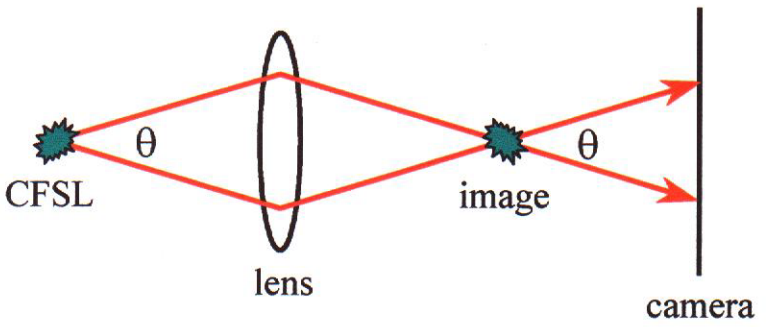


Figure 16. This set-up was used to measure the beam divergence of the axial emission from a quasi-vertical CFSL. A 50.8 mm focal length, achromatic lens in a times 1 magnification configuration moved the image closer to the camera. This reduced the EMI produced by the CFSL and allowed greater range in the camera position without reducing the energy absorbed per pixel. Images of the beam were recorded 199-205 mm from the CFSL. Because the lens inverts the image, it actually increases the beam divergence slightly, but this is a small effect as long as the object diameter (60 microns) is small compared to the focal length of the lens.

valid for those devices because their device cavities were 1 mm long compared to the 0.5 mm long cavities tested more recently.

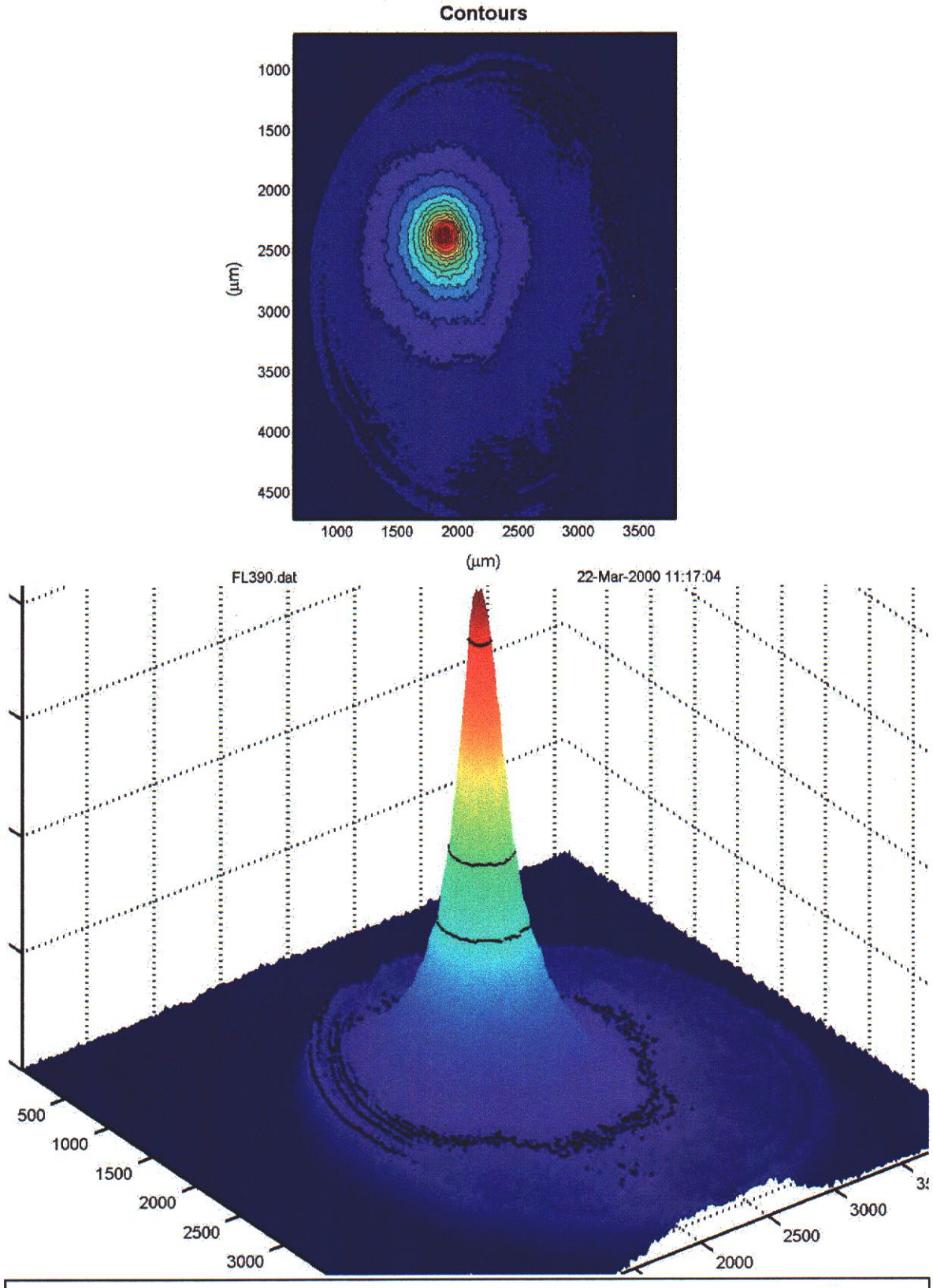


Figure 17. Contour and profile images for the beam from a quasi-vertical CFSL. X and Y axes are beam dimensions in microns. Such images and their locations (figure 16) determined beam diameter and divergence (figure 18). Profile contours are marked at 1/e, 0.5, and 0.9 of peak. The beam diameter is the circle that fits the 1/e contour.

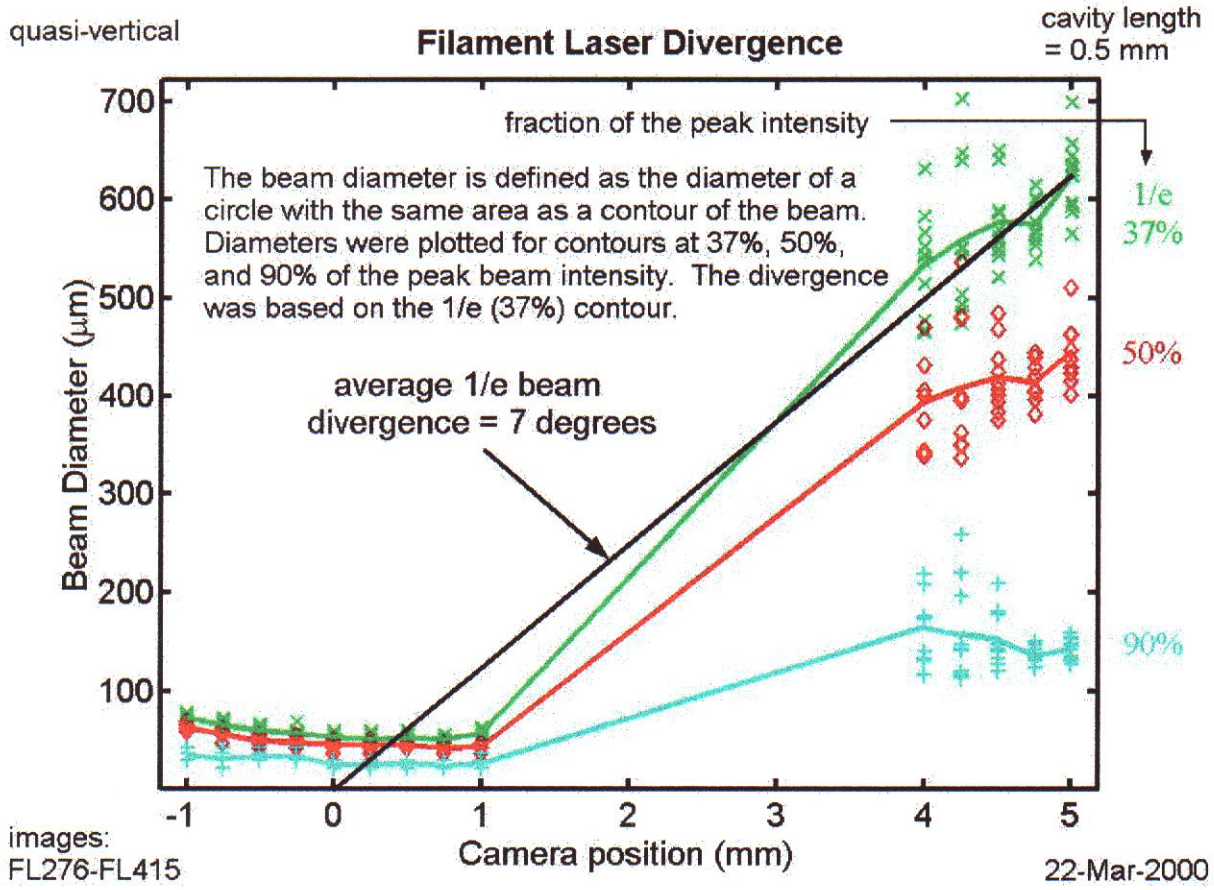


Figure 18. The divergence of the axial emission from 0.5 mm long, quasi-vertical CFSL is calculated in the above plot. Each point of a particular color or shape represents the beam diameter obtained from an image (see figure 16) recorded with the camera at the specified position (see figure 15). The points grouped by color or shape represent the diameters corresponding to contours at 37%, 50%, and 90% of the beam intensity (see figure 16). The segmented lines connect the average diameters of ten shots at each camera position. The straight line shows the average slope (change in diameter divided by change in camera position) from the center of the facet image.

Interpretation of Results

The measurements shown in the previous section show many properties that are characteristic of lasing: high on-axis optical intensity, spectral narrowing, pulse compression, lasing thresholds vs reflectivities, and small beam divergence. These measurements demonstrate a new class of semiconductor laser that can potentially produce a brighter, higher energy optical pulse than conventional semiconductor lasers (CSL). Because these lasers have a very different carrier generation mechanism, they also have very distinct operating properties.

Unlike CSL, this new laser is not limited in volume or aspect ratio by the depth of a p-n junction or current injection that is controlled by the diffusion length of the carriers in GaAs (1-2 μm). In CFSL the population inversion for lasing is produced by carrier generation that creates an electron-hole plasma in the form of a current filament. The current filaments have been shown to form at nearly the speed of light[7]. In large high gain PCSS, they have been as long as 3.4 cm with diameters of several hundred microns. At high currents (up to 2.5 kA per device), many filaments form, filling the active region of the switch which can be several centimeters wide. It is difficult to determine a specific diameter as the filaments merge together. With much smaller filaments in 2.5 mm long lateral devices we have recorded 75nJ of 890nm radiation from one end in 1.5ns (50W peak). This is approximately ten times more energy than has been produced by CSL in the same time interval.

Evidence for lasing is found in the spectra of optical emission. Figure 14 shows that the surface or spontaneous emission is approximately 50 nm wide while the axial or stimulated emission is 0.1 nm. This spectral narrowing occurs because the probability for stimulated emission is much higher than for spontaneous emission over a very small spectral range. The excitations that would normally lead to spontaneous emission in other regions of the spectrum participate in stimulated emission when a significant optical path or optical feedback is present. Multiple shot spectra from the CFSL show that the location of the peak in axial emission varies within a 5 nm range. This is probably due to the statistical and chaotic nature of filament formation. Since the carrier density and effective cavity shape varies from shot to shot, the optimum lasing wavelength and effective cavity lengths change, causing the observed variation in wavelength peak emission.

Non-uniform carrier density within individual filaments also explains other features of the CFSL axial spectra. Even with line triggering (figures 2 and 9), to direct the filament formation, the carrier density is probably highly non-uniform in the axial and radial directions with fine structure that changes in every pulse. Such non-uniformity effects cavity gain and effective path length due to carrier density induced variation in the refractive index. A 10% variation in the average carrier density along the light path can cause a full mode shift in the spectrum due to the induced change in the refractive index. Some axial spectra from the CFSL are vary narrow (0.1 nm) while others are as wide as a few nanometers. Also no clear evidence of mode structure has been observed. Mode

spacing for the 1 mm long CFSL should be 0.2 nm. Our spectral resolution was 0.05 nm. Although fine structure was observed in each spectrum, there was no evidence of a 0.2nm period. Single mode cavities typically show mode structure if the modes are closely spaced (relatively long cavities) and multiple modes fit in the spectral range of lasing. The absence of mode structure in the spectra from the CFSL (with relatively long cavities 0.5-2.5 mm) and the variation in line width indicate that multiple modes are present in the cavity. This is probably due to multiple paths in the cavity with varying effective path lengths due to the non-uniform carrier density which wash out the mode structure and broaden the spectral peaks.

More evidence for lasing is found in the optical output energy versus current plot shown in figure 15. Light versus current curves from lasers show thresholds and slopes that depend on cavity losses and optical feedback from features such as mirror reflectivities. Curves are shown in this figure for three quasi-vertical CFSL with different combinations of facet reflectivities: device A - no coatings, device B - a thin gold coating on the back facet and no front facet coating, and device C - a thick gold back facet and a thin gold front facet. Device A shows the smallest slope which is consistent with being the device with the least optical feedback and greatest cavity loss. Its threshold location is too uncertain to make a valid comparison to the other devices. Device C shows the greatest slope compared to devices A and B and lowest threshold compared to device B. This is consistent with device C having the most optical feedback and smallest cavity loss. Shot to shot variations make it difficult to get accurate data for these comparisons. The data shown in this figure represent the highest values obtained at each current, i.e. the pulses that produce the optimum conditions for lasing.

Confinement of carriers and photons has improved carrier density uniformity and device performance in CSL. The development of double heterojunction lasers from 1971-1980 from the original single heterojunction devices reduced lasing thresholds, increased intensity, and limited the lasing modes. Physical confinement produced by the ribbed device (figure 7) should produce similar improvements in the operating properties of CFSL. Less shot-to-shot variation is desirable for almost all potential applications and would produce more accurate measurements of light versus current thresholds and slopes. More uniform carrier density distribution should produce narrower spectra and increase the possibility of observing mode structure,

The relatively small divergence of these lasers may simply be a result of their cavity length to diameter ratio. Similar to miniature solid-state lasers, the cavities formed by current filaments tend to be cylindrical. Even though carrier density non-uniformity may cause multi-mode spectra, the spatial properties of the cavities may limit spatial modes and hence lower divergence. Since these cavities can be much larger, lower divergences may be obtained, and the diffraction limit (determined by the smallest aperture) can also be orders of magnitude smaller than the micron-size dimension of CSL.

The beam profile shown in figure 17 is also a measure of the current filament average cross-section. These measurements represent the first observations down the axis of a high gain current filament. Prior to these results their had been some speculation that the shape of the current filament below the surface of the substrate might be elliptical or distorted by the presence of the semiconductor surface and exponential absorption of the trigger photons. The circular shape of the beam profile leads one to conclude that the current filament cross-section is also probably circular.

In addition to testing the ribbed devices, several other experiments could give more useful information about the CFSL. The fact that the optical pulse only lasts 1.5-2.5 ns independent of the current pulse width (6-15 ns) presents questions about what spoils the lasing conditions. There has been some indication that higher current filaments produce longer optical pulses. This observation should be carefully explored with experiments that look closely at the pulse width dependence on filament current to help us understand the conditions of lasing in a filaments. For example, most explanations that use thermal mechanisms would predict a decreasing optical pulse width as the filament current is increased. Time-resolved measurements of surface (spontaneous) emission and axial (stimulated) emission could also provide essential information to model filament formation and lasing. Beam polarization is another aspect of lasing that was not explored. Even though a cylindrical filament channel may not produce a polarized beam, the nearby surface and other features in the cavity may affect the polarization. The maximum optical energy achievable with higher currents and multiple filaments was not determined. Multiple filament lasing was not tested. All of these experiments and tests can be achieved with systems that have been set up to monitor filament formation. However, they did not fit within the resources of this project and are being proposed for future activities.

(This page intentionally left blank)

Theory for Current Filament Semiconductor Lasers

The theory for the CFSL is based on the collective impact ionization (CII) theory for the filaments. In this section, the essence of this theory and its revisions are described.[8, 23, 24]

The CII theory was formulated to explain the formation of filaments. There are at least three important empirical observations addressed by the theory. One, the electrical field, approximately 10 to 30 kV/cm, required to trigger the filaments, is low compared with the field, approximately 500 kV/cm, associated with single particle avalanche breakdown of GaAs. Two, electrical current is carried by highly conductive filaments. Three, the highly conductive filaments are created rapidly, within nanoseconds of optical triggering. This empirical information led to the development of the CII theory. In agreement with other theories, the electron-hole plasma in the filaments is assumed to be generated by impact ionization involving electrons and holes heated by the electric field. The hot electrons, once they reach energies approximately one bandgap above the conduction band edge, excite electrons from the valence band to the conduction band. This process generates two charge carriers, the electron and the hole left remaining in the valence band.

The CII theory extends earlier theories by considering the density dependence of this phenomenon. The density dependence follows because energy redistribution occurs by carrier-carrier scattering. This energy redistribution broadens the distribution and extends it to higher energies. Furthermore, the larger occupation of the high energy portion of the distribution leads to more impact ionization compared with the low density distribution.

The essence of the theory is due to the more efficient impact ionization at high carrier density. This allows the formation of current filaments because a low density and a high density plasma can coexist at a given electric field. The high density plasma inside a filament causes rapid generation of electrons-holes; this phenomenon is consistent with the existence of the high density plasma. Outside the filament, the low density plasma is unable to generate carriers by impact ionization, a situation consistent with the low density plasma. The tendency of these carrier densities, inside and outside the filament, to remain the unchanged is consistent with the formation of a stable filament.

The first implementation of this theory assumed that energy redistribution within the dense plasma is very effective. Thus the plasma could be regarded as having reached a quasi-equilibrium temperature much higher than the lattice temperature. Continuum calculations based on this theory were found to produce surprisingly good agreement with current and voltage measurements of mature filaments.

This first implementation of the theory predicts that lasing cannot occur in filaments because quasi-equilibrium is assumed. This conclusion also follows from the

lasing criterion expressed in terms of quasi-Fermi levels. This criterion states that the difference of the Fermi levels must exceed the semiconductor bandgap. However, the quasi-equilibrium approximation leads to identical quasi-Fermi levels that lie within the semiconductor bandgap. Thus the first implementation of the theory could not predict lasing.

Further theoretical work showed that this implementation of the theory is inconsistent with the formation of filaments in GaAs. To be specific, this new analysis used the empirical observation of the field that sustains the filaments, the lock-on field, to do a consistency test of the theory. The assumption of quasi-equilibrium allowed a calculation of the carrier temperature associated with the lock-on field. This carrier temperature, however, was found to be much too low to produce enough carriers to justify a distribution characterized by a single temperature. The lock-on field, approximately 5 kV/cm, was thus found to be inconsistent with the formation of a filament characterized by a single temperature. This analysis predicted that the lock-on field should be much higher.

These steady state calculations stimulated a revised theory for GaAs. This revised theory relies on the details of the GaAs bandstructure to produce efficient redistribution of carriers within the conduction band. In particular, this theory relies on the presence of the indirect heavy-mass L and X points as collectors of field heated electrons. These electrons then boost other electrons to higher energies, leading to occupation of the states that can cause impact ionization. The net result is a non-thermal distribution that is more effective at impact ionization than the thermal distribution characterized by a single temperature. Furthermore, the initial quasi-equilibrium theory is now assumed to be the basis for the formation of filaments following electrical breakdown in generic semiconductors. It is thought that this breakdown and filament formation may lead to filaments that then lead to thermal breakdown and subsequent destruction of samples.

The new theory is being tested in time-dependent solutions of the Boltzmann transport equation by using Monte Carlo methods. Two types of carrier-carrier scattering are being considered in this theory. The generic scattering of two carriers with small momentum transfer is a basic mechanism that is nearly independent of bandstructure. In addition, the carrier-carrier scattering, usually involving larger momentum transfer, that follows from the unique features of the bandstructure is also computed. However, these latter types of scattering have been difficult to include in the simulations. It is likely that the Monte Carlo calculations will lead to more insight into the filament lasing. For example, the optical properties of the filaments are computed as a function of electric field, carrier density, lattice temperature, and other parameters. The new calculations will be compared with measured optical spectra in future work.

Finally, a continuum theory of the filament laser is also being developed. The essence of this theory is that population inversion occurs because carrier-carrier scattering populates high energy states in the GaAs conduction band. The energy source for this inversion is the electric field heating of electrons to the L and X points of GaAs.

Future Tests and Improvements

During the course of this project, there were many ideas generated to test and improve the operation of the CFSL. These ideas and some of the work initiated to pursue them are documented in this section. Whether they can be completed depends on future research and development of this device, for they are beyond the scope and/or resources of this project.

To test the potential for a much larger active region in the CFSL, compared to conventional semiconductor lasers, longer cavities and higher currents (which produce larger diameter filaments) should be pursued. The quasi-vertical cavities of figure 6 were limited to 0.5 mm in length due to the standard thickness of the substrate. Some 1 mm material has been ordered, p and n type growth has been completed on opposite surfaces, and some devices have been patterned, but they still need to be cleaved and tested. Cleaving is more difficult with thicker substrates, saw cutting and polishing may be necessary for even thicker material. Higher currents are more likely to damage contacts and produce optical pulses that may damage the reflective coatings and surfaces at the ends of the cavities. Shorter current pulses and multiple closely space filaments are suggestions to allow increased current and produce a larger lasing volume without increase contact or facet damage. Another approach to longer cavities is to return to the modified lateral PCSS design of figure 4. These cavities can be made arbitrarily long as we have created filaments up to 34 mm long. The contact damage is not as critical to lasing because it is not in the lasing path. However, it is harder to make long lifetime contacts in this configuration, and the device may be limited by contact damage. Operation of this type of device was not as stabile as the QV devices, but they did produce the highest pulse energies. The ribbed device, which will be discussed below, is a variation of this configuration that may help stabilize operation.

There is also a potential Q-switching effect caused by the contact "shadowed" regions near the ends of the cavities in the modified lateral PCSS configuration. The trigger light does not penetrate the contact metal, so a filament is not "seeded" below the contacts. The since the electric field curves sharply up from the substrate to reach the contacts, the filaments also bend up to the contacts and do not reach the cleaved facets of the device. This means there are small regions near both contacts that are strongly absorbing (no carriers) and could act as passive Q-switches in the lasing cavity because they must be bleached before net cavity gain can be realized. The effect of this potential Q-switching could be explored by testing devices with varying contact widths to effectively vary the volume of the Q-switch or delay to bleaching. Q-switching produces shorter, higher intensity optical pulses, which may improve operation and be useful for short pulse applications that are non-linear in optical intensity.

Several suggested measurements or concepts require new test apparatus (figure 19) for precision alignment of the CFSL. On-axis triggering is a method to produce a straight line trigger with a small circular beam profile. The idea is to adjust the absorption depth

and focus of a circular trigger laser so that the beam penetrates the device, leaving seed carriers from the incident to exit surfaces. This approach would produce a very straight line of seed carriers and may produce more uniform carrier density in the filament. Testing this idea requires optical access to both surfaces of the device and precision alignment and tuning of the trigger laser and/or substrate temperature. Another suggestion is to make a direct measurement of the amplifier gain from a CFSL to obtain more fundamental information about the lasing properties of these devices. This would require propogating a small 890 nm beam down the axis of a straight line filament that could be triggered with an on- or off-axis trigger laser. If the same wavelength could trigger the device and also be amplified, then a wide or double on-axis pulse could be used to test on-axis triggering and amplifier gain. Finally, the possibility of using the current filament as the active region in a larger cavity with off-substrate components has been proposed. This would allow more freedom in selecting facet reflectivities, higher quality reflecting surfaces, lasing mode control with spatial and spectral filters, and beam expansion to avoid facet damage. Although such a cavity may not be practical for many compact or environmentally harsh applications, it would provide an excellent opportunity to learn more about the potential of this device and determine which aspects should be commercially developed.

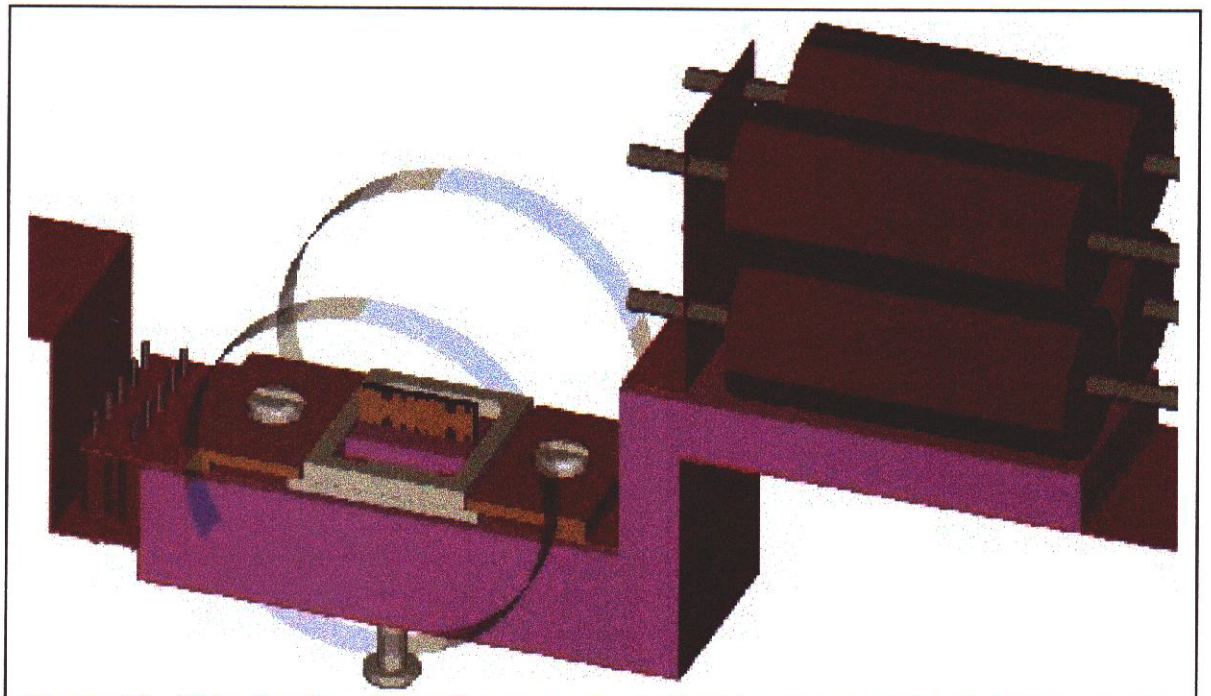


Figure 19. This drawing shows the apparatus designed to test on-axis triggering, optical amplification (gain), and lasing cavities consisting of off-substrate components. The vertical walls of the holder to contain the coolant are cut-away to show the lenses or windows (transparent cylinders), the device, and its holder. The opaque cylinders on the left and right are low inductance resistors that limit t and monitor the filament current.

Probably the most critical aspect of CFSL operation is its stability. Amplitude variation and/or time jitter generally cause severe problems for most applications. Single mode operation is also desirable for some applications. The best idea that has been proposed to improve CFSL stability and promote single mode operation is to increase the gain and carrier density uniformity with electrical and optical confinement. The development of conventional semiconductor lasers went through an eight year process change (roughly 1971-1979) in going form single to double heterojunction lasers to improve confinement and hence operating stability. The ribbed device discussed previously (figure 7) is a simple approach to improve confinement. These devices have been fabricated and should be tested to determine the significance of this type of physical confinement. Other types of confinement, that have been tested with conventional semiconductor laser, should also be tried. Some development will probably be required to insure that these processes are compatible with the high electric field and high resistivity that must be maintained by the CFSL. If the dramatic improvements that were achieved with confinement in conventional semiconductor lasers can also be achieved with the new CFSL, then their potential as compact, short pulse lasers can be realized in many military and commercial applications.

Other experiments and tests which were suggested while interpreting the results from this project include measurement of: the optical pulse width versus filament current, time-resolved emission intensities and spectra, beam polarization dependences, and devices with multiple filaments for lasing.

(This page intentionally left blank)

Conclusions

Compact short-pulse lasers are important for: active optical sensors (LADAR, range imaging, imaging through clouds, dust, smoke, or turbid water), triggering high power electrical switches, direct optical ignition of fuels and explosives, permanent optical recording, and micro-machining. We have invented a new class of semiconductor laser that can potentially produce brighter and higher energy optical pulses than conventional injection-pumped semiconductor lasers because this new laser is not limited in volume or aspect ratio by the depth of a p-n junction. We have tested current filament semiconductor lasers that have produced 75nJ of 890nm radiation in 1.5ns (50W peak), approximately ten times more energy than conventional semiconductor lasers. These lasers are created from current filaments in semi-insulating GaAs and, in contrast to conventional semiconductor lasers, are not based on current injection. Instead, low-field avalanche carrier generation produces a high-density, charge-neutral plasma channel with the required carrier density distribution for lasing.

Lasing has been demonstrated by measuring five characteristic properties: high beam intensity, pulse compression, spectral narrowing, lasing thresholds, and low beam divergence. The distinct properties of filament formation influence some of the properties such as multi-mode spectral and low beam divergence. Devices have been designed, fabricated, but not tested to improve operation by physical carrier and photon confinement. The development of conventional semiconductor lasers has taken forty years. Our results were achieved in less than three years.

In future experiments, carrier confinement and other fabrication techniques that have been successfully applied to CSL should be applied to CFSL. Much larger devices should be tested to see if high total energies and lower divergences are achieved. GaN and other semiconductors should tested to see if filament formation and similar light emitting and lasing devices are possible. The possibilities tested during the two and one half year duration of this project were very limited and the potential of the class of semiconductor lasers has only begun to be realized. Further diagnostics to resolve the electrical and optical properties of the current filaments in space, time, and energy are being developed. The evolution and spatial and energy distribution of the filament carrier density is a difficult problem to model and more information is critical to the understanding of these phenomena. Some of this information could be obtained with measurements of the optical pulse width versus filament current, time-resolved emission intensities and spectra, beam polarization dependences, and devices with multiple filaments for lasing. Plasmas in general are very difficult to control because of their statistical and chaotic formation and their high density and energy. The current filaments in high gain PCSS are a unique exception because their timing and location can be controlled very precisely with the optical trigger. They are an ideal environment for exploring and influencing the properties plasmas.

(This page intentionally left blank)

References

1. S. Williamson, G. F. Albrecht, G. Mourou, "Laser triggered Cr:GaAs HV sparkgap with high trigger sensitivity," *Rev. Sci. Instrum.*, vol. 53, June, 1982, pp 867-870.
2. D. H. Auston, "Picosecond optoelectronic switching and gating in silicon," *Appl. Phys. Lett.*, Vol. 26, Feb., 1975, pp 101-103.
3. C. H. Lee (ed.), *Picosecond Optoelectronic Devices*, Academic Press, New York, 1984.
4. A. Rosen and F. J. Zutavern, (eds.), *High-Power Optically Activated Solid-State Switches*, Artech House, Boston, 1993.
5. See proceedings from Pulsed Power, Power Modulator, and Optically Activated Switching Conferences, IEEE and SPIE, 1985-2001.
6. F. J. Zutavern, G. M. Loubriel, M. W. O'Malley, W. D. Helgeson, D. L. McLaughlin, "High gain photoconductive semiconductor switching," *Proc. 8th IEEE International Pulsed Power Conference*, R. White and K. Prestwich, eds., (IEEE, NY, 1991), San Diego, CA, June 16-19, 1991, pp 23-28.
7. R. Aaron Falk, Jeff C. Adams, Gail Bohnhoff-Hlavacek, "Optical Probe Techniques for Avalanching Photoconductors," *Proc. 8th IEEE International Pulsed Power Conference*, R. White and K. Prestwich, eds., (IEEE, NY, 1991), San Diego, CA, June 16-19, 1991, pp 29-32.
8. H. P. Hjalmarson, F. J Zutavern, G. M. Loubriel, L. A. Romero, A. G. Baca, K. Khachaturyan, D. R. Wake, "An Impact Ionization Model for Optically-Triggered Current Filaments in GaAs," SNL Report, *SAND 96-3972*, Albuquerque, NM, December, 1996.
9. G. M. Loubriel, F. J. Zutavern, H. P. Hjalmarson, R. R. Gallegos, W. D. Helgeson, and M. W. O'Malley, "Measurement of the Velocity of Current Filaments in Optically Triggered, High Gain GaAs Switches," *Applied Physics Letters,* Vol. 64, No. 24, 13 June, 1994, pp. 3323-3325.
10. F. J. Zutavern, G. M. Loubriel, H. P. Hjalmarson, W. D. Helgeson, M. W. O'Malley, M. H. Ruebush, T. A. Plut, A. Mar, and R. A. Falk, "Properties of High Gain GaAs Switches for Pulsed Power Applications," *Proceedings of 11th IEEE Pulsed Power Conference,* Baltimore, MD, June 29- July 2, 1997, pp 959-964.
11. R. Aaron Falk, F. J. Zutavern, M. W. O'Malley, "Carrier Density and Thermal Images of Transient Filaments in GaAs Photoconductive Switches," SPIE v3322. Conf. On Diagnostic Techniques for Semiconductor Materials and Devices, Montreal, 1997, pp 243-254.
12. Weng W. Chow, Stephan W. Koch, Murray Sargent III, *Semiconductor-Laser Physics*, Springer-Verlag, New York, 1994.
13. Bogdankevich, O.V., "Electron-beam-pumped semiconductor lasers," *Quantum Electronics*, 24, 1994, pp 1031-1053.
14. A. G. Baca, H. P. Hjalmarson, G. M. Loubriel, D. L. McLaughlin, and F. J. Zutavern, "High Current Density Contacts for GaAs Photoconductive Semiconductor Switches," *Proc. 9th IEEE International Pulsed Power Conference*, K. R. Prestwich and W. L. Baker, (IEEE, NY, 1993), Albuquerque, NM, June 21-23, 1993, pp. 84-87.

15. Fred J Zutavern, Albert G. Baca, Weng W. Chow, Michael J. Hafich, Harold P. Hjalmarson, Guillermo M. Loubriel, Alan Mar, Martin W. O'Malley, G. Allen Vawter, "Current Filament Semiconductor Lasers", post-deadline paper presented at CLEO, San Francisco, 2000.
16. Fred J Zutavern, Albert G. Baca, Weng W. Chow, Michael J. Hafich, Harold P. Hjalmarson, Guillermo M. Loubriel, Alan Mar, Martin W. O'Malley, G. Allen Vawter, "Semiconductor e-h Plasma Lasers", International Semiconductor Laser Conference, Monterey, CA, 2000.
17. G. Allen Vawter, Albert G. Baca, Weng W. Chow, Michael J. Hafich, Harold P. Hjalmarson, Guillermo M. Loubriel, Alan Mar, Martin W. O'Malley, Fred J. Zutavern, "Performance of Semiconductor Current Filament Lasers", Laser and Electro-Optic Society Annual Meeting, Puerto Rico, 2000.
18. Weng W. Chow, Albert G. Baca, Michael J. Hafich, Harold P. Hjalmarson, Guillermo M. Loubriel, Alan Mar, Martin W. O'Malley, G. Allen Vawter, "Lasing in a Semiconductor Current Filament," *Winter Colloquim for Quantum Electronics*, Jan 9-11, 2001, Snowbird, Utah, 2001.
19. Fred J Zutavern, "Switches and Lasers from Lightning–like Current Filaments in GaAs," Physics-Electrical Engineering Colloquium, Texas Technical University, Lubbock, TX, March, 2001.
20. Fred J Zutavern, Albert G. Baca, Weng W. Chow, Michael J. Hafich, Harold P. Hjalmarson, Guillermo M. Loubriel, Alan Mar, Martin W. O'Malley, G. Allen Vawter, "Semiconductor Lasers from Current Filaments", APS March Meeting (C17.1), Seattle, WA, 2001.
21. H. P. Hjalmarson, F. J. Zutavern, G. A Vawter, W. Chow, A. Mar, D. R. Wake, "Current Filament Semiconductor Laser Spectra," APS March Meeting (C17.2), Seattle, WA, 2001.
22. K. Kambour, Charles W. Myles, Harold P. Hjalmarson, "Carrier-Carrier Scattering in GaAs PCSS's," APS March Meeting (C17.7), Seattle, WA, 2001.
23. H.P. Hjalmarson, G. M. Loubriel, F. J. Zutavern, and D. R. Wake, "A Collective Impact Ionization Theory of Lock-on," in *Proceedings of 12th IEEE Pulsed Power Conference*, Monterey, CA, pp 299-302.
24. K. Kambour, Samsoo Kang, Charles W. Myles, and H. P. Hjalmarson, "Steady-State Properties of Lock-On Current Filaments in GaAs," *IEEE Trans. Plasma Sci.*, Vol. 28, Oct. 2000, pp 1497-1499.

Appendices: Laboratory Signal Analysis Software for Matlab©

The following appendices contain some of the software which was developed to process and analyze data taken to explore the properties and operating characteristics of PCSS and CFSL. They are written in the form of macro ("m") files for Matlab©, a high level language produced by The Mathworks, Inc. (copyright 1984-2001) and widely used by engineers and other scientists to process lists and tables of numerical data.

A. Beam profiling: beampars and beamlist

B. Beam divergence: divplot and beamdiver

C. Spectral sensitivity and background corrections: specfix, specgain, specdir, plotsfix, flspplot, and flanyl

D. Waveform digitizer data analysis: getdata, plotdata, addata, plotcurve, addcurve, plotlist, and plotshif

(This page intentionally left blank)

Appendix A. Beam profiling: beampars and beamlist

BEAMPARS – calculate laser profile parameters from an image file

```
%bpar=beampars(imf,ptitle,bcut,plims,skipplts,pdims,punits,epz,eunits) - FJZ 3/20/00
%
% plots and calculates laser profile parameters from an image file
% (bit-mapped or BEAMCODE file)
% See also:  beamlist, divplot, and beamdiver
%
% input:  imf = image file name (if omitted a selection window opens)
%                  Presently, the only formats are:
%                  1.  bit-mapped, 8-bit (BEAMCODE image only)
%                  2.  BEAMCODE data file (includes header & bkg)
%         ptitle = an optional plot title
%         bcut = background cut-off. (zero if omitted)
%                  This value is subtracted from the image and all
%                  negative intensities are zeroed.
%         plims = the pixel range to be imaged [rowmin,rowmax,colmin,colmax]
%         skipplots = non-zero, not empty to skip plots
%         pdims = the pixel dimensions [horizontal,vertical]
%                  If omitted, obtained from file header or [1,1].
%         punits = pixel dimension units
%         epz = energy per z (image height) or the unit energy of a pixel
%                  If omitted, 1 is used.
%         eunits = energy units
%
% output:
%         bpar.image=the 2-d matrix image
%         bpar.bkg=the background image (if it was saved)
%         bpar.header=the header info
%         bpar.imdims=the image dimensions [rows,columns]
%         bpar.imsize=the image size [horizontal,vertical]
%         bpar.pdims=the pixel dimensions [horizontal,vertical]
%         bpar.punits=the pixel dimension units
%         bpar.epz=the energy per z
%         bpar.eunits=the energy units
%         bpar.bcut=the background cut-off
%         bpar.centroid=the centroid ($intens[x y]dxdy/A) [horiz.,vert.]
%         bpar.centpix=the centroid [row,col]
%         bpar.peakloc=peak location ($intens^4[x y]dxdy/A) [horiz.,vert.]
%         bpar.peakpix=peak location [row,col]
%         bpar.panxy=the BEAMCODE pan x & y offsets
%         bpar.max=max value
%         bpar.min=the min value
%         bpar.fract=[0 .01 .1 1/e .5 .9]
%         (the fraction of the amplitude used for diameters and energies)
%         bpar.ht=bpar.fract*(max-min) + min
%         (the amplitude used for diameters and energies)
%         bpar.dia=the equivalent diameter at bpar.ht or bpar.fract
%         bpar.diap=the equivalent diameter at ... in pixels
%         bpar.energy=the energy inside the bpar.ht contours
%         bpar.rise=the average 10-90 spatial rise length
%
%bpar=beampars(imf,ptitle,bcut,plims,skipplts,pdims,punits,epz,eunits) - FJZ 3/20/00
function bpar=beampar(imf,ptit,bcut,plims,skipp,pdims,punits,epz,eunits)
if ~exist('eunits') eunits=' '; end;
if notarg(eunits,0) eunits=' '; end;
if ~exist('epz') epz=1; eunits=' '; end;
if notarg(epz,0) epz=1; eunits=' '; end;
if ~exist('punits') punits=' '; end;
if notarg(punits,0) punits=' '; end;
```

```
if ~exist('pdims') pdims=''; punits=' '; end;
if notarg(pdims,0) pdims=''; punits=' '; end;
if ~exist('skipp') skipp=0; end;
if notarg(skipp) skipp=0; end;
if ~exist('plims') plims=[]; end;
if notarg(plims,0) plims=[]; end;
if ~exist('bcut') bcut=0; end;
if notarg(bcut) bcut=0; end;
if ~exist('ptit') ptit=''; end; if notarg(ptit,0) ptit=''; end;
if ~isempty(ptit) ptit=[':  ',ptit]; end
if ~exist('imf') imf=''; end;
bkg=[]; pans=[];
[im,bkg,h]=getbcim(imf);
if isempty(h)
  disp('Image file not found.  BEAMPARS exiting.');
  return;
end
if bcut~=0 im=(im-bcut).*(im>bcut);end
if ~isempty(plims) % limit the image size
  plims=max(1,plims);
  plims(2)=min(plims(2),size(im,1));
  plims(4)=min(plims(4),size(im,2));
  im=im(plims(1):plims(2),plims(3):plims(4));
end
bpar.image=im;
bpar.bkg=bkg;
bpar.header=h;
bpar.imdims=size(im);
if isempty (plims)
  bpar.plims=[1 bpar.imdims(1) 1 bpar.imdims(2)];
else
  bpar.plims=plims;
end
if strcmp(upper(h{5}),'BEAMCODE')
   if isempty(pdims) pdims=h{9}; end
   punits=h{10};
   pans=h{13};
end
if isempty(pdims) pdims=[1,1]; end
bpar.imsize=bpar.imdims([2 1]) .* pdims;
bpar.pdims=pdims;
bpar.punits=punits;
x=((1:bpar.imdims(2))-.5)*pdims(1); %column values [horizontal]
y=((1:bpar.imdims(1))-.5)*pdims(2); %row values [vertical]
bpar.xax=[x(1),x(bpar.imdims(2))];
bpar.yax=[y(1),y(bpar.imdims(1))];
bpar.epz=epz;
bpar.eunits=eunits;
bpar.bcut=bcut;

% calculate parameters
centr=centroid(im);
bpar.centroid=centr.*pdims;
bpar.centpix=centr([2 1]);
peakloc=centroid(im.^4);
bpar.peakloc=peakloc.*pdims;
bpar.peakpix=peakloc([2 1]);
bpar.panxy=pans;

bpar.max=max(max(im));
bpar.min=min(min(im));
bpar.fract=[0,0.01,.1,1/exp(1),.5,.9];
bpar.ht=bpar.fract*(bpar.max-bpar.min) + bpar.min;
```

```
for i=1:length(bpar.ht)
area=sum(sum(im>=bpar.ht(i)))*pdims(1)*pdims(2);
bpar.dia(i)=2*sqrt(area/pi);
end
bpar.diap=bpar.dia/sqrt(pdims(1)*pdims(2));

for i=1:length(bpar.ht)
bpar.energy(i)=sum(sum(im.*(im>=bpar.ht(i))))*epz;
end
bpar.rise=(bpar.ht(6)-bpar.ht(3))*epz/(bpar.dia(3)-bpar.dia(6));

if skipp return; end

%create graphs
if punits(1)=='u' punits=['\mu',punits(2:length(punits))]; end

% image
if any(get(0,'Children')==1); delete(1); end;
figure(1);
imagesc(x,y,im);
axis image
fpos=get(gcf,'position'); ipos=get(gca,'position');
set(gcf,'position',[fpos(1:2),fpos(4)*ipos(3)/ipos(4),fpos(4)]);
if pdims(1)~=1 & punits~=' '; xlabel(['(',punits,')']);end
if pdims(2)~=1 & punits~=' '; ylabel(['(',punits,')']);end
title(['2-D Image',ptit]);
cornlab(h{1:2});
boldplot(2,14,12,10);

% contours
if any(get(0,'Children')==2); delete(2); end;
figure(2);
contourf(x,y,im,15);
axis image; set(gca,'ydir','reverse');
fpos=get(gcf,'position'); ipos=get(gca,'position');
set(gcf,'position',[fpos(1:2),fpos(4)*ipos(3)/ipos(4),fpos(4)]);
if pdims(1)~=1 & punits~=' '; xlabel(['(',punits,')']);end
if pdims(2)~=1 & punits~=' '; ylabel(['(',punits,')']);end
title(['Contours',ptit]);
cornlab(h{1:2});
boldplot(2,14,12,10);

% 3-d linear profile
if any(get(0,'Children')==3); delete(3); end;
figure(3);
surf(x,y,im); shading interp; hold on
contour3(x,y,im,bpar.ht([3 4 5 6]),'k'); hold off
set(gca,'ydir','reverse');
a=axis; axis image; b=axis; b(5:6)=a(5:6); axis normal; axis(b);
if pdims(1)~=1&punits~=' '; xlabel(['(',punits,')']);end
if pdims(2)~=1&punits~=' '; ylabel(['(',punits,')']);end
title(['3-D Profile',ptit]);
cornlab(h{1:2});
boldplot(2,14,12,10);

% 3-d logarithmic profile
if any(get(0,'Children')==4); delete(4); end;
figure(4);
surf(x,y,im); shading interp; hold on
contour3(x,y,im,bpar.ht([3 4 5 6]),'k'); hold off
set(gca,'ydir','reverse');
set(gca,'zscale','log');
```

```
axisnear;
a=axis; axis image; b=axis; b(5:6)=a(5:6); axis normal; axis(b);
if pdims(1)~=1&punits~=' '; xlabel(['(',punits,')']);end
if pdims(2)~=1&punits~=' '; ylabel(['(',punits,')']);end
title(['3-D Logarithmic Profile',ptit]);
cornlab(h{1:2});
boldplot(2,14,12,10);

figspred(1:4,[10 230],[448 10]);

%bc=cm(:,cm(2,cm(2,1)+2)+2+cm(2,1)+2:cm(2,cm(2,1)+2)+2+cm(2,1)+1+...
%     cm(2,cm(2,cm(2,1)+2)+2+cm(2,1)+1));
%cm=contourc(c,cv);
%cv=[mn+.1*r,mn+.5*r,mn+.9*r]
%mn=min(min(c));mx=max(max(c));r=mx-mn;
```

BEAMLIST – calculates and saves beam parameters using beampars for a list of files

```
%[lpars,pls,dias,es]=beamlist(filelist,filesave)  FJZ 3/20/00;
%   calculates beam parameters using "beampars" for a list of files
%   "filelist" is a list (or matrix) of filenames
%   "filesave" is an optional filename to save the results
%   "lpars" is a structured variable containg the list of parameters
%   "pks" is a matrix of peak locations (pixels) with one row per file
%   "dias" is a matrix of beam diameters with one row per file
%   "es" is a matrix of beam energies with one row per file
%   (The "fract" or "ht" substructures define the diameters and energies.)
%   (See also:  beampars, divplot, beamdiver)
function [lp,pks,dias,es]=beamlist(filelist,filesave)
if ~exist('filesave','var') filesave=''; end
if notarg(filesave,0) filesave=''; end
if ~exist('filelist','var') filelist=''; end
if notarg(filelist,0) filelist=''; end

if isempty(filelist)
  disp('Select filenames.  Press cancel to exit.');
  [fn,path]=getfile('*.dat',-1);
  if isempty(fn) disp('No filenames selected.  "beamlist" exiting.'); return; end
  filelist=[path lower(fn)];
end
if iscell(filelist) filelist=strvcat(filelist); end;
if isempty(filelist) disp('No filenames found.  "beamlist" exiting.'); disp(filelist);
...
    return; end

[nr,nc]=size(filelist);
if nr<1 disp('Empty filelist.  "beamlist" exiting.'); return; end
if isempty(findstr('.',filelist(1,:)))
  filelist=[filelist repmat('.dat',nr,1)];
end
lp=[];
for i=1:nr
  bp=beampars(filelist(i,:),'','','',-1) %calculate beam parameters w/o plotting
  i1=max(findstr('\',filelist(i,:)))+1;
  if isempty(i1) i1=1; end
  i2=min(findstr('.',filelist(i,:)))-1;
  if ~isempty(bp)
    if i1<1 | i2<1 | i1>i2 | i1>nc | i2>nc
      disp(['Could not creat a field name from:  ',filelist(i,:)]); return;
    end
    disp(['Creating lpars.',filelist(i,i1:i2)]);
    disp(' ');
    bp=rmfield(bp,'image');
    bp=rmfield(bp,'bkg');
    lp=setfield(lp,filelist(i,i1:i2),bp);
  end
end
fn=fieldnames(lp);
pks=[];  for i=1:nr pks=[pks;getfield(lp,fn{i},'peakpix')]; end
dias=[];  for i=1:nr dias=[dias;getfield(lp,fn{i},'dia')]; end
es=[]; for i=1:nr es=[es;getfield(lp,fn{i},'energy')]; end
if ~isempty(filesave)
  save(filesave,'lp');
  disp(['Results saved.  To retreive enter "load ',filesave,'".']);
end
```

(This page intentionally left blank)

Appendix B. Beam divergence: divplot and beamdiver

DIVPLOT – creates divergence plots from data saved with beamlist and beampars

```
%divplot  FJZ 3/21/00
%   generates divergence plots (see also:  beamdiver, beampars, beamlist)
%   There are four possible calling sequences:
%
%1.  divplot(y,n1,n2)    e.g.  divplot([-3,-2,-1,0,1,2,3],220,289)
%Assuming the images are in groups at the same camera positions "y",
%the file names will be generated FL[n].dat where n=n1:n2.  There must be
%a set of image files for each value of y (ng=length(y)), and the number
%of images in each group must be npg=(n2-n1+1)/ng.
%
%2.  divplot(y,n1,n2,ny)    e.g.  divplot([-2,-1,0,1,2],220,242,[5,5,4,4,5])
%Same as 1. except the number of images at each location (npg) varies and
%is given in ny.  Rules are:  length(ny)=length(y) and sum(ny)=n2-n1+1
%
%3.  divplot(y,fl)    e.g. divplot([-2,-1,0,1],{{'a1','a2','a3'},{'b1','b2'},...})
%When the above naming convention is not followed, then the names can be
%passed in a double cell array.  In this case, y contains the camera positions
%and fl is a a double cell array with the image file names at each position
%grouped together in one cell.
% e.g. fl={{'f1p1.dat','f2p1.dat','f3p1.dat'},...
%          {'f1p2.dat','f2p2.dat','f3p2.dat','f4p2.dat'},...
%               ...
%          {'f1pn.dat','f2pn.dat'}}
% The file names can be arbitray lengths and the number of files at each location
% can vary.  The only requirement is that the number of values in y corresponds to
% the number of groups of names (i.e. camera positions).
%
%4.  divplot
%When the results from a previous run exist (diverge.mat), a call without
%arguments will reproduce the plots and tables.
function divplot(y,n1,n2,ny)

if exist('ny')
  caseno=2;
  ng=length(y);
  npg=ny;
  if length(npg)~=length(y)
     disp('The 1st (y) and 4th (ny) arguments do not have the same length.');
     help divplot
    return;
  end
  %generate the filenames
  nlast=n1-1;
  for ig=1:ng
     n=nlast+(1:npg(ig));
     nlast=n(end);
     if max(n)>n2
        disp('The file number generated from "ny" exceeds "n2".');
        disp(['last group numbers = ',num2str(n)]);
        disp(['n1, n2 = ',num2str([n1 n2])]);
        help divplot
        return;
     end
     fl{ig}=[repmat('FL',npg(ig),1),num2str(n.'),repmat('.dat',npg(ig),1)];
  end
elseif exist('n2')
  caseno=1;
  ng=length(y);
```

```
  npg=(n2-n1+1)/ng;
  if fix(npg)~=npg
    disp('The number of files does not allow equal size groups.');
    return;
  end
  %generate the filenames
  for ig=1:ng
    n=n1-1+(ig-1)*npg+(1:npg);
    fl{ig}=[repmat('FL',npg,1),num2str(n.'),repmat('.dat',npg,1)];
  end
elseif exist('n1')
  caseno=3;
  fl=n1;
  if length(y)~=size(fl,2)
    disp('The number of filename groups ~= the number of positions.');
    return;
  end
  npg=[];
  ng=length(y);
  for i=1:ng
    npg(i)=length(fl{i});
  end
elseif exist('y')
  disp('Camera positions (y-values) must be accompanied by file names or numbers.');
  return;
else
  if exist('diverge.mat','file')
    caseno=4;
    load diverge   %if already saved
  else
    disp('The results file (diverge.mat) was not found.');
    return;
  end
end

%calculate beam parameters for all files and save in groups
if caseno<4
  for ig=1:ng [l{ig},p{ig},d{ig},e{ig}]=beamlist(fl{ig},fl{ig}(1,1:end-4)); end
end

[sy,isy]=sort(y);
ly=length(y);  %number of camera positions
fnames=[fl{1}(1,1:end-4),'-',fl{end}(end,1:end-4)];
fdate=getfield(l{1},subsref(fieldnames(l{1}),substruct('{}',{2})),'header',{2});
fdate=fdate{1};
ng=length(fl); %number of image groups (should equal ly)
if (ng~=ly)
   disp('Number of camera positions (y) ~= number of image groups (fl).');
   return;
end
for ig=1:ng
   npg(ig)=size(fl{ig},1); %number of images per group
end
maxe=[];
for iy=1:ly maxe=[maxe,max(max(e{iy}))]; end;
maxe=max(maxe);

% plot diameters
figure(1); clf; box; hold on;hp=[];
col='bmkgrcbmkgrc';
for id=3:6 for iy=1:ly plot(repmat(y(iy),npg(iy),1),d{iy}(:,id),['x',col(id)]); end;
end
for id=1:6 md{id}=[]; for ig=1:ng md{id}=[md{id},mean(d{ig}(:,id))]; end; end
```

```
for id=3:6 hp(id)=plot(sy,md{id}(isy),['-',col(id)]); end
title('Filament Laser Divergence');
xlabel('Camera position (mm)');
ylabel('Beam Diameter (\mum)');
axisnear;  ax(inf,inf,0);
cornlab(fnames,fdate,'0.5 mm','quasi-vertical');
labelcurve(['10%';'1/e';'50%';'90%'],0,hp(3:6));
boldplot(1,16,14,12);

% plot energies
figure(2); clf; box; hold on
for id=3:6 for iy=1:ly plot(repmat(y(iy),npg(iy),1),e{iy}(:,id)/maxe,...
        ['+',col(id)]); end; end
for id=1:6 me{id}=[]; for ig=1:ng me{id}=[me{id},mean(e{ig}(:,id))]; end; end
for id=3:6 hp(id)=plot(sy,me{id}(isy)/maxe,['-',col(id)]); end
title('Filament Laser Energies');
xlabel('Camera position (mm)');
ylabel('Beam Energy (relative)');
axisnear;  ax(inf,inf,0);
cornlab(fnames,fdate,'0.5 mm','quasi-vertical');
labelcurve(['10%';'1/e';'50%';'90%'],0,hp(3:6));
boldplot(1,16,14,12);

figspred(1:2,[5,275],[460,10]);

if caseno<4
  save diverge fl l p d e y md me
end

%  tabulate mean diameters
fns=[];
for iy=isy
   fns=[fns;fl{iy}(1,1:end-4),'-',fl{iy}(end,1:end-4)];
end

disp(' ');
disp(' ');
disp(' ');
disp(['            mean diameters (mm):  ',fnames,'  ',fdate]);
disp(' ');
disp('     files position \ energy-> 0%       1%      10%      1/e      50%
90%');
dtab=cat(1,md{:}).';
disp([repmat(' ',ly,5),fns,repmat(' ',ly,1),num2str(sy.','%5.2f'),...
      repmat(' ',ly,7),num2str(dtab(isy,:)/1000,'%9.3f')]);

%  tabulate mean energies
disp(' ');
disp(' ');
disp(' ');
disp(['            mean energies (/1000):  ',fnames,'  ',fdate]);
disp(' ');
disp('     files position \ energy-> 0%        1%      10%      1/e      50%
90%');
mtab=cat(1,me{:}).';
disp([repmat(' ',ly,5),fns,repmat(' ',ly,1),num2str(sy.','%5.2f'),...
      repmat(' ',ly,7),num2str(mtab(isy,:)/1000,'%9.3f')]);

%  calculate and tabulate divergences from mean diameters & y
for iy=1:ly-1
   diva(iy,1:5)=360/pi*atan(abs((dtab(isy(iy+1),2:end)-dtab(isy(iy),2:end))...
      /(sy(iy+1)-sy(iy)))/2000);
end
```

```
disp(' ');
disp(' ');
disp(' ');
disp( '          divergence angles from mean diameters for adjacent positions');
disp(['          ',fnames,'  ',fdate]);
disp(' ');
disp('     files position \ energy-> 1%      10%      1/e      50%      90%');
disp([repmat(' ',ly-1,5),fns(1:end-1,:),repmat(' ',ly-1,1),...
      num2str(sy(1:ly-1).','%5.2f'),repmat(' ',ly-1,7),num2str(diva,'%9.3f')]);
disp(' ');
disp(' ');
                                        disp(' ');
```

BEAMDIVER – tanslates beamdivergence between NA, f, angle, and slope

```
%div=beamdiver(var,val)    FJZ 3/21/00
%    calculates NA, f, angle, or slope from any one parameter
%    see also: divplot, beamlist, beampars
%    (angles are in degrees)
%
%  var = one of 'NA', 'f', 'angle', 'slope'
%  val = the value (in degrees for angle, otherwise dimensionless)
%  div = a structured list of all four parameters
%
%  angle = full angle of the diverging cone = 2*asin(NA) = 2*atan(slope/2)
%     NA = numerical aperature = sin(angle/2) = sin(diameter/(2*focallength))
%      f = "f-number" = focallength/diameter = 1/(2*tan(angle/2)) = 1/slope
%  slope = (change in diameter)/(change in distance) = 2*tan(angle/2)= 1/f
%
%  Note:  NA and f are defined for the smallest lens rquired to collect
%  all the light and form an image at infinity (nearly parallel beam).
%  For magnification, M, NA or f must be changed to:
%                 NA'= sin((1+M)/M*asin(NA)) ~ NA*(1+M)/M
%                  f'= f*M/(1+M)
function out=beamdiver(var,val)
if ~exist('var') | ~exist('val')
  out=[];
  disp('This routine requires a variable name and value.');
  help divergence
  return;
end
var=lower(var);
eval([var,'=val;']);

if strcmp(var,'slope')
  angle=2*atan(slope/2);
elseif strcmp(var,'angle')
  angle=pi/180*angle;
elseif strcmp(var,'na')
  angle=2*asin(na);
elseif strcmp(var,'f')
  angle=2*atan(1/(2*f));
else
  out=[]; disp('Variable name not recognized.'); return;
end

  slope=2*tan(angle/2);
  out.angle=angle*180/pi;
  out.NA=sin(angle/2);
  out.f=1/slope;
  out.slope=slope;
```

(This page intentionally left blank)

Appendix C. Spectral sensitivity and background corrections:

SPECFIX - corrects spectra for background and gain (see specgain)

```
%[s,h,b,g]=specfix(spec,bkg,gain,temp,iedge,cosmic)  11/29/99 FJZ
%
%   corrects spectra for background noise and gain
%
%inputs:
%   spec - either:
%             1.) the spectra filename ('' opens a window)
%          or 2.) an n X 2 array of wavelength and intensity
%          (the x,y values of the spectrum to be corrected)
%   bkg - either:
%             1.) a text matrix of filenames of "dark" spectra to
%        average for a background, that is subtracted from the gain
%       and spectra files
%          or 2.) an n X 2 array of wavelength and background
%       to subtract from the spectra (originally from "dark" files)
%           (if omitted, empty or zero, no background is subtracted)
%   gain - either:
%             1.) a text matrix of filenames to average and calculate
%       the gain assuming a black body spectrum at temperature, "temp"
%          or 2.) an n X 2 array of wavelength and gain to scale
%       the spectra files (originally from black body files)
%          (if omitted, empty, or zero, a gain of 1 is used)
%   temp - the filament temperature to remove the black blody shape
%          (only used when gain is a text matrix of black body spectra)
%          (if omitted, empty, or zero, a zero slope is used)
%   iedge - number of pixels to omit from each edge of the spectra
%           (bkg and gain of zero will be returned for these pixels)
%   cosmic - remove cosmic rays from data files with a double pass filter
%            (see cosmicut)
%            If >0 cosmic rays will be removed with r=cosmic.
%            If =0 no cosmic rays will be removed.
%            If omited or <0 cosmic rays will be removed with r=5.5
%
%outputs:
%   s - a 2 X n array of wavelength (nm) and corrected intensities given by
%         y=(y0-bkg)*(max(gain-bkg)/(gain-bkg))*
%         (y0(1)/y0)^5*(exp(hc/(kTy0(1)))-1)/(exp(hc/(kTy0))-1)
%   h - the spectra file header
%   b - an n X 2 array of wavelength and background to subtract from spectra
%   g - an n X 2 array of wavelength and gain to scale spectra
%         assuming wavelength values match, corrected spectra:  y = (y0-b).*g
%[s,h,b,g]=specfix(spec,bkg,gain,temp,iedge,cosmic)  11/29/99 FJZ
function [s,h,b,g]=specfix(spec,bkg,gain,T,iedge,cosmic)

if ~exist('cosmic') cosmic=-1; end; if notarg(cosmic) cosmic=-1; end;
if cosmic<0 cosmic=5.5; end
if ~exist('T') T=0; end; if notarg(T) T=0; end;
if ~exist('gain') gain=''; end; if notarg(gain,0) gain=''; end;
if ~exist('bkg') bkg=''; end; if notarg(bkg) bkg=''; end;
if ~exist('spec') spec=''; end; if notarg(spec,0) spec=''; end;
if ~exist('iedge') iedge=[]; end; if notarg(iedge) iedge=0; end

if isempty(spec)
   disp('Select the spectrum to be corrected.'); pause(2);
   [x,y,h]=getdata(spec,0,'','','',-1);
   ny=length(y);
   if isempty(x) disp('No spectrum selected.  Exiting ...'); end
   disp(spec);
```

```
    if cosmic
       [yc,yfc,ibad]=cosmicut(y(iedge+1:ny-iedge),cosmic,2);
       y(iedge+1:ny-iedge)=yc;
       if ~isempty(ibad)
          disp([num2str(length(ibad)), 'points removed to eliminate cosmic rays.']);
          npdisp=min(length(ibad),15); disp(ibad(1:npdisp));
       end
    end
elseif ischar(spec)
    disp(spec);
    [x,y,h]=getdata(spec,0,'','','',-1);
    ny=length(y);
    if cosmic
       [yc,yfc,ibad]=cosmicut(y(iedge+1:ny-iedge),cosmic,2);
       y(iedge+1:ny-iedge)=yc;
       if ~isempty(ibad)
          disp([num2str(length(ibad)), 'points removed to eliminate cosmic rays.']);
          npdisp=min(length(ibad),15); disp(ibad(1:npdisp));
       end
    end
else
    [nr,nc]=size(spec);
    if min([nr,nc])~=2
       disp('Spectra variable must reference a spectra filename or a 2 X n');
       disp('array containing the x,y - values of the spectra.  Exiting ...');
       return;
    end
    h='';
    if (nr==2)
       x=spec(1,:); y=spec(2,:);
    else
       x=spec(:,1); y=spec(:,2);
    end
end

if isempty(x) disp('No spectrum.  Exiting ...'); end
disp('bkg and gain corrections');
[xbg,yb,yg]=specgain(bkg,gain,T,iedge,cosmic);
if isempty(xbg)
   disp('Background or gain corrections not calculated.  Exiting ...');
   b=''; g=''; return;
end
if length(xbg)~=length(x)&length(xbg)~=1
   disp('Wavelength (X) values of the spectra do not agree with the');
   disp('background and gain corrections.  Exiting ...');
   b=''; g=''; return;
end
if length(xbg)~=1
   nx=length(x);
   if x(1)~=xbg(1) & x(1)==xbg(nx) xbg=xbg(nx:-1:1); yb=yb(nx:-1:1); yg=yg(nx:-1:1);
end
   if any(xbg~=x)
      disp('Wavelength (X) values of the spectra do not agree with the');
      disp('background and gain corrections.  Exiting ...');
      b=''; g=''; return;
   end
end
b=[xbg,yb];
g=[xbg,yg];

% apply corrections to the data
y=(y-yb).*yg;
s=[x,y];
```

SPECGAIN - background and gain from black body and dark noise files

```
%[xbg,yb,yg]=spegain(bkg,gain,temp,iedge,cosmic)  11/29/99 FJZ
%
%   calculates background and gain corrections to spectra
%   from dark and blackbody spectra at the same wavelength
%
%inputs:
%   bkg - either:
%            1.) a text matrix of filenames of "dark" spectra to
%         average for a background, that is subtracted from the gain
%        and spectra files
%            2.) an n X 2 array of wavelength and background
%        to subtract from the spectra (originally from "dark" files)
%            3.) a spectra string, e.g.:  'DNae[3,5:12]'
%             (if omitted, empty or zero, no background is subtracted)
%   gain - either:
%            1.) a text matrix of filenames to average and calculate
%        the gain assuming a black body spectrum at temperature, "temp"
%            2.) an n X 2 array of wavelength and gain to scale
%        the spectra files (originally from black body files)
%            3.) a spectra string, e.g.:  'BBcd[3,5,9]'
%             (if omitted, empty, or zero, a gain of 1 is used)
%   temp - the filament temperature to remove the black blody shape
%          (only used when gain is a text matrix of black body spectra)
%          (if omitted, empty, or zero, a zero slope is used)
%   iedge - number of pixels to omit from each edge of the spectra
%           (bkg and gain of zero will be returned for these pixels)
%   cosmic - remove cosmic rays from data files with double pass filter
%            (see cosmicut)
%            If >0 cosmic rays will be removed with r=cosmic.
%            If =0 no cosmic rays will be removed.
%            If omited or <0 cosmic rays will be removed with r=5.5
%
%outputs:
%   xbg - wavelength values for background and gain corrections
%   yb - background offset values
%   yg - gain correction
%        corrected data: y=(y0-yb).*yg  where:
%   yg = (min(g-yb)./(g-yb))*
%        (y0(1)./y0).^5*(exp(hc/(kT*y0(1)))-1)./(exp(hc/(kT*y0))-1)
%
%[xbg,yb,yg]=spegain(bkg,gain,temp,iedge,cosmic)  11/29/99 FJZ
function [xbg,yb,yg]=specgain(bkg,gain,T,iedge,cosmic)
if ~exist('cosmic') cosmic=-1; end; if notarg(cosmic) cosmic=-1; end;
if cosmic<0 cosmic=5.5; end
if ~exist('T') T=0; end; if notarg(T) T=0; end;
if ~exist('gain') gain=''; end; if notarg(gain,0) gain=''; end;
if ~exist('bkg') bkg=''; end; if notarg(bkg) bkg=''; end;
if ~exist('iedge') iedge=0; end; if notarg(iedge) iedge=0; end

% check to see if background correction must be created from files
if isempty(bkg)
   disp('Select the background files.  (Select "cancel" when complete.)');
   pause(2);
   bkg=getlist;
end
if isempty(bkg)
   xb=0; yb=0;
elseif ischar(bkg)
   % average background files
   [nbr,nbc]=size(bkg); yb=[];
   if ~any(bkg(1,:)=='.')
```

```
      sd=specdir('',0,0);
      bkg=specstr(bkg,sd);
      if isempty(bkg) disp('Background spectra not found.  SPECGAIN exiting...');
return; end
      [nbr,nbc]=size(bkg);
   end
   for i=1:nbr
      disp(bkg(i,:));
      [x,y]=getdata(bkg(i,:),0,'','','',-1);
      ny=length(y);
      if cosmic
         [yc,yfc,ibad]=cosmicut(y(iedge+1:ny-iedge),cosmic,2);
         y(iedge+1:ny-iedge)=yc;
         if ~isempty(ibad)
            disp([num2str(length(ibad)), 'points removed to eliminate cosmic rays.']);
            npdisp=min(length(ibad),15); disp(ibad(1:npdisp));
         end
      end
      if isempty(yb)
         yb=y; xb=x;
      else
         if length(x)~=length(xb)
            disp('Length of wavelength vectors for background spectra do not agree.
SPECGAIN exiting...');
            xbg='';yb='';yg=''; return;
         end
         nx=length(x);
         if x(1)~=xb(1) & x(1)==xb(nx) xb=xb(nx:-1:1); yb=yb(nx:-1:1); end
         if any(x~=xb)
            disp('Wavelength values for background spectra do not agree.  SPECGAIN
exiting...');
            xbg='';yb='';yg=''; return;
         end
         yb=yb+y;
    end
   end
   yb=yb/nbr;
   if iedge>0
      ny=length(yb); yb(1:iedge)=zeros(1,iedge); yb(ny-iedge+1:ny)=zeros(1,iedge);
   end
   disp(['Background averaged from ',num2str(nbr),' files.']);
   displist(bkg);
else
   [nr,nc]=size(bkg);
%   if min([nr,nc])~=2
%      disp('Warning - ');
%      disp('Background variable should reference dark spectra filenames or a 2 X n');
%      disp('array containing the (x,y) - values (wavelength,darknoise).');
%      xbg='';
%   end
   if (nr==2)
      xb=bkg(1,:); yb=bkg(2,:);
   else
      xb=bkg(:,1); yb=bkg(:,2);
   end
end

if isempty(gain)
   disp('Select the black body files.  (Select "cancel" when complete.)');
   pause(2);
   gain=getlist;
end
if isempty(gain)
```

```
   xg=xb; yg=ones(length(xg),1);
elseif ischar(gain)
   % average gain files
   [ngr,ngc]=size(gain); yg=[];
   if ~any(gain(1,:)=='.')
      if ~exist('sd') sd=specdir('',0,0); end
      gain=specstr(gain,sd);
      if isempty(gain) disp('Gain spectra not found.  SPECGAIN exiting...'); return;
end
      [ngr,ngc]=size(gain);
   end
   for i=1:ngr
      disp(gain(i,:));
      [x,y]=getdata(gain(i,:),0,'','','',-1);
      ny=length(y);
      if cosmic
         [yc,yfc,ibad]=cosmicut(y(iedge+1:ny-iedge),cosmic,2);
         y(iedge+1:ny-iedge)=yc;
         if ~isempty(ibad)
            disp([num2str(length(ibad)), 'points removed to eliminate cosmic rays.']);
            npdisp=min(length(ibad),15); disp(ibad(1:npdisp));
         end
      end
      nx=length(x);
      if length(xb)==1 xb=x; yb=zeros(length(nx),1); end
      if x(1)~=xb(1) & x(1)==xb(nx) xb=xb(nx:-1:1); yb=yb(nx:-1:1); end
      if any(x~=xb)
         disp('Wavelength values of background files do not agree.  SPECGAIN
exiting...');
         xbg='';yb='';yg=''; return;
      end
      y=y-yb;
      if isempty(yg)
         yg=y;xg=x;
      else
         if length(x)~=length(xg)
disp(['Length of wavelength vectors for gain spectra do not agree.  ',...
'SPECGAIN exiting...');
            xbg='';yb='';yg=''; return;
         end
         nx=length(x);
         if x(1)~=xg(1) & x(1)==xg(nx) xg=xg(nx:-1:1); yg=yg(nx:-1:1); end
         if any(x~=xg)
            disp('Wavelength values of gain files do not agree.  SPECGAIN
exiting...');
            xbg='';yb='';yg=''; return;
         end
         yg=yg+y;
      end
   end
   yg=yg/ngr;
   disp(['Gain averaged from ',num2str(ngr),' files.']);
   displist(gain);
   i=find(yg==0);
   if ~isempty(i)
      yg(i)=max(yg)/100;
      disp([num2str(length(i)),' zeros replaced in gain curve with max/100.']);
   end
   yg=min(yg)./yg;
   % include black body correction
   if T>0
      h=6.62608E-34;   %Planck's Constant (Joule Second)
      c=2.99792E8*1e9;     %speed of light (nm/Second)
```

```
      k=1.38066E-23;       %Boltzmann Constant (Joule/Kelvin)
      yg=yg*xg(1)^5*(exp(h*c/(xg(1)*k*T))-1)./(xg.^5.*(exp(h*c/(xg*k*T))-1));
   end
   %s=cos(th)*2*h*c^2./(lam.^5.*(exp(h*c*10^6./(lam*k*T))-1))*10^12;
   %   (Joules/(second microns^3))
   %For a non-ideal emitter, multiply by: Absorb(lam,th)
   %Peak(T)=espd(h*c/(4.96511*k*T)*1e6,0,T)
   %Edge(x,a,b,alpha)=((b-a)*tanh(alpha*x)+(b+a))/2
   if iedge>0
      ny=length(yg); yg(1:iedge)=zeros(1,iedge); yg(ny-iedge+1:ny)=zeros(1,iedge);
   end

else
   [nr,nc]=size(gain);
%   if min([nr,nc])~=2
%      disp('Warning -');
%      disp('Gain variable should reference blackbody filenames or a 2 X n');
%      disp('array containing the (x,y) - values (wavelength,gain).');
%      xbg='';
%   end
   if (nr==2)
      xg=gain(1,:); yg=gain(2,:);
   else
      xg=gain(:,1); yg=gain(:,2);
   end
end
xbg='';
if length(xg)~=length(xb)
   disp('Length of wavelength vector for background is not the same as for the gain.
SPECGAIN exiting ...');
   return;
end
if any(xg~=xb)
   disp('Wavelength values of gain and background values do not agree.  SPECGAIN
exiting ...');
   return;
end
xbg=xg;
```

SPECDIR - creates and displays a summary of the spectra in a directory

```
%sum=specdir(direct,nochron,width)   FJZ 11/24/99
%
%   summarizes directories of spectra (.SPC files).
%
%input:
%   direct - directory to be summarized (current if omitted)
%   nochron - If non-zero, the chronological listing is omited.
%   width - width of summary listing (default is 80)
%
%outputs:
%   .SPEC - text matrix of all spectra filenames
%   ...
%   .FL - text matrix of all filament laser (FL) filenames
%         sorted by run & shot number
%   .FLr - text matrix of spectral runs
%   .FLis - indeces of all the FL spectra in sum.SPEC:  sum.SPEC(sum.FLis,:)
%   .FLir - a cell array of the indeces of the spectra in each run relative to FL:
%             sum.FL([sum.FLir{runs}],:)
%   .FLtime - a cell array of the time of the 1st spectrum in each run
%             (a 2 element time vector is generated to include unnumbered spectra)
%              datestr([sum.FLtime{runs}])
%   ...
%   .DN, .DNr, .DNis, .DNir, .DNtime - same info for dark noise files
%   .BB, .BBr, .BBis, .BBir, .BBtime - same info for black body files
%   .XX, .XXr, .XXis, .XXir, .XXtime - same info all other spectra
%   ...
%   .TIME - a vector of the time of the 1st spectrum in all of the runs
%   .ITIME - the indeces of the 1st specrum in all of the runs
%           [s.SPEC(s.ITIME,:),blanks(length(sum.TIME))',datestr(sum.TIME)]
%   .SUM - a cell string array of the output summary
%   .COMMENT - a text matrix of the comment line from the 1st spectrum in each run
%
%sum=specdir(direct,nochron,width)   FJZ 11/24/99
function sum=specdir(direct,nochron,width)
global pcssdir colors
types=upper({'FL','DN','BB','LD','BS'});
types=strvcat(types);
typename={'Filament Laser','Dark Noise','Black Body','Laser Diode','Big Sky Laser'};
oldpcss=pcssdir;
if ~exist('direct') direct=cd; end; if notarg(direct,0) direct=cd; end
if ~exist('nochron') nochron=0; end; if notarg(nochron) nochron=0; end
if ~exist('width') width=80; end; if notarg(width) width=80; end
if width<0 width==0; end
if ~strcmp(direct,pcssdir) pcssdir=direct; end
olddir=cd; cd(direct);

dspec=dir('specdir_*.*');
if isempty(dspec)
   nsavf=1;
   newdata=-1;
else
   newdata=0;
   dspecm=strvcat(dspec.name);
   nsavf=max(str2num(dspecm(:,9:10)));
   if isempty(nsavf) nsavf=1; newdata=-1; end
   if nsavf<=0; nsavf=1; newdata=-1; end
end

if ~newdata   load(['specdir','_',num2str(nsavf)]);
   [rspec,cspec]=size(sum.SPEC);
   dspc=dir('*.spc');
```

```
   [rspc,cspc]=size(dspc);
   if rspc==rspec
      if (width) disp(strvcat(sum.SUM)); end
      cd(olddir); pcssdir=oldpcss; return;
   end
   % This table is out of date, get a new table.
   delete(['specdir','_',num2str(nsavf),'.mat']);
   newdata=-1;
end % ~newdata

sum='';
sumry=sprintf('\n%s   Summary of Spectra found in:',datestr(now));
sumry=strvcat(sumry,sprintf('      %s',direct));
if width fprintf('%s\n',sumry(1,:)); fprintf('%s\n',sumry(2,:)); end
specall=getlist('*.SPC'); i=smartsort(specall,-1); specall=specall(i,:);
sum.SPEC=specall;
[su,ir]=unique(upper(specall(:,1:2)),'rows');
type=upper(specall(ir,1:2));
ialltime=[]; alltime=[]; alltitl=[];
for it=1:length(type)
   isp{it}=find(upper(specall(:,1))==type(it,1)&upper(specall(:,2))==type(it,2));
   spec=specall(isp{it},:);
   %  spec=getlist([type{it},'*.SPC']);
   if ~isempty(spec)
      %     i=smartsort(spec,-1);
      %     spec=spec(i,:);
      sum=setfield(sum,type(it,:),spec);
      [ns,ncs]=size(spec);
      [run,ir]=unique(upper(spec(:,3:4)),'rows');
      [nr,ncr]=size(run);
      sum=setfield(sum,[type(it,:),'r'],run);
      k=find(type(it,1)==types(:,1)&type(it,2)==types(:,2));
      if isempty(k)
         tname=type(it,:);
      else
         tname=typename{k};
      end
      nextline=sprintf('\n%g %s Spectra ---',ns,tname);
      if (width) fprintf('%s\n',nextline); sumry=strvcat(sumry,nextline); end
      tim=[];
      itim=[];
      rdx=[];
      titl=[];
      for i=1:nr
         rdx{i}=find(run(i,1)==upper(spec(:,3))&run(i,2)==upper(spec(:,4)));
         tim{i}=[];
         itim{i}=[];
         titl{i}=[];
         ni=length(rdx{i});
         if spec(rdx{i}(ni),5)=='.'
            nis=ni-1;
         else
            nis=ni;
         end
         if nis>0
            nextline=sprintf('%3d.  %s%s%s-%s%s%s',i,type(it,:),...
               lower(run(i,:)),deblank(spec(rdx{i}(1),5:ncs)),type(it,:),...
               lower(run(i,:)),deblank(spec(rdx{i}(nis),5:ncs)));
            [x,y,h]=getdata(spec(rdx{i}(1,1),:),[],[],[],[],-1);
            [nh,nch]=size(h);
            eval([deblank(h(23,:)),';']);eval([deblank(h(24,:)),';']);
            nextline=[nextline,sprintf('  %s%s',fcmnt1,fcmnt2)];
            titl{i}=[fcmnt1,fcmnt2];
```

```
      nl=length(nextline); nl2=min(nl,width); if width==0 nl2=nl; end
      if (width) fprintf('%s\n',nextline(1:nl2)); end
      sumry=strvcat(sumry,nextline(1:nl2));
      if (nl>nl2 & width)
         fprintf('      %s\n',nextline(nl2+1:nl));
         sumry=strvcat(sumry,[blanks(6),nextline(nl2+1:nl)]);
      end
      %     fprintf('  %s\n',deblank(h(64,9:nch)));
      eval([deblank(h(9,:)),';']); eval([deblank(h(10,:)),';']);
      tims=deblank(h(2,13:nch));
      nrit=strmatch('Integration Time',h);
      if isempty(nrit)
         ncit=[];
      else
         ncit=findstr(h(nrit,:),'=')+1;
      end
      if isempty(ncit)
         integ=0;
      else
         integ=str2num(h(nrit,ncit:nch));
      end
      nextline=sprintf('      %s   range:  %7g - %7g nm  integ=%6.3g s',...
         tims,ffirst,flast,integ);
      tim{i}=datenum(tims);
      itim{i}=isp{it}(rdx{i}(1));
      if (width) fprintf('%s\n',nextline); end
      sumry=strvcat(sumry,nextline);
   end
   if nis<ni
      nextline=sprintf('%3d*  %s%s%s           ',i,type(it,:),...
         lower(run(i,:)),deblank(spec(rdx{i}(ni),5:ncs)));
      [x,y,h]=getdata(spec(rdx{i}(ni,1),:),[],[],[],[],-1);
      [nh,nch]=size(h);
      eval([deblank(h(23,:)),';']);eval([deblank(h(24,:)),';']);
      nextline=[nextline,sprintf('  %s%s',fcmnt1,fcmnt2)];
      nl=length(nextline); nl2=min(nl,width); if width==0 nl2=nl; end
      if (width) fprintf('%s\n',nextline(1:nl2));  end
      sumry=strvcat(sumry,nextline(1:nl2));
      if (nl>nl2 & width)
         fprintf('      %s\n',nextline(nl2+1:nl));
         sumry=strvcat(sumry,[blanks(6),nextline(nl2+1:nl)]);
      end
      %     fprintf('  %s\n',deblank(h(64,9:nch)));
      eval([deblank(h(9,:)),';']); eval([deblank(h(10,:)),';']);
      tims=deblank(h(2,13:nch));
      if isempty(tim{i})
         tim{i}=datenum(tims); itim{i}=isp{it}(rdx{i}(1));
         titl{i}=[fcmnt1,fcmnt2];
      else
         tim{i}=[tim{i},datenum(tims)]; itim{i}=[itim{i},isp{it}(rdx{i}(ni))];
         titl{i}=strvcat(titl{i},[fcmnt1,fcmnt2]);
      end
      nrit=strmatch('Integration Time',h);
      if isempty(nrit)
         ncit=[];
      else
         ncit=findstr(h(nrit,:),'=')+1;
      end
      if isempty(ncit)
         integ=0;
      else
         integ=str2num(h(nrit,ncit:nch));
      end
```

```
            nextline=sprintf('      %s   range:  %7g - %7g nm  integ=%6.3g s',...
               tims,ffirst,flast,integ);
            if (width) fprintf('%s\n',nextline); sumry=strvcat(sumry,nextline); end
         end

      end
      sum=setfield(sum,[type(it,:),'is'],isp{it});
      sum=setfield(sum,[type(it,:),'ir'],rdx);
      sum=setfield(sum,[type(it,:),'time'],tim);
      alltime=[alltime,tim{:}];
      ialltime=[ialltime,itim{:}];
      alltitl=strvcat(alltitl,titl{:});
   end
end
sum.TIME=alltime;
sum.ITIME=ialltime;
sum.SUM=cellstr(sumry);

if ~nochron
   [t2,it2]=sort(alltime);
   if (width) disp(' '); end
   sumry=strvcat(sumry,' ');
   for i=it2
      nextline=sprintf('%s  %s
%s',specall(ialltime(i),:),datestr(alltime(i)),alltitl(i,:));
      if (width) fprintf('%s\n',nextline); end
      sumry=strvcat(sumry,nextline);
   end
   sum.COMMENT=alltitl;
   sum.SUM=cellstr(sumry);
end % ~nochron

while(exist(['specdir','_',num2str(nsavf),'.mat'])==2) nsavf=nsavf+1; end
specdf=['specdir','_',num2str(nsavf)];
savetime=now;
save(specdf,'savetime','sum');
if (width) disp(' '); end
if (width)
   disp(['Results of "specdir" are saved in:  ',specdf,'  ',datestr(savetime)]);
end

cd(olddir); pcssdir=oldpcss;
```

PLOTSFIX - plots corrected spectra (see specfix, plotsfix_eg)

```
%[h,b,g]=plotsfix(spec,shift,bkg,gain,temp,iedge,cosmic)     FJZ 12/6/99
%        plots corrected spectra (see:  specfix)
%        corrects spectra for background noise and gain
%
%   Examples using plotsfix to remove background and gain variations
%   from spectra.
%
%   [h,b,g]=plotsfix('SRaa2.spc',0,'DNaa[2:21]','BBaa[2:11]');
%   h=plotsfix('SRab2.spc',0,b,g);
%
%   [h,b,g]=plotsfix(sum.FL(sum.FLir{2},:),250,'DNca[2:11]','BBca[2:11]');
%
%inputs:
%   spec - 1.) a text matrix of the spectra filenames to plot
%          2.) an FL run number, shot numbers (see specdir)
%          3.) a spectra string, e.g.:  'FLae[3,5:12]'
%   shift - y-offset,shifts each curve vertically, (default 0)
%   bkg - either:
%          1.) a text matrix of filenames of "dark" spectra to
%        average for a background, that is subtracted from the gain
%       and spectra files
%          2.) an n X 2 array of wavelength and background
%       to subtract from the spectra (originally from "dark" files)
%       (if omitted, empty or zero, no background is subtracted)
%          3.) a DN run number (see specdir)
%          4.) a spectra string, e.g.:  'DNae[3,5:12]'
%   gain - either:
%          1.) a text matrix of filenames to average and calculate
%       the gain assuming a black body spectrum at temperature, "temp"
%          2.) an n X 2 array of wavelength and gain to scale
%       the spectra files (originally from black body files)
%       (if omitted, empty, or zero, a gain of 1 is used)
%          3.) a BB run number (see specdir)
%          4.) a spectra string, e.g.:  'BBae[3,5:12]'
%   temp - the filament temperature to remove the black blody shape
%          (only used when gain is a text matrix of black body spectra)
%          (if omitted, empty, or zero, a zero slope is used)
%   iedge - pixels to omit from each edge of the spectra (default=8)
%           (bkg and gain of zero will be returned for these pixels)
%   cosmic - remove cosmic rays from data files with a double pass filter
%            (see cosmicut)
%            If >0 cosmic rays will be removed with r=cosmic.
%            If =0 no cosmic ray removal.
%            If omitted or <0 cosmic rays will be removed with r=5.5 (default).
%
%outputs:
%   h - the last spectra file header
%   b - an n X 2 array of wavelength and background to subtract from spectra
%   g - an n X 2 array of wavelength and gain to scale spectra
%       assuming wavelength values match, corrected spectra:  y
%[h,b,g]=plotsfix(spec,shift,bkg,gain,temp,iedge,cosmic)     FJZ 12/6/99
function [h,b,g]=plotsfix(spec,shift,bkg,gain,T,iedge,cosmic)

global colors curvestyle
if ~exist('cosmic') cosmic=-1; end; if notarg(cosmic) cosmic=-1; end;
if cosmic<0 cosmic=5.5; end
if ~exist('T') T=0; end; if notarg(T) T=0; end;
if ~exist('gain') gain=[0,1]; end; if notarg(gain,0) gain=[0,1]; end;
if ~exist('bkg') bkg=[0,0]; end; if notarg(bkg) bkg=[0,0]; end;
if ~exist('shift') shift=0; end; if notarg(shift) shift=0; end;
if ~exist('spec') spec=''; end; if notarg(spec,0) spec=''; end;
```

```
if ~exist('iedge') iedge=8; end; if notarg(iedge) iedge=8; end
spectitl='';
sd=specdir('',0,0);
cssave=curvestyle; if isempty(curvestyle) curvestyle='-'; end
specrun=0;
savspec=spec(1,:);
if ~ischar(spec)
   specrun=spec(1);
   nshots=length(spec)-1;
   if nshots>0
      spec=sd.FL(sd.FLir{specrun}(spec(2:nshots+1)),:);
      if nshots<savspec(nshots+1)-savspec(2)+1
         spectitl=['FL',sd.FLr(specrun,:),'[',num2str(savspec(2:nshots+1)),']'];
      else
         spectitl=['FL',sd.FLr(specrun,:),'[',num2str(savspec(2)),'-
',num2str(savspec(nshots+1)),']'];
      end
   else
      spec=sd.FL(sd.FLir{specrun},:);
      [nsr,nsc]=size(sd.FL(sd.FLir{specrun},:));
      spectitl=['FL',sd.FLr(specrun,:),'[1-',num2str(nsr),']'];
   end
end
bkgrun=0; if ~ischar(bkg)&length(bkg)==1 bkgrun=bkg;  bkg=sd.DN(sd.DNir{bkgrun},:);
end
gainrun=0; if ~ischar(gain)&length(gain)==1 gainrun=gain;
gain=sd.BB(sd.BBir{gainrun},:); end

if isempty(spec)
   disp('Select the spectra files.  (Select "cancel" when complete.)');
   pause(2);
   spec=getlist;
   if isempty(spec) disp('No spectra selected.  PLOTSFIX exiting...'); return; end
   [nsr,nsc]=size(spec);
   if nr==1
      spectitl=spec;
   elseif nr>1
      spectitl=[spec(1,:),'-',spec(nsr,:)];
   end
end

[nsr,nsc]=size(spec);
if ~any(spec(1,:)=='.')
   spec=specstr(spec,sd);
   if isempty(spec) disp('Spectra not found.  PLOTSFIX exiting...'); return; end
   [nsr,nsc]=size(spec);
   spectitl=savspec;
end
b=bkg; g=gain; yshift=0;
clf; hold on; box on;
xlabel('Wavelength (nm)');
ylabel('Intensity (arb.)');
for i=1:nsr
   [s,h,b,g]=specfix(spec(i,:),b,g,T,iedge,cosmic);
   if isempty(s)
      disp('No corrected spectra.');
      bkg
      gain
      disp('PLOTSFIX exiting ...');
      return;
   end
   plot(s(:,1),s(:,2)+yshift,colors(mod(i-1,length(colors))+1));
   yshift=yshift+shift;
```

```
end
if isempty(spectitl)
   if nsr==1
      spectitl=spec;
   else
      spectitl=[spec(1,:),'-',spec(nsr,:)];
   end
end
axis tight;
[nhr,nhc]=size(h); rtitl='(';
if specrun>0
   rtitl=[rtitl,num2str(specrun)];
else
   if nargin>0
      if isempty(inputname(1))
         rtitl=[rtitl,savspec];
      else
         rtitl=[rtitl,inputname(1)];
      end
   end
end
if bkgrun>0
   rtitl=[rtitl,',',num2str(bkgrun)];
else
   if nargin>2
      if isempty(inputname(3))
         rtitl=[rtitl,',',bkg];
      else
         rtitl=[rtitl,',',inputname(3)];
      end
   end
end
if gainrun>0
   rtitl=[rtitl,',',num2str(gainrun)];
else
   if nargin>3
      if isempty(inputname(4))
         rtitl=[rtitl,',',gain];
      else
         rtitl=[rtitl,',',inputname(4)];
      end
   end
end
rtitl=[rtitl,')'];
eval([deblank(h(23,:)),';']);eval([deblank(h(24,:)),';']);
nrit=strmatch('Integration Time',h);
if isempty(nrit)
   ncit=[];
else
   ncit=findstr(h(nrit,:),'=')+1;
end
if isempty(ncit)
   integ=0;
else
   integ=str2num(h(nrit,ncit:nhc));
end
cornlab([fcmnt1,fcmnt2],h(2,13:nhc),...
   [rtitl,', shift=',num2str(shift),', iedge=',num2str(iedge)],...
   [deblank(spec(1,:)),'-',deblank(spec(nsr,:)),' integ=',num2str(integ),' s']);
n1=findstr(spec(1,:),'.')-1;
n2=findstr(spec(nsr,:),'.')-1;
sptypes={'Filament Laser','Dark Noise','Black Body'};
spshort={'FL','DN','BB'};
```

```
nshort=strmatch(upper(spec(1,1:2)),spshort);
if isempty(nshort)
   titl1=upper(spec(1,1:2));
else
   titl1=sptypes{nshort};
end
if nsr>1
   title([titl1,' Spectra: ',spectitl]);
else
   title([titl1,' Spectrum: ',spectitl]);
end
boldplot(0.5,14,14,14,10);
curvestyle=cssave;
```

FLSPPLOT - plots the results from flanyl

```
%flspplot(nfl,sumfile) - plots results from "flanyl"
function flspplot(nfl,sumfile)
global colors
if ~exist('nfl') nfl=0; end; if notarg(nfl,0) nfl=max(get(0,'children'))+1; end
if isempty(nfl) nfl=1; end
if ~exist('sumfile') sumfile=[]; end; if notarg(sumfile,0) sumfile=[]; end
if isempty(sumfile)
   d=dir('summary_*.mat');
   tn=str2mat(d(:).name);
   ns=max(str2num(tn(:,9:10)));
   if isempty(ns) disp('No summary files found.  Exiting...'); return; end
   sumfile=['summary_',num2str(ns)];
end
disp(['Loading spectral analysis file:  ',sumfile]);
s=load(sumfile);
pfn={'energy','centroid','width','peak','peakposition'};
ylb={'Energy (arb.)','Wavelength (nm)','Width (2*2nd moment) (nm)',...
      'Peak Intensity (arb.)','Position of Peak Intensity (nm)'};
nplts=length(ylb);
nfig=nfl-1;
for ifig=1:nplts
   nfig=nfig+1;
   figure(nfig);
   hold on
   y=getfield(s,pfn{ifig});ncrvs=length(y);
   lastc=0;
   for i=1:ncrvs
      plot((2:length(y{i}))+lastc,[y{i}(2:length(y{i}))],[colors(mod(i,6)+1)]);
      plot((1:length(y{i}))+lastc,[y{i}],[colors(mod(i,6)+1),'+']);
      lastc=lastc+length(y{i});
   end
%   for i=1:ncrvs
plot(current(i)*ones(1,length(centroid{i})),[centroid{i}],[colors(mod(i,6)+1),'+']);
end
%   for i=1:ncrvs
plot(current(i)*ones(1,length(centroid{i})),[centroid{i}],[colors(mod(i,6)+1)]); end
   axis tight
   title(['Filament Laser Spectral ',upper(pfn{ifig})]);
   ylabel(ylb{ifig});
   xlabel('Shot Number')
   cornlab(s.comment1,s.comment2,s.specdir,['FL:',s.runname(1,:),'-',...
        s.runname(ncrvs,:),'  shift=',num2str(s.shift),'  iedge=',num2str(s.iedge)]);
   boldplot(1,16,14,14,10)
   hold off
end
figspred(1:nfig,[1,200],[450,20],[600,500]);
prtfigset(1:nfig,'l');  %landscape orientation for printing
```

FLANYL - analyzes filament laser spectra and stores spectra and run names, energies, centroids, widths, peak values, and peak positions.

```
%[specname,runname,rdx,energy,centroid,width]=flanyl(specdir,pdx,shift,bkg,iedge)
%
%   "flanyl" (Filament Laser ANALysis) processes filament laser spectra.
%
%outputs:
%  specname - a text matrix of all the filenames, FL*.SPC, in "directory"
%  runname - a text matrix of the 4 letter run names, e.g. ['FLIJ';'FLIK'; ...
%  rdx - a cell array of the indeces of the spectra for each run
%        specname(rdx{i}(j),:) is the filename of jth spectrum in the ith run
%  energy - is a cell array of the energy in each spectrum of each run:  $y
%  centroid - is a cell array of the centroids ...:  $(x*y)/$y
%  width - is a c. a. of twice the 2nd moments (~width):  2*sqrt($((x-c)^2*y)/$y)
%        the energy, centroid, or width of the jth spectrum in the in the ith run
%        is given by _____{i}(j)
%
%inputs:
%  specdir - is the directory of the spectra to be analyzed (default is cd)
%  pdx - indeces of runs to process.  If pdx=-1 all runs are analyzed.  (default 0)
%  shift - is a vertical offset to add to each spectrum for plotting
%  bkg - is a text matrix of background filenames to be averaged and subtracted
%        from all the spectra (bkg assumes the same x-values for all spectra)
%
%Inputs and outputs are saved in a file named:  summary_n.mat
function
[specname,runname,rdx,energy,centroid,width]=flanyl(specdir,pdx,shift,bkg,iedge)
global pcssdir colors
oldpcss=pcssdir;
if ~exist('specdir') specdir=cd; end; if notarg(specdir,0) specdir=cd; end
if ~strcmp(specdir,pcssdir) pcssdir=specdir; end
specname=getlist('FL*.SPC'); i=smartsort(specname,-1); specname=specname(i,:);
pcssdir=oldpcss;
[ns,nc]=size(specname);
[runname,ir]=unique(specname(:,3:4),'rows');
[nrs]=length(ir);
if ~exist('pdx') pdx=0; end; if notarg(pdx) pdx=0; end
if pdx==-1 pdx=1:nrs; end
if ~exist('shift') shift=0; end; if notarg(shift) shift=0; end
% average background
if ~exist('iedge') iedge=[]; end; if notarg(iedge) iedge=0; end
if ~exist('bkg') bkg=[]; end; if notarg(bkg,0) bkg=[]; end
if ~isempty(bkg)
   [nbr,nbc]=size(bkg); yb=[];
   for i=1:nbr
      [x,y]=getdata(bkg(i,:));
      y=y(iedge+1:length(y)-iedge);
      x=x(iedge+1:length(x)-iedge);
      if isempty(yb)
         yb=y;
      else
         yb=yb+y;
      end
   end
   yb=yb/nbr;
else
   yb=0;
end
for i=1:nrs
   rdx{i}=find(runname(i,1)==specname(:,3)&runname(i,2)==specname(:,4));
```

```
    disp(['------------------  RUN:  FL',runname(i,:),'  --------------------']);
    s=displist(specname(rdx{i},:)); disp(s);
    if any(i==pdx)
       [x,y,h]=getdata(specname(rdx{i}(1),:));
       [nr,nc]=size(h);
       disp([deblank(h(2,13:nc)),' - ',deblank(h(64,9:nc))]);
       eval([deblank(h(9,:)),';']);eval([deblank(h(10,:)),';']);
       disp(['range:  ',num2str(ffirst,7),' - ',num2str(flast,7),' (nm)']);
       figure(i);clf;
       shift0=0;
       nj=length(rdx{i});
       for k=1:nj
          j=rdx{i}(k);
          [x,y]=getdata(specname(j,:));
          y=y(iedge+1:length(y)-iedge);
          x=x(iedge+1:length(x)-iedge);
          npts=length(y);
          y=y-yb; hold on;
          energy{i}(k)=sum(y);
          centroid{i}(k)=x'*y/energy{i}(k);
          width{i}(k)=sqrt(abs((x-centroid{i}(k))'.^2*y/energy{i}(k)))*2;
          [peak{i}(k),peakpnt]=max(y);
          peakposition{i}(k)=x(peakpnt);
          icol=mod(j,length(colors))+1;
          plot(x,y+shift0,colors(icol));
          shift0=shift0+shift;
       end
       title('Filament Laser Spectra');
       xlabel('Wavelength (nm)');
       ylabel('Intensity (arb.)');
       eval([deblank(h(23,:)),';']);eval([deblank(h(24,:)),';']);
       axis tight;
       box on;
       cornlab(fcmnt1,fcmnt2,h(2,13:nc),['FL',runname(i,:),'1-',num2str(nj),...
             '  shift=',num2str(shift),'  iedge=',num2str(iedge)]);
       if i==1 comment1=fcmnt1; comment2=fcmnt2; end
       boldplot(1,16,14,14,10);
       hold off;
    end
end
figspred(1:nrs,[1,200],[450,20],[600,500]);
prtfigset(1:nrs,'l');  %landscape orientation for printing
nf=1;
while(exist(['summary','_',num2str(nf),'.mat'])==2) nf=nf+1; end
sumfile=['summary','_',num2str(nf)];
savetime=now;
save(sumfile,'savetime','specdir','pdx','shift','bkg','iedge',...
   'comment1','comment2','specname','runname','rdx',...
   'energy','centroid','width','peak','peakposition');
disp(['Results of spectral analysis are saved in:  ',sumfile,'  ',datestr(savetime)]);
```

(This page intentionally left blank)

Appendix D. Waveform digitizer data analysis: getdata, plotdata, addata, plotcurve, addcurve, plotlist, and plotshif

GETDATA – read data files saved by various waveform digitizers

```
%[x,y,h,trig,all,par,iev]=getdata(file,xlim,sv,envelope,ncpe,quiet);
%                fjz 12/14/00
%
%  opens a file or file menu and returns x, y, and h (header) information.
%  PCSS plot routines "plotcurv", "addcurve", "plotlist", "plotshif",
%  "matxplot", ...  use "getdata" to input and scale x and y and "plotdata"
%  to create plots with the parameters shown in "showdefs".
%
%input parameters:
%    file = an optional file name.  If empty/omitted, a file menu is opened.
%           Valid file types are: TEK SCD, HP 54111, TEK TDS, LeCroy 6894,
%           spectra, DADisp column, PPat 2 column, GENerated data.  Files from
%           TEK TDS must be converted to ASCII (.ASC) with CNVRTWFM.
%    xlim = [xmin,xmax] an optional range for the scaled x values in the plot.
%           and event # for sequenced data
%      sv = a list of events for "sequenced" data files (LeCroy).
%envelope = an optional flag to input envelope data only (inductive hardening)
%           If non-zero, envelope data will be extracted from the raw data
%           and stored in x, y, par, and iev.
%    ncpe = the average number of cycles per event (used with envelope).
%           If 0, empty, or omitted, ncpe=nppt=length(x)/length(trig).
%   quiet = an optional flag to eliminate standard output messages.
%
%returned values:
%       x,y = arrays of x and y coordinates.  The original values,
%             X and Y, are scaled by default factors (help showdefs).
%                 x = X*xscale+xoff   and   y = Y*yscale+yoff
%         h = an text matrix of "header" information from the data file.
%trig & all = 3 column arrays (tables) of trigger times, offsets, event numbers
%             for each event in "sequenced" data files (LeCroy).
%             Trigger matrices contain a row in the matrix for each event.
%             "trig" returns this information only for the events
%             which are specified in "sv".  "all" returns it for all events.
%             The data in "x" and "y" corresponds to the triggers stored in
%             "trig".  There are nppt=length(x)/length(trig) points per trigger
%             and x includes the time offsets (time from the 1st trigger).
%       par = parameters about the raw data (see envelope).
%       iev = the indeces of the 1st envelope point for each event.
%             Cycle starting points indeces (grouped by event) are defined by:
%                                  start{ev}=iev(ev)+offset:8:iev(ev-1)-1;
%             where "ev" is the index of an event in the sequence vector
%             and "offset" is defined in envelop1 (offset=1 for start points).
%             All starting point indeces (not grouped by event) are also given
%             by:     offset:8:length(x)      See:  help envelop1
%
%[x,y,h,trig,all,par,iev]=getdata(file,xlim,sv,envelope,ncpe,quiet);
function [x,y,header,triggers,alltrig,par,iev] =
getdata(file,xlim,svin,envel,ncpe,quiet)
global ptitle xlab ylab xscale yscale xoff yoff integrate pcssdir pcssfiles;
global invertsp tekbug tekfix mindelt cosmic;
if ~exist('file','var')   | notarg(file,0) file=''; end;
if ~exist('xlim','var')   | notarg(xlim,0) xlim=[-inf,inf]; end
if (length(xlim)~=2) xlim=[-inf,inf]; end;
```

```
if ~exist('svin','var')   | notarg(svin)    svin=1; end;
if ~exist('envel','var')  | notarg(envel)   envel=0; end
if ~exist('ncpe','var')   | notarg(ncpe)    ncpe=0; end
if ~exist('quiet','var')  | notarg(quiet)   quiet=0; end
if iscell(file) file=strvcat(file); end;
tic;
sd(pcssdir);
if (length(file)>0)
   fullname=file;
   pa='';
   fn=deblank(file);
   if length(findstr(file,'*'))+length(findstr(file,'?'))>0
      [fn,pa] = uigetfile(file,'GetData');
      if (fn==0)
         disp(['No filename selected in "getdata".  Exiting ...']);x=[];y=[];
         header=[];triggers=[];alltrig=[];return;
      end
   else
      pa='';
   end
else
   if isempty(pcssfiles)
      [fn,pa] = uigetfile('*.*','GetData');
   else
      [fn,pa] = uigetfile(pcssfiles,'GetData');
   end
end
itype=1+max(find(fn=='.'));
typ=fn(itype:end);
fullname=[pa,fn];
ifn1=find(fullname=='\'); if isempty(ifn1) ifn1=0; end;
shortname=fullname((ifn1+1):end);
if ~quiet disp(' '); disp(['Reading:  ',pa,fn]); end;
%
% Generated (simulated) data for software testing (See: LeCroyGen)
%
if strcmp(upper(typ),'GEN')
    load(fullname,'-mat');
    % loads ds - struct var: name, ringfreq, phase, mag, eventime, nevt, resolution,
    %                        nskip, offset, pretrig, header
    %        y - y-values of generated data
    %       at - trigger matrix (nevt X 3) all events
    disp(['getdata - generated data:  ',ds.name]);
    nppt=round(ds.eventime/ds.resolution);  % number of points per trigger (event)
    nevt=ds.nevt;                           % number of events (triggers)
    npt=nppt*ds.nevt;                       % number of points (total)
    tbe=ds.nskip/ds.ringfreq+ds.eventime;   % time between events
    ie=1:nevt;                              % event index
    ip=1:nppt;                              % point index
    x=(ip-1)*ds.resolution*xscale+xoff;
    at(:,[1 2])=at(:,[1 2])*xscale+xoff;
    alltrig=at;
    at=alltrig(svin',1)+alltrig(svin',2);
    triggers=alltrig(svin',:);
    %size(at)
    %size(t)
    %size(x(ones(length(sv),1),:))
    %size(squeeze(at(sv',1,ones(nppt,1))))
    x=reshape((x(ones(length(svin),1),:)+at(:,ones(nppt,1)))',1,nppt*length(svin));
    if isfield(ds,'chirp')
       for i=1:length(svin)
          y((i-1)*nppt+ip)=y(nppt*(svin(i)-1)+ip);
       end
```

```
      y=y(1:(nppt*(length(svin))));
    else
      y=reshape(y(svin,:)',1,nppt*length(svin))*yscale+yoff;
    end
    if ~isempty(ds.header) header=ds.header; else header=''; end
    header=strvcat(shortname,header(2:end,:));
    par=ds; par.npt=npt; par.nppt=nppt; par.tbe=tbe; par.sv=svin;
    if ~quiet
      disp(['Generated data:  ',num2str(ds.nevt),' events, ',num2str(nppt),...
            ' pts/event, ', num2str(npt),' pts, ']);
      disp(['                 extends from ',num2str(alltrig(1,1)+alltrig(1,2)),...
            ' to ',num2str(alltrig(nevt,1)+alltrig(nevt,2)+nppt*ds.resolution),'.']);
      disp(['                 Time between events is ',num2str(tbe),'.']);
        if xscale==1
          disp( '                 (Time is in seconds.)');
        else
          disp(['                 (Time is in ', num2str(1/xscale),' seconds.)']);
        end;
      end;
    return;
end

[fid,msg] = fopen([pa,fn],'r');

if (fid==-1)
   disp(['ERROR:  Filename, "',pa,fn,'" not found.  "getdata" exiting...']);
   disp(msg);x=[];y=[];header=[];triggers=[];alltrig=[];
   return;
end
fullname=[pa,fn];
ifn1=find(fullname=='\'); if isempty(ifn1) ifn1=0; end;
shortname=fullname((ifn1+1):end);
header=shortname;
%
% Ocean Optics Inc. - Spectrometer files
%
if strcmp(upper(typ),'OOI')
   if ~quiet disp('Ocean Optics Inc. (OOI) spectrometer file found.'); end;
   header=str2mat(header,'(spectrum)','Ocean Optics Inc.');
   actchan=fread(fid,1,'int16');
   header=str2mat(header,['active channel = ',num2str(actchan)]);
   darksav=fread(fid,1,'int16');
   header=str2mat(header,['dark saved = ',num2str(darksav)]);
   refsav=fread(fid,1,'int16');
   header=str2mat(header,['reference saved = ',num2str(refsav)]);
   samsav=fread(fid,1,'int16');
   header=str2mat(header,['sample saved = ',num2str(samsav)]);
   viewmode=fread(fid,1,'int16');
   header=str2mat(header,['view mode = ',num2str(viewmode)]);
   fseek(fid,42,'bof');
   if darksav ydark=fread(fid,2048,'float')*yscale+yoff; end
   if refsav yref=fread(fid,2048,'float')*yscale+yoff; end
   x=fread(fid,2048,'float')*xscale+xoff;
   y=fread(fid,2048,'float')*yscale+yoff;
   if darksav y=[y;ydark]; end
   if refsav y=[y;yref]; end
   fclose(fid);
   if isempty(xlab) xlab='Wavelength (nm)'; end
   if isempty(ylab) ylab='Counts'; end
   return
end
shotcom(fn,-1);
s=fscanf(fid,'%c',1024);
```

```
l=length(s);
cr=findstr(s,setstr(13));
if (length(cr)==0) cr=0; end;
%
% Tektronix SCD5000 data files
%
if strcmp(s(1:7),'SCD5000')
   s=[s,fscanf(fid,'%c',10240)];
   l=length(s);
   fclose(fid);
   h1=cr(5);
   npts=getvalue(s,'NR.PT:',',');
   xm=getvalue(s,'XINCR:',',')*xscale;
   if (tekbug)
      x0=xm*getvalue(s,'PT.OFF:',',')*tekfix+xoff;
      disp('Warning - a Tek SCD5000 bug in the trigger delay is being corrected.');
      disp('Set global variable tekbug=0 to inhibit this correction.');
   else
      disp('Warning - some Tek SCD5000 scopes have a bug in the trigger delay.');
      disp('If the starting time for your data is wrong, set global variable');
      disp('tekbug=-1 to fix this bug (multiplies the time offset by
tekfix=1.8011).');
      disp('Not all SCDs have this problem.');
      x0=xm*getvalue(s,'PT.OFF:',',')+xoff;
   end
   ym=getvalue(s,'YMULT:',',')*yscale;
   y0=(getvalue(s,'YZERO:',',')*yscale-ym*getvalue(s,'YOFF:',','))+yoff;
   imin=max(fix((xlim(1)-x0)/xm+1),1);
   imax=min(fix((xlim(2)-x0)/xm+1),npts);
   imax=max(imax,imin);
   x=(x0+(imin-1)*xm):xm:(x0+(imax-1)*xm);
   y=sscanf(s(h1+2:l),'%d',imax);
   y=y(imin:imax);
   y=ym*y'+y0;
   ln=wrap(s,20,74,5);
   title=ln(2,1:findstr(ln(2,:),'Rec ')-1);
   header=str2mat(header,title,ln);
   %
   %Tektronix TDS files which have been converted to ASCII with CNVRTWFM
   %
elseif strcmp(s(1:16),':WFMPRE:BYT_NR 1')
   s=[s,fscanf(fid,'%c',1024000)];
   l=length(s);
   fclose(fid);
   npts=getvalue(s,'NR_PT',';');
   xm=getvalue(s,'XINCR',';')*xscale;
   x0=xm*getvalue(s,'PT_OFF',';')+xoff;
   ym=getvalue(s,'YMULT',';')*yscale;
   y0=(getvalue(s,'YZERO',';')*yscale-ym*getvalue(s,'YOFF',';'))+yoff;
   imin=max(fix((xlim(1)-x0)/xm+1),1);
   imax=min(fix((xlim(2)-x0)/xm+1),npts);
   imax=max(imax,imin);
   x=(x0+(imin-1)*xm):xm:(x0+(imax-1)*xm);
   h1=findstr(s,':CURVE');
   k1=h1+6;
   %  t0 = clock;
   y=sscanf(s(k1:l),'%g,%g,',imax);
   y=y(imin:imax);
   %  etime(clock,t0)
   y=ym*y'+y0;
   k1=findstr(s,'WFID')+6;
   k2=findstr(s(k1:length(s)),'"');
   k2=min(k2,50-length(fullname))+k1-2;
```

```
    title=s(k1:k2);
    header=str2mat(header,title,wrap(s(1:h1-1),10,74,0));
    %
    %   Hewlett Packard 54111 data files (from Lotus data acquisition)
    %
elseif strcmp(s(1:6),'ASCII:')
    s=[s,fscanf(fid,'%c',102400)];
    l=length(s);
    fclose(fid);
    cr=findstr(s,setstr(13));
    nlh=getvalue(s(1:cr(1)),'ASCII:',',');
    last=getvalue(s(cr(nlh-2):cr(nlh-1)),'-',setstr(13));
    first=getvalue(s(cr(nlh-2):cr(nlh-1)),'points:','-');
    npts=last-first+1;
    xm=getvalue(s,'xinc=',setstr(13))*xscale;
    x0=getvalue(s,'x0=',setstr(13))*xscale+xoff;
    ym=getvalue(s,'yinc=',setstr(13))*yscale;
    y0=getvalue(s,'y0=',setstr(13))*yscale+yoff;
    imin=max(fix((xlim(1)-x0)/xm+1),1);
    imax=min(fix((xlim(2)-x0)/xm+1),npts);
    imax=max(imax,imin);
    x=(x0+(imin-1)*xm):xm:(x0+(imax-1)*xm);
    [y,count,ermsg,nextindx]=sscanf(s((cr(nlh)+2):l),'%u',imax);
    %  ['|',s((cr(nlh)+2):(cr(nlh)+20)),'|']
    %  count
    %  ermsg
    %  nextindx
    %  size(y)
    y=y(imin:imax);
    y=ym*y'+y0;
    ln=wrap(s,57,74,57);
    header=str2mat(header,ln(7,:),ln);
    %
    %          *.SPC spectra from ISA 500M spectrometer (GRAMS software)
    %
elseif strcmp(upper(typ),'SPC')
    % read header SPCHDR
    blockoff=1;
    header=str2mat(header,'(spectrum)','Instruments SA (Gramms)');
    header=dumpvar(s,header,0,blockoff,'ftflgs',0,'uint',1);
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,blockoff,'fversn',1,'uint',1);
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,blockoff,'fexper',2,'uint',1);
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,blockoff,'fexp',3,'uint',1);
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,blockoff,'fnpts',4,'int*4');
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,blockoff,'ffirst',8,'double');
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,blockoff,'flast',16,'double');
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,blockoff,'fnsub',24,'int*4');
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,blockoff,'fxtype',28,'uint',1);
    header=dumpvar(s,header,0,blockoff,'fytype',29,'uint',1);
    header=dumpvar(s,header,0,blockoff,'fztype',30,'uint',1);
    header=dumpvar(s,header,0,blockoff,'fpost',31,'uint',1);
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,blockoff,'fdate',32,'uint*4');
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    spcmn=bitand(fdate,2^6-1);
```

```
spchr=bitand(bitshift(fdate,-6),2^5-1);
spcda=bitand(bitshift(fdate,-11),2^5-1);
spcmo=bitand(bitshift(fdate,-16),2^4-1);
spcyr=bitand(bitshift(fdate,-20),2^12-1);
spcdn=datenum(spcyr,spcmo,spcda,spchr,spcmn,0);
spcdate=datestr(spcdn,2);
spctime=datestr(spcdn,13);
header(2,13:32)=datestr(spcdn);
if ~quiet disp(header(1:3,:)); end;
header=str2mat(header,['spcdate=',spcdate],['spctime=',spctime]);
header=dumpvar(s,header,0,blockoff,'fres',36,'string',9);
header=dumpvar(s,header,0,blockoff,'fsource',45,'string',9);
header=dumpvar(s,header,0,blockoff,'fpeakpt',54,'word');
header=dumpvar(s,header,0,blockoff,'fspare',56,'string',32);
% header=dumpvar(s,header,0,blockoff,'fcmnt',88,'string',130);
spcmnt=deblank(s(89:88+130)); lcmnt=length(spcmnt);
if lcmnt>0
  lc1=fix((lcmnt+1)/2);
  spcmnt1=spcmnt(1:lc1);
  if lcmnt>lc1
    spcmnt2=spcmnt(lc1+1:lcmnt);
  else
    spcmnt2=' ';
  end
else
  spcmnt1=' ';
end
header=str2mat(header,['fcmnt1=''',spcmnt1,'''']);
header=str2mat(header,['fcmnt2=''',spcmnt2,'''']);
if ~quiet disp(['comment:  ',spcmnt1,spcmnt2]); end;
header=dumpvar(s,header,0,blockoff,'fcatxt',218,'string',30);
[nrh,nch]=size(header);
if length(header(nrh,:))>9;
 spclab=deblank(header(nrh,9:nch));
 lslab=length(spclab);
  spclab=spclab(1:lslab-1); lslab=lslab-1;
  ilbs=find(spclab==0);
  if length(ilbs)==0
    spcxlab=spclab(1:lslab);
  elseif length(ilbs)==1
    spcxlab=spclab(1:ilbs(1)-1);
    spcylab=spclab(ilbs(1)+1:lslab);
  elseif length(ilbs)==2
    spcxlab=spclab(1:ilbs(1)-1);
    spcylab=spclab(ilbs(1)+1:ilbs(2)-1);
    spczlab=spclab(ilbs(2)+1:lslab);
  end
end
if exist('spcxlab','var')
  if ~quiet disp(['x-axis range:  ',num2str(ffirst),' - ',num2str(flast)]); end;
  if ~quiet disp(['x-axis label:  ',spcxlab]); end
  header=str2mat(header,['spcxlab=''',spcxlab,'''']);
end
if exist('spcylab','var')
  if ~quiet disp(['y-axis label:  ',spcylab]); end;
  header=str2mat(header,['spcylab=''',spcylab,'''']);
end
if exist('spczlab','var')
  if ~quiet disp(['z-axis label:  ',spczlab]); end;
  header=str2mat(header,['spczlab=''',spczlab,'''']);
end
header=dumpvar(s,header,0,blockoff,'flogoff',248,'int*4');
[nrh,nch]=size(header); eval([header(nrh,:),';']);
```

```
    header=dumpvar(s,header,0,blockoff,'fmods',252,'int*4');
    header=dumpvar(s,header,0,blockoff,'fprocs',256,'uint',1);
    header=dumpvar(s,header,0,blockoff,'flevel',257,'uint',1);
    header=dumpvar(s,header,0,blockoff,'fsampin',258,'word');
    header=dumpvar(s,header,0,blockoff,'ffactor',260,'float');
    header=dumpvar(s,header,0,blockoff,'fmethod',264,'string',48);
    header=dumpvar(s,header,0,blockoff,'fzinc',312,'float');
    header=dumpvar(s,header,0,blockoff,'fwplanes',316,'int*4');
    header=dumpvar(s,header,0,blockoff,'fwinc',320,'float');
    header=dumpvar(s,header,0,blockoff,'fwtype',324,'uint',1);
    header=dumpvar(s,header,0,blockoff,'freserv',325,'string',187);
% read x-values
    blockoff=513;  fseek(fid,blockoff-1,'bof');
    x=fread(fid,fnpts,'float')*xscale+xoff;
%       x=2^(fexp-32)*fread(fid,fnpts,'int32')*xscale+xoff;
if all([abs((ffirst-x(1))/ffirst)>1e-7,abs((ffirst-x(fnpts))/ffirst)>1e-7])
  if ~quiet disp('First x-value different from header.'); end
  return;
end
if all([abs((flast-x(fnpts))/flast)>1e-7,abs((flast-x(1))/flast)>1e-7])
  if ~quiet disp('Last x-value different from header.'); end;
  return;
end
% read sub header
blockoff=blockoff+4*fnpts;
    s=fscanf(fid,'%c',32);
    header=dumpvar(s,header,0,01,'subflgs'   , 0,'uint',1 );
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,01,'subexp'    , 1,'uint',1 );
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,01,'subindx'   , 2,'word' );
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,01,'subtime'   , 4,'float');
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,01,'subnext'   , 8,'float');
    header=dumpvar(s,header,0,01,'subnois'   ,12,'float');
    header=dumpvar(s,header,0,01,'subnpts'   ,16,'int*4');
    header=dumpvar(s,header,0,01,'subscan'   ,20,'int*4');
    header=dumpvar(s,header,0,01,'subwlevel',24,'float');
    header=dumpvar(s,header,0,01,'subresv'   ,28,'string',4);
% read y-values
blockoff=blockoff+32;
  y=2^(subexp-32)*fread(fid,fnpts,'int32')*yscale+yoff;
  if cosmic
    y2=y(4:fnpts);
    if cosmic>0
      [y2,yfc,ibad]=cosmicut(y2,cosmic);
    else
      [y2,yfc,ibad]=cosmicut(y2);
    end
    y=[y(1:3);y2];
    lbad=min(50,length(ibad));
    if lbad>0
      disp('cosmic rays removed at x = ');
      disp(x(ibad(1:lbad)));
    end
  end
% read log header
blockoff=blockoff+fnpts*4;
    s=fscanf(fid,'%c',64);
    header=dumpvar(s,header,0,01,'logsizd'   , 0,'int*4');
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    spcsize=flogoff+logsizd;
```

```
    if ~quiet disp(['size:  ',num2str(spcsize),' bytes']); end;
    header=dumpvar(s,header,0,01,'logsizm'  , 4,'int*4');
    header=dumpvar(s,header,0,01,'logtxto'  , 8,'int*4');
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,01,'logbins'  ,12,'int*4');
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,01,'logdsks'  ,16,'float');
    [nrh,nch]=size(header); eval([header(nrh,:),';']);
    header=dumpvar(s,header,0,01,'logspar'  ,20,'string',44);
    header=str2mat(header,['spcsize=',num2str(spcsize)]);
    % read log values
    blockoff=blockoff+64;
    s=fscanf(fid,'%c',logsizd-64);
    logtxt=s(logtxto-64+1:logsizd-64);
    llog=find(logtxt==0);
    if llog>0
      logtxt=logtxt(1:llog(1)-1);
      crlog=find(logtxt==13);
      if length(crlog)>0
        icr1=1;
        for icr=crlog
          if icr-icr1<=nch
            header=str2mat(header,logtxt(icr1:icr-1));
          else
            header=str2mat(header,logtxt(icr1:icr1+nch-1),...
            logtxt(icr1+nch:icr-1));
          end
          icr1=icr+2;
        end
      else
        disp('No end-of-lines (13) found in log text.');
      end
    else     disp('No end-of-text (0) found in log text.');
    end
    fclose(fid);
    %
    %spectra from Anita's spectrometer
    %
elseif strcmp(s(17:23),'0.00000')
    s=[s,fscanf(fid,'%c',102400)];
    fclose(fid);
    spec=sscanf(s,'%g%g',[2,1152]);
    n=length(spec(1,:));
    x=spec(1,:);
    if (invertsp)
       y=spec(2,n:-1:1);
    else
       y=spec(2,:);
    end
    header=str2mat(header,'(spectrum, Anita''s)');
    %
    %BH loops from KJS Associates (via Bruce Kelley & Doug Adkins)
    %
elseif strcmp(s(1:6),'Header')
    if strcmp(s(9:13),'Label') | all(s(20:26)==char(196))
       ls=length(s);
       s=[s,fscanf(fid,'%c',6000)];
       l=length(s);
       ns=findstr(s,'Data Points');
       crnext=findstr(s(ns:ns+180),setstr(13));
       if all(s(20:26)==char(196))
          ns=ns+crnext(3)+1;
          fseek(fid,ns,-1);
```

```
            bh=fscanf(fid,'%g :%g',[2,1024000]);
        else
            ns=ns+crnext(2)+1;
            fseek(fid,ns,-1);
            bh=fscanf(fid,'%g:%g',[2,1024000]);
        end
        fclose(fid);
%        disp(['|',s(ns:ns+25),'|'])
        [n,nc]=size(bh);
        x=bh(1,:);
        y=bh(2,:);
        header=str2mat(header,'(BH loop)');
        head2=wrap(s(1:ns-2),100,80,100);
        header=str2mat(header,head2);
    end
    %
    %DADISP single column version 1 (Mitch)
    %
elseif strcmp(s(1:4),'    ') & cr(1)==21;
    s=[s,fscanf(fid,'%c',102400)];
    fclose(fid);
    y=sscanf(s,'%g');
    npts=length(y);
    imin=fix(max(xlim(1)-xoff,0)/xscale)+1;
    imax=fix(min((xlim(2)-xoff)/xscale+1,npts));
    imax=max(imax,imin);
    x=(xoff+(imin-1)*xscale):xscale:(xoff+(imax-1)*xscale);
    y=y(imin:imax)*yscale+yoff;
    header=str2mat(header,'(DADISP column)');
    %
    %DADISP single column version 2(Gary)
    %
elseif strcmp(s(1:7),'DATASET');
    s=[s,fscanf(fid,'%c',102400)];
    fclose(fid);
    header=str2mat(header,'(DADISP column version 2)');
    header=str2mat(header,s(1:cr(1)-1));
        for i=1:14
    header=str2mat(header,s(cr(i)+2:cr(i+1)-1));
            end
y=sscanf(s(cr(15)+2:length(s)),'%g');
    npts=length(y)
    interval=sscanf(s(cr(7)+10:cr(8)),'%g')
    x_offset=sscanf(s(cr(8)+10:cr(9)),'%g')
    imin=fix(max(xlim(1)-(xoff+x_offset*xscale),0)/(xscale*interval)+1);
    imax=fix(min((xlim(2)-(xoff+x_offset*xscale))/(xscale*interval)+1,npts));
    imax=max(imax,imin);
    x=((xoff+x_offset*xscale)+(imin-1)*xscale*interval):xscale*interval:...
    ((xoff+x_offset*xscale)+(imax-1)*xscale*interval);
    y=y(imin:imax)*yscale+yoff;
    %
    %DADISP double column
    %
elseif strcmp(s(1),' ')& strcmp(s(21),' ') & cr(1)==41
    s=[s,fscanf(fid,'%c',102400)];
    fclose(fid);
    m=sscanf(s,'%g%g',[2,length(cr)]);
    x=m(1,:)*xscale+xoff;
    [m,imin]=min(abs(x-xlim(1)));
    if (m==inf)
        imin=1;
    end
    [m,imax]=min(abs(x-xlim(2)));
```

```
    if (m==inf) imax=length(x); end;
    x=x(imin:imax);
    y=m(2,imin:imax)*yscale+yoff;
    header=str2mat(header,'(PPat 2 columns)');
    %
    %     LeCroy
    %
elseif strcmp(s(str2num(s(2))+3:str2num(s(2))+10),'WAVEDESC')
    blockoff=str2num(s(2))+3;
    filesize=str2num(s(3:blockoff-1));
    tfname=tempname;
    tfid=fopen(tfname,'w+');
    maxheadcol=60;
    l=length(fullname);
    shortname=fullname(max(1,l-maxheadcol+1):l);
    header=dumpvar(s,header,tfid,blockoff,'template',16,'string',10);
    header=dumpvar(s,header,tfid,blockoff,'comtype',32,'int*2');
    [nrh,nch]=size(header);
    eval([header(nrh,:),';']);
    stemp=['''byte''';'''word'''];
    comtyp=comtype;
    header(nrh,9:14)=stemp(comtype+1,:);
    header=dumpvar(s,header,tfid,blockoff,'comord',34,'int*2');
    [nrh,nch]=size(header);
    eval([header(nrh,:),';']);
    stemp=['''hifirst''';'''lofirst'''];
    header(nrh,8:16)=stemp(comord+1,:);
    header=dumpvar(s,header,tfid,blockoff,'ndesc',36,'long');
    header=dumpvar(s,header,tfid,blockoff,'nuser',40,'long');
    header=dumpvar(s,header,tfid,blockoff,'nresdesc',44,'long');
    header=dumpvar(s,header,tfid,blockoff,'ntrig',48,'long');
    header=dumpvar(s,header,tfid,blockoff,'nrise',52,'long');
    header=dumpvar(s,header,tfid,blockoff,'nres1',56,'long');
    header=dumpvar(s,header,tfid,blockoff,'nwave1',60,'long');
    header=dumpvar(s,header,tfid,blockoff,'nwave2',64,'long');
    header=dumpvar(s,header,tfid,blockoff,'nres2',68,'long');
    header=dumpvar(s,header,tfid,blockoff,'nres3',72,'long');
    header=dumpvar(s,header,tfid,blockoff,'instrname',76,'string',16);
    header=dumpvar(s,header,tfid,blockoff,'serialnum',92,'long');
    header=dumpvar(s,header,tfid,blockoff,'tracelab',96,'string',16);
    header=dumpvar(s,header,tfid,blockoff,'rw1',112,'int*2');
    header=dumpvar(s,header,tfid,blockoff,'rw2',114,'int*2');
    header=dumpvar(s,header,tfid,blockoff,'nwavepts',116,'long');
    header=dumpvar(s,header,tfid,blockoff,'pps',120,'long');
    header=dumpvar(s,header,tfid,blockoff,'firstp',124,'long');
    header=dumpvar(s,header,tfid,blockoff,'lastp',128,'long');
    header=dumpvar(s,header,tfid,blockoff,'firstpoff',132,'long');
    header=dumpvar(s,header,tfid,blockoff,'sparsefact',136,'long');
    header=dumpvar(s,header,tfid,blockoff,'segindex',140,'long');
    header=dumpvar(s,header,tfid,blockoff,'nsubarray',144,'long');
    header=dumpvar(s,header,tfid,blockoff,'swperacq',148,'long');
    header=dumpvar(s,header,tfid,blockoff,'obso1',152,'long');
    header=dumpvar(s,header,tfid,blockoff,'vgain',156,'float');
    header=dumpvar(s,header,tfid,blockoff,'voff',160,'float');
    header=dumpvar(s,header,tfid,blockoff,'maxval',164,'float');
    header=dumpvar(s,header,tfid,blockoff,'minval',168,'float');
    header=dumpvar(s,header,tfid,blockoff,'nombits',172,'int*2');
    header=dumpvar(s,header,tfid,blockoff,'nomsub',174,'int*2');
    header=dumpvar(s,header,tfid,blockoff,'hzint',176,'float');
    header=dumpvar(s,header,tfid,blockoff,'hzoff',180,'double');
    header=dumpvar(s,header,tfid,blockoff,'pixoff',188,'double');
    header=dumpvar(s,header,tfid,blockoff,'vunit',196,'string',48);
    header=dumpvar(s,header,tfid,blockoff,'hunit',244,'string',48);
```

```
header=dumpvar(s,header,tfid,blockoff,'rw3',292,'int*2');
header=dumpvar(s,header,tfid,blockoff,'rw4',294,'int*2');
header=dumpvar(s,header,tfid,blockoff,'trigtsec',296,'double');
header=dumpvar(s,header,tfid,blockoff,'trigtmin',304,'byte');
header=dumpvar(s,header,tfid,blockoff,'trigthr',305,'byte');
header=dumpvar(s,header,tfid,blockoff,'trigtday',306,'byte');
header=dumpvar(s,header,tfid,blockoff,'trigtmon',307,'byte');
header=dumpvar(s,header,tfid,blockoff,'trigtyr',308,'int*2');
header=dumpvar(s,header,tfid,blockoff,'acqdur',312,'float');
header=dumpvar(s,header,tfid,blockoff,'rectype',316,'int*2');
[nrh,nch]=size(header);
eval([header(nrh,:),';']);
stemp=str2mat('single sweep','interleave','histogram','trend',...
      'filter coef.','complex','extrema','sequence_obs');
header(nrh,9:22)=['''',stemp(rectype+1,:),''''];
header=dumpvar(s,header,tfid,blockoff,'procdone',318,'int*2');
[nrh,nch]=size(header);
eval([header(nrh,:),';']);
stemp=str2mat('none','fir filer','interpolated','sparsed','autoscaled',...
      'no result','rolling','cumulative');
header(nrh,10:23)=['''',stemp(procdone+1,:),''''];
header=dumpvar(s,header,tfid,blockoff,'rw5',320,'int*2');
header=dumpvar(s,header,tfid,blockoff,'risswps',322,'int*2');
header=dumpvar(s,header,tfid,blockoff,'timebase',324,'int*2');
[nrh,nch]=size(header);
eval([header(nrh,:),';']);
s125='125';
tmp3=fix(timebase/3);
tbase=['timebase=''',s125(timebase+1-tmp3*3),'e',num2str(tmp3-12),' sec/div'''];
header=str2mat(header(1:nrh-1,:),tbase);
header=dumpvar(s,header,tfid,blockoff,'vertcoup',326,'int*2');
[nrh,nch]=size(header);
eval([header(nrh,:),';']);
stemp=str2mat('DC 50 Ohm','ground','DC 1 MOhm','ground','AC 1 MOhm');
header(nrh,10:20)=['''',stemp(vertcoup+1,:),''''];
header=dumpvar(s,header,tfid,blockoff,'probatt',328,'float');
header=dumpvar(s,header,tfid,blockoff,'fvgain',332,'int*2');
[nrh,nch]=size(header);
eval([header(nrh,:),';']);
tmp3=fix(fvgain/3);
temps=['fvgain=''',s125(fvgain+1-tmp3*3),'e',num2str(tmp3-6),' V/div'''];
header=str2mat(header(1:nrh-1,:),temps);
header=dumpvar(s,header,tfid,blockoff,'bwlim',334,'int*2');
[nrh,nch]=size(header);
eval([header(nrh,:),';']);
stemp=str2mat('on','off');
header(nrh,7:11)=['''',stemp(bwlim+1,:),''''];
header=dumpvar(s,header,tfid,blockoff,'vertvern',336,'float');
header=dumpvar(s,header,tfid,blockoff,'avertoff',340,'float');
header=dumpvar(s,header,tfid,blockoff,'wavesrc',344,'int*2');
[nrh,nch]=size(header);
eval([header(nrh,:),';']);
stemp=str2mat('channel 1','channel 2','channel 3','channel 4',' unknown?');
header(nrh,9:19)=['''',stemp(wavesrc+1,:),''''];
fclose(tfid);
eval(['delete ',tfname]);
[nheadrow,nheadcol]=size(header);
for i=2:nheadrow
   eval([header(i,:),';']);
end
if hzint<mindelt
   compfact=fix(mindelt/hzint);
   hzintlim=hzint*compfact;
```

```
    hzofflim=hzint*round(compfact/2);
    if ~quiet disp(['    Data compressed by a factor of ',num2str(compfact)]); end;
    if ~quiet disp('    (See "mindelt" in "help showdefs".)'); end;
  else
    compfact=1;
    hzintlim=hzint;
    hzofflim=0;
  end
  if ~quiet disp(['    Retrieved data temporal resolution =',...
     num2str(hzintlim),' s/pt']); end;
  if ~quiet disp('    Scope settings:'); end;
  if ~quiet disp(['      ',tbase,' (10 div = full scale)']); end;
  vpdiv=vgain*(maxval-minval+1)/8;
  if ~quiet disp(['      vertical amplitude was:  ',...
       num2str(vpdiv),' V/div (full scale = ',...
       num2str(-vpdiv*4-avertoff),' to ',num2str(vpdiv*4-avertoff),' V )']); end;
  if ~quiet disp(['      vertical offset was:  ',num2str(avertoff),' V']); end;
  time=sprintf('%2d:%2d:%2.2g',trigthr,trigtmin,trigtsec);
  time=strrep(time,' ','0');
  date=sprintf('%2d/%2d/%2d',trigtmon,trigtday,mod(trigtyr,100));
  date=strrep(date,' ','0');
  lin=length(instrname);
  serial=num2str(serialnum,10);
  lsn=length(serial);
  lsn1=max(lsn,1);
  title=['s',serial(lsn1:lsn1),'ch',wavesrc(9),' ',time,' ',date];
  header=str2mat(shortname,title,header(2:nheadrow,:));
  nheadrow=nheadrow+2;
  if (nuser~=0)
    usertext=dumpstr(s,blockoff+ndesc,1,nuser);
    header=str2mat(header,wrap(usertext,30,max(nheadcol,maxheadcol),50));
    [nheadrow,nheadcol]=size(header);
  end
  imin=fix(max(xlim(1)-hzoff,0)/hzint)+1;
  imax=fix(min((xlim(2)-hzoff)/hzint+1,nwavepts));
  imax=max(imax,imin);
  nt=0;
  if fix(ntrig/16)>0
    fseek(fid,blockoff+ndesc+nuser+nresdesc-1,'bof');
    triggers=fread(fid,ntrig/8,'double');
    triggers=[triggers(1:2:ntrig/8)';triggers(2:2:ntrig/8)';1:ntrig/16]';
    alltrig=triggers;
    nt=size(triggers,1);
    nppt=nwavepts/nt;
    if ~quiet
      disp(['    Sequenced data:  ',num2str(nt),' events, ',num2str(nppt),'...
          pts/event, ', num2str(nwavepts),' pts, ']);
      disp(['                      extends from ',...
          num2str(triggers(1,1)+triggers(1,2)),' to ',...
          num2str(triggers(nt,1)+triggers(nt,2)+nppt*hzint),' seconds.']);
disp(['                    Total acquire time was ',num2str(acqdur),' seconds.']);
    end;
    t1=fix((imin-2+nppt)/nppt+1);
    t2=fix(imax/nppt);
    if length(svin==1) & svin<t1 svin=t1; end;
    t2=min(t2,max(svin));
    t1=max(1,min(max(t1,min(svin)),t2));
    if (t2>=t1)
      sv=setdiff(svin,find(t1>svin|t2<svin));
      if envel
        %  ignore x-values for envelope data processing
        x=[];
      else
```

```
            if (length(sv)*nppt>1.2e6)
               disp('This takes too long.  Number of points limited to 1.2e6');
               sv=sv(1:fix(1.2e6/nppt)+1);       %    return;
            end
            x=[];
            for t=sv
               x=[x,triggers(t,1)+triggers(t,2)+(round(compfact/2)...
                     :compfact:(nppt-1+round(compfact/2)))*hzint];
            end
            imin=nppt*(t1-1)+1;
            imax=nppt*t2;
         end
         triggers=triggers(sv,:);
      else
         if (imax-imin>1.2e6)
            disp('This takes too long. Number of points limited to 1.2e6');
            imax=imin+1.2e6;
         end
         t=fix((imin-1)/nppt)+1;
         x0=triggers(t,1)+triggers(t,2)+(imin-1-nppt*(t-1))*hzint;
         x=(x0+hzofflim):hzintlim:(x0+hzofflim+(imax-imin)*hzint);
         triggers=triggers(t,:);
         t1=t;t2=t;sv=t1:t2;
         nppt=length(x);
      end
   else
      x=(hzoff+(imin-1)*hzint+hzofflim):hzintlim:(hzoff+(imax-1)*hzint-hzofflim);
   end
   dataoff=blockoff+ndesc+nuser+nresdesc+ntrig+nrise+nres1-1;
   prec=['schar';'short'];
   iprec=comtyp+1;
   y=[];
   if (imax>=imin)
      if (nt<=0)
         if (imax-imin>1.2e6)
     disp('This will take too long to read.  The array is limited to 1.2e6 points.');
            imax=imin+1.2e6;
%           return;
         end
         fseek(fid,dataoff+(imin-1)*(comtyp+1),'bof');
         y=fread(fid,imax-imin+1,prec(iprec,:));
         if compfact>1
            ytemp=y(1:compfact:imax-imin+1+1-compfact);
            for icf=2:compfact
               ytemp=ytemp+y(icf:compfact:imax-imin+1+icf-compfact);
            end
            y=ytemp/compfact;
         end
      else
         if envel
%save envelope data
            par.fn=fn; par.npts=nwavepts; par.nppe=nppt;
            if ncpe==0 ncpe=nppt; end;
            nppc=nppt/ncpe;
            par.nevs=nt; par.evmin=min(sv); par.evmax=max(sv);
            nppc2=nppc; ncpe2=ncpe; nsv=size(sv,2); x=[]; y=[]; iev=[1];
            sumy=0; sumy2=0; navg=0;
            for it=1:nsv
               t=sv(it);
               xtemp=alltrig(t,1)+alltrig(t,2)+(0:nppt-1)*hzint;
               i1=(alltrig(t,3)-1)*nppt*iprec; fseek(fid,dataoff+i1,'bof');
               ytemp=fread(fid,nppt,prec(iprec,:))'*vgain*yscale-voff*yscale+yoff;
               [xtemp2,ytemp2,nppc2,envoff]=envelop1(xtemp,ytemp,t,it,nppt,...
```

```
                fix(.5+(nppc+3*nppc2)/4),fix(.76+(ncpe+3*ncpe2)/4),5);
            x=[x,xtemp2]; y=[y,ytemp2]; iev=[iev,length(xtemp2)];
            neppc=size(envoff,2); ncpe3=length(xtemp2)/neppc;
            if(ncpe3~=0) ncpe2=ncpe3; end
            sumy=sumy+sum(ytemp); sumy2=sumy2+sum(ytemp.*ytemp);
         end
         navg=nsv*nppt; par.nppc=nppc2; iev=cumsum(iev);
         par.yavg=sumy/navg; par.yrms=sqrt(sumy2/navg); par.navg=navg;
         par.envoff=envoff;
      else
         for t=sv
            i1=(alltrig(t,3)-1)*nppt*iprec; fseek(fid,dataoff+i1,'bof');
            ytemp2=fread(fid,nppt,prec(iprec,:))';
            if compfact>1
               ytemp=ytemp2(1:compfact:nppt+1-compfact);
               for icf=2:compfact
                  ytemp=ytemp+ytemp2(icf:compfact:nppt+icf-compfact);
               end
               y=[y ytemp/compfact];
            else
               y=[y ytemp2];
            end
         end
      end
   end
   if ~isempty(y)&~envel y=y*vgain*yscale-voff*yscale+yoff; end
else
   y=0;
end
x=x*xscale+xoff;
if exist('triggers','var')
   triggers(:,1:2)=triggers(:,1:2)*xscale+xoff;
   alltrig(:,1:2)=alltrig(:,1:2)*xscale+xoff;
end
y=y';
fclose(fid);
else
   fclose(fid);
   %
   %invalid file type
   %
   disp(['***** The file, ',fullname])
   disp('***** was not recognized as valid data file type. *****')
   disp(' ')
   disp('Valid file types are: TEK SCD, HP 54111, TEK TDS, LeCroy 9384L,  spectra,')
   disp('DADisp column, and PPat 2 column.')
   disp('     Files from TEK TDS must be converted to ASCII (.ASC) with a')
   disp('routine called CNVRTWFM.')
end
eltime=toc;
if eltime>5 disp(['Elapsed time = ',num2str(eltime),' seconds.']); end
```

PLOTDATA – plot data acquired with getdata for higher level plotcurv and plotshif

```
%hans=plotdata(x,y,h,[xmin,xmax],[xs,xo,ys,yo],triggers,sv,offlab,trigpar)
%                fjz 12/14/00
%
%    plots y vs. x.  Called by PCSS plot routines "plotcurv", "addcurve",
%  "plotlist", "plotshif", "matxplot", ...  These use "getdata" to input
%  and scale x and y and "plotdata" to create plots with the parameters
%  shown in "showdefs".
%
%           x,y = arrays of x and y coordinates to be plotted.
%             h = an optional matrix of text from the data file.
%    [xmin,xmax] = an optional range for the scaled x values in the plot.
% [xs,xo,ys,yo] = an optional array of scale factors.
%                 If included, they scale the plotted values X and Y:
%                                    X=x*xs+xo
%                                    Y=y*ys+yo
%                 (This scaling is applied after the default PCSS scaling
%                 and before integration.  Use global variable "inteshift"
%                 for a y-offset after integration.  See showdefs.)
%      triggers = a 3 column array (table) of trigger times, offsets,
%                   and event # for sequenced data
%            sv = a list of events for sequenced data (LeCroy).
%        offlab = start times are multiplied by "offlabel"
%                   will be printed to the left of each curve
%       trigpar = the parameters to synchronize the curves at a new trigger
%                   point following the start of each curve (event).
%                   "trig" is a structured variable with fields for arguments of
%     "digtrig":  trig.level, trig.slope, trig.slim, trig.holdoff,
%                   trig.xres, trig.minper, trig.prefilter
%          hans   = the handles for the plotted curves
%
%         colors      &      line styles
%        r     red            -     solid
%        g     green          :     dotted
%        b     blue           -.    dashdot
%        c     cyan           --    dashed
%        m     magenta        .     point
%        y     yellow         o     circle
%        k     black          x     x-mark
%                                +     plus
%                             *     star
%  colors='rgbcmyk' is a global variable defined by pcssdefs.
%
%hans=plotdata(x,y,h,[xmin,xmax],[xs,xo,ys,yo],triggers,sv,offlab,trigpar)
function p=plotdata(x,y,h,xlim,f,triggers,sv,offlab,tp)
global ptitle xlab ylab integrate curvecolor curvestyle curvecount inteshift colors;
if ~exist('x','var') | notarg(x,0) | ~exist('y','var') | notarg(y,0)
   disp('x & y values must be passed to "plotdata".  Exiting ...'); p=[]; return; end;
if ~exist('xlim','var')      | notarg(xlim,0)   xlim=[]; end;
if ~exist('f','var')         | notarg(f,0)      f=[]; end;
if ~exist('triggers','var') | notarg(triggers) triggers=[]; end;
if ~exist('sv','var')        | notarg(sv)       sv=[]; end;
if ~exist('offl','var')      | notarg(offl)     offl=0; end;
if ~exist('tp','var')        | notarg(tp)       tp=[]; end;
%colors='mcrgbywk' set with PCSSDEFS
ncolr=length(colors);
styles=['- ';': ';'-.';'--';'. ';'o ';'x ';'+ ';'* '];
if length(curvecolor)==0
   colr=colors(rem(curvecount-1,ncolr)+1);
else
   colr=curvecolor;
end
```

```
if length(curvestyle)==0
   styl=styles(rem(curvecount-1,4)+1,:);
elseif strcmp(upper(curvestyle(1:1)),'P')
   styl=styles(rem(curvecount-1,5)+5,:);
elseif strcmp(upper(curvestyle(1:1)),'L')
   styl=styles(rem(curvecount-1,4)+1,:);
else
   styl=curvestyle;
end
n=length(x);
imin=1; imax=n;
if size(f,2)==4
   xs=f(1,1); xo2=f(1,2); ys=f(1,3); yo=f(1,4);
else
   xs=1; xo2=0; ys=1; yo=0;
end
if (length(xlim)==2)
   xmin=(xlim(1)-xo)/xs;   xmax=(xlim(2)-xo)/xs;
   for i=1:n
      if (x(i)< xmin) imin=i+1; end;
      if (x(i)<=xmax) imax=i;   end;
   end
   if (imax<=imin) imin=1; imax=n; end;
end
iminsav=imin; imaxsav=imax;
seqnote=''; seqoff='';
if ~isempty(triggers) & ~isempty(sv)
nt=size(triggers,1);
nppt=length(x)/nt;
   nev=length(sv);
   s=[];
   for i=1:nev
      s=[s,find(triggers(:,3)==sv(i))];
   end
if (length(s)==0)
                     if min(sv(:))>max(triggers(:,3)) s=nt; end
                     if max(sv(:))<min(triggers(:,3)) s=1; end
   end
   nev=min(nev,length(s));
imin=max(imin,(s(1)-1)*nppt+1);
imax=min(imax,s(1)*nppt);
   if ~isempty(tp)
      [itr,xtr]=digtrig(x(imin:imax),y(imin:imax),tp);
      if ~isempty(xtr) xo=-xs*xtr(1)+xo2;
      else xo=-xs*x(imin)+xo2;
      end;
   else
      xo=-xs*x(imin)+xo2;  %-triggers(s(1),2)
   end;
      if nev==1
                     seqnote=['#',num2str(triggers(s(1),3))];
   elseif nev<4
                     seqnote=['#',sprintf('%i ',triggers(s(1:nev),3))];
elseif (abs(triggers(s(nev),3)-triggers(s(1),3))+1)/nev==1
                     seqnote=['#',num2str(triggers(s(1),3)),' to
',num2str(triggers(s(nev),3))];
elseif abs(triggers(s(nev),3)-triggers(s(1),3))/(nev-1)==...
         abs(triggers(s(2),3)-triggers(s(1),3))
seqnote=['#',num2str(triggers(s(1),3)),' to ',num2str(triggers(s(nev),3)),...
            ' by ',num2str(triggers(s(2),3)-triggers(s(1),3))];
  elseif nev<11
                     seqnote=['#',sprintf('%i ',triggers(s(1:nev),3))];
           else
```

```
                    seqnote=['#',num2str(triggers(s(1),3)),' to ',...
            num2str(triggers(s(nev),3)),' by ?'];
   end
   seqoff=[' at ',num2str(-xo,7),' (ev ',num2str(triggers(s(1),3)),')'];
   disp([seqnote,seqoff]);
   if (size(s,2)>1)
      if ((abs(xo)<abs(triggers(s(2),2)-triggers(s(1),2))/10000) | offlab)
         seqoff=[]; end
   end
else
   xo=xo2;
end
if integrate==0
    intnote='';
   p=plot(xs*x(imin:imax)+xo,ys*y(imin:imax)+yo,[colr,styl]);
elseif integrate==1
intnote='(integ.)';
dx=xs*(x(imax)-x(imin))/(imax-imin);
p=plot(xs*x(imin:imax)+xo,...
      dx*cumsum(ys*y(imin:imax)+yo)+(curvecount-1)*inteshift,[colr,styl]);
else
 intnote='(diff.)';
dx=xs*(x(imax)-x(imin))/(imax-imin);
p=plot(xs*x(imin:imax-1)+xo+dx/2,diff(ys*y(imin:imax)+yo)/dx,[colr,styl]);
end;
if offlab
   axlims=axis;
   xspace=(axlims(2)-axlims(1))/60; %yspace=(axlims(4)-axlims(3))*length(s)/80;
   xofflab=xs*x(imin)+xo-xspace/5; yofflab=yo;
   text(xofflab,yofflab,sprintf('%5.3f',-xo*offlab),'HorizontalAlignment',...
      'right','color',colr);
end
pt='    ';
if length(ptitle)>0
pt=[ptitle,' ',' ',' '];
end
if length(ptitle)>0&~strcmp(upper(pt(1:4)),'NONE')&~strcmp(upper(pt(1:4)),'FILE')
   title([ptitle,deblank([' ',intnote]),deblank([' ',seqnote])]);
   if ~strcmp(h(1,1:7),'AddData')
      if size(h,1)>1 cornlab(h(1,:),h(2,:)); end;
   end
elseif size(h,1)>0
   if ~strcmp(h(1,1:7),'AddData')
      bs=findstr(h(1,:),'\');
      if (length(bs)>0)
         ih1=bs(length(bs))+1;
      else
         ih1=1;
      end
      ih2=length(deblank(h(1,:)));
      if strcmp(upper(pt(1:4)),'FILE')
         title([intnote,deblank([' ',seqnote]),deblank([' ',h(1,ih1:ih2)])]);
      else
         if ~strcmp(upper(pt(1:4)),'NONE')
            ttl=[intnote,deblank([' ',seqnote]),deblank([' ',h(1,ih1:ih2)]),...
                  deblank([' ',h(2,:)])];
            nttl=min(47,length(ttl));
            title(ttl(1:nttl));
         end
      end
      if size(h,1)>1 cornlab(h(1,:),h(2,:)); end;
   end
end
```

```
bh=0; spect=0; addd=0;
if ~isempty(h)
  if strcmp(h(2,1:7),'(BH loo')
          bh=-1;
  end
  if strcmp(h(2,1:7),'(spectr')
       spect=-1;
  end
  if strcmp(h(1,1:7),'AddData')
        addd=-1;
  end
end
if addd==0
  if length(xlab)>0
    if length(xlab)==4
if ~strcmp(upper(xlab),'NONE')
  xlabel([xlab,seqoff]);
             end
    else
xlabel([xlab,seqoff]);
    end
  else
    if spect==0 & bh==0
    xlabel(['Time (s)',seqoff]);
   elseif spect
       xlabel('Wavelength (nm)');
     elseif bh
       xlabel('Applied Field (Oe)');
           end
  end
  if length(ylab)>0
    if length(ylab)==4
if ~strcmp(upper(ylab),'NONE')
  ylabel(ylab);
             end
    else
  ylabel(ylab);
    end
  else
    if spect==0 & bh==0
    ylabel('Voltage (V)');
   elseif spect
       ylabel('Intensity')
     elseif bh
       ylabel('Magnetic Field (kG)');
           end
  end
end
if ~isempty(sv) & ~isempty(triggers)
   hold on;
   rf=size(f,1);
   for i=2:length(s)
      xo2=0;
      if size(f,2)==4 & rf>=i
         xs=f(i,1); xo2=f(i,2); ys=f(i,3); yo=f(i,4);
      end
      imin=max(iminsav,(s(i)-1)*nppt+1);
      imax=min(imaxsav,s(i)*nppt);
   if ~isempty(tp)
      [itr,xtr]=digtrig(x(imin:imax),y(imin:imax),tp);
      if ~isempty(xtr) xo=-xs*xtr(1)+xo2;
      else xo=-xs*x(imin)+xo2;
      end;
```

```
    else
       xo=-xs*x(imin)+xo2;  %-triggers(s(1),2)
    end;
       disp([' seq',num2str(i),'=',num2str(triggers(s(i),3)),...
             ' t0(',num2str(s(i)),')=',num2str(-xo)]);
       curvecount=curvecount+1;
       if length(curvecolor)==0
          colr=colors(rem(curvecount-1,ncolr)+1);
       else
          colr=curvecolor;
       end
       if length(curvestyle)==0
          styl=styles(rem(curvecount-1,4)+1,:);
       elseif strcmp(upper(curvestyle(1:1)),'P')
          styl=styles(rem(curvecount-1,5)+5,:);
       elseif strcmp(upper(curvestyle(1:1)),'L')
          styl=styles(rem(curvecount-1,4)+1,:);
       else
          styl=curvestyle;
       end

       if integrate==0
          p=plot(xs*x(imin:imax)+xo,ys*y(imin:imax)+yo,[colr,styl]);
       elseif integrate==1
          dx=xs*(x(imax)-x(imin))/(imax-imin);
          p=plot(xs*x(imin:imax)+xo,...
             dx*cumsum(ys*y(imin:imax)+yo)+(curvecount-1)*inteshift,[colr,styl]);
       else
          dx=xs*(x(imax)-x(imin))/(imax-imin);
          p=plot(xs*x(imin:imax-1)+xo+dx/2,diff(ys*y(imin:imax)+yo)/dx,[colr,styl]);
       end
       if offlab
          xofflab=xs*x(imin)+xo-xspace/5;
          yofflab=yo;
          if integrate yofflab=(curvecount-1)*inteshift; end
          text(xofflab,yofflab,sprintf('%5.3f',-xo*offlab),...
             'HorizontalAlignment','right','color',colr);
          if (i==length(s))
             if offlab==1
                startlab='start';
             else
                startlab=['start (x ',num2str(1/offlab),')'];
             end
             text(0.01,1.001,startlab,'HorizontalAlignment','left',...
                'VerticalAlignment','bottom','units','normalized',...
                'color','k','FontSize',10);
          end
       end
    end
    hold off
end
axisloose;
figure(gcf);
```

ADDDATA – uses plotdata to add addition curves to an existing plot

```
%han=adddata(x,y,[xmin,xmax],[xs,xo,ys,yo],triggers,sv,offlab,trigpar)
%              fjz 12/14/00
%
%adds a curve to an existing plot. Similar to "plotdata" it is
%  called by PCSS plot routines "plotcurv", "addcurve",
%  "plotlist", "plotshif", "matxplot", ...  These use getdata to input
%  and scale x and y and "plotdata" to create plots with the parameters
%  shown in "showdefs".
%
%           x,y = arrays of x and y  oordinates to be plotted.
%             h = an optional matrix of text from the data file.
%    [xmin,xmax] = an optional range for the scaled x values in the plot.
% [xs,xo,ys,yo] = an optional array of scale factors.
%                 If included, they scale the plotted values X and Y:
%                                    X=x*xs+xo
%                                    Y=y*ys+yo
%                (This scaling is applied after the default PCSS scaling
%                and before integration.  Use global variable "inteshift"
%                for a y-offset after integration.  See showdefs.)
%      triggers = a 3 column array (table) of trigger times, offsets,
%                 and event # for sequenced data
%            sv = a list of events for sequenced data (LeCroy).
%        offlab = start times are multiplied by "offlabel"
%                 will be printed to the left of each curve
%       trigpar = the parameters to synchronize the curves at a new trigger
%                 point following the start of each curve (event).
%                 "trig" is a structured variable with fields for arguments of
%     "digtrig":  trig.level, trig.slope, trig.slim, trig.holdoff,
%                 trig.xres, trig.minper, trig.prefilter
%         hans  = the handles for the plotted curves
%
%han=adddata(x,y,[xmin,xmax],[xs,xo,ys,yo],triggers,sv,offlab,trigpar)
function p=adddata(x,y,xlim,factors,triggers,seq,offl,tp)
global ptitle xlab ylab xscale yscale xoff yoff curvecolor curvestyle curvecount
if ~exist('x','var') | notarg(x,0) | ~exist('y','var') | notarg(y,0)
   disp('x & y values must be passed to "adddata".  Exiting ...'); p=[]; return; end;
if ~exist('xlim','var')      | notarg(xlim,0)   xlim=[]; end;
if ~exist('factors','var')   | notarg(factors)  factors=[]; end;
if ~exist('triggers','var') | notarg(triggers) triggers=[]; end;
if ~exist('sv','var')        | notarg(sv)       sv=[]; end;
if ~exist('offl','var')      | notarg(offl)     offl=0; end;
if ~exist('tp','var')        | notarg(tp)       tp=[]; end;
curvecount=curvecount+1;
hold on;
p=plotdata(x,y,str2mat('AddData',' '),xlim,factors,triggers,seq,offl,tp);
hold off;
```

PLOTCURV – uses getdata and plotdata to build high level plots from data files

```
%han=plotcurv(file,[xmin,xmax],[xs,xo,ys,yo],sv,offlab,trigpar)    fjz 12/14/00
%
%  plots the selected data file as a curve in a new plot. Calls PCSS
%  plotting routines to input, scale, and plot the data.
%  Default plot settings (scales, offsets, labels, color, and style)
%  are saved in global variables.  See:  showdefs, savedefs, loaddefs
%  Related functions:  addcurv, getdata, plotdata, plotlist, matxplot
%
%         file = an optional file name.  If empty or omitted, a file menu is opened.
%                Valid file types are: TEK SCD, HP 54111, TEK TDS, LeCroy 6894,
%                spectra, DADisp column, PPat 2 column, GENerated data.  Files from
%                TEK TDS must be converted to ASCII (.ASC) with CNVRTWFM.
%  [xmin,xmax] = an optional range for the scaled x values of the curve.
%[xs,xo,ys,yo] = an optional array of scale factors.
%                One row of factors can be included for each event for sequenced data.
%                If they are included, the plotted values X and Y will be:
%                                    X=x*xs+xo
%                                    Y=y*ys+yo
%                (This scaling is applied after the default PCSS scaling
%                and before integration.  Use global variable "inteshift"
%                for a y-offset after integration.  See showdefs.)
%           sv = an optional list of events for sequenced data (LeCroy).
%                If omitted, only the first event will be processed.
%       offlab = time offset multiplier.  If non-zero, start times are printed
%                to the left of each curve.
%      trigpar = the parameters to synchronize the curves at a new trigger
%                point following the start of each curve (event).
%                "trig" is a structured variable with fields for arguments of
%                "digtrig":  trig.level, trig.slope, trig.slim, trig.holdoff,
%                            trig.xres, trig.minper, trig.prefilter
%          han = the handle for the plot created by plotcurv (0 if no plot).
%
%han=plotcurv(file,[xmin,xmax],[xs,xo,ys,yo],sv,offlab,trigpar)    fjz 12/14/00
function plothan=plotcurv(file,xlim,factors,sv,offl,tp)
global curvecount
if ~exist('file','var')     | notarg(file,0) file=''; end;
if ~exist('xlim','var')     | notarg(xlim,0) xlim=[]; end;
if ~exist('factors','var') | notarg(factors)   factors=[]; end;
if ~exist('sv','var')       | notarg(sv)     sv=[]; end;
if ~exist('offl','var')     | notarg(offl)   offl=0; end;
if ~exist('tp','var')       | notarg(tp)     tp=[]; end;
curvecount=1;
plothan=0;
[x,y,l,triggers]=getdata(file,xlim,sv);
if isempty(x) disp('"getdata.m" did not find a file.'); plothan=[]; return; end;
plothan=plotdata(x,y,l,0,factors,triggers,sv,offl,tp);
```

ADDCURVE – add files (curves) to an existing plot initiated with plotcurv

```
%han=addcurve(file,[xmin,xmax],[xs,xo,ys,yo],sv,offlab,trigpar)    fz 12/14/00
%
%  adds the selected data file as a curve to an existing plot. Calls
%  PCSS plotting routines to input, scale, and plot the data.
%  Default plot settings (scales, offsets, labels, color, and style)
%  are saved in global variables.  See:  showdefs, savedefs, loaddefs
%  Related functions:  addcurv, getdata, plotdata, plotlist, matxplot
%
%         file = an optional file name.  If empty or omitted, a file menu is opened.
%                Valid file types are: TEK SCD, HP 54111, TEK TDS, LeCroy 6894,
%                spectra, DADisp column, PPat 2 column, GENerated data.  Files from
%                TEK TDS must be converted to ASCII (.ASC) with CNVRTWFM.
%  [xmin,xmax] = an optional range for the scaled x values of the curve.
%[xs,xo,ys,yo] = an optional array of scale factors.
%                One row of factors can be included for each event for sequenced data.
%                If they are included, the plotted values X and Y will be:
%                                    X=x*xs+xo
%                                    Y=y*ys+yo
%              (This scaling is applied after the default PCSS scaling
%              and before integration.  Use global variable "inteshift"
%              for a y-offset after integration.  See showdefs.)
%         "sv" = an optional list of events for sequenced data (LeCroy).
%                If omitted, only the first event will be processed.
%       offlab = time offset multiplier.  If non-zero, start times are printed
%                to the left of each curve.
%      trigpar = the parameters to synchronize the curves at a new trigger
%                point following the start of each curve (event).
%                "trig" is a structured variable with fields for arguments of
%                "digtrig":  trig.level, trig.slope, trig.slim, trig.holdoff,
%                            trig.xres, trig.minper, trig.prefilter
%          han = the handle for the plot created by plotcurv (0 if no plot).
%
%han=addcurve(file,[xmin,xmax],[xs,xo,ys,yo],sv,offlab,trigpar)    fz 12/14/00
function phan=addcurve(file,xlim,factors,seq,offl,tp)
global ptitle xlab ylab xscale yscale xoff yoff;
if ~exist('file','var')     | notarg(file,0)   file=''; end;
if ~exist('xlim','var')     | notarg(xlim,0)   xlim=[]; end;
if ~exist('factors','var') | notarg(factors) factors=[]; end;
if ~exist('seq','var')      | notarg(seq)      seq=[]; end;
if ~exist('offl','var')     | notarg(offl)     offl=0; end;
if ~exist('tp','var')       | notarg(tp)       tp=[]; end;
if iscell(file) file=strvcat(file); end;
[x,y,l,triggers]=getdata(file,xlim,seq);
                 phan=adddata(x,y,xlim,factors,triggers,seq,offl,tp);
```

PLOTLIST – plot multiple files (curves) using plotcurv and addcurve

```
%hans=plotlist(fns,[xmin,xmax],[xs,xo,ys,yo],sm,offlab,trigpar);    fjz 12/14/00
%
%  plots a list of curves on one graph.  Calls "plotcurv".
%
%           fns = a list or matrix which contains the file name list.
%                 See: lists, getlist, makelist, fixlist, dirlist, makefns
%                 to generate the list of file names
%                 or enter:  plotlist({'name1','name2','name3'},[xmin,xmax],...)
%                 where name1, ... are data filenames of arbitrary length.
%    [xmin,xmax] = optional limiting range of x values.
%                 If [xmin,xmax] is empty or 0, it is ignored.
% [xs,xo,ys,yo] = an optional array of scale factors and offsets for each
%                 curve.  There is one row for each file or event (curve).
%                 For each curve, i, the points (X,Y) are defined by:
%                                 X = x*xs(i)+xo(i)
%                                 Y = y*ys(i)+yo(i)
%               (This scaling is applied after the default PCSS scaling
%               and before integration.  Use global variable "inteshift"
%               for a y-offset after integration.  See showdefs.)
%                 Scale factors can be defined as follows:
%                    [ones(10,1),zeros(10,1),ones(10,1),[0:9]']
%               The transpose symbol (') converts rows to columns.
%               "ones" or "zeros" define a column (10x1) of ones or zeros.
%                0:9 defines a row (0,1,2,...9) for vertical offsets.
%            sm = an optional "sequence matrix" or table of events for
%                 "sequenced" data (LeCroy).  Each row corresponds to the
%                 events to be processed for each filename in the list, fn.
%                 If sm is omitted, only the first event will be processed.
%        offlab = time offset multiplier.  If non-zero, start times are printed
%                 to the left of each curve.
%       trigpar = the parameters to synchronize the curves at a new trigger
%                 point following the start of each curve (event).
%                 "trig" is a structured variable with fields for arguments of
%                 "digtrig":  trig.level, trig.slope, trig.slim, trig.holdoff,
%                             trig.xres, trig.minper, trig.prefilter
%          hans = the graphics handles for all the curves in the plot.
%
%hans=plotlist(fns,[xmin,xmax],[xs,xo,ys,yo],sm,offlab,trigpar);    fjz 12/14/00
function plothan=plotlist(mfn,xlim,fact,sv,offl,tp)
if ~exist('mfn','var')   | notarg(mfn,0)  mfn=''; end;
if ~exist('xlim','var')  | notarg(xlim,0) xlim=[]; end;
if ~exist('fact','var')  | notarg(fact)   fact=[]; end;
if ~exist('sv','var')    | notarg(sv)     sv=[]; end;
if ~exist('offl','var')  | notarg(offl)   offl=0; end;
if ~exist('tp','var')    | notarg(tp)     tp=[]; end;
if iscell(mfn) mfn=strvcat(mfn); end;
n=size(mfn,1);
[rf,cf]=size(fact);
[rs,cs]=size(sv);
if rf>=cs & cs>0
   plothan=plotcurv(mfn(1,:),xlim,fact(1:cs,:),sv(1,:),offl,tp);
   nfused=cs;
elseif rf>0
   plothan=plotcurv(mfn(1,:),xlim,fact(1,:),sv,offl,tp);
   nfused=1; cs=1;
else
   plothan=plotcurv(mfn(1,:),xlim,fact,sv,offl,tp);
   nfused=1; cs=1;
end
% add sucessive filenames to the plot
if isempty(mfn) return; end;
```

```
i=2;
while(i<=n)
   addhan=0;
   if rs>=i seq=sv(i,:); elseif rs>0  seq=sv(rs,:); else seq=''; end
   if (rf>=nfused+cs)
      addhan=addcurve(mfn(i,:),xlim,fact(nfused+1:nfused+cs,:),seq,offl,tp);
      nfused=nfused+cs;
   elseif (rf>nfused)
      addhan=addcurve(mfn(i,:),xlim,fact(nfused+1,:),seq,offl,tp);
      nfused=nfused+1;
   else
      addhan=addcurve(mfn(i,:),xlim,fact,sv,offl,tp);
   end
   plothan=[plothan addhan];i=i+1;
                                          end
```

PLOTSHIF – plots multiple files (curves) shifted vertically for comparison

```
%hans=plotshif(flist,shift,sm,offlab,trigpar);    fjz   12/14/00
%
%  plots a list of curves on one graph where each curve is shifted
%  by a constant vector from the previous one.  Calls "plotlist".
%
%   flist = a list or matrix (multiple rows) of data filenames.
%           See: lists, getlist, makelist, fixlist, dirlist, makefn
%                to generate the list off filenames or
%           enter:  plotshif({'name1','name2','name3'},shift,sm)
%                   where name1, ... are filenames of arbitray lengths.
%   shift = a y offset or an [x,y] offset vector.
%           If integrating, "inteshift" is set to "shift".  See showdefs.
%      sm = an optional "sequence matrix" or table of events for
%           "sequenced" data (LeCroy).  Each row corresponds to the
%           events to be processed for each filename in the list, fn.
%           If sm is omitted, only the first event will be processed.
%  offlab = time offset multiplier.  If non-zero, start times are printed
%           to the left of each curve.
% trigpar = the parameters to synchronize the curves at a new trigger
%           point following the start of each curve (event).
%           "trig" is a structured variable with fields for arguments of
%           "digtrig":  trig.level, trig.slope, trig.slim, trig.holdoff,
%                       trig.xres, trig.minper, trig.prefilter
%    hans = the graphics handles for all the curves in the plot.
%
%hans=plotshif(flist,shift,sm,offlab,trigpar);    fjz   12/14/00
function han=plotshif(fn,sh,sv,offl,tp)
global integrate inteshift
if ~exist('fn','var')     | notarg(fn,0)    fn=''; end;
if ~exist('sh','var')     | notarg(sh,0)    sh=[0,0]; end;
if ~exist('sv','var')     | notarg(sv,0)    sv=[]; end;
if ~exist('offl','var')   | notarg(offl,0)  offl=[]; end;
if ~exist('tp','var')     | notarg(tp,0)    tp=[]; end;
if iscell(fn) fn=strvcat(fn{:}); end % convert cell list to a string matrix
if length(sh)==1 sh=[0 sh]; end;
if integrate==1 inteshift=sh(2); sh(2)=0;      end;
if ~isempty(sv)
   nf=size(sv,1)*size(sv,2);
   sf=[ones(nf,1) [0:nf-1]'*sh(1) ones(nf,1) [0:nf-1]'*sh(2)];
else
   if isempty(fn) disp('Plotshif must have a list of filenames.  Exiting ...');
      han=[]; return; end;
   [nf,cf]=size(fn);
  sf=[ones(nf,1) [0:nf-1]'*sh(1) ones(nf,1) [0:nf-1]'*sh(2)];
end
hans=plotlist(fn,0,sf,sv,offl,tp);
```

(This page intentionally left blank)

Dr. John Aurand
ITT Industries
6400 Uptown Blvd., NE Suite 300E
Albuquerque, NM 87110

Dr. Peter Barnes
Auburn University
206 Allison Laboratory
Department of Physics
Auburn, AL 36849-3501

Dr. Collins Clark
Mission Research Corp.
1720 Randolph Rd., SE
Albuquerque, NM 87106

Dr. Tim Dallas
Department of Electrical Engineering
MS3102
Texas Tech University
Lubbock, TX 79409

Dr. James C. Dickens
Department of Electrical Engineering
MS 3102
Texas Tech University
Lubbock, TX 79409

Dr. Roger Dougal
University of South Carolina
500 S. Main St.
Columbia, SC 29208

Dr. William Donaldson
Lab for Laser Energetics
Univ of Rochester
250 East River Rd
Rochester, NY 14623-1299

Dr. Robert Druce
Lawrence Livermore Nat. Laboratory
P. O. Box 808, L-281
Livermore, CA 94550

Dr. R. Aaron Falk
Optometrix
4134 Lind Ave SW
Renton, WA 98055

Dr. Carl Franck
Dept. of Physics, Clark Hall
Cornell University
Ithaca, NY 14853

Dr. Charles Frost
Pulse Power Physics Inc.
1039 Red Oaks Loop, NE
Albuquerque, NM 87122

Dr. Shubhra Gangopadhyay
Department of Physics MS1051
Texas Tech University
Lubbock, TX 79409

Dr. John Gaudet
AFRL/DEH
Kirtland AFB
87117

Dr. Bernard Gerstman
Dept of Physics
Florida Intl University
University Pk
Miami, FL 33199

Dr. Martin Gundersen
University of Southern California
Rm 420, Seaver Science Center
Los Angeles, CA 90089-0484

Dr. Lynn Hatfield
Department of Physics MS1051
Texas Tech University
Lubbock, TX 79409

Dr. Tony F. Heinz
Dept of Physics
Columbia University
550 W 120th St
New York NY 10027

Dr. Wayne Hofer
Lawrence Livermore Nat. Laboratory
P. O. Box 808, L-169
Livermore, CA 94550-0808

Dr. Mark Holtz
Department of Physics MS1051
Texas Tech University
Lubbock, TX 79409

Dr. Erich Kunhardt
Stevens Institute of Technology
Department of Physics
807 Castle Point Terrace
Hoboken, NJ 07030

Dr. Chi H. Lee
University of Maryland
Dept. EE, A.V. Williams, Bldg 115
College Park, MD 20742

Dr. David C. Stoudt
Code B20
NSWC, Dahlgren Division
17320 Dahlgren Road
Dahlgren, Virginia 22448

Dr. Roger Lichti
Department of Physics MS1051
Texas Tech University
Lubbock, TX 79409

Dr. Henryk Temkin
Department of Electrical Engineering
MS3102
Texas Tech University
Lubbock, TX 79409

Dr. Mike Mazzola
Dept. Electrical and Computer Eng
P.O. Box 9571
MS State
MS 39762

Dr. Douglas R. Wake
3816 49th Ave NE
Seattle, WA
98105

Dr. Charles Myles
Department of Physics MS1051
Texas Tech University
Lubbock, TX 79409

Dr. D. H. Auston
Case Western Reserve University
Office of the President
10900 Euclid Ave
Cleveland, OH 44106

Dr. Michael Pocha
Lawrence Livermore Nat. Laboratory
L-222
P. O. Box 808
Livermore, CA 94550-0808

Dr. Venkatesh Narayanamurti
DEAS 217 Pierce Hall
Harvard University
29 Oxford St
Cambridge, MA 02138

Dr. Brian Ridley
Dept. of Electrical Engineering
Cornell University
Ithaca, NY 14853

Dr. Gerard Mourou
Ctr for Ultrafast Optical Sci
Univ of Michigan
2200 Bonisteel Blvd
Ann Arbor, MI 48109

Dr. Arye Rosen
David Sarnoff Research Center, SRI
CN 5300
Princeton, NJ 08543-5300

Dr. Steven R J Brueck
CHTM
Univ of New Mexico
1313 Goddard SE
Albuquerque, NM 87106

Dr. Edl Schamiloglu
Dept. Elect. & Comp. Eng.
Room 110, EECE Bldg
University of New Mexico
Albuquerque, NM 87131-1356

Dr. Stephen E Schnatterly
Dept of Phys
Univ of Virginia
Charlottesville VA 22901

Dr. Karl Schoenbach
Old Dominion University
Rm 231, Krausman-Duckworth Hall
Norfolk, VA 23529-0246

Dr. Peter Herczfeld
Electrical & Computer Engineering Dept.
Drexel University
Philadelphia, PA 19104

Dr. Paul J. Stabile
David Sarnoff Research Center, SRI
CN 5300
Princeton, NJ 08543-5300

Dr. Robert R. Rice
Mail Code 055-WA05
The Boeing Company
6633 Canoga Avenue
Canoga Park, CA 91309

Kenneth E Kambour
Dept of Physics
Texas Tech University
Stop 1051
Lubbock, TX 79409

Dr. Charles H Townes
Phys Dept
U. C. Berkeley
557 Birge Hall
Berkeley CA 94720

Dr. William C. Nunnally
Department of Electrical Engineering
303 Engineering Bldg West
University of Missouri
Columbia, MO 65211

George Yates
P-23,MS H803
Los Alamos National Laboratory PO 1663
Los Alamos, NM 87545

Dr. Mike Abdahlla
ASR Corporation
7817 Bursera Dr. NW
Albuquerque, NM
87120

Dr. Thomas E. McDonald
P-23,MS H803
Los Alamos National Laboratory PO 1663
Los Alamos, NM 87545

Dr. Mike Skipper
ASR Corporation
7817 Bursera Dr. NW
Albuquerque, NM
87120

Dr. C. David Capps
2733 37th Avenue SW
Seattle, WA 98126

Dr. Kent Choquette
312 Microelectronics Laboratory
University of IL at Urbana-Champaign
208 N. Wright Street
Urbana, IL 61801

Dr. Brian K. Ridley
School of Electrical Engineering
Phillips Hall
Cornell University
Ithaca, NY 14853-5401

Dr. Dan Bailey
Compaq Computer Corp.
334 S. St.
MS SHR3-1/S30
Shrewsbury, MA 01545

Sam Kang
Dept of Physics
Texas Tech University
Stop 1051
Lubbock, TX 79409

Mr. Kenneth Prestwich
12201 Cedar Ridge NE
Albuquerque, NM 87112

Dr. William Baker
Air Force Research Lab/DEH
3550 Aberdeen Ave, SE
Kirtland AFB, NM 87117-5776

Dr. Steve Sampayan
Lawrence Livermore Nat. Lab.
P.O. Box 808 L-645
Livermore, CA 94551-9900

Dr. Walter J. Sarjeant
University of Buffalo – SUNY/AB
Dept. of Elect. and Comp. Eng.
P.O. Box 601900
Buffalo, NY 14260-1900

Dr. Jon Mayes
Applied Physical Electronics, L.C.
602 Explorer
Austin, TX 78734

Mr. Ian Smith
Titan Pulse Sciences, Inc.
600 McCormick St.
San Leandro, CA 94577

Distribution - Foreign

Dr. Brian K. Ridley
Deptartment of Physics
University of Essex
Colchester CO4 3SQ
UK

Dr. Anthony Holden
Marconi Optical Components
Caswell, Towcester
Northants NN12 8EQ
UK

Distribution - Internal

Qty	Org	MS	Name	Qty	Org	MS	Name
1	1000	0513	Al Romig	1	1744	0603	Peter Esherick
1	1100	1427	Tom Picraux	1	1822	0886	Bonnie McKenzie
1	1111	1056	Barney Doyle	1	1846	0888	Bob Anderson
1	1111	1413	Carl Seager	1	1900	0323	Don Cook
1	1111	1056	Dave Follstaedt	1	1903	0323	Regan Stinnett
1	1118	1423	Joe Woodworth	1	2131	0482	Kent Meeks
1	1118	1423	Randy Schmitt	1	2131	0482	Scott Holswade
1	1122	1421	George Samara	1	2612	0328	Jim Wilder
1	1123	0601	Art Fischer	1	2612	0328	John Sackos
1	1123	0601	Eric Jones	1	2612	0328	Lou Weichman
1	1123	0601	Jerry Simonds	1	2612	0328	Pat Smith
1	1123	0601	Steve Kurtz	1	2618	0859	Bob Nellums
3	1123	0601	Weng Chow	1	2618	0859	Collin Smithpeter
1	1126	0601	Bob Biefeld	1	2618	0859	Ed Hoover
1	1140	1413	Terry Michalske	1	5911	1202	Tim Drummond
1	1600	1190	Jeff Quintenz	1	6201	0752	James Gee
1	1630	1178	Doug Bloomquist	1	6218	0753	Jerry Ginn
1	1639	1178	Henry C. Harjes	1	6245	0710	Bruce Kelley
1	1640	1194	Dillon McDaniel	1	6524	0974	Phil Dreike
1	1640	1194	Ken Prestwich	3	9235	1111	Harry Hjalmarson
1	1640	1194	Tom Martin	1	9720	0136	Juan Ramirez
1	1642	1152	Dave Seidel	1	11500	0161	John Hoheimer
1	1642	1152	Mark Kiefer	1	15000	1221	Jim Tegnelia
1	1643	1152	James Solberg	1	15003	1221	Dan Rondeau
1	1643	1152	Kim Reed	1	15300	1165	William Guyton
1	1643	1152	Larry Schneider	1	15300	1188	Roy Hamil
1	1644	1194	David E. Bliss	1	15309	1168	Richard O'Rourke
1	1644	1194	Mike Mazarakas	1	15310	1221	Russ Skocypec
1	1644	1194	Rick Spielman	1	15311	1188	Brian F. Clark
1	1644	1194	William Stygar	1	15311	1188	David Wick
1	1645	1193	John Maenchen	1	15311	1188	John S. Wagner
1	1670	1191	Keith Matzen	1	15311	1188	Stephenson Tucker
1	1674	1186	Barry Marder	1	15322-1	1157	John Montoya
1	1674	1186	Steve Slutz	1	15322	1157	Larry R. Shapnek
1	1674	1186	Thomas Mehlhorn	1	15330	1153	Malcolm Buttram
1	1677	1196	Ramon Leeper	1	15331	1153	Dale Coleman
1	1700	1079	Dave Williams	1	15331	1153	Gerald Rohwein
1	1707	1425	Marion Scott	1	15331	1153	Jeff Alexander
1	1738	1073	Steve Rohde	1	15331	1153	Mitchell Ruebush
1	1740	1077	Tom Zipperian	1	15331	1153	Nancy L. Ries
3	1742	0603	Albert Baca	1	15331	1153	Nancy Ries
3	1742	0603	Allen Vawter	1	15331	1153	Paull Patterson
1	1742	0603	Charles Sullivan	1	15331	1153	Richard P. Toth
1	1742	0603	Darwin Serkland	1	15331	1153	Steven Dron
3	1742	0603	Michael Hafich	1	15331	1153	Talbot Smith
1	1742	0603	Olga Spahn	1	15331	1153	Wayne Crowe
1	1742	0603	Ron Hadley	3	15333	1153	Alan Mar
1	1742	0603	Tom Plut	50	15333	1153	Fred Zutavern
1	1743	0603	Mial Warren	1	15333	1153	Gary Denison
1	1744	1425	Carol Ashby	3	15333	1153	Guillermo Loubriel

1	15333	1153	Joe Lundstrom		1	15335	1182	Timothy J. Renk
1	15333	1153	Larry Bacon		1	15336	1153	Robert Walko
1	15333	1153	Larry Rinehart		1	15336	1153	Steve Dron
1	15333	1153	Lars Roose		1	15336	1153	Stewart Cameron
1	15333	1153	Luis Molina		1	15341	1179	David E. Beutler
3	15333	1153	Martin O'Malley		1	15341	1179	Roque R. Gallegos
1	15333	1153	Steve Babcock		1	15343	1167	Larry D. Posey
1	15333	1153	Thomas Nelson		1	15345	1166	Ted F. Wrobel
1	15333	1153	Ting S. Luk		1	15351	0859	Terry Stalker
1	15333	1153	Willie Luk		1	15353	1170	Phil Van Buren
1	15334	1153	Ronald Pate		1	15403	1164	Tom Hitchcock
1	15335	1182	Bob Turman		1	16000	0105	Bob Asher
1	15335	1182	Ronald J. Kaye		1	16000	0839	Gerry Yonas
1	15335	1182	Steven L. Shope					

1	8945-1	9018	Central Technical Files
2	9616	0899	Technical Library
1	9612	0612	Review & Approval Desk, for DOE/OSTI
1	11500	0161	Patent and Licensing Office
1	4001	0188	LDRD Office

Total number of copies is: 257