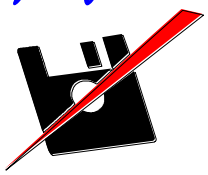
User's Manual

Core Technologies for Laser Markers



AMERICAN
LASERWARE inc

PO BOX 845 **222 Clearview Road**
Goldenrod, FL 32733 **Chuluota, FL 32766**
Phone: (407) 366-2237 - Fax: (407) 366-4691
www.laserware.com

Limited Warranty:
American Laserware Inc. will repair or replace this product, for one (1) year from the date of original purchase in the event of a defect in materials or workmanship.

Limits and Exclusions:
No warranties are expressed or implied except for the expressed warranty listed above. There is no warranty of merchantability or fitness for a particular purpose. American Laserware shall not be liable for incidental or consequential damages resulting from the use of this product, the information in this manual, or arising out of any breach of this warranty.

Some states do not allow the exclusion or limitation of incidental or consequential damages, or limitations on how long an implied warranty lasts, so the above exclusions or limitations may not apply to you.

This warranty gives you specific legal rights and you may have other rights that vary from state to state.

American Laserware, Inc.
P.O. Box 845
Goldenrod, FL 32733
Phone support:
(407) 366-2237

Table of Contents

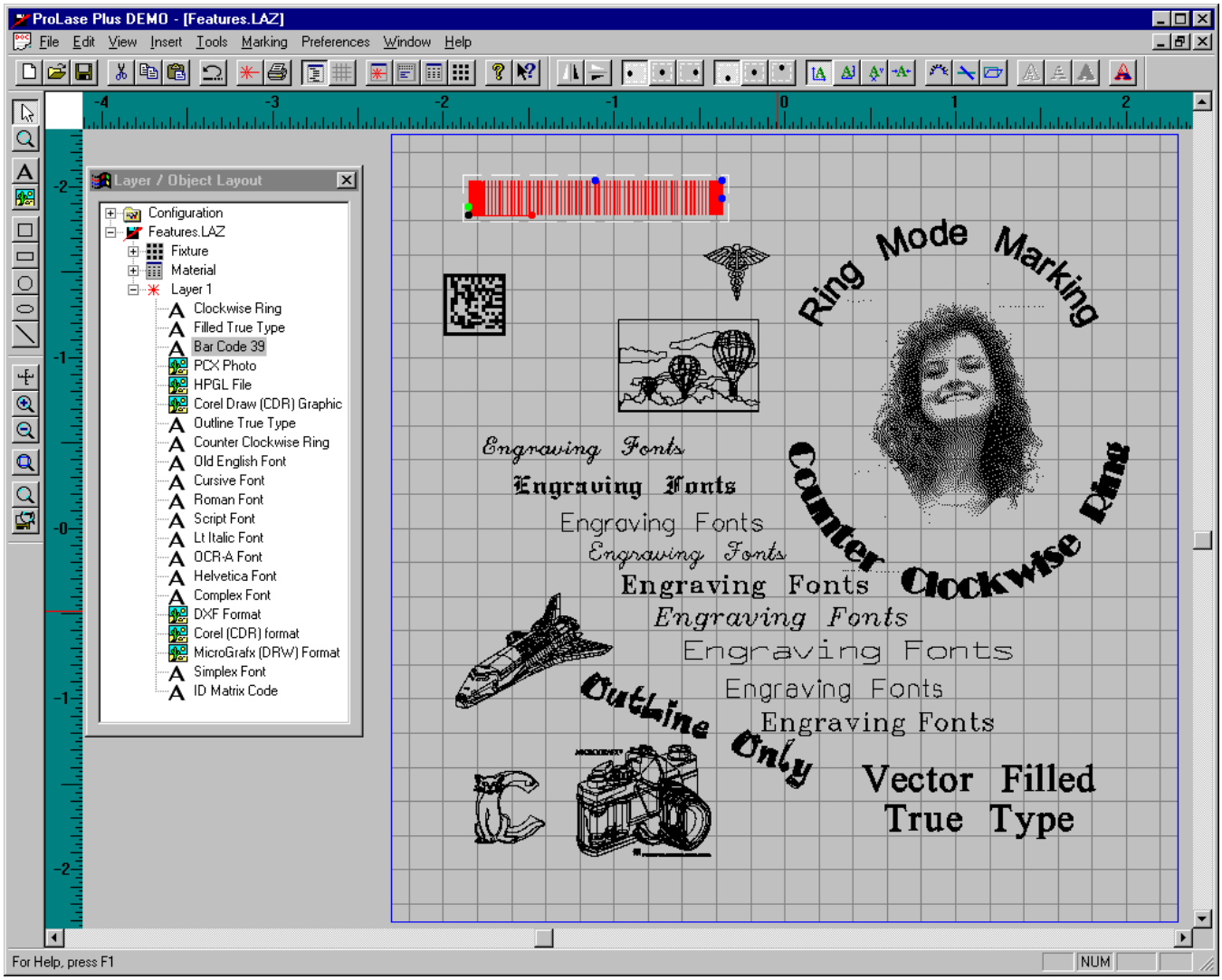
Chapter 1 Introduction

Welcome to ProLase 7! ProLase has been designed to meet the needs of all types of users of laser marking systems at a reasonable price. ProLase was developed to be original software on new systems as well as a retrofit package for existing systems. The package provides significant advancements over previous laser marking control systems, while remaining extremely user-friendly. It's an object oriented, graphically interactive, PC control system providing a user the ability define and execute laser marking jobs (LAZ files). Multiple hardware interfaces are supported giving ProLase the ability to control most laser marking systems, including Nd:YAG, CO_2, VO_4, and fiber lasers.

File linkages to several internal datasets makes ProLase programs flexible and powerful. These datasets include a materials application system, a fixture file system, and a translation database. The materials application system allows a user to define a laser process, give the process a unique name (usually the substrate name, e.g. '304 stainless steel') and subsequently link the process to job files. A material process can include up to seven passes using different values for power, frequency, and speed on each pass. The file system can contain and manage many thousands of different process 'recipes'.

The fixture files allow the user to control fixture offsets and define high level step and repeat (S&R) processes. Just like the material file system, any LAZ job can use any fixture defined in a fixture file. The links allow all appropriate graphic and process information to be automatically loaded when the operator selects the LAZ file. At any time the operator can change the links, for example a LAZ job that is normally marked on stainless steel, can be marked on brass by selecting the brass process file prior to executing the job program file.

Unlike some marking software, the operator never has to remember what fonts and logos need to be loaded for a particular job. ProLase automatically performs all required graphic loading. ProLase does not require users to learn any programming languages or special codes, and yet ProLase provides all of the flexible, graphic control users are accustomed to, including radial marking, aspect control, character spacing, angular rotations, and full justification. Text to be marked can be fixed or variable. Variable text can be retrieved at runtime from a variety of sources including, the keyboard, a bar code reader, and disk files. Automatic date coding and alphanumeric serialization are included as variable text types. Available fonts include specialized laser engraving fonts, normal Windows TrueType fonts, and a large number of bar code formats. True Type fonts can be vector filled with user specified density, angle and kerf values. Graphics (sometimes called 'logos' on other systems) can be imported from a large variety of common vector or bitmap formats. All graphic features are either menu controlled or graphically controlled via the mouse and keyboard.



The best way to become familiar with ProLase is to read this manual while running the software. Experiment with different combinations of graphic controls while reading about them. The menu maps and the table of contents are useful for finding a particular function. If at any time a question comes up which is not answered by the manual, please call American Laserware (407-366-2237). We want to help you get the most out of ProLase. Free software updates are available on our web site (**www.laserware.com**). It is recommended that the user periodically check the dates posted for the most recent update. This software is heavily supported and new versions are released on a regular basis. The latest version may contain new features, bug fixes, and other improvements that could enhance the user's production.

American Laserware highly recommends a strong backup routine such as making a copy of the entire ProLase installation folder. If the new version does not work correctly, restore the old version, and notify American Laserware so the problem can be solved.

ProLase is quite flexible and meets many needs. Sometimes, however the system must be customized to meet specific customer requirements. The ProLase7 COM Server option allows

programming savvy customers to design their own software to meet their specific needs, whether to limit the user interface for production only marking or to implement a unique part serialization scheme. Non-programmers or persons who feel a modification is required should contact American Laserware, Inc. directly to implement customizations. Normally this can be done quickly and at a very reasonable price.

Chapter 2 Hardware requirements

Security Key:

ProLase requires a software security device sometimes known as a lock or 'dongle'. This device plugs into the USB port on the PC. Any attempt to run the ProLase software without a properly attached dongle will generate an error. The type of lock is written upon it. Each version of ProLase is coded to a particular type of lock. For example, a ProLase7 PLUS lock will not activate the ProLase7 software. A ProLase7 lock is required.

WARNING: The security device is a valuable piece of hardware. Its value is the cost of the ProLase license fee! Please treat it that way. Damaged devices can be replaced for a small fee. LOST DEVICES WILL NOT BE REPLACED! Protect your software investment: keep track of your dongle!

CPU:

The ProLase software was developed on an Intel Processor with standard speed and memory. In general, the faster the CPU and more memory available, the better the performance.

A better computer system with faster processors and bussing and more memory will improve the performance and response of the software. However, this improvement is limited to the design process. In general, only a limited reduction in the cycle times for marking parts can be made by improvements to the computer. New lasers, marking heads, etc. are required for such effects.

Operating System:

ProLase was developed for Microsoft Windows 7 and will run in Windows 95, Windows 98, Windows Millennium Edition, Windows 2000, or Windows 7 operating systems. ProLase is an unusual program in that it must disable the normal operation of the OS while marking or waiting for a start signal. Specifically, ProLase will not process any Windows messages in these cases. While the laser is on during marking, all operating interrupts are disabled. To avoid problems it is best to not use other user interface applications while ProLase is in any mark mode (manual, auto, etc.). Applications and communications that happen in the background like network control do not have any adverse effects on ProLase, even while marking. It is best if the screen saver is disabled on a PC using ProLase for marking as it can interfere with the normal operation of ProLase.

Graphic Card:

ProLase is a high resolution, 16 color, graphic program requiring the use of a VGA graphic card and monitor. 640x480, 16 colors is the MINIMUM resolution and color depth supported. Like most graphic programs, ProLase is best when used with higher resolutions and larger monitors.

Control I/O:

ProLase can be configured to drive several different types of laser engraving/marking equipment.

When configured for an HPGL device such as the Universal Laser Systems laser plotters, the PC Centronics port is all the I/O capability required for normal manual operation. Other digital I/O can be optionally provided for use of parts handling equipment.

When configured for a galvanometric device (see Fig. 1) such as General Scanning or Cambridge equipment, a Techmar DADIO or Measurement Computing PCIM-DDA06/16 card is required. Examples of existing equipment that might use this I/O include Quantrad, Control Laser, General Laser, Lumonics, AB, JEC, etc.

ProLase uses all I/O addresses from 300 hex to 34F hex and possibly others if PCI cards are used. No single system will use all these addresses, but between different interface hardware options, all of this address space is utilized. Many times we do multiple outputs for the same function, like outputting the Synrad power command and the PWM from the PCI-CTR card at the same time. The proper thing for all ProLase customers to do is to make sure that there are no hardware devices (except ProLase devices) addressed between 300 and 34F hex. You can do this by checking the RESOURCES tab for all devices listed in DEVICE MANAGER. If you find any devices within the ProLase address space you should change those devices so that they use addresses outside of ProLase address space. You can RESERVE ProLase address space in Windows so that no PLUG & PLAY devices are assigned those addresses. To reserve ProLase I/O space:

Go to Control Panel and click on the SYSTEM icon.
Click on the DEVICE MANAGER tab.
Right Click on the COMPUTER icon.
Click on the PROPERTIES menu selection.
Click on the RESERVE RESOURCES tab.
Check Input/Output I/O.
Click on the ADD button.
Enter STARTING VALUE 300 Hex
Enter ENDING VALUE 34F Hex
Reboot System

ProLase address map (ISA cards).

240 Hex - 24F Hex	**SCANLAB RTC2 card**
300 Hex - 30F Hex	**Cambridge Digital interface**
310 Hex - 31F Hex	**CIO-CTR05 counter Timer card**
320 Hex - 32F Hex	**RESERVED (custom interfaces)**
330 Hex - 33F Hex	**DDA06, DADIO or Synrad FLCC card**
340 Hex - 34F Hex	**SCANLAB PC1000 card**

DAC Cards:

ProLase supports several off-the-shelf DAC cards. All ISA DAC cards should be addressed to 330 hex. The DADIO (Scientific Solutions or Techmar) card provides 4 12-bit DAC outputs, and 16 bits of digital I/O. The CIO-DDA06/12 (Measurement Computing) provides 6 12-bit DAC outputs and the same 16 bits of digital I/O. On either card, DACs 2 and 3 are configured for +/-10 volt outputs (X and Y) for connection to standard analog servo electronics. DACs 0 and 1 normally set for 0-5 volt outputs for analog control of power and Q-Switch frequency. In the case of the Synrad lasers with a UC1000, DAC 0 should be set for 0-10 volt outputs. A male DB9 connector provides the DAC outputs on the

Techmar card. A DB37 connector provides all signals on the CIO-DDA06/12 board. These cards must be addressed by dipswitch to 330 hex(see card manuals for setting address and connector pin assignments). Remember only one of these cards should be used.

Currently the most used analog configuration is the Measurement Computing PCIM-DDA06/16 DAC card combined with the PCI-CTR05 counter timer card.

On any of these cards, the digital I/O is configured in two banks of eight bits each. The first bank is inputs and the second outputs. These I/O points are provided by a male DB25 connector on the DADIO card and by the DB37 on the DDA06 cards. It is recommended that the digital I/O is optically isolated by use of an OPTO 22 rack and modules (or equivalent). Currently the following I/O definitions are implemented in the software (see tables below for connector pin assignments):

DADIO CARD:

This section describes the pin outs for the Scientific Solutions Inc. DADIO DAC Card. All digital signals are on the DB25 connector and all analog signals are on the DB9 connector.

DB25 Digital I/O

INS:	PIN#:	DESCRIPTION:
#0	17	Start write signal.
#1	4	
#2	16	
#3	3	
#4	15	
#5	2	
#6	14	
#7	7	Optional Hardware Abort signal, can be programmed to abort run on high or low input.

OUTS:	PIN#:	DESCRIPTION:
#0	25*	Laser On, connects to gate input on laser system.
#1	12	
#2	24*	NOT Laser ON, reverse of #1, connects to gate input
#3	11	
#4	23	System ready for start signal - for auto control
#5	10	
#6	22	
#7	9	
GROUND	13	Common signal ground for all ins and outs

* Use either OUT#0 or OUT#2 for gate control, not both. NOTE: Synrad customers should use pin 24 for the UC1000 gate control.

DB9 DAC Outputs

DAC	PIN#:	DESCRIPTION:
#0	6	Power Control DAC (0-5 or 0-10 VDC)
#1	7	Frequency Control DAC (0-5 or 0-10 VDC)
#2	8	X Galvo Control DAC (+/- 5 or +/- 10 VDC)

#3	**9**	**Y Galvo Control DAC (+/- 5 or +/- 10 VDC)**
GROUND	**1-5**	**Signal grounds for DACs**

CIO-DDA06/12 or /16 or PCIM-DDA06/16:

This section describes the pin outs for the Measurement Computing DDA06 DAC Card. This card uses a single DB37 connector for both digital and analog signals. The UPDATE/XFER jumper can be set either way for the ISA versions of the card, but the PCIM-DDA06/16 needs that jumper set to XFER mode.

Digital Inputs and Outputs:

INS:	**PIN#:**	**DESCRIPTION:**
#0	**37**	**Start write signal.**
#1	**36**	
#2	**35**	
#3	**34**	
#4	**33**	
#5	**32**	
#6	**31**	
#7	**30**	**Optional Hardware Abort signal, can be programmed to abort run on high or low input**

OUTS:	**PIN#:**	**DESCRIPTION:**
#0	**10***	**Laser On, connects to gate input on laser system.**
#1	**9**	
#2	**8***	**NOT Laser ON, reverse of #1, connects to gate input**
#3	**7**	
#4	**6**	**System ready for start signal - for auto control**
#5	**5**	
#6	**4**	
#7	**3**	**Last Part in series being marked**
GROUND	**11**	**Common signal ground for all digital ins and outs**
CW	**25**	**High when CW mode is active/ Low when not CW (pulsing)… PORT C, bit 4**

* Use either OUT#0 or OUT#2 for gate control, not both. NOTE: Synrad customers should use pin 8 for the UC1000 gate control.

Analog Outputs (DACs):

DB37 (Cont) DAC

DAC#	**PIN#:**	**DESCRIPTION:**
#0	**18**	**Power Control DAC (0-5 or 0-10 VDC)**
#1	**16**	**Frequency Control DAC (0-5 or 0-10 VDC)**
#2	**14**	**X Galvo Control DAC (+/- 5 or +/- 10 VDC)**
#3	**12**	**Y Galvo Control DAC (+/- 5 or +/- 10 VDC)**
#4	**2**	**Focus**
#5	**1**	**Spare**
LLGND	**13,15,17,19,20,21**	**Signal grounds for DACs**

PCI-DDA02/12 or /16, PCI-DDA04/12 or /16, PCI-DDA08/12 or /16:

This section describes the pin outs for the Measurement Computing PCI-DDAxx DAC Cards. These cards use a single 100-pin connector for both digital and analog signals.

Digital Inputs and Outputs:

INS:	PIN#:	DESCRIPTION:
#0	58	Start write signal.
#1	57	
#2	56	
#3	55	
#4	54	
#5	53	
#6	52	
#7	51	Optional Hardware Abort signal, can be programmed to abort run on high or low input

OUTS:	PIN#:	DESCRIPTION:
#0	66	Laser On, connects to gate input on laser system.
#1	65	
#2	64	NOT Laser ON, reverse of #1, connects to gate input
#3	63	
#4	62	System ready for start signal - for auto control
#5	61	
#6	60	
#7	59	Last Part in series being marked
GROUND	100	Common signal ground for all digital ins and outs

* Use either OUT#0 or OUT#2 for gate control, not both. NOTE: Synrad customers should use pin 8 for the UC1000 gate control.

Analog Outputs (DACs):

100-pin (Cont) DAC			
DAC#	PIN#:	DESCRIPTION:	
#0	1	Power Control DAC (0-10 VDC)	NOTE: X GALVO CONTROL DAC ON DDA02
#1	3	Frequency Control DAC (0-10 VDC)	NOTE:Y GALVO CONTROL DAC ON DDA02
#2	5	X Galvo Control DAC (+/- 5 or +/- 10 VDC)	
#3	7	Y Galvo Control DAC (+/- 5 or +/- 10 VDC)	
#4	9	Focus Control DAC (0-10 VDC)	
#5	11	Spare	
#6	13	Spare	
#7	15	Spare	
LLGND	2,4,6,8,10,12,14,16	Signal grounds for DACs	

DIO48:

This section describes the use of the Measurement Computing Corporation's (MCC) CIO-DIO48 card for the following ProLase drivers: Cambridge 603x (16 Bit MUX) and 32 Bit X/Y. These drivers use digital signals to communicate the galvo positions to the marking head (or servo controller). The pin assignments are detailed below. Note that the Cambridge 603x does NOT connect to the DY0-DY15 pins. This driver uses the strobe signals to indicate which galvo receives the position specified by DX0-DX15.

50-pin Connector

Pin#	Func	Pin#	Func
50	Ground	49	N.C.
48	N.C.	47	N.C.
46	N.C.	45	N.C.
44	N.C.	43	N.C.
42	N.C.	41	N.C.
40	DY8	39	DY9
38	DY10	37	DY11
36	DY12	35	DY13
34	DY14	33	DY15
32	DY0	31	DY1
30	DY2	29	DY3
28	DY4	27	DY5
26	DY6	25	DY7
24	N.C.	23	N.C.
22	N.C.	21	N.C.
20	Gate	19	NOT Gate
18	Strobe X	17	Strobe Y
16	DX8	15	DX9
14	DX10	13	DX11
12	DX12	11	DX13
10	DX14	9	DX15
8	DX0	7	DX1
6	DX2	5	DX3
4	DX4	3	DX5
2	DX6	1	DX7

PC1000:

This section describes the use of the PC1000 XY-Head Interface board from SCANLAB AG. This card allows control of the SK1000 Scan Head family from SCANLAB, the XY2/100 head from General Scanning, Inc. and other hardware. The user should consult with SCANLAB about any compatibility issues concerning the PC1000 card. ProLase requires the PC1000 card to be jumper addressed to hex 340. All wiring should be done as specified by SCANLAB. The PC1000 supplies the X/Y galvo control, however a CIO-DDA06/12 (ISA) and CTR05 (either ISA or PCI) are used for all other control including digital I/O, laser power, focus, pulsing, and timing.

Laser Control:

The LASER ON signal can be used to gate the internal Q-switching frequency generator provided by most Nd:YAG laser manufacturers (set frequency mode to Internal/Gated). When this signal is gated ON the Q-switching pulse train is sent to the RF electronics, when gated OFF the pulse train will be prevented from reaching the RF driver (Q-switch hold-off). It can also be used to gate pulse width modulation electronic equipment supplied by RF excited CO_2 laser manufacturers. In either case, the laser input for this signal is normally labeled as GATE IN. The signal is used by the computer to control laser firing. It is set ON while a vector is being drawn and OFF during non-marked movements (between letters for example). The GATE signal can also be used to initiate the First Pulse Quencher circuit on many older Nd:YAG lasers. Newer lasers have a more sophisticated First Pulse Suppression logic which is supported using a counter/timer board as explained later in this manual.

The START signal is used for parts handler interfaces. It initiates a marking sequence just as the keyboard <ENTER> does. The START signals are active only when ProLase is in one of it's marking modes. The READY signal is used to tell a parts handler when the computer is ready to accept a START command. These two signals are all that are required for most part handler system handshaking. The LAST PART signal is used for queuing handler systems. It informs the handler that the last part in a sequence is being marked. ProLase adds user programmability to this hardware for customer specific control requirements.

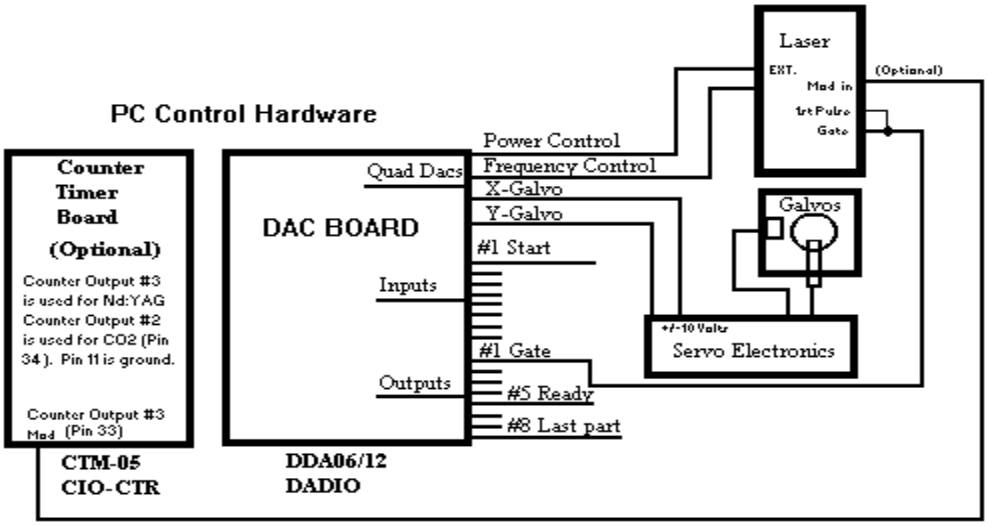


Figure 1: Connections for a typical Nd:YAG galvo system.

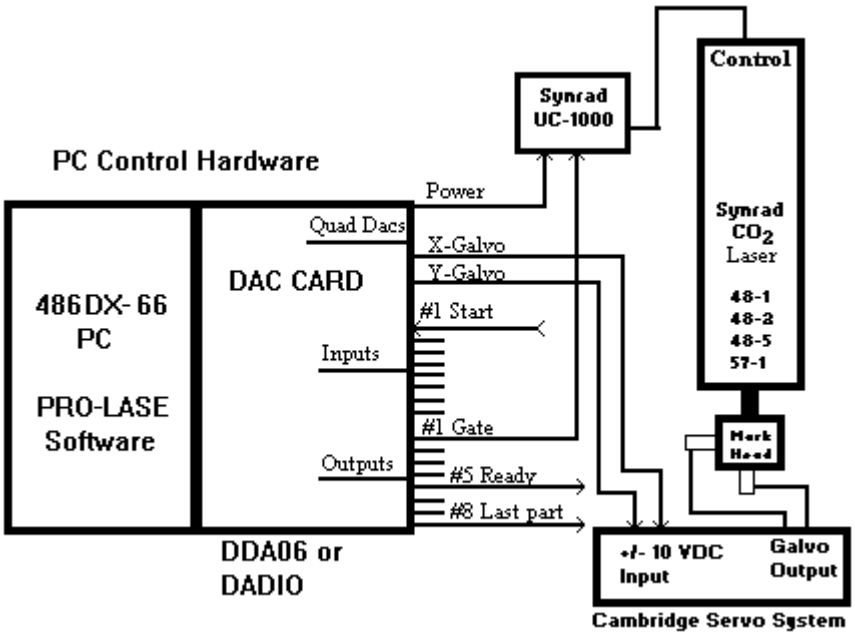


Figure 2: Connections for a typical CO_2 galvo system.

Optional Pulse Control:

Actual pulsing signals can be optionally provided. This TTL signal connects directly to the RF driver MOD IN, providing Q-switch control (set frequency mode to External). When using this option, the FPS (pin 32) signal is used to provide first pulse suppression logic to Nd:YAG lasers with this feature. The interface is accomplished using a counter/timer card. ProLase supports 3 models: CTM-05, CIO-CTR05, or PCI-CTR05. This option is also required for the use of the ProLase raster (bitmap marking) capability. The card is used to precisely coordinate the galvo positioning with the laser pulsing. The DAC frequency control output is not used with this option. To use this option:

1. If ISA card address the CTM-05 board to hex 310.
2. Use Pin 33 for Nd:YAG MOD OUT.
3. Use Pin 11 as digital ground.
4. Jumper (connect with wire) Pin 35 to P18 on Counter/Timer Board
5. (Optional) If your laser supports First Pulse Suppression connect pin 32 to the laser input for this function.

A signal configured to provide pulse width modulation (PWM) power control of RF excited CO_2 lasers, like those produced by Synrad Inc. of Bothell, Washington, or Universal Laser Inc. of Scottsdale AZ, is also provided. In this case, the DAC power control is not used. The DAC power control and GATE signal can be used without this option by providing an external PWM controller such as the UC-1000 series manufactured by Synrad Inc. To use this option:

1. If ISA card address the CTM-05 board to hex 310.
2. Use Pin 34 as PWM Control. This connects directly to the Synrad Laser control input.
3. Use Pin 11 as digital ground.
4. Jumper Pin 35 to Pin 18 on Counter/Timer Board.

All of these signals include the 'StartSeg TC' logic that minimizes hot spots created by galvo acceleration. Pin 35 can be used as a preferred GATE signal that also provides this logic. The GATE signals on the DAC boards do not include this feature.

A Metra-Byte PC bus counter/timer board (Model CTM-05) or Computer Boards Inc. (Model CIO-CTR or PCI-CTR05) is required for any of these options. The Computer Boards Inc. product, CIO-CTR05 is about $150 USD.

Laser control signals:

A fully implemented system using a DDA (or similar) driver and a CTR05 card supports 5 laser outputs: 1) Gate, 2) Delayed Gate, 3) First Pulse Suppression (FPS), 4) Pulse-width modulation (PWM), 5) Q-switch modulation. All 5 signals are generated for all marks, however only those require should be used. A typical Q-switched YAG system will require the FPS and Q-switch signals but not the PWM, which is used to control RF excited CO_2 lasers. The PWM signal is used for CO2 lasers. The FPS and Q-switch modulation are used on YAG lasers. The Gate (or NOT Gate) signals originate from the general purpose I/O. The remaining signals are from the CTR05 card. The following figure illustrates the relative timing of each of the signals:

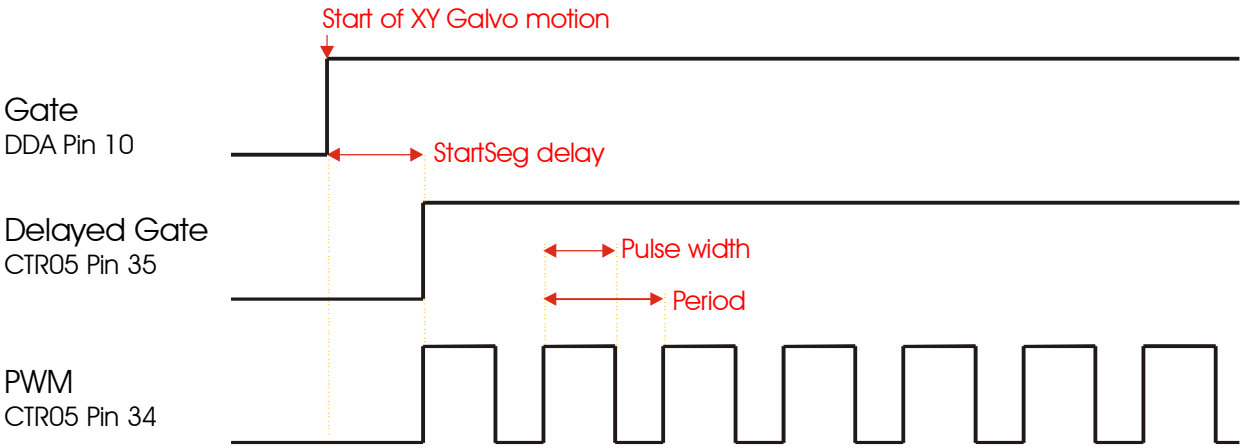


Figure 3: Typical PWM CO_2 laser signal timing diagram.

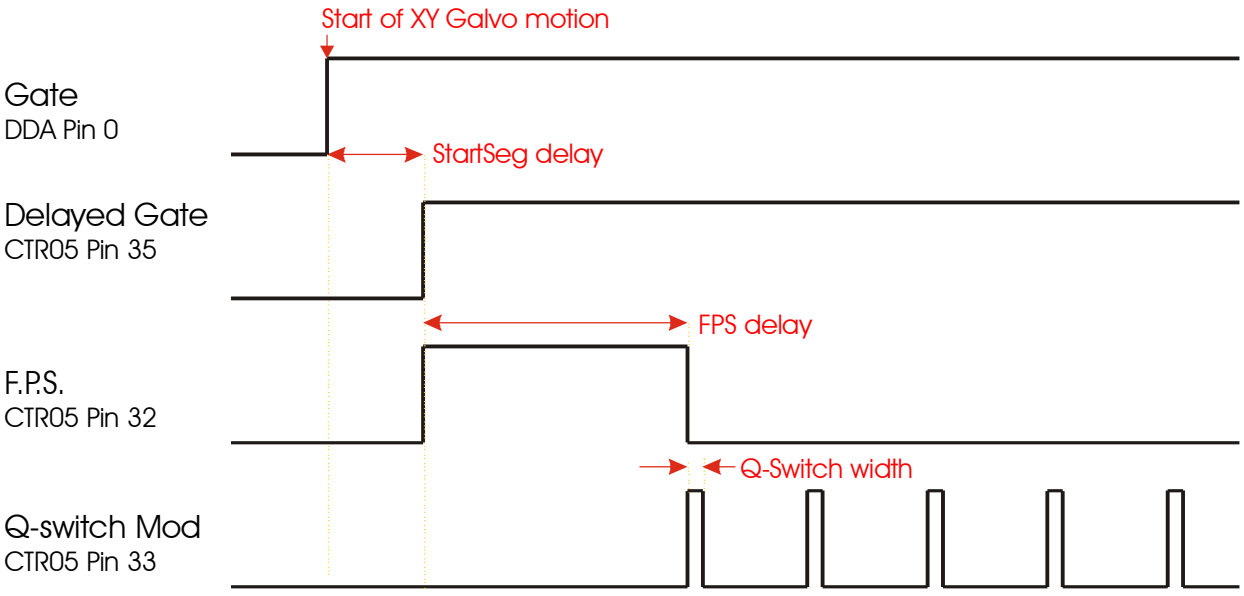


Figure 4: Typical Q-switched YAG (normal FPS mode) laser signal timing diagram.

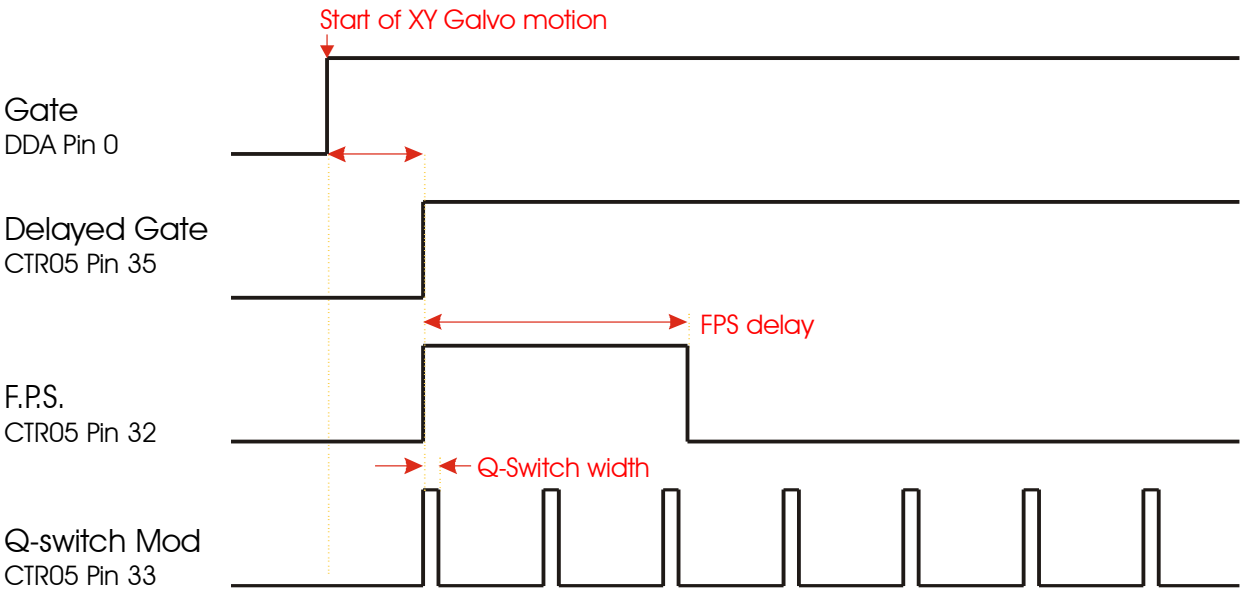


Figure 5: Typical Q-switched YAG (in Alt. FPS Mode) laser signal timing diagram.

In the preceeding figures, the period for the PWM and the Q-switch mod. signals vary with the frequency laser parameter setting of the object being marked. The PWM pulse with varies with the object's power level. The StartSeg delay, the Q-Switch width, FPS delay and FPS mode toggle are parameters set in the Configuration pages (see Config. below).

For laser system power supplies that require indication of continuous wave (CW) or pulsed operation, a CW Signal is available. This output signal is only available for systems that use the DDA card / 8255 chip for digital I/O, including some configurations of SCANLAB PC1000, RTC2 and RTC3 cards. The CW Signal is provided by the 8255 chip in Port C, bit 4. On the DDA06 card, this maps to pin 25. The CW Signal is active-high to indicate CW mode. A low state indicated pulsing mode.

SCANLAB RTC2 , RTC3, RTC4, RTC5:

The RTC cards from SCANLAB are independent, digital motion/laser control system that resides on a PC bus. The RTC card provides X/Y galvo positioning control and laser control signals. As options, the RTC can provide Z-axis focus control, digital I/O, and On-The-Fly marking features. When this card is used, ProLase acts as a 'front-end' system only and is not directly involved in the actual real-time control. From the End-User's point of view use of ProLase is very much the same as when using one of our direct control interfaces discussed above. When control or implementation of a feature varies for RTC users, it is documented with the feature to which it applies. Generally these changes are minor. There is one notable exception: arcs are not supported completely by the RTC card motion control. Therefore, ProLase converts all arcs into a series of connected line segments (polylines) before passing them on to the RTC card. Although not as precise as actual arcs, this approximation works for the vast majority of applications. The user can control the 'fineness' of the line segments using the *Arc Res* parameter in the RTC configuration dialog. Setting the *Arc Res* to 1 results in the greatest number of line segments to approximate the arc and is the most precise. However, this also means that is may take longer to mark. As *Arc Res* is increased fewer and fewer line segments are used. Eventually, only a single line segment will connect the end points. The default value of 100 works well for the majority of applications and should only be modified when more precise or faster marks are required.

On-The-Fly marking capabilities are not part of standard ProLase or ProLase PLUS. These features are only available using RTC driver with the ProLase RTC version of the software. (Of course, the RTC card must have the On-The-Fly features hardware enabled from SCANLAB as well.) In versions of ProLase lacking this feature, the controls are removed or disabled.

The RTC card handles X/Y galvo control, laser power control and laser gating/pulsing control. A Counter/Timer board is not required. If digital I/O is not provided on the RTC card, it can be provided by either an optional SCANLAB External I/O add-on card or via a CIO-DDA06/12 card. This selection is made on the RTC configuration dialog. ProLase DAC focus control requires the DDA06/12 card be used. The DDA06/12 MUST be configured as described above for the normal DDA06/12 driver. When the Digital I/O is provided by the SCANLAB RTC hardware, ProLase is mapped to the higher number inputs and outputs. Simply add 8 to the ProLase input/output number to identify the corresponding pin on the SCANLAB card. For example, ProLase input 0 = SCANLAB Input 8 = Ext.IO pin # 13. This is done to allow ProLase user to access the built in optically isolated inputs (SCANLAB inputs 12-15). The output channels are mapped the same way for continuity.

All RTC cards also come with additional External Start/Stop inputs. These maybe enabled or disabled in the RTC configuration dialog. When enabled, these signals work in conjunction (logical OR operation) with the normal ProLase digital I/O. For example, ProLase will begin marking when either the Start Mark signal from normal digital I/O is seen OR when the SCANLAB External Start signal is seen.

RTC boards support up to 6 analog output channels. RTC2 boards support only Analog 1. All RTC3/RTC4 cards can support Analog 1 and Analog 2. Analog 3 – Analog 6 (also called Output A-D)

are only available with the RTC3/RTC4 I/O Extension card installed and selected for use as the Digital I/O source for the RTC3/RTC4. The function of each of these channels is as follows:

Analog 1	Power control of YAG-type laser (similar to DDA06/12 DAC0)
Analog 2	Power control for grayscale marking
Analog 3	RESERVED
Analog 4	RESERVED
Analog 5	Focus control (similar to DDA06/12 DAC 5)
Analog 6	RESERVED

If a DDA card is chosen as the digital I/O source for the RTC3/RTC4 card, then its DAC5 can be used for focus control. Since Analog 1 is always available with the RTC cards, it should be used for power control and not DDA DAC0.

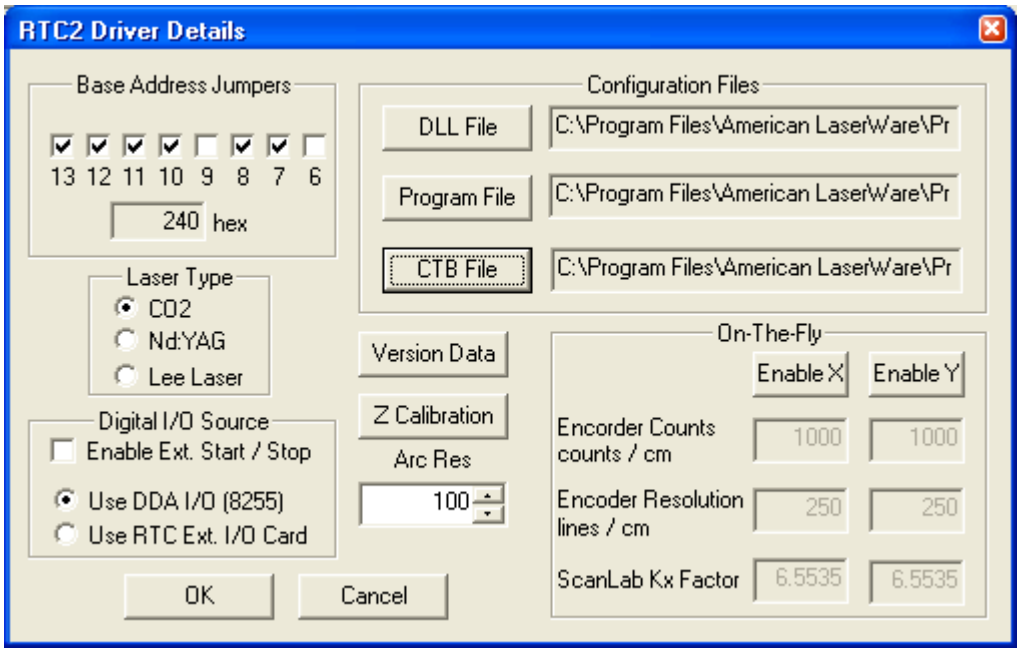
The first step in configuring the RTC driver in ProLase is to follow the complete installation instructions for the RTC hardware and accompanying software as specified by SCANLAB. **This must be done PRIOR to using ProLase**. The ProLase installation does NOT install any of the SCANLAB RTC software. The RTC software includes a test application (HPGL.exe) that should be used to verify that the RTC card and driver software as installed properly and will identify the proper support files (*.DLL, *.HEX, and *.CTB) for your hardware and options. The RTC board comes with a diskette that contains a series of files. The *.HEX files contains programming information for the hardware on the RTC card. The *.CTB files are correction tables use to correct for optical distortions and are specific to the lens used. Consult the SCANLAB documentation for a more complete description. At least one of each type of these files is required by ProLase. The locations of the files are user selectable in the ProLase RTC configuration. The user should specify the same files that were used by the SCANLAB software to verify the card's operation.

The Version Data button (RTC2) or group (RTC3/RTC4/RTC5) displays information about the version of SCANLAB hardware and files being used. This information is provided purely for user benefit to assist SCANLAB in solving any problems the may occur in your system.

NOTE: NT systems using ProLase RTC and ProLase RTC DLL require the following:
Installation of GIVEIO driver using IOINST.EXE. Example: IOINST.EXE GIVEIO GIVEIO.SYS
Installation of Sentinel driver.
Installation of SCANLAB RTC driver using RTCSetup.exe from SCANLAB.
The NT versions of the SCANLAB *.DLL and *.HEX files must be used (For example, select RTC24NT4.DLL rather than RTC24W95.DLL). The location of these files are parameters in the ProLase configuration when the RTC2, RTC3, RTC4, RTC5 driver is selected.

The differences and features between the RTC2, RTC3, RTC4, RTC5 cards as applicable to ProLase are itemized below:

RTC2:



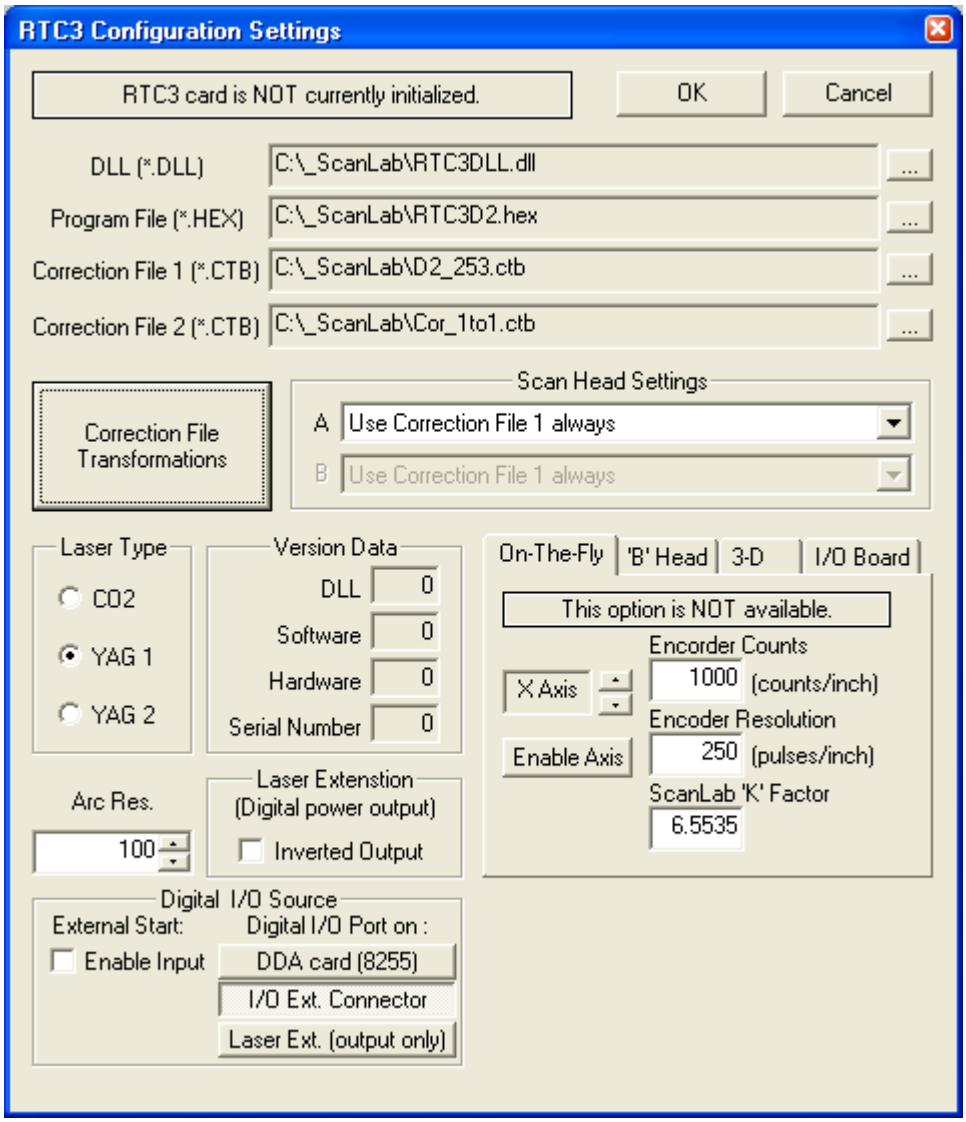
The SCANLAB RTC2 card is an ISA bus card and is NOT Plug&Play compatible. Therefore, its address setting must be set using hardware jumpers and this setting must be specified in the ProLase RTC2 configuration dialog using the Base Address Jumpers controls. The default for the card and the ProLase configuration is **240** hex. In most cases this will work fine

NOTE: ProLase RTC for Windows NT users must also install the RTC2 Driver using the SCANLAB provided RTC2SETUP.EXE program. The address of the card is a parameter when installing this driver and must match the address entered in the Base Address Jumpers parameter of the ProLase RTC2 configuration dialog.

The card must be the right type for the kind of laser used. The RTC2 comes in 3 basic types: CO2, Nd:YAG, and LEE. This will be factory set by SCANLAB based on your needs. You must tell ProLase the type of laser you are using, in the Laser Type parameter on the RTC2 configuration dialog. The laser output signals for the RTC2 card is fully defined in the SCANLAB documentation.

The RTC2 card also imposes a minimum speed limit for galvo motion. This is based on the field size used and is roughly given by FieldSize / 65 seconds. For a typical 4 inch square field, the minimum speed then is 4 inch / 65 sec = 0.0615 inch/sec.

RTC3:



The RTC3 card is designed for the PCI bus and will be assigned an address by the computer automatically. There is no user programmable addressing required.

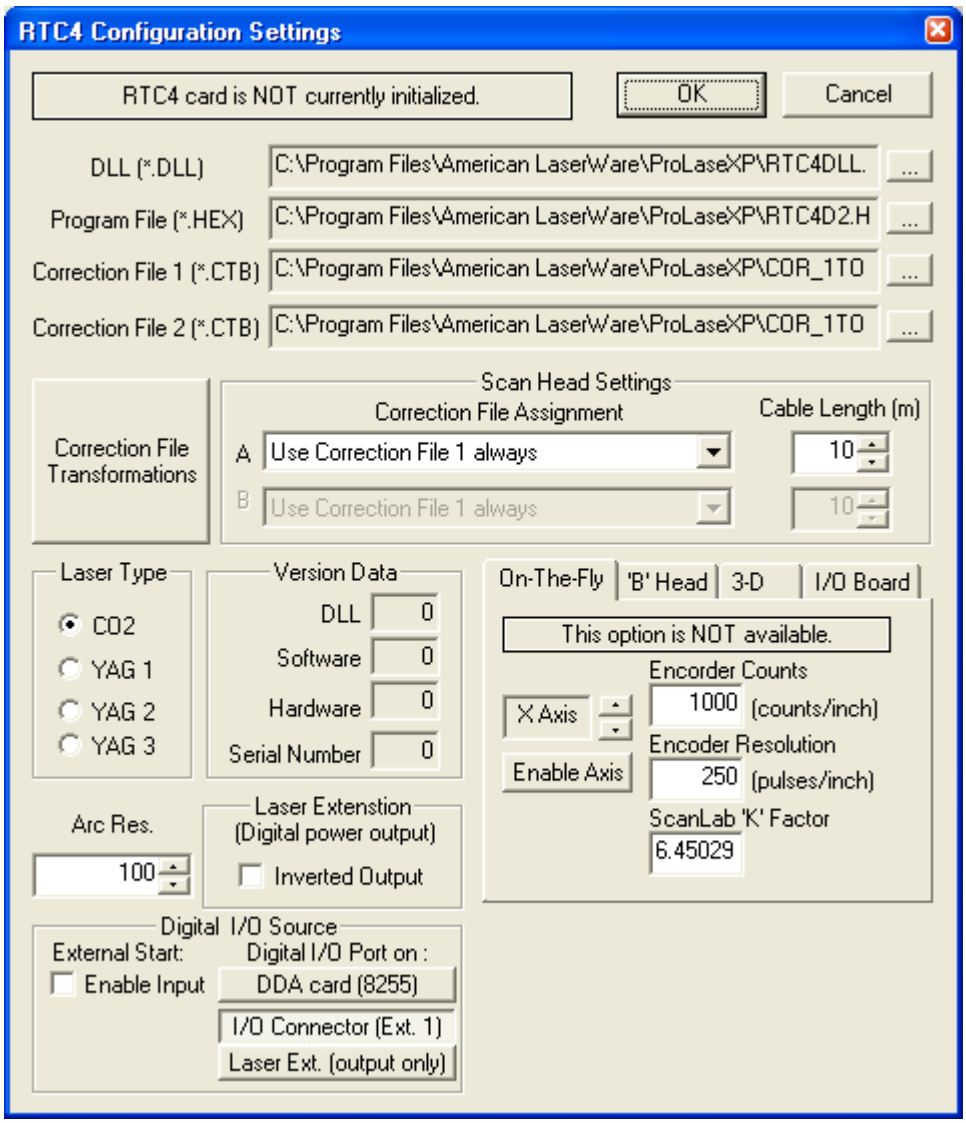
The RTC3 driver is required by ALL operating systems, not just NT. It is installed using the RTCSetup.exe program supplied with the RTC3 card and is fully described by SCANLAB.

The RTC3 does not have a minimum speed limit.

Laser output mode is completely software configurable. This should be set the laser type being used, but there is no longer a hardware setting that it must match. If you change your laser type (and associated optics), simply select the new type. The differences between the settings are described in the SCANLAB documentation.

The RTC3 Laser Extension connector can be used to provide digital power control signals. See the SCANLAB documentation for hardware settings of these signals. The RTC3 configuration dialog allows the user to 'negate' the logic level of these signals as needed for their particular laser hardware.

RTC4:



The RTC4 card is designed for the PCI bus and will be assigned an address by the computer automatically. There is no user programmable addressing required.

The RTC4 driver is required by ALL operating systems, not just NT. It is installed using the RTCSetup.exe program supplied with the RTC4 card and is fully described by SCANLAB.

The RTC4 does not have a minimum speed limit.

Laser output mode is completely software configurable. This should be set the laser type being used, but there is no longer a hardware setting that it must match. If you change your laser type (and associated optics), simply select the new type. The differences between the settings are described in the SCANLAB documentation.

The RTC4 Laser Extension connector can be used to provide digital power control signals. See the SCANLAB documentation for hardware settings of these signals. The RTC4 configuration dialog allows the user to 'negate' the logic level of these signals as needed for their particular laser hardware.

As used by ProLase, the RTC4 card is nearly identical to the RTC3 to provide backward compatibility to existing RTC3 systems. The differences include:

- Additional support for YAG 3 laser types. See SCANLAB documentation for more information.
- The Digital I/O Port selection is defaulted to 'I/O Connector (Ext. 1)'. Unlike the RTC3, the RTC4 supports the digital I/O without the I/O Extension card. Therefore, most new users should use the default, 'I/O Connector (Ext. 1)' port for digital I/O. This same port should also be used if the I/O Extension card is in fact used with the RTC4 (if for example the additional analog channels are desired). The selections for the alternate I/O ports remain for existing customers wishing to replace a damaged RTC3 with an RTC4.

Synrad FLCC:

ProLase supports the Synrad Fiber Link Controller Card (FLCC) for use with the Synrad *digital* marking heads (DH, FH or Fenix). Synrad customer's with SH or analog mode DH heads should use the DADIO or CIO-DDA06/12 card configuration (see above). The FLCC must be addressed to 330 hex. Digital I/O connections through the head should done in accordance with Synrad's instructions. ProLase only supports the 4 bits of digital output and 6 bits of digital input typically available on the FH / Fenix series heads, regardless of which head is actually used.

Special configuration features are available for many Synrad lens models to aid in field size calibration and enable the head's internal distortion correction. The lens selection and effects are discussed in more detail under Synrad FLCC in the configuration section of the manual.

Fiber Lasers:

ProLase supports pulsed fiber lasers with a hardware interface matching that of the IPG Photonics Corporation model pulsed lasers. (Laser Photonics fiber lasers also use this interface.) While operating at the same wavelength as YAG and Vanadate lasers, the control interface is different. Additional control signal are required for these lasers. A number of fiber laser manufactures and resellers exist. Check with your laser supplier to determine if your laser is compatible with the interfaces ProLase supports.

IPG:
IPG pulsed fiber lasers operate only at repetition rates between 20kHz-80kHz range. Additionally, a 'pre-amplifier' signal must be supplied prior to gating the laser and it must be removed shortly after the laser is gated off. The laser control interface also has an 8 bit digital port control input port and a latch input to accept the current power setting.

ProLase supports this IPG laser control interface will all interface options, both analog and digital. When using the RTC interfaces, the RTC card must support Laser Mode 4 and have general purpose

digital outputs. (The RTC5, RTC 4 and RTC ScanAlone 4 cards have this I/O built in.) Interfaces using off-the-shelf I/O cards (DDA06, DIO-48, etc) MUST use the CTR-05 counter timer card to control the laser as both external frequency generation and digital power control are needed.

To use the IPG lasers with ProLase, you must use the following settings in the configuration:

1. Enable the fiber laser signals with the 'Laser Photonics Fiber Laser Signal' checkbox on the Calibration, Laser Control page. This will limit the frequency output to 20-80kHz regardless of the setting in the material file or the object laser settings.
2. Activate the Laser Enable Signal on the Calibration, Laser Control page. This is used as the pre-amplifier signal. When the laser needs to be turned on, ProLase will activate the Laser Enable signal, then wait the 'Pre' delay before the laser gate signal is activated. After the gate is closed ProLase will wait a maximum of the 'Post' delay before the Laser Enable signal is cleared. If the gate is opened again before the 'Post' delay completes, it is reset until the gate closes again. If the Laser Enable signal is cleared, then ProLase will wait the 'Cycle' delay before it will again set the Laser Enable signal. Check the laser manuial for the magnitude and polarity of these signals.
3. If using RTC cards, in the RTC Configuration Settings window, change the Laser Type setting to 'Mode 4'. This setting allows the Laser 1 signal from the Laser Connector on the RTC card to be used as the free running (asynchronous) modulation input signal to select the pulse frequency.

Control wiring for the IPG fiber laser is as follows:

1. Gate signal as usual: ProLase output bit 0 (Gate) or bit 2 (NOT Gate) or LASERON signal from RTC card.
2. Laser Enable signal: Extended I/O Output 1 on Port C (bit 5) or output bit 1 with RTC card.
3. Modulation input as usual: CTR-05 pin 32 or LASER1 signal from RTC card.
4. Digital power control as usual: CTR-05 outputs 0-7 or L0-L7 on LaserExtension connector with RTC card.
5. Power latch signal: Extended I/O Output 0 on Port C (bit 4) or output bit 0 with RTC card.

Chapter 3 Basic software controls

Mouse Control:

ProLase uses the mouse like most other Windows based graphic programs. The mouse can be used to select from lists, menus, toolbar buttons, etc. The mouse can be used to 'drag and drop' objects in the drawing area. Simply place the mouse cursor over the current object (the cursor will become a four headed arrow), hold down the left mouse button and move. The mouse can also be used to manipulate the individual graphic objects via the graphic handles. The graphic handles allow the user to directly change the position, size, aspect, rotation angle and tilt angle of any object. Simply place the mouse cursor over a graphic handle (the cursor will change according to the handle selected), hold down the left mouse button and move the handle. The current object is drawn in red. A white dotted box, called a bounding box, surrounds it. The current object also has a series of colored dots around it. The black graphic dot is called the anchor point. This is the point that corresponds to the (X,Y) position of the object as displayed in the Sizing/Orientation property page. The object is drawn about the (X,Y) position depending on the justification. The blue dots are the graphic handles for stretching. Only the blue dots that allow the object to stretch without moving the anchor point are displayed. If the object is justified 'Center X, Center Y' then all of the graphic handles are shown since the anchor point is the center of the object. The red line with a red dot is a rotation graphic handle, used to rotate the object about the anchor point. The green line with green dot is the tilt angle graphic handle.

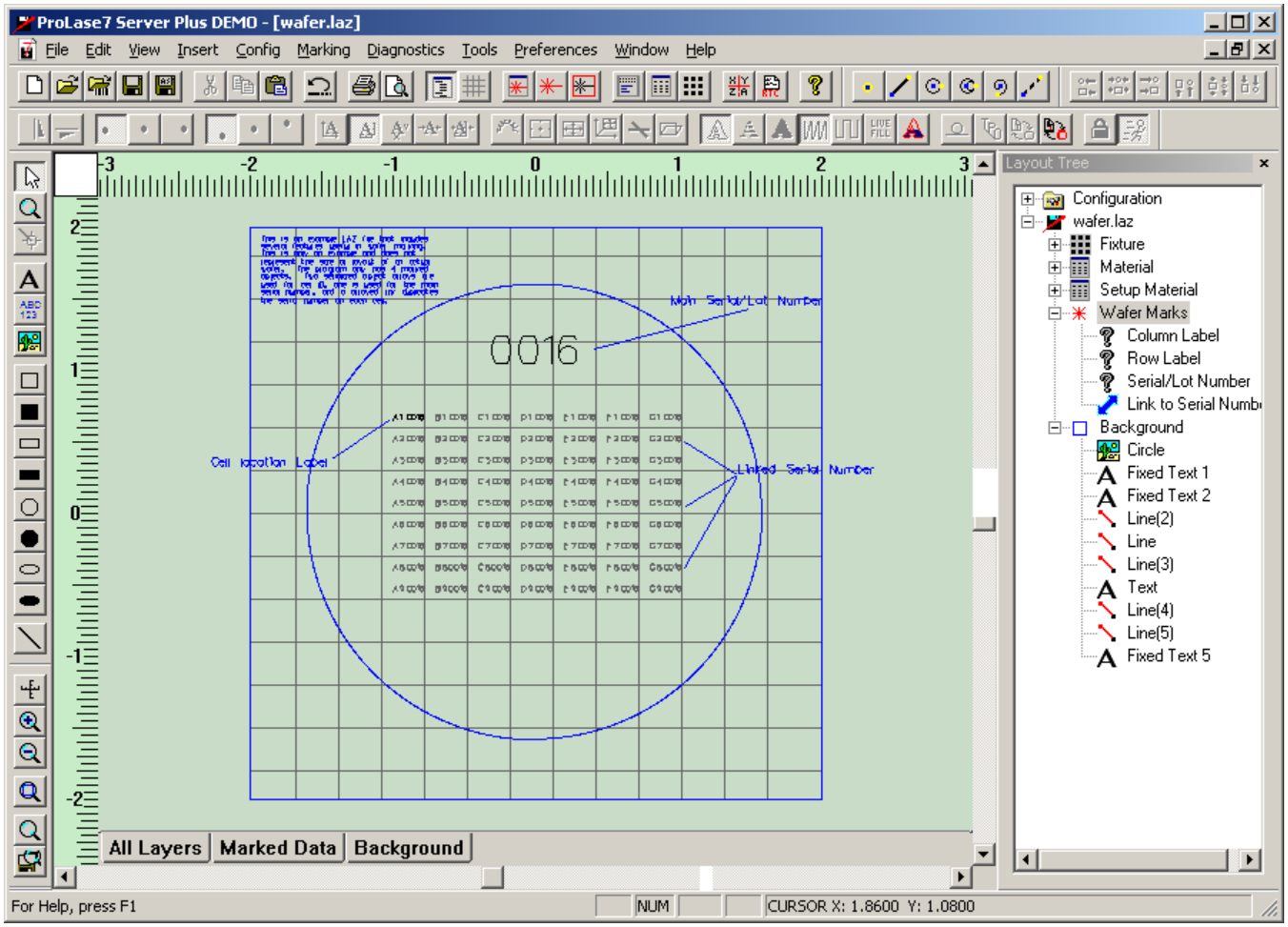
In ring mode, the bounding box is a ring and there is an additional rotation handle. A ring mode object can be rotated about the anchor point or about the center of the ring. The extra blue dot in ring mode is for adjusting the radius of the ring. Ring mode is used for marking around ring type parts like bearing races.

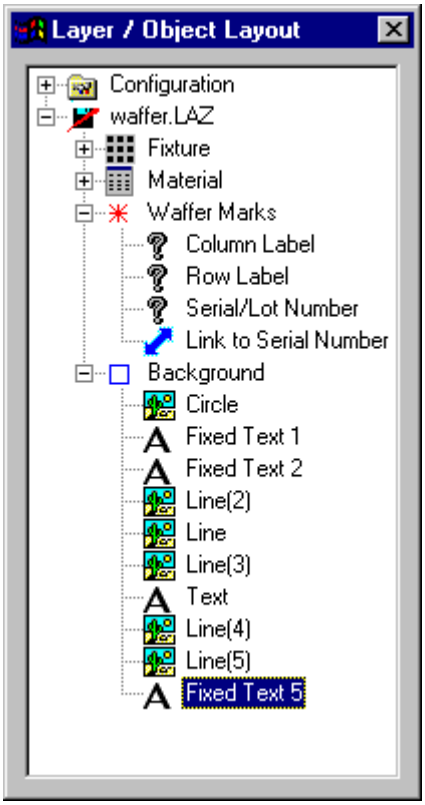
The mouse can also be used to group several objects together so they can be move, sized, rotated as a unit. Simply position the mouse to an extreme corner of the desired group, press and hold left button, drag mouse to opposite corner, forming a square that contains all the objects to be included in the group.

ProLase extensively uses the 'right click' mouse logic to give the user quick access to properties and power menus. For example, 'right click' on a blank area of the drawing window and a power menu allows you to create an object, or create a layer. 'Right click' on an object and directly access the objects properties. A tree structure (Layer/Object Layout) displays the names of all data used to control the marking process. Simply right click on any item within the tree structure for direct access to that item's properties.

Layer / Object Control:

Marked objects are grouped into a layer/object structure. Objects within a layer are marked together. Machine control and automation can occur between layers. An object can be a paragraph of fixed text, a line of variable text, one of our stock objects (circle, square, line, etc.) or an imported graphic. Each object has its own set of parameters that control position, size, character spacing, font, orientation, etc. Each layer can contain many objects so a marking program could consist of many paragraphs, and many different graphic files. ProLase only marks 'active' layers, leaving other layers to be used for operator instructions and information.





In this example:

"Wafer Marks" is the first layer, the laser spot indicates that the objects within this layer get marked.

"Column Label", "Row Label", "Serial/Lot Number", and "Link to Serial Number" are marked objects in the first layer.

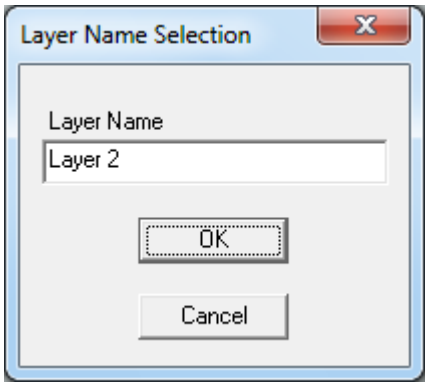
"Background" is the second layer, the blue box indicates that the objects in this layer are drawn on the screen, but do not get marked by the laser (background layer).

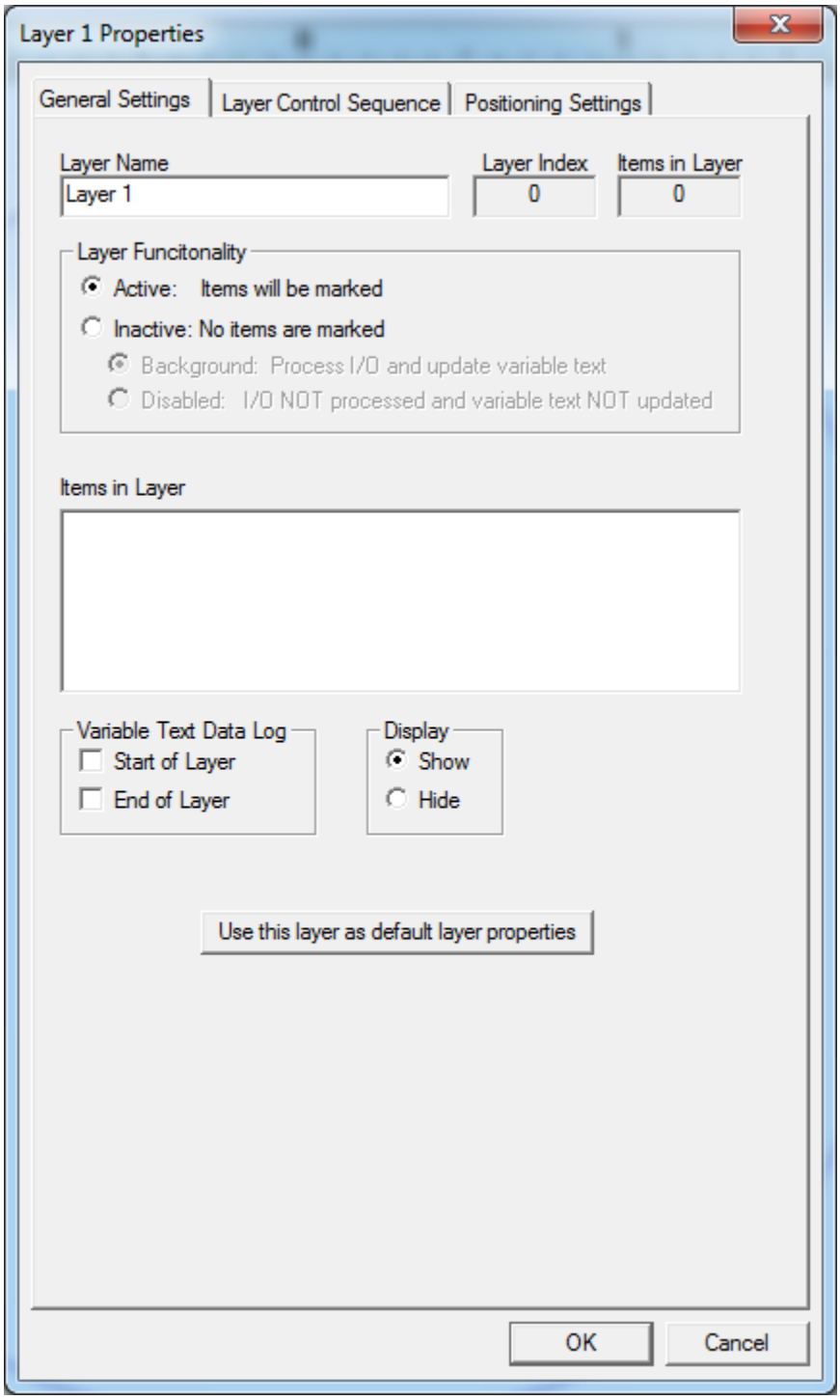
"Circle", "Fixed Text 1", etc. are the background objects on the second layer.

The current object is highlighted within the tree structure and drawn in red within the drawing area. The selected object is available for editing of mark properties such as position, size, and orientation. There are several ways to access the object properties including the INSERT menu, right clicking the object in the drawing area, or right clicking the name of the object within the Layer/Object Layout tree structure.

Layer Creation:

The Layer creation process can be initiated by right clicking in the Layer/Object Layout tree structure, or a blank section of the drawing area. The Insert Menu also allows for insertion of a layer. After selecting NEW LAYER, ProLase will prompt the operator for the name of the new layer. ProLase will insert a default name that can be changed. At anytime the operator can change the name of a layer by accessing and editing the Layer properties.





After creating the Layer, the operator should access the Layer properties and change as needed.

Layer General Settings:

The General Setting Property sheet is used to define the name of the Layer and to select ACTIVE or background. Additionally, it displays the total number of Objects in the layer and lists them all by name.

Layer I/O Settings:

The Layer I/O Settings property sheet is used to control external I/O for automation. Each Layer within a LAZ document can be configured to set TTL outputs and wait for TTL inputs.

This allows the user to configure interfaces between ProLase and parts handling equipment. The programmable interface is two-way, allowing for setting of outputs and waiting for input combinations before marking the next layer.

Sequential processing example:

A box is to be marked on four sides. A rotary indexer is used to present the four sides to the marking field. An output signal from ProLase is defined to be the 90-degree 'move' command to the indexer. An input is selected to be the 'in position' signal from the indexer, another input is selected to be the 'operator start sequence' signal. In ProLase, four layers are programmed, containing the objects to be marked on each of the sides. All four layers are defined by the user as active (to be marked) by use of the Layer properties *General Settings* Page. All four layers are programmed to mark on all conditions (*Conditional Mark* = XXXXXXXX). Layer 1 is programmed to start marking when the operator start button is pushed, Layers 2,3,4 are programmed to start marking when an 'in position' signal is received. This is accomplished by setting the *Start Mark Input* for each layer. All layers are programmed to send a 'move' signal after marking by setting the *Post Mark Output*. This marks all sides and returns the indexer to the 0-degree position before the next cycle. As an alternative to the 'in position' input, a delay (*Post Mark Delay*) can be programmed so that the system will wait a user defined time period between the 'move' signal and the marking. Many layers using up to six bits of output and eight bits of input can be programmed for advanced multi-step interfaces.

Layer I/O:

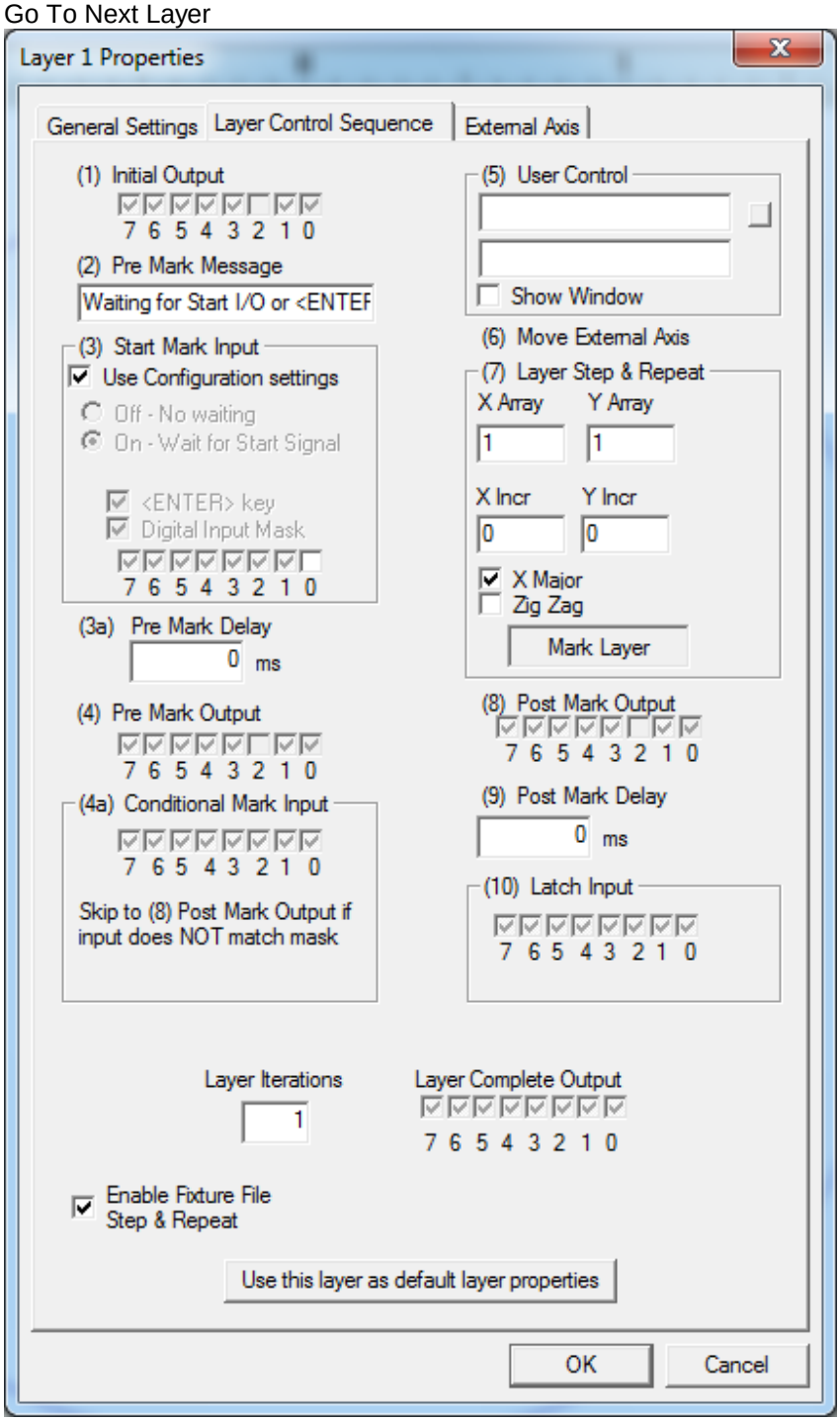
Layer IO provides a user programmable method of controlling digital I/O between program file layers. The output selections allow the user to set a bit pattern on the digital port. The input selection allows the user to program a bit pattern that the system will wait for before marking the layer. The sequencing of the I/O is the same as the order of the menu, and the sequence is repeated for each active layer. The *Layer Active* selection is used to program a layer as *Active* or *Background.* Each layer has its own set of programmed outputs and input.

Each item in the sequence is labeled with a number to indicate its order in the sequence. The simple flow listing below shows the time sequence of events.

I/O Sequence:

Start Layer
- Set *Init Output* on port
- Display Pre Mark Message for the operator
- Read Port until Port == *Wait Input*
- Set *Premark Output* on port
- Check Layer Conditional I/O (go to 8 (Post Mark Output) if NOT Condition)
- Call User Control (user EXE file)
- Move External Axis (Focus or ProLase Plus axis)
- Read Port If Port == Object Conditional I/O, then mark object
  - Next Object
- Step & Repeat Mark Layer
- Set *Postmark Output* on port

Wait *Postmark Delay* time
Wait for Input Latch (change in Input(s) specified
Repeat Layer x Interactions
Set Port to Layer Complete
Go To Next Layer



Selecting any of the digital I/O functions will present the user with a control used to define the bit settings. This control shows the number of each bit on the port and the programmed setting. The mouse is used to select the bit to program. The mouse button is used to toggle the possible settings. There are three possible settings, a black check (1 or bit on), an empty box (0 or bit off), or a grayed out check (X or don't care). The grayed out check means leave that bit unchanged when setting the outputs, and it means 'don't care' or ignore when checking inputs. For example: an output like XXXX1001 will set bits 0,3 to one, 1,2 to zero, and leave 4,5,6,7 unchanged. Programmed as an input XXXX1001 would mean that the system would keep reading the port until bits 0,3 equal 1 and bits 2,3 equal zero, then mark the part. The input routine would ignore the port values of bits 4,5,6,7.

User Control:

The User Control feature allows the user to extend the capabilities of ProLase by providing a user created BAT or EXE function to provide custom communication or parts handler control. Every time the layer is processed the user defined BAT or EXE file is called by ProLase in a synchronous manner. This means that the user program is executed to completion before ProLase continues to the next step (actual marking). The user provided program can be used to communicate with various interfaces (network connections, RS232, etc.), get data to be marked from the interface and create files to provide ProLase variable text objects with the data. (see Variable Text type, Dynamic File). User Control programs can create any number of text files used by Variable Text. Name, address, and phone number, for example could be marked as three objects, with one User Control call to create all three Dynamic Files. The User Control can also be used to move external motion systems for part placement. If used in this way it is important that all motion is completed before the User Control program quits, otherwise ProLase will begin marking while the part is still moving.

The User Control has three properties: the name of the .BAT or .EXE file to be executed, any desired command line parameters, and whether or not the user wishes the program to have its own viewing window. The first edit box is for the name of the program and includes a browse button for easy navigation. The second edit box is for any desired command line parameters. The Show Window check box is to indicate that the program should have a viewing window. The viewing window is a simple DOS box for a .BAT file. Visual Basic, C++, etc. type programs can have full blown user interfaces. For simple .BAT file control the Show Window feature is a good way to debug the interface. Simply enable Show Window while testing the interface, and disable during production. Note: All batch files used by ProLase must have their CLOSE ON EXIT property set. To set this property in Windows use explorer or My Computer to find the icon for the .BAT file. Right click on the icon, and select Properties from the menu. Go to the Program tab. Make sure "Close on exit" is checked.

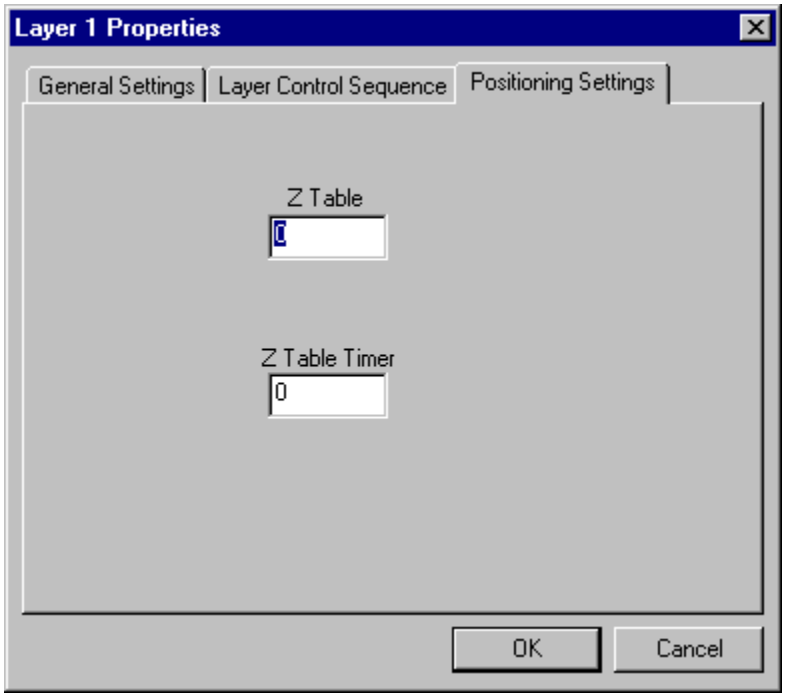
Digital I/O Default Settings:

The *Init Output* is set to XXX1XXXX (bit 4 on = READY). The *Wait Input* is programmed as XXXXXXX1 (bit 0 on = START). The Object C*onditional Mark* is programmed as XXXXXXXX for every object meaning all objects will always be marked. The *Premark Output* is set to XXX0XXXX (bit 4 off = BUSY or NOT READY signal). The *Postmark Output* is set to XXX0XXXX (bit 4 off). When layer 1 is finished, the whole thing starts again.

The *Abort Output* is the Output setting that will be applied when a run is aborted. This becomes the "standby" state of the output port when ProLase is not in run mode.

Positioning Settings:

The positioning settings page is used to set the Z Table (External focus) for each layer. This allows

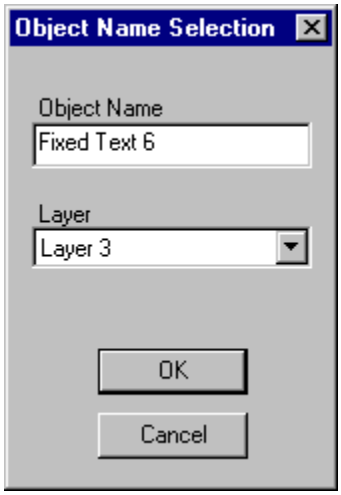


for the marking of “Stepped” parts. Z Table Timer is a delay time used to insure that the Z axis has completed its motion before the system begins to mark.

Object Creation:

The Object creation process can be initiated by right clicking in the Layer/Object Layout tree structure, or a blank section of the drawing area. The Insert Menu also allows for insertion of an object. After selecting NEW Object, ProLase will prompt the operator for the type of object to create: fixed text, variable text, graphic, link, or empty. Next, the system will prompt for the name of the new object. ProLase will insert a default name that can be changed. At any time, the operator can change the

name of an Object by accessing and editing the Object properties.



After creating the Object, the operator should access the Object properties and edit as desired. At the very least, the operator should enter the desired text or graphic file name. After that the operator can continue editing the property sheets or can enter OK to the property pages and start using the mouse drag and drop, or toolbar buttons to manipulate the object.

Default Object Properties:

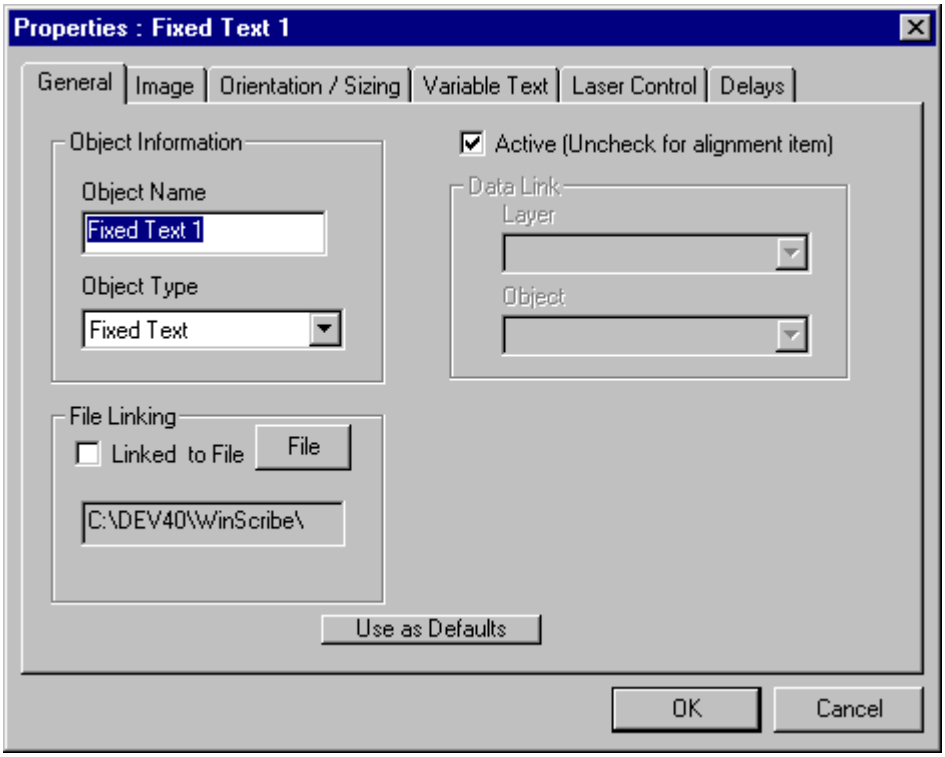
ProLase allows the user or OEM to set the default values for each object type. This allows the user to decide what set of values should be used to create a new object. To use this feature simply create a new object and set the properties to whatever values you want that particular object type to start with. For example if you want new graphic objects to be created with the Aspect Mode property set to NATURAL, you simply create a graphic object and set the Aspect mode to NATURAL. Then go to the General Property sheet and press the USE AS DEFAULTS button. From that point on all newly created graphic objects will be set to NATURAL Aspect Mode. This feature can be used to set default sizes, fonts, positions, laser settings and all other object properties. Each object type (FIXTED TEXT, GRAPHIC, etc.) has its own set of default properties. One of the best things about this feature is the ability to define default folders for each object type. For example Fixture files, Material Files, Fonts, and Graphics can be organized in different folders, and the defaults can be used to set the locations of these folders. The default information is stored in the Preferences file (DEFAULT.PRE). If DEFAULT.PRE is deleted, ProLase will set all defaults to the original factory settings.

General Object Properties:

The ACTIVE check box allows a user to define an object as a data object to be marked on the part (ACTIVE checked) or as an alignment object (ACTIVE unchecked). An alignment object can be used in the MARK DIALOG control to show where particular alignment points will occur on the part. LIGHTSHOW mode and SETUP are often used with alignment objects to align a part correctly before actual marking. Alignment objects can be bounding boxes around the objects, a box of representing the lens field or a point on the part such as a corner or a outline graphic of the part itself. Any type of ProLase object can be used as an Alignment object. Alignment objects are only marked if the ALIGN

check box is checked in the MARK DIALOG control, otherwise actual data objects (ACTIVE objects) are marked.

The General Property sheet also allows the user to define the name and type of Object. There are five possible Object types:



Fixed Text is a paragraph of fixed text. Fixed text is text that is known at the time the object is created and will not change between marks. Fixed Text objects can have up to 10 lines (each up to 255 characters long) in a paragraph. Multiple Fixed text Objects can be used to create paragraphs longer than 10 lines.
Text Plus
A paragraph of text can optionally include imbedded text from other Fixed Text or Variable Text objects using the Text Plus feature (see Text Plus below).

Variable Text is a line of variable Text. Variable text is text that is not specifically known at the time the object is created and could change between marks. There are five types of Variable Text:

Keyboard variable text is used when the operator can type in the variable text while marking.

Date Code variable text is a code based on the PC date and/or time. Date Codes will be automatically computed on every mark.

Code Reader variable text is text received from an external source such as a bar code reader

or the RS232 port.

Serial Number variable text is a sequence of alpha and/or numeric text that is serialized on each mark. The Serial increment can be any positive or negative number.

List File variable text comes from a disk file such as a mailing list. The file must consist of a list of ASCII text with each item in the list separated by a carriage return/line feed pair. The next item in the list will be used on each mark. Multiple Objects can use the same list file.

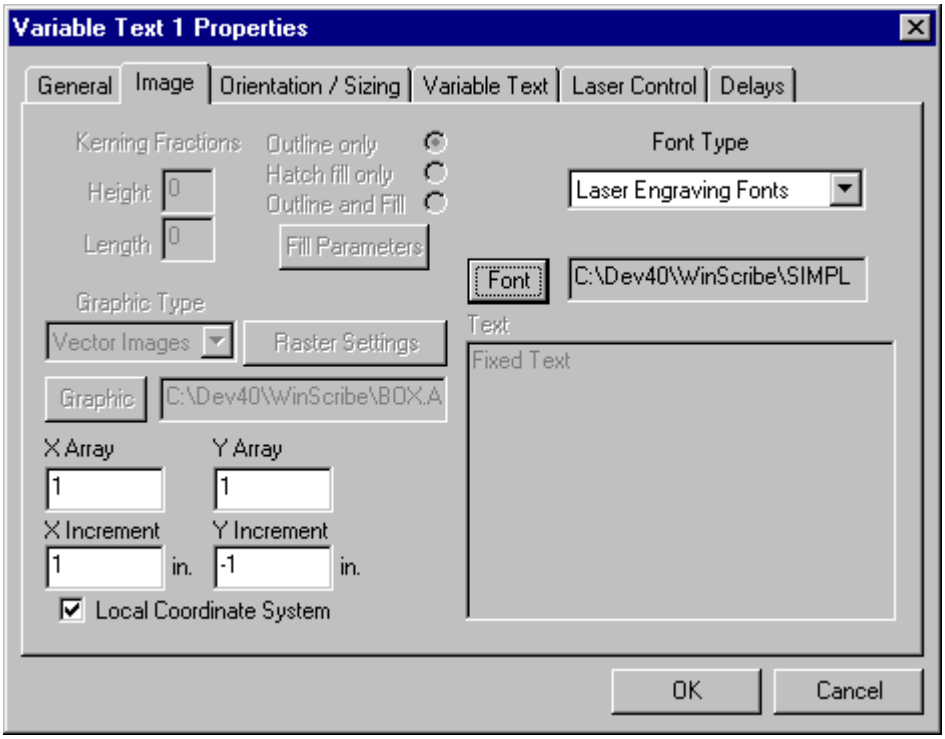
Graphic is a graphic or logo imported from one of the supported graphic formats. There are two basic types of Graphic objects: vector and raster.

Data Link is an object that marks the text or graphic defined by another object. The link object can specify a size, position, angle, font, etc. for marking, but the data being marked comes from the object linked to. An example use of this is printing human readable text below a serialized bar code. The first object would be the serial number, variable text object using a bar code font. The second, Data Link object would be linked to the first, but would use a human readable font. As the first object serializes the second object will display the same number.

Empty is object is used as a simple placeholder in the Layer's list of items.

Image Object Properties:

The Image property sheet allows the user to select characteristics about the to be marked.



Different aspects of this page are available for selection based on the type of Object.

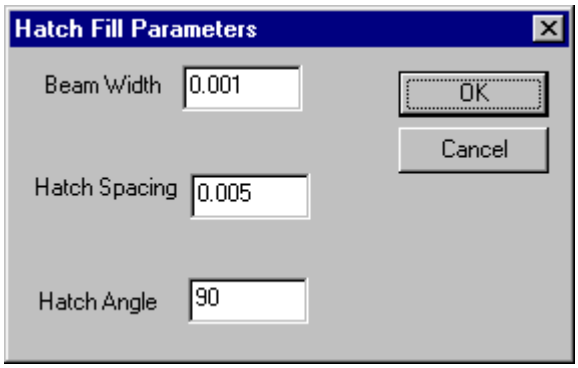
Text Objects:

For Text Objects, this page is used to select the font type, and font properties. There are three types of fonts including Laser Engraving Fonts, True Type Fonts, and Bar Code Fonts. Laser Engraving fonts are usually single line without fills. These fonts can be marked very fast and at small sizes. True Type Fonts are vector outlines that can be filled by ProLase. Bar Code Fonts can also be filled by ProLase.

True Type Fonts and Hatch Fills:

When True Type Fonts are selected several additional properties become available for selection. Kerning fractions allow the user to adjust the spacing between characters. The Height parameter sets the character spacing to a number proportional to the height of the character. The number is a percentage, so 5 means that the character spacing will be 5% of the text height. If only height is specified then the character spacing will be the same for all characters. Length adds a number to the character spacing that is proportional to the width of each character. For example if 5% is selected for length then the spacing around wide characters like “W” will be more than the space for narrow characters like the letter “I”. In general, most True Type Fonts look good with a height adjustment around 5 and a length adjustment of zero.

True Type Fonts can be hatch filled by ProLase according to the operator specified hatch properties. Hatch properties include hatch spacing (in the selected distance units), hatch angle (in degrees), and kerf compensation (beam size in distance units). Kerf compensation means ProLase will only fill to within one beam width of the original outline. This keeps the fill from overlapping the outline when both fill and outline are being marked. When the user is manipulating a filled object with the graphic handles, the fill vectors are not displayed. The operator can command the system to display the fill by using the VIEW/RENDER FILLS menu selection. This forces ProLase to calculate and display the fill using the current object data. ProLase automatically calculates the fills based on the current size, aspect etc., insuring the fill density remains constant. From the Image property sheet the user can select OUTLINE ONLY, FILL ONLY, or BOTH when marking True Type Fonts.



Bar Code Fonts:

When Bar Code Fonts is selected the user can access the type of bar code and the Bar Code Parameters by pressing the FONT button. From Bar Code Parameters, the user can select the desired Bar Code format from the drop list. ProLase currently supports 25 of the most common standardized

linear bar code formats, the Data Matrix 2-D format, plus a customizable user-defined format. The DETAILS button can be used to see what kind of data is valid for the selected format. Some formats only accept numbers, others only numbers and uppercase letters, some only pairs of numbers, and still others accept all ASCII characters. If the current text is invalid for the current bar code font, ProLase will display an error and the drawing area will display a box with an X in it. The REVERSE check box is used to create reverse contrast image bar codes (white marks on a dark background). The CHECKSUM check box is used to include a checksum digit in those formats that can use this feature.

The Data Matrix bar code type specifies the public domain 2-D matrix symbology from RVSI Acuity CiMatrix. The BORDER and MATRIX SIZE controls are only available with the Data Matrix bar code type. A Data Matrix symbol can encode up to 3116 numeric or 2335 alphanumeric characters. Any ECC200 standard size can be selected, from 10x10 to 144x144, including 6 rectangular formats. The Smallest size selection will have ProLase create the symbol with the fewest cells that can encode the given data. This will always yield a square matrix.

User Defined is the final bar code type. It can be used to have ProLase render any 2-D public or proprietary format. Pressing the DETAILS button with the User Defined type selected will display the User Defined Bar Code Properties window. These parameters are specified on an object basis and can be saved using the object default settings. They are used as follows:

Name: This is a text description of the format for the benefit of the user. This label will be appended to the User Defined name in the format list. For example, if the name is “My new format”, then the drop list of available bar code formats will end with “User Defined – My new format”.

Encoding Program: This is the fully qualified path name of the executable program capable of encoding the data and generating the proper output file. This file shall accept a single command line argument of an input file name and must output a single data file. Each of these files is an ASCII text file with data separated by CRLF as described below. Acquisition, use and maintenance of the Encoding Program is the responsibility of the user. The call to the encoding program will be made as

EncodingProgramPathName “Input file name”

ProLase will add the quotes to the command line parameter to ensure that the entire path name of the input file will be treated as a single argument despite any spaces.

Input File: This is the name (including full path) of the ASCII text file that will be generated by ProLase for use as input to the Encoding Program. The data is CRLF delimited and is in the following format:

1) Version number, currently 0
2) Output file name
3) Number of characters to be encoded
4) Data to be encoded. This will be either all text of fixed text objects or the first line (variable data) of variable text objects.
5) Active additional parameters (up to 5) in increasing order.

Output File: This is the name (including full path) of the ASCII text file that will be generated by the Encoding Program for use by ProLase. The data is CRLF delimited and is in the following format:

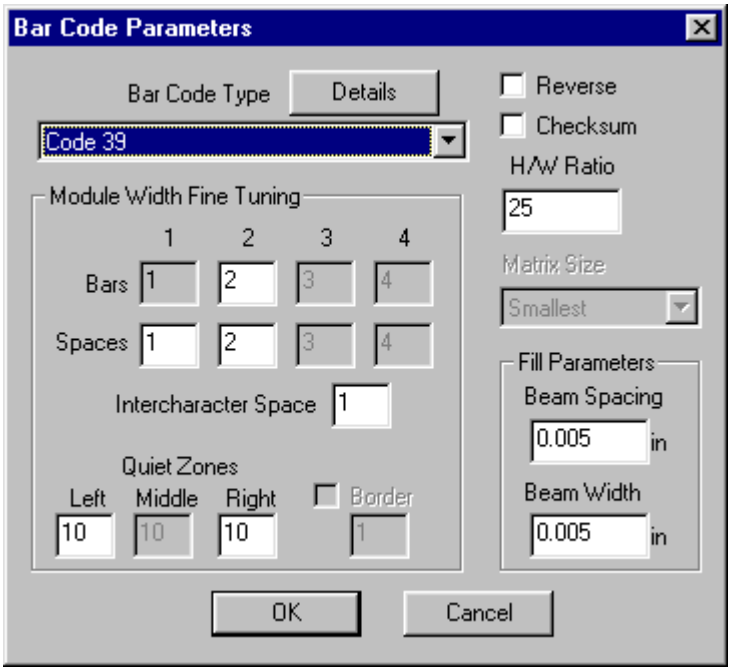
1) Version number, currently 0
2) Number of rows of data, R
3) Number of columns of data, C
4) Data. Each module (cell) of the encoded data is represented by a 1 or 0 digit indicating either black or white mark in normal (non-reversed) printing. There MUST be R CRLF delimited lines of C data per line.

Additional Parameters: These are additional parameters that can be used to specify additional settings that the encoding program may support. Note that normal ProLase bar code parameters still apply and ProLase will handle the black-on-white or white-on-black setting, kerfing, fills, and sizing/orientation geometry. The Encoding

Program using the Additional Parameters must handle other settings. Examples of such parameters include specifying the dimensions of the resulting code or a particular encoding scheme or data redundancy level. Up to 5 parameters may be specified with each parameter activated or deactivated as desired. When fewer than 5 parameters are activated, the input file will list only the active ones in numerical order.

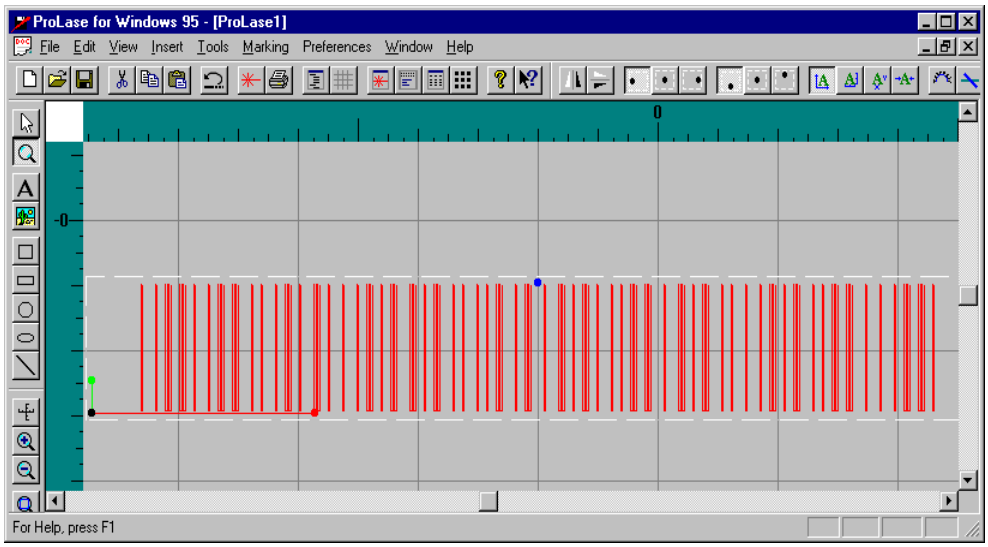
Included in the installation folder of ProLase is a sample of an encoding program named 'GenCode.exe'. This program works as described above to generate a DataMatrix representation of the specified text. Using GenCode.exe will produce the same marks as the DataMatrix bar code type with the 'Smallest' size selected. GenCode.exe does not accept any additional parameters.

NOTE: *Please have patience when using larger 2-D codes. These codes can contain many thousands of vectors (144x144 has 20736 elements). Even on fast computers, this will take considerable time to draw or mark.*



Tuning Bar Codes:

The Module Width Fine Tuning is used to control the ratio between different elements in a bar code. Most bar codes are made up of bars and spaces of various widths. Common bar codes have two widths, narrow and wide, where the ratio between narrow and wide is about 2. Bar Code 128 uses four widths, all based on a ratio to the narrowest width. In ProLase the narrowest width bar is the basic unit, all other bars and spaces are defined in terms of a ratio to the narrow bar. In other words, the



narrow bar always has a ratio of 1:1. Normally the narrow space also has a ratio of 1:1 since it should be the same width as the narrow bar. However, since the laser mark has a finite width to it, the bars tend to 'bleed over' into the adjacent space area. When the marked part is carefully inspected, the spaces are narrower than the narrow bar. One way to compensate for this 'bleed over' effect is to vary the narrow space ratio (for example 1.095:1) until the narrow space width marks equal to the narrow bar width. Typically, this value is doubled to get the wide space ratio. Optimally, the wide bar will have a ratio of 2.000:1. Each of the ratios can be adjusted until the desired effect is achieved. Another way to compensate for the 'bleed over' effect is to tell ProLase the width of the laser beam mark across the material. This parameter (Beam Width) is used by ProLase to do kerf compensation, similar to the hatch fill feature. The bars are created by a series of parallel, vertical lines spaced according to the user supplied Beam Spacing value. ProLase will adjust the beam spacing slightly to force a consistent number of laser passes per bar, all exactly evenly spaced. This prevents inconsistent pass spacing due to rounding errors, present on many laser-marking systems. The H/W Ratio is the default Height to Width ratio for the bar code. ProLase uses this default ratio when the object is in "Natural Aspect" mode.

Rasterized Bar Codes

If either Data Matrix or User Defined bar code format is selected, the fill method can be specified. Lines indicates each cell is filled by a series of vertical lines. Circles fills using a series of concentric circles. Raster will result marking the bar code as a raster image using the settings found in the Raster Parameters window. When Raster is selected, the Single Dot Per Cell checkbox becomes available. The Single Dot Per Cell setting will result in the raster bar code being marked as small as possible, with only a single laser pulse for each cell in the code. The raster dot density is based on the beam width parameter.

In the experience of ALI, it has been found that the fill method generally has little effect on the readability of the resulting code. However, it can have a significant on marking times. In some cases, vector filled bar codes mark faster. In others, rasterized codes are better. Because the time is highly dependent on the size of the code and the material on which it is marked, the user should experiment with all the fill methods and laser settings to determine the combination that yields the best performance.

Text Plus

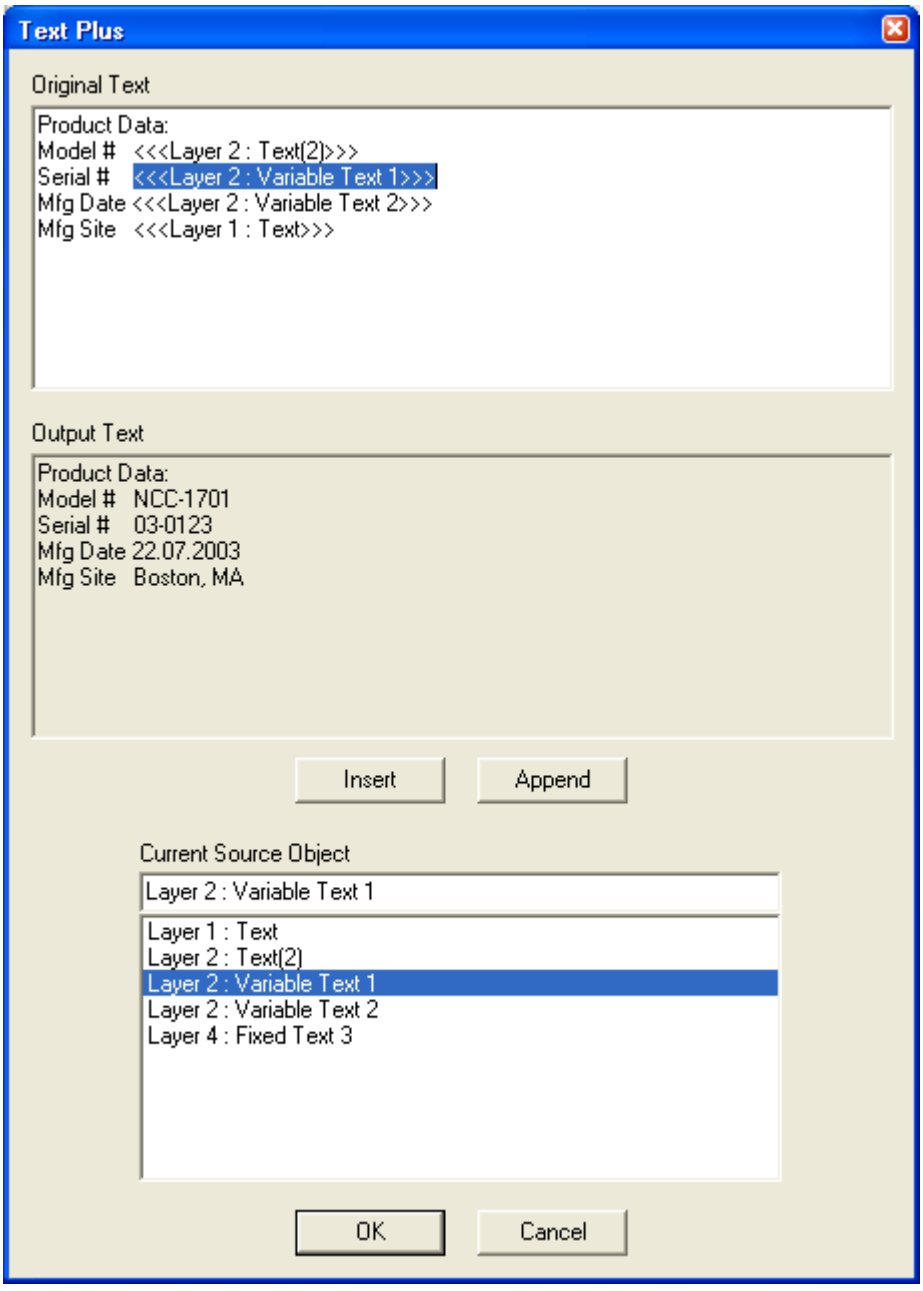
All Fixed Text type text objects automatically have the Text Plus feature enabled. This feature allow for the embedding of text from other text objects into the fixed text of the current object. Using this feature, a single complex text paragraph can be constructed from a number of separate parts including serial numbers or date codes. This can be especially useful if the text will be marked using a high density, 2-D bar codes, such as Data Matrix, where the desired result is a single large code containing 3 lines of text rather than 3 smaller codes with only a single line each.

The Text Plus embedding is accomplished simply by including an insertion code anywhere into the fixed text of the current object. The current text of a Variable Text object or the first line of a Fixed Text object will be replace the insertion code in the current object's text. The insertion code has the following form:

<<<*Layer Name* : *Object Name*>>>

Where *Layer Name* and *Object Name* are the names for the layer and object, respectively, of the Fixed Text or Variable Text object that is the source of the embedded text. The insertion code must EXACTLY MATCH the desired source object information and format, including the 'space, colon, space' separator. If no EXACT MATCH is found, the characters are marked as normal text.

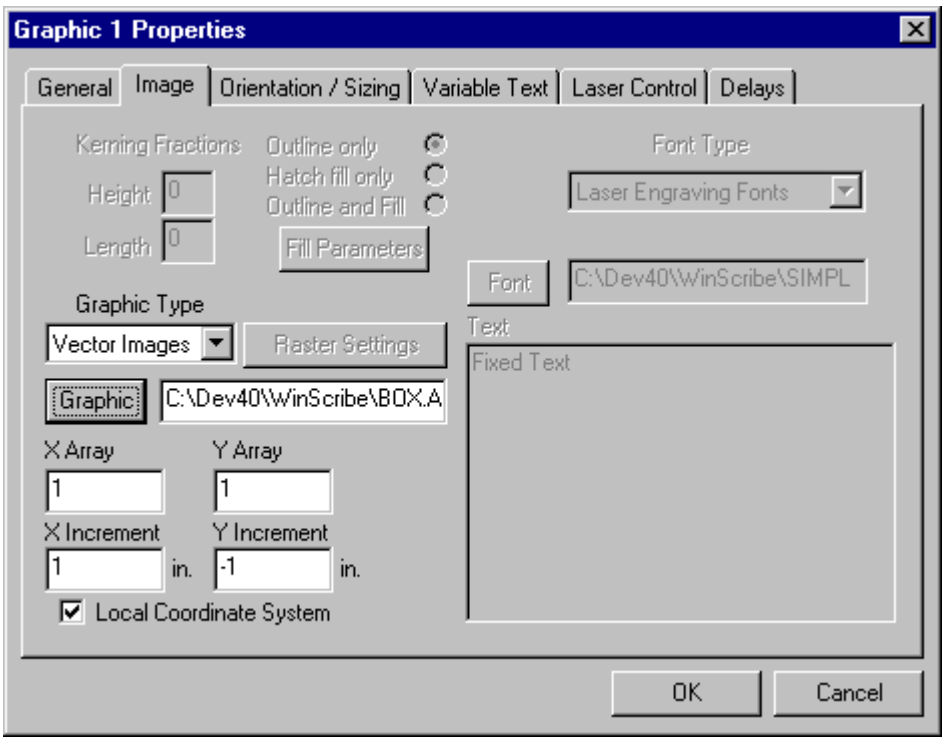
The *Text Plus* window provides a helpful tool for creating correct insertion codes. The text of the object can be manipulated in the *Original Text* region. Below this, in the *Output Text* region, the current text is displayed as it would currently be marked. Near the bottom, the *Current Source Object* list itemizes all the currently valid objects that may be used as a source of data. When an insertion code is selected in the text box, that object is highlighted. Changing the list selection will modify the insertion code. The



Insert and *Append* buttons replace the current selected text with an insertion code to the first source object in the list.

NOTE: The Text Plus feature is very powerful, but also very demanding. The insertion code must EXACTLY MATCH a valid source object otherwise those characters will be marked exactly as they

read. Additionally, only the current value of variable text is used. Variations in the Variable Text object's parameters can change the frequency at which the text is updated. Care must be taken to ensure that the update is accomplished prior to the Text Plus object being marked. Finally, the insertion codes in the Text Plus object are normal fixed text that may, or may not, be substituted with text from another object. Insertion, deletion, or re-ordering of any objects and layers may invalidate any and all insertion codes leading to undesired text being marked. Therefore it is highly recommended that Text Plus objects only be defined at the end of document design (perhaps even on a separate layer to ensure that no variable text will be modified) and that Simulated marking is performed to verify that the resulting text gets marked as desired.



Graphic Objects:

For Graphic Objects, this page is used to select the graphic file. ProLase support two types of Graphic Objects: vector and raster. ProLase loads the required information from graphic files as needed. If the graphic has already been loaded once during a session and the user creates a new LAZ file that uses the graphic, the graphic is not re-loaded since it already exists in the ProLase graphic memory structure. This structure keeps track of all graphics and fonts during a session by keeping a list of the names. This means the user should keep graphic file names unique. ProLase performs all graphic loading automatically. Whenever the user opens a LAZ file, ProLase loads all required graphics and fonts not already in graphic memory. In many cases, ProLase produces a temporary version of a graphics file in a ProLase friendly format. This allows ProLase to load such a file faster than re-importing from the original source. If a file is not already loaded in graphic memory, ProLase checks the date on the original file to see if it has changed since ProLase made the temporary version. If the date of the original file has changed ProLase re-imports the file and recreates the temporary file. This insures that any changes made to the graphic file will be reflected in the laser marks.

The GRAPHIC TYPE drop list control allows the user to change between vector and raster graphics. The GRAPHIC button that allows selection of the graphic file to be use will only list files with type extensions that correspond to the chosen graphic type. Selecting *Raster* as the graphic type enables the RASTER SETTINGS button that shows a window displaying all the raster specific parameters. New graphic objects are set to the *Vector* graphic type by default.

Vector graphics consists of a series of graphic primitives: lines, positions (lines with the laser off), and circular arcs. Sources for vector graphics are many including CAD, artwork and other drawing software. ProLase supports 23 of the most common vector formats. Some formats generate Arc commands while others simply form approximate Arcs using PolyLines. ProLase performs true arc generation, so it is better to use formats that support circular arcs whenever possible. One of the most popular formats, DXF/DWG, does not explode its text into lines and arcs, therefor these formats can not be used if the drawing includes text. HPGL files explode everything into lines and positions, but does not produce arcs. Other formats, including Adobe Illustrator or Encapsulated PostScript, explode text into the primitives and support Arcs. The user should try several of the formats and select whatever works best for the particular application.

ProLase uses dynamic scaling which means graphics are drawn at whatever size the user enters in ProLase. Scaling and size information from the drawing program have no meaning in ProLase. For example, the user may define a square in AutoCAD that is 1 mile on each side. Once imported into ProLase this square will be drawn at whatever size the user specifies in the Object property pages.

Raster graphics, or bitmaps, are made up of many rows of tiny dots or pixels (picture elements). All pixels have the same size and have 3 unique properties: raster (or row), column, and color. Raster graphic sources include artwork, drawing and scanner/digital camera software. ProLase supports 12 of the most common raster formats.

Pressing the RASTER SETTINGS button allows the user to control the way ProLase modifies the source raster image for marking. Laser markers cannot mark in color, they make monochrome images. ProLase can mark bitmaps in one of three possible Render Modes: B/W, Connect the Dots, or Grayscale.

B/W:

ProLase must convert the image from the source color depth (the number of colors available) to B/W. ProLase must also scale the image to the size the user has specified. These two processes are accomplished by dithering. ProLase supports 3 common dithering algorithms: Bayer, Burkes, and Floyd-Stienberg. The *Dither Scheme* property allows the user to specify which dither algorithm to use. The quality of the resulting image for each algorithm depends strongly on the subject matter (portraits, landscapes, logos, etc.). The user should experiment to decide which method works best for a particular image. Dithering means that grays are simulated by dot patterns.

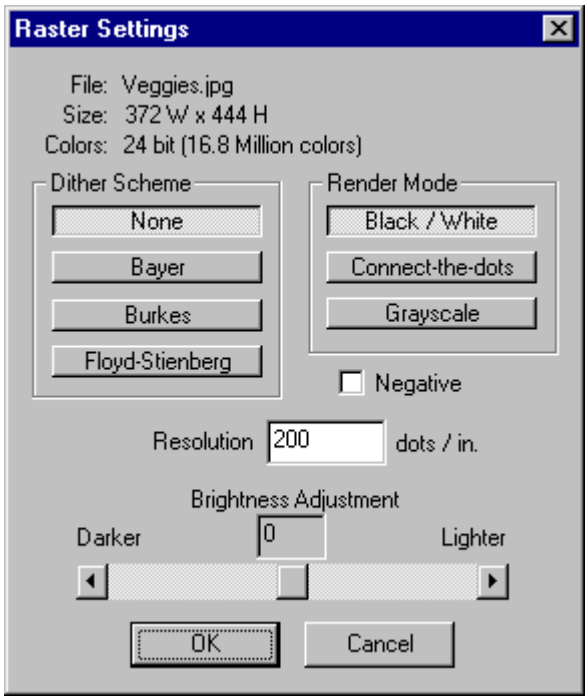
Connect the dots:

Connect the dots means that ProLase will draw vector lines for a series of “on” dots. This mode is for systems where ProLase can not control the individual laser pulses. Examples of these kinds of systems would be any kind of laser that only uses GATE control instead of pulse control, such as some older Nd:YAG lasers or the Synrad FLCC card where the interface controls the PWM and ProLase only sends a GATE. This mode should be used for any laser run in CW mode.

Grayscale:

This is the most advanced bit map control where grays are accomplished by varying the laser power for every dot in the bitmap. This mode only works on lasers with high speed power control such as PWM controlled CO2 lasers and diode pumped Nd:YAG lasers. (Currently American Laserware does not have a diode pumped YAG so this technique has not been tested on these lasers. In the near future we expect to acquire such a laser. The technique has been tested on a PWM controlled CO2 laser with excellent results.) This mode should never be used on flashlamp pumped Nd:YAG lasers since the power supplies and flashlamps take a long time to actually effect the laser power. In general flashlamp pumped, Q-Switched YAG lasers should use the B/W mode with dithering to simulate grays. The Grayscale mode produces high quality images at lower resolutions compared to the B/W mode, which means the speed of producing an image is much higher.

The overall size of an image is determined by the ProLase editor. The *Resolution* property allows the user to set the resolution for the resulting image following the dither modifications. This value is given



in ‘dots per unit’, where the unit is the base units the user specified in the Configuration (see Config Units). If inches are selected, then the resolution is in ‘dots per inch’ or DPI. The total number of dots is the product of the graphic size and resolution. A 2” wide x 1.5” tall graphic with 200 DPI resolution will be marked with 300 raster lines, each 400 across. The best results are achieved when the resolution corresponds to the laser beam width. Thus, a system with a 5 mil spot size should use a resolution close to 200 DPI. The *resolution* only applies to the marked image. Because the screen resolution is much less than the DAC/galvo resolution (typically 1024 x 768 versus 4095 x 4095), a different method is used to display the image to the screen. For this reason, *Mark Simulate* shows a much darker image than will be marked. It should be remembered that the screen images are only to give the user feedback for general object properties such as position and size.

The *Negative* checkbox inverts the colors, producing a photographic negative. This is useful when

making light marks in darker materials. *Brightness Adjustment* alters the overall brightness level of the image to improve the image quality. For example, 'washed out' marks can be darkened.

A Note about Editing Raster Images

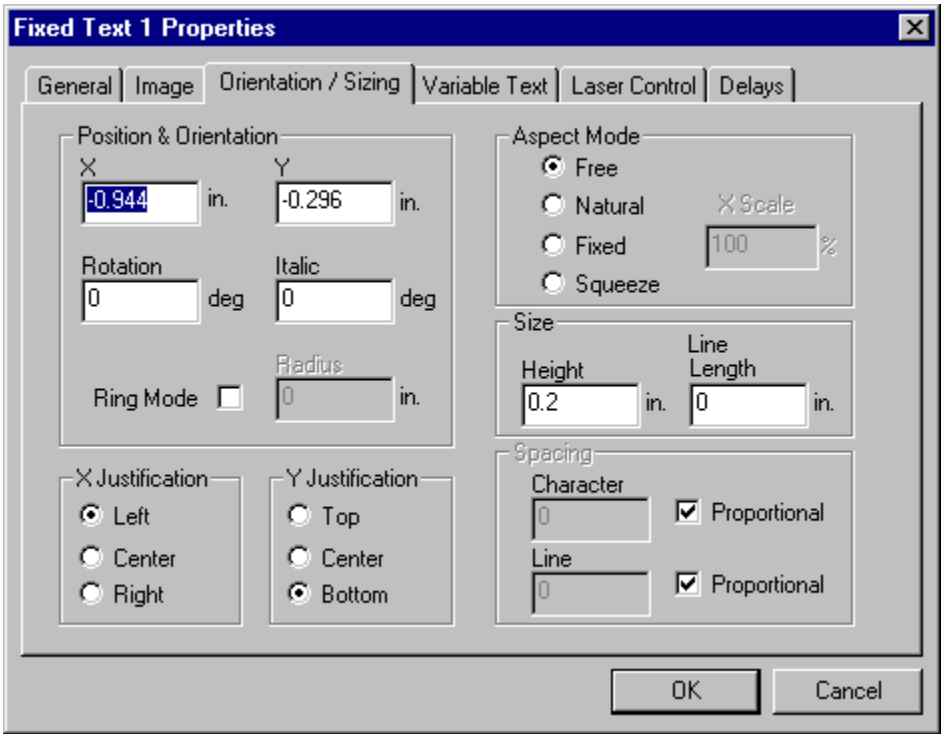
ProLase is NOT an image editor. Because marking raster images is a very complicated issue, ProLase provides a limited ability to modify the source image as a convenience only. ProLase quickly produces high quality marks from high-contrast, simple, block images. Portraits and other high fidelity images often require considerable time to achieve perfect results. Scanned images often have artifacts that should be removed. This MUST be done outside of ProLase. In general, complicated raster images should first be modified by professional image editing software.

A Technical Note about Marking Raster Images

ProLase always marks a raster image from bottom to top in the reference frame of the mark field. ProLase positions the laser at each pixel for a short time. When ProLase has direct control over the laser pulsing, ProLase allows exactly one laser pulse for each pixel. Because of this, the marking speed is governed by the laser frequency and the speed parameter is ignored. Thus, a Q-switched Nd:YAG at 5kHz will mark a 100 pixel raster in 20ms. Additionally, to achieve high quality marks, ProLase forces all pixels to have exactly the same size. Consequently, ProLase uses the specified graphic size and resolution to determine an approximate number of dots and a galvo step size is determined for a dot. Then the total number of dots is determined by how many of these dots fill the specified graphic size. The result is a 'perfect' image whose size and resolution vary only slightly from the specified values. Resolution, laser power, pulse frequency, and brightness all affect the quality of the image. Because these parameters interact, tuning an image is a complicated and somewhat subjective issue. The user should experiment with these parameters on different images to get a feel for the control.

Orientation and Sizing:

The Orientation and Sizing property sheet allows the user to control the geometric aspects of an object. Properties like X/Y Position, Height, Aspect ratio, character spacing, etc. can be controlled form this



page.

Position X/Y:

The *Position Fields are* used to specify the X and Y starting position for the object. The object is actually drawn relative to this position as a function of *Spacing* and *Justification*. The anchor point, a small, black circle, indicates the position of the position. X/Y Position can be specified to 0.0001 resolution of whatever distance units are being used. If inches are being used, the Position can be specified to .0001" resolution (0.1 mil). The maximum position is +/- 32 of the distance units (+/- 32 inches, if using inches).

The user interface allows the position to be dragged using the mouse, or the numeric X/Y values can be manually entered via the property sheet. A four-headed cursor is displayed to indicate Object dragging of Position. If Cancel is pressed while the property sheet is visible, then the object position will return to its pre-edited condition.

Rotation:

The Rotation property is used to control the writing angle of an object. The object is rotated about the *Position* values. For example when left justified, the left side of the object remains stationary, while the rest of the object is rotated about the position point. Orientation of 180 degrees produces upside down writing. The rotation angle can be specified in .1 degree increments.

Italic:

The *Italic* property is use to control the slant angle of an object. Slanting a text object to the right gives the characters an italic look.

Ring Mode:

The *Ring Mode* property is used to control the writing of text on the radius of a ring. Positive values will produce clockwise ring writing, meaning the bottom of the characters point towards the ring center. Negative values produce counter-clockwise writing. A value of zero produces normal linear marking. The *Position* property is used to define the X/Y position of the marking. The Rotation property is used to define the angle on the ring to be marked. In other words *Position*, and *Rotation* are use to back calculate the center of the ring. *Justify*, *Spacing*, and *Italic* work on ring writing just as they do on linear writing. The *Size/Length* property refers to arc length when writing on a ring. A paragraph of ring writing will produce a series of curved text lines, with the ring center moving down by the value of *Spacing/Line*.

Justification:

The *Justification* properties control how the object is drawn about the *Absolute* position. The most common *Justification* is horizontal and includes *Left*, *Right*, and *Center*. *Left Justify* is the way that most people and machines write text, with each line of text beginning at a left hand margin specified by the *X Position* property. *Right Justify* text is written so that the end of text lines up on a right hand margin, specified by the *X Position*. *Center Justify* text is written so that the center of each line falls at the X position value.

Examples:

Left Justify text
is written so that
the beginnings line
up.

Right Justify text is
written so that
the ends line up.

Center Justify text is
written so that
the middles line up.

Spacing:

The *Spacing* properties are used to control character and line spacing of text objects. When *Character Spacing* is set to zero, the character will be drawn using the natural (fixed or proportional) spacing, built into the font. Otherwise, the characters will be drawn with a fixed spacing defined by the value entered. ProLase character *Spacing* refers to the spacing from character center to character center, not the intra-character space (space between characters). For fixed spaced fonts like the Quantrad fonts, the ProLase *Spacing* value is set equal to the desired intra-character space plus the programmed character width (*Length property)*. Normally the built-in font spacing should be used, (*Spacing* set to zero); exceptions would be rulers and dials where characters need to be marked at specific locations. When in fixed spacing mode (any *Spacing* value other than zero), the *Size/Length* parameter controls the width of each character. When in natural spacing mode (*Spacing* set to zero) the Size/*Length* parameter controls the length of the entire line of text. *Spacing* is easiest to understand by setting *Length* to zero, adjusting *Character Spacing*, and finally adjusting character width (*Length)*. *Character Spacing* must be greater than *Length*, otherwise the characters will overlap (except when *Character Spacing* is set to zero in which case proportional natural is used and *Length* refers to sentence length). In fixed spacing mode, characters are automatically centered at the defined locations. X Position and Y Position define the first character position and the *Spacing* value is used to calculate subsequent positions. This greatly simplifies graduation applications like rulers, calipers, meters, and dials.

If a limited "fixed space only" system (like the Quantrad) has been used in the past, this interaction between *Spacing* and *Length* may seem awkward at first, but once understood, it provides much more flexibility.

Size:

The size properties are used to control object height and length. When *Length* is set to zero the graphic length is made proportional to the graphic height. When *Length* is greater than zero then the defined length is used to adjust the object. The object is adjusted to fit in a rectangle with dimensions of *Length* X *Height*. This feature can be used to compress, or change the aspect ratio, of an object. It can also be used to ensure that variable text will fit in a defined area no matter how many characters it contains. To mark "blocked" text, the text in this manual for example, a *Length* is selected that is somewhere between shortest and longest sentences in a paragraph. Each sentence in the paragraph is placed in a separate object. Minimizing the difference in length between the shortest and longest sentences will optimize the appearance.

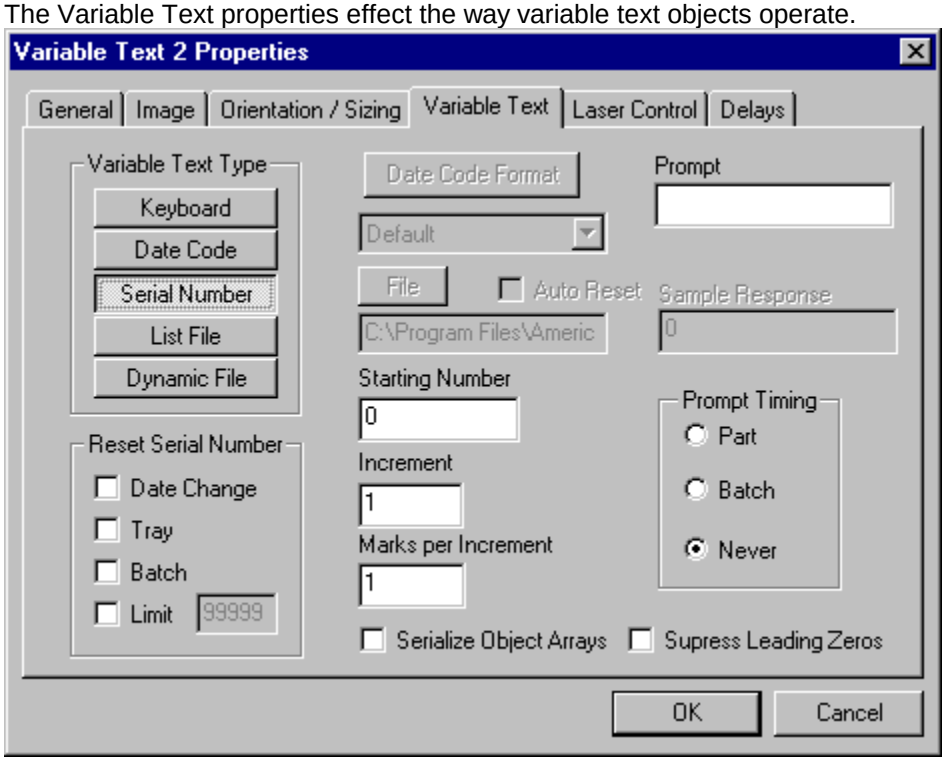
Remember that when a *Character Spacing* is defined (*Character Spacing* is set to anything but zero), *Length* refers to the length of each individual character not the length of the sentence. Conversely, if *Character Spacing* is set to zero (natural character spacing) then *Length* refers to the sentence length.

This makes sense if you consider that fixed spaced characters must have a constant sentence length. For example, 4 characters with a 0.5" spacing will allows have a sentence length (center of first character to center of last character) of 2" regardless of the character size.

In character sets, the height refers to the height of a rectangle completely enclosing a normal capital letter, like "A". The bottom of the box is called the baseline. Some letters, like "acemnorsuvwxz," are smaller than the box and letters like ‘gjqy" descend below the bottom of the box.

Variable Text:

The Variable Text properties effect the way variable text objects operate.



Variable text is text information that changes during processing. Examples of changing text are serialized numbers, date codes, and operator inputs. When a *Variable Text* type is selected, the system will accept a prompt property for the required data; this prompt is issued to the operator when the Document is selected for run. At runtime, the prompt will display a default entry. In the case of *Date Code,* the default is the current date. In the case of a *Serial Number*, the default is the next number to mark. The prompt, along with a Sample Response can be entered as Properties. Variable Text Objects are limited to 1 line of marked data. At runtime, the operator input can be entered or the operator can press <Enter> to use the default input. The Prompt Timing property can be used to disable operator prompting for *Date Code* or *Serial Number.* If the Document is saved after a run, then the last used input data becomes the new default data. If a Sample Response is entered then it will be displayed when manipulating the object, otherwise the prompt will be displayed.

Keyboard:

The *Keyboard* input function is used for manual entry of data to be used while running a mark program. For example, if parts are marked with a company logo and a person's name, the names can be manually input by the operator. If Prompt Timing/Batch is selected then any number of parts can be marked with the same data, and then another batch can be marked with different data.

When *Keyboard* is selected, the system will ask for an operator prompt; this prompt is used to inform the operator that data is required and should indicate what type data is to be provided.

Date Code:

This is used to mark the current (or operator input) date. This date is automatically updated based on the PC date variable.

When selected, *Date Code* will request an operator prompt, just like *Keyboard*. The actual prompt at runtime can be suppressed using the Prompt Timing/Never selection..

Serial Number:

Serial Number is used to select automatic serialization of a string of alphanumeric data. When selected, *Serial Number* will request an operator prompt and an increment amount. The actual prompt at runtime can be suppressed using the Prompt Timing/Never selection.

To serialize with an operator prompt for a starting serial number, set Prompt Timing to Batch. If the current default data is null, then the operator will be prompted for a starting serial number no matter how the system is configured. The marked serial number always contains the same number of characters as the starting or default data. For example, if the starting data is "0000" then the number will be incremented until it reaches "9999", at this point it will wrap around to "0000" again. "0" wraps at "9", "A" wraps at "Z" and "a" wraps at "z". Mixed data like "Ab5cD" is valid and each character will wrap as described above.

An object in a non-marked layer can be used to keep a running total of parts marked.

List File:

The *List File* selection is used to set up sequential input of data from a disk file. At runtime, *List File* allows a text string to come from a disk file such as a mailing list. Each time the program is run for marking, the next record in the file is used to fill in the text. Thus, a series of parts with different data can be run without operator input of data. For example customized gift pens can be marked using a list of names from a simple ASCII mailing list file. If multiple Variable Text objects in a program are defined as *List File* and are linked to the same text file, then the next record is read for each object. For example if the file is set up with a name, address, and telephone number for each person, then 3 records must be read from the file for each name required. To read a record without marking it, simply define a Variable Text *List File* object in an unmarked layer and use the same file name. Multiple files can be used as input for one program. For example, object #1 might get a name from NAME.LST and object #2 could get an address from another file called ADDRESS.LST. In this case, the files must match, meaning the 101st name in NAME.LST goes with the 101st address in ADDRESS.LST. This combination of mixed data in a file, and multiple files used by a mark program, provides a powerful link for variable data.

List Files can also contain ProLase modifiers. Modifiers allow the user to change the properties of a list file object before marking. This means that the size, location and other object properties can be adjusted by commands within the list file record. The commands are in the form "/c=pppp/" where c is the one character command and pppp is the parameter. In the case of size, position, and angle the parameter is a number that can contain decimal points and sign. For example to move the X position to 1.5" left of center the command would be "/x=-1.5/". Some parameters such as file names are simple ascii strings, for example to mark a graphic use the command "/g=C:\proplus\graphic.dxf/". The graphic command must include the complete path to the graphic. Notice that every ProLase modifier command ends with a "/" character. Everything after the "/" is either another command or part of the list file data to be marked. If the user wants to use the "/" character in the marked data, simply put two of them in a row (i.e. "//" = "/", "///" = "//", etc. when marking). Make sure that there is a space between command sets. When marking the list file data all modifier commands are removed. All object properties remain at their last set value until changed by another modifier command. The original property settings are as programmed in the LAZ file for the list file object. The following is a list of the modifiers:

/a=pppp/	Object angle in degrees	/a=90.0/
/x=pppp/	Object X position	/x=.053/
/y=pppp/	Object Y position	/y=-1.2359/
/h=pppp/	Object Height	/h=.5/
/l=pppp/	Object Length /l=2.0	
/g=pppp/	Mark graphic	/g=C:\proplus\columbia.dxf/
/m=pppp/	Use Material File	/m=FALSE/ , /m=TRUE/
/f=pppp/	Object Material File Freq adj.	/f=100.0/ (in percent, /m=TRUE/)
/F=pppp/	Object Frequency	/F=5.0/ (in KHz, /m=FALSE/)
/p=pppp/	Object Material File Power adj.	/p=100.0/ (in percent, /m=TRUE/)
/P=pppp/	Object Power	/P=25.0/ (power units, /m=FALSE/)
/s=pppp/	Object Material File Speed adj.	/s=100.0/ (in percent, /m=TRUE/)
/S=pppp/	Object Speed	/S=5.0/ (in /sec, /m=FALSE/)

EXAMPLE LIST FILE:

/h=.2/ /x=.5/ /y=-.5/ American Laserware
/y=-1/ Any Data
/y=-1.5/ More Data
.
.
.

In this example the first record will mark the word "American Laserware" 0.5" to the right and 0.5" below center, with a character height of 0.2". The second record will mark the word "Any Data" 1.0" below center, the x position remains at 0.5" to the right and the height remains 0.2". Although modifier commands can be place anywhere within the data string, it is best to put all commands at the front of the string, and the data at the back.

The prompt option is available with the *List File* selection. This prompt is used to allow the operator to change the file name at runtime, allowing the same program to be run with different mailing lists. After the prompt, the *List File* selection will request a file name. This file name will be used as the default file

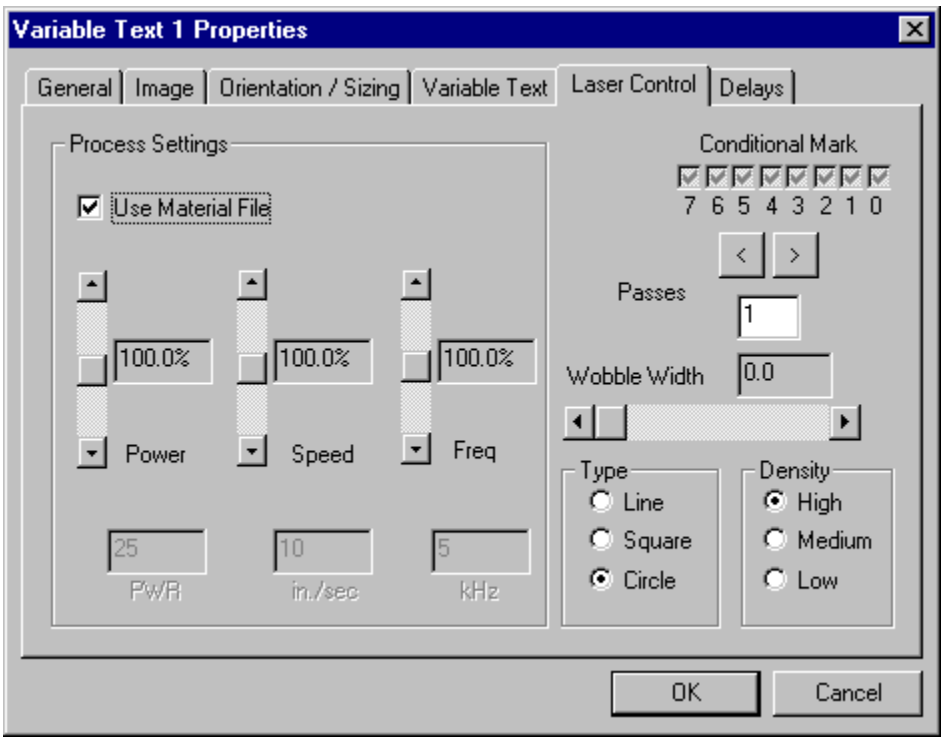
name on the next run.

Dynamic File:

The Dynamic file selection is used to read data from a changing disk file. Each time the object is marked, the first line of a text file is used to fill in the text for the object. Unlike the *List File*, the text file is not kept open from mark to mark. The file specified with the *Dynamic File* selection is closed after the data is acquired from it. This allows the file to be changed dynamically from mark to mark. The file must be changed by an external application. This can be done with the Call User Control step of the Layer IO (see Layer IO for more information). The executable file specified must open, modify, and close the same text file to which the object is linked. Multiple objects can be linked to the same text file. These objects will mark the same data unless the file is modified between them, such as if they are on different layers.

Laser Control:

The Laser Control Property sheet is used to control the major process parameters in laser marking. The main process parameters are writing speed, laser power, and pulsing frequency. If Use Material File is selected then the values for these parameters come from whatever material file the Document is linked to. These values can be adjusted for each Object using the provided slider controls. The controls can be set between 1% and 200% of the material file values. This gives the operator individual object control, while maintaining a connection to the material database. If the user prefers to enter actual values (speed, power, and frequency) for each object and not use the database, then simply uncheck Use Material File and enter the desired values in the provided entry boxes.



Conditional Mark:

The Conditional Mark property provides conditional marking based on the digital I/O port (DIO). Each object can be programmed to mark only if the DIO is in a specific state. Programming this value to XXXXXXXX (grayed checks) means the object will always be marked (the default). This might be used is a marker connected to a test station. The station tests the parts and sets the DIO port to a value corresponding to an object to mark. For example, if we have 4 possible results from the tester we can program four different objects at the same position and size in the mark field. We can give each of these objects a unique *Conditional Mark* value:

Object	Conditional Mark	Marked Data
1	XXXXX00X	A
2	XXXXX01X	B
3	XXXXX10X	C
4	XXXXX11X	D
5	XXXXXXXX	Logo, text, etc.

.
.
.

When the marking is initiated, only the object whose Condition Mark value matches the state of the DIO port will be marked. In this example, text and logos common to all parts can be programmed with a Conditional Mark value of XXXXXXXX (don't care). This program will mark parts with A, B, C or D depending on the state of the I/O when the START signal is received.

Wobble:

Wobble provides a way of altering the beam path/shape by superimposing an additional ‘wobble’ signal to the normal path taken by the laser spot. Most frequently this is an older method of ‘fattening’ lines, but it can be used to achieve other aesthetic effects. The wobble settings vary greatly with the desired effect, response of the galvos, and marking speed used. Users should experiment to become familiar with wobble settings for their machine. Generally, the marking speed will decrease and the mark time will increase when wobble is used Wobble Width determines the amplitude of the wobble signal. A wobble width of zero indicates no wobble. The units for the wobble width are in 1/1000 of the selected units. The actual increase in line width might vary from the setting, as it is dependant upon hardware response.

The wobble Type setting determines the shape of the superimposed wobble signal. For example, the circle type will add a circular signal to the normal linear motion, resulting in a spiral shape. The wobble Density setting corresponds to the frequency of the superimposed signal. From the circle type example above, with a low density, you may be able to distinguish the individual spiral path of the galvos (provided the wobble width is large enough and the marking speed is slow enough). At a higher density, the resulting mark might look like a completely solid line.

NOTE: The SCANLAB RTC cards have built in hardware wobble generators that operate slightly different that the regular ProLase wobble. If a SCANLAB RTC driver is selected in ProLase, a different set of wobble controls is displayed: Wobble Width and Wobble Freq. The Wobble Width is the same. The Wobble Freq sets an oscillation frequency and acts similarly to the Density setting above but has

units of Hertz. The SCANLAB RTC cards only support the circle wobble type.

Chapter 4 Touring the Menus

ProLase is an interactive graphics editing system used to define marking Documents. It includes provisions for defining a series of objects, which can include imported graphic images, fixed text, and variable text. This series of objects can be stored on disk, as a LAZ document file and later recalled for further editing or execution. The system is menu driven, using the keyboard or mouse for control.

Main Menu:

FILE Edit View Insert Tools Marking Preferences Window Help

File Menu:

New	***Ctrl+N***
Open…	***Ctrl+O***
Close	
Save	***Ctrl+S***
Save As	
Import Graphic File…	
Import ALI File…	
Convert Multiple ALI Files…	
Select Scanner	
Acquire Image…	
Mark	***Ctrl+M***
Print	***Ctrl+P***
Print Preview	
Print Setup	
MRU List	

The *File* menu is used for common file functions such as opening, saving, importing, etc. Each function is described in detail below.

New:

New is used to load the 'LAZ.NEW' document from the disk. This is essentially a template document containing user defined default information. LAZ.NEW can be created and changed like any other mark Document, but it must be saved using another name with the standard LAZ extension, and then renamed LAZ.NEW outside of ProLase. This prevents accidental modifications to the template.

Open:

Open is used to load a document file for editing, execution, or adding to the queue system. Each document (.LAZ) file contains the graphic information that defines every object in every layer. This file also contains links to the material process database, the fixture database system, fonts, and imported graphic files. Many sub-directories of LAZ files can be on a hard drive. Therefore, many thousands of

documents with corresponding links can be managed by ProLase.

When selected, the *Open* function will prompt the user to save the current file (if changed) and then display a directory of existing mark documents. The directory starts with the current file displayed in the file input box. Once the desired file is highlighted or entered, the OPEN button is used to load the file.

The system will check for the existence of all external links, such as PLT or DXF graphic files. If a required file is not present, then the system will inform the user and request a replacement file. The user can select a new file or CANCEL to replace the file with a rectangle at the position and size of the missing object.

After resolving external file references, the system will draw the loaded document. Because ProLase is a multi-document interface, many LAZ documents may be loaded at one time.

Close:

Close is used to close the current document. If the current document has been modified since it was last saved, then the ProLase ask if those changes should be save or simply discarded. If multiple documents are open, the next document in the list will be displayed as the current document. If the last document is closed, all document related functions will be disabled until another document is loaded or created.

Save:

Save is used to store the current state of a mark document (LAZ file) to disk. The file is always saved under the name that was used to open the file, or the name last used in the *Save As* function.

Save As:

Save As is used to write the current mark document to disk. When selected, the system will prompt the user for the file name to use when writing the file. This name becomes the current document file name until changed by another *Save As*. Writing a file to its current name is the same as the *Save* function. A copy can be accomplished by using *Open* to read a file and *Save As* to copy it under another name. Sometimes it is easiest to create a new document by copying and modifying an existing one.

Import Graphic File:

Import Graphic File is one method of inserting a graphic object into the currently selected document. Right-clicking in the graphic area and selecting the graphics tool bar button are other methods of accomplishing the same thing.

Import ALI File:

Import ALI File is used to import a mark document (ALI) file from the DOS version of ProLase. The Importation process includes converting the ALI file to the LAZ format. The importer automatically converts linked DOS fixture and material files to their Windows counterparts. The converted files are saved at the same sub-directory as the original ALI, FIX, and APS files. The extensions are different for the Windows versions, so there is no interference between the two systems. It is possible to have and run both the Windows and DOS versions of ProLase on the same PC, using the same sub-

directories.

Convert Multiple ALI Files:

Convert Multiple ALI Files is used similar to *Import ALI File* except that an entire folder of ALI files can be converted to LAZ files and none of them are actually loaded as the current ProLase document. Following this operation, any or all of them may be loaded using the *Open* command. The same rules regarding support files with the *Import ALI File* command apply here.

Select Scanner:

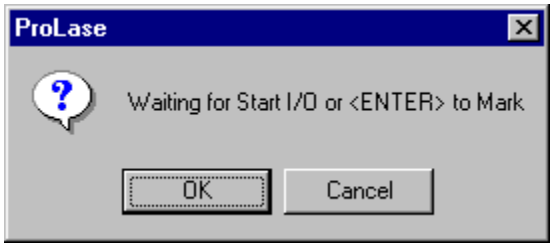
Select Scanner allows the user to select the TWAIN device used to import images into ProLase from the list available to the operating system.

Acquire Image:

Acquire Image will cause the selected TWAIN device (see *Select Scanner* above) to acquire a bitmap image using the device's support software. This image is then inserted into the current ProLase document as a graphic object in raster mode.

Mark:

Mark is used to mark the current document on the laser marking system. This menu selection is the same function as the MARK tool bar button. This is designed for testing or low volume marking. Only one part is marked and the external I/O control start signal is ignored for the first layer. Marking begins as soon as the selection is made. If marking a multilayer document the process will stop between layers, display a dialog and wait until the operator wishes to continue, by either software selection or hardware I/O.



Complete automation control and batch marking is accomplished using the MARKING/MARK CONTROL dialog.

Print:

Print will print the current ProLase window to the default printer device.

Print Preview:

Print Preview provides a preview of the image to be printed.

Print Setup:

Print Setup allows the user to modify the settings of the printer devices.

MRU List:

MRU List displays a list of the 4 most recently opened documents. Selecting one of these files will open it, providing quick access to frequently used files.

Main Menu:

File **Edit** View Insert Tools Marking Preferences Window Help

Edit Menu:

Undo
Cut
Copy
Replace
Insert
Delete
Duplicate

A note about ProLase Edit Menu features:
ProLase differs from most Windows programs in that its edit features do not include a paste feature. In most programs, paste serves two functions. Either it can replace selected items with the clipboard data, or it can insert the clipboard data into the document without removing any existing items. In applications such as word processors, the cursor indicates where the clipboard data will be inserted, or highlights the text that will be replaced. However, the ProLase environment is not conducive to this format as there is no text cursor. ProLase breaks the paste feature into its two sub-components: Insert and Replace. These two items effectively replace the functionality of the paste command.

Undo:

Undo will reverse the last action that the user made in ProLase. For example, if the user deleted an object accidentally, selecting *Undo* will bring the program back one step to where that item still existed.

Cut:

Cut will copy the currently selected object to the clipboard and delete the object from the document. It can be inserted back into the LAZ file later. Note that it will be lost the next time that an object is copied to the clipboard.

Copy:

Copy will replicate the currently selected object to the clipboard. Only objects can be copied to the clipboard.

Replace:

Replace will replace the currently selected object with the object in the clipboard. Note that the

currently selected object will be lost when this feature is used.

Insert:

Insert will place the clipboard object into the Layer / Object list of the current document ahead of the currently selected object. If the layer itself is selected in the Layer / Object Layout window, the clipboard object will be inserted as the last item in the layer.

Delete:

Delete will remove the currently selected object. It will not copy the object to the clipboard. Any object currently stored in the clipboard will remain valid.

Duplicate:

Duplicate will create a new object identical to the currently selected object. It will insert the new object at the end of the current layer.

Main Menu:

File Edit **View** Insert Tools Marking Preferences Window Help

View Menu:

- ***Render Fills***
- ***Zoom >***
  - ***Center***
  - ***Zoom In***
  - ***Zoom Out***
  - ***Center At***
  - ***Zoom To***
  - ***Zoom From***
  - ***Zoom In/Out***
  - ***Zoom Field***
  - ***Zoom Default***
  - ***Zoom Save***
- ***Standard Toolbar***
- ***Object Controls Toolbar***
- ***Stock Objects Toolbar***
- ***Drill Objects Toolbar***
- ***Zoom Toolbar***
- ***Status Bar***
- ***Layer/Object Info***
- ***Layer Tabs***
- ***Layer Tabs Setup***
- ***Graphic Library Info***

Render Fills:

Render Fills is used to force ProLase to calculate the hatch fills for the current document. ProLase will

automatically calculate the hatch whenever they are required for actual marking. Whenever the user modifies an object property that changes the hatch (size, hatch density, hatch angle, etc.) the hatch is removed from the drawing area, and only the object outline is displayed. This allows the user much smoother object manipulation as the hatch calculations take some time. At anytime the user can select Render Fills to calculate and display the hatch.

Zoom:

Zoom functions allow the user to adjust the drawing display. Zooming in allows for viewing of drawing details. Zooming out allows for viewing of the entire drawing area and beyond.

Center is used to redraw the graphics screen about the real world coordinates defined by the user. The cursor function is used to provide the position input. The cursor will only redraw about a point currently on the screen.

Zoom In is used to magnify the graphics, centering on the current cursor location. When selected, the magnifying glass cursor will be shown, and can be controlled by use of the mouse. When the desired center is selected, the left mouse button is used to initiate the Zoom.

Zoom Out works as the reverse of *Zoom In*.

Zoom Default restores the last zoom level selected prior to the execution of a *Zoom Save* command. This zoom level is stored with the file when the file is saved.

Zoom Field displays the physical area of the marking device as defined by the *Field Size* configuration parameter. In DAC devices, this is the lens field area, and in HPGL systems, this is the plotter area.

Zoom Save is used to save the current zoom, center and grid information as the *Zoom Default* settings. This information will be saved on disk with the mark file information if a subsequent *File/Save* or *File/Save As* is executed. When first drawn after a *File/Open*, a file will be displayed using the stored *Default Zoom* information.

Standard Toolbar:

This menu control allows the user to toggle on/off the display of the standard toolbar. This common function toolbar includes buttons for several often used functions:

File Tools:
- New
- Open
- Save

Edit Tools:
- Cut
- Copy
- Insert
- Undo

Output Tools:

Mark
Print
Print Preview

Display Tools:

Layer/Object Layout
Grid Settings

Modeless Dialog Displays

Mark Dialog
I/O Monitor
Material File Editor
Fixture File Editor
ProLase Plus Capable

Help

Help System Contents
Context Help

Stock Objects Toolbar:

This menu control allows the user to toggle on/off the display of the Stock Objects toolbar. This cursor control toolbar includes buttons to switch the cursor mode from select/drag to zoom or object creation.

Selecting the Fixed Text or Graphic cursor mode will create a default Fixed Text or Graphic object, respectively, at the cursor location after left-clicking the mouse. The object property pages will be displayed to further edit the new object.

Stock objects including filled and outline squares, rectangles, circles, and ellipses or simple lines are created by selecting the corresponding button then left-clicking and dragging the object to size.

Control Modes:

Select cursor
Zoom cursor

Create Object Modes:

Fixed Text Object
Graphic Object
Square
Filled Square
Rectangle
Filled Rectangle
Circle
Filled Circle
Ellipse
Filled Ellipse
Line

Object Control Toolbar:

This menu control allows the user to toggle on/off the display of the Object Control toolbar. This toolbar includes common functions used in control of object properties. Each of these buttons corresponds to an object property. The current value of the properties can be viewed or adjusted using the object property sheets.

Mirror:
- Mirror about X axis
- Mirror about Y axis

X Justification:
- Left
- Center
- Right

Y Justification:
- Bottom
- Center
- Top

Aspect Mode:
- Free
- Natural
- Fixed
- Squeeze

Angles:
- Ring Mode
- Center at Origin
- Real Size
- Real Size & Position
- Rotation
- Tilt

Fill Mode:
- Outline Only
- Fill Only
- Both
- Unidirectional Fill
- Bidirectional Fill
- Live Fill
- Render Fills

Graphic:
- Show Blanking
- Tool Path Optimizer
- Reload Graphic

Object Control:
- Lock Object Settings
- Active / Inactive Object

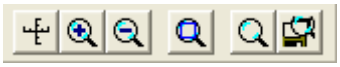
Drill Objects Toolbar:

This menu control allows the user to toggle on/off the display of the Drill Objects toolbar. This cursor control toolbar includes buttons to switch the cursor mode to object creation for various micromachining, drill object. The Spot tool has location only. The others are located with a left-click and dragged to size. After creation, the object property pages are displayed for further editing.

Create Drill Object Modes:

- Spot
- Line
- Circle
- Trepan
- Spiral
- Move

Zoom Toolbar:



This menu control allows the user to toggle on/off the display of the Zoom Control toolbar. This toolbar includes common Zoom functions:

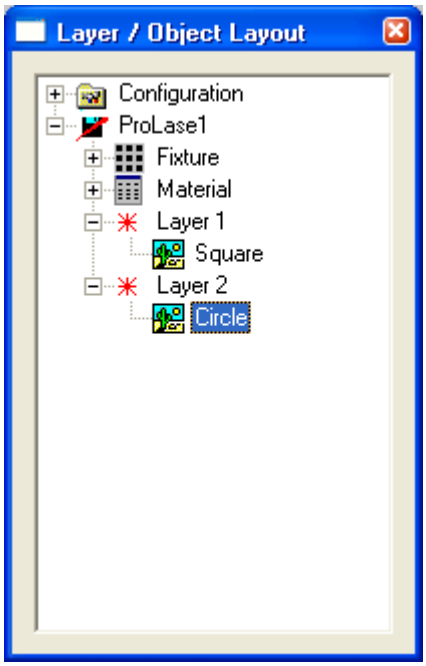
Zooms:

- Center
- Zoom In
- Zoom Out
- Zoom Field
- Zoom Default
- Zoom Save

Status Bar:

This menu control allows the user to toggle on/off the display of the Status bar. The Status bar shows many current values, including:

Layer / Object Info:

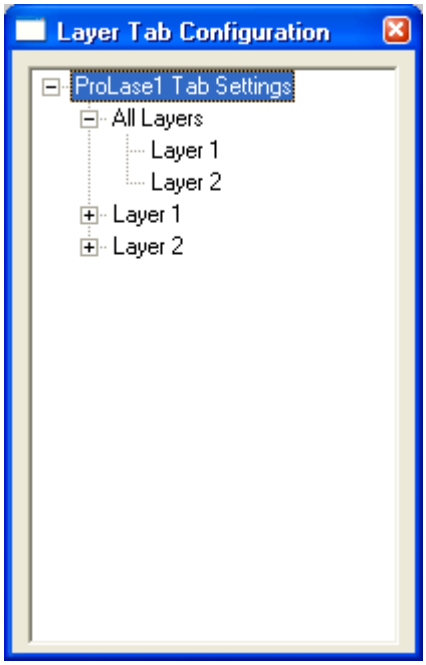


This menu control allows the user to toggle on/off the display of the Layer / Object Layout window which lists the document, fixture and material files, and the layer and object organization.

Layer Tabs:

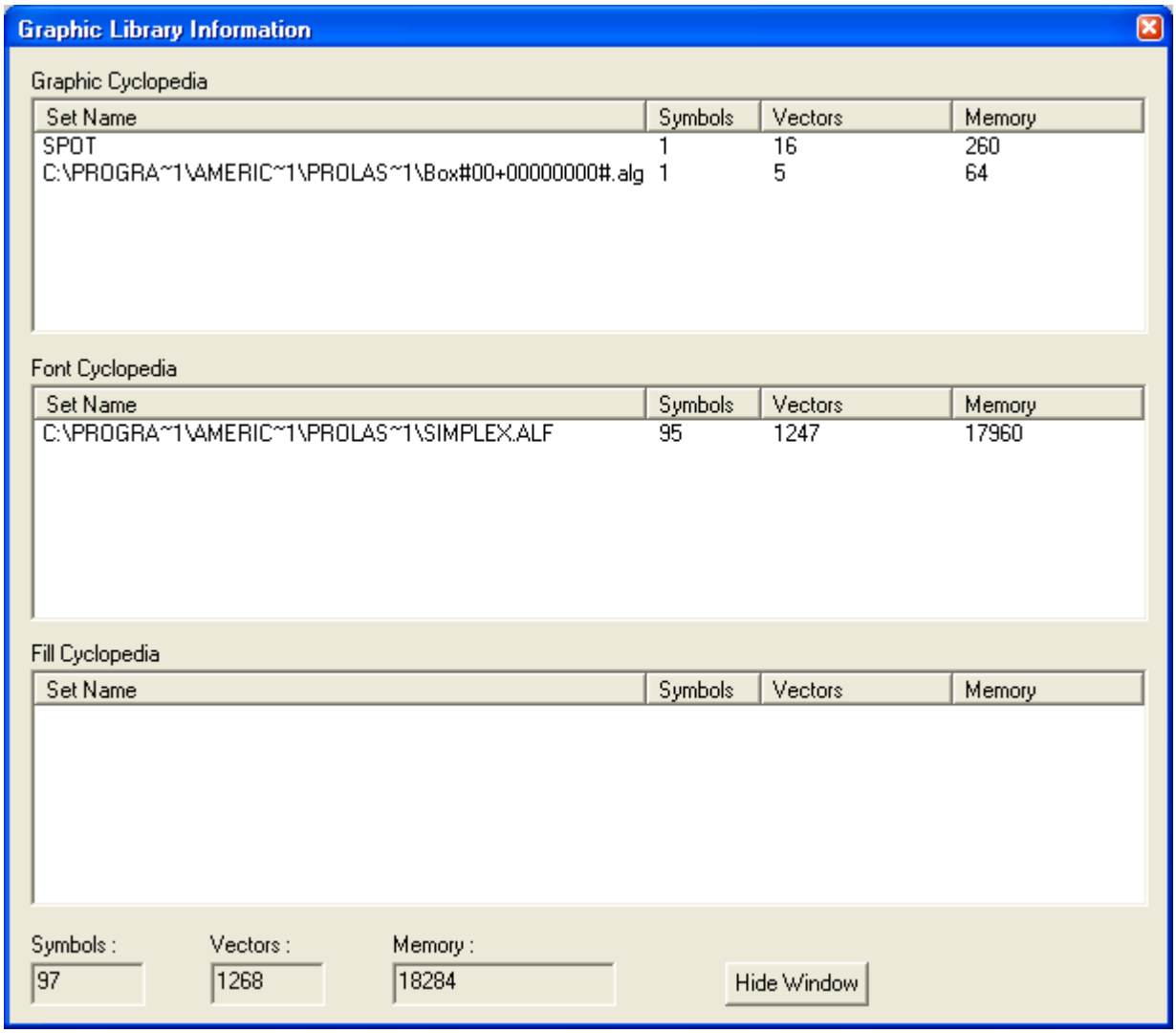
This menu control allows the user to toggle on/off the display of the Layer Tabs which allow the user to quickly switch between views that contain only layers specified by the user.

Layer Tabs Setup:



This menu control allows the user to toggle on/off the display of the Layer Tabs Configuration window. This window is used to create tabs and specify which layers are shown by each tab.

Graphic Library Info:



This menu control allows the user to toggle on/off the display of the Graphic Library Information window. This window displays the current content and memory usage of the graphic library.

Main Menu:

File Edit View **Insert** Tools Marking Preferences Window Help

Insert Menu:

New Layer
Delete Layer
Layer Properties
New Object >
Fixed Text

Variable Text
Graphic
Data Link
Empty
Line
Drill

Delete Object
Object Properties

New Layer:

The New Layer menu item will insert a new layer into the current document. A dialog asking the user to name the layer will appear. A unique layer name must be used (or leave as the default name). The Layer is then inserted into the document. See the section above on creating layers and objects for further details.

Delete Layer:

The Delete Layer menu item will delete the currently selected layer from the document. This will remove the layer and all objects within that layer from the document.

Layer Properties:

The Layer Properties menu item will open the Layer Properties window. This gives the user access to all of the layer features, such as I/O settings, activation flags, layer name, etc. See the section above on creating new layers and objects for further details

New Object:

The New Object menu item allows the user to create a new object in the document. The object will be one of the following types:

Fixed Text:

This item displays text in any desired font. The text is static and will not change unless the user explicitly alters it in the object properties window.

Variable Text:

This item displays text in any desired font. The single line of text can change between marks. The text is generated by a number of sources including serial numbers, text files, or sources external to ProLase.

Graphic:

This item displays a graphic image. Numerous graphic formats are supported by ProLase.

Data Link:

This item will use “data” from another item (either text or graphics). The sizing, orientation, font, etc. are independent of the data source, but it will display the same text or graphic as the other item. Any modifications to the source item (either by the user or by external sources) will be reflected in the data link item.

Empty:

This item has no display. It is a placeholder and can later be modified to be a display item.

Line:

This item is a simple line. It can be specified by a start point, length, and angle.

Drill:

This item displays a micromachining tool. Spots, circles, trepans, spirals, and velocity controlled moves are supported.

Delete Object:

The Delete Object menu item will delete the currently selected object from the document.

Object Properties:

The Object Properties menu item will open the object properties window. This gives the user access to all of the object features, such as position, orientation, graphics, fonts, text, etc. See the section above on creating new layers and objects for further details.

File Edit View Insert **Tools** Marking Preferences Window Help

Tools Menu:

Object Is NOT Locked
Object Is ACTIVE
Snap Settings >
- ***Grid***
- ***Rotation***
- ***Italic***
- ***General***

Mirror >
- ***X***
- ***Y***

Justification >
- ***Left***
- ***Center X***
- ***Right***
- ***Top***
- ***Center Y***
- ***Bottom***

Aspect Mode >
- ***Free***
- ***Natural***
- ***Fixed***
- ***Squeeze***

Ring Mode
Center At Origin
Rotate
Italicize
Real-World Size

Real-World Position
Fill Type >
Outline Only
Hatch Only
Outline & Hatch
Fill Direction >
Uni-Directional
Bi-Directional
Live Fill
Show Tool Path
Tool Path Optimization
Reload Object
Reload All
Convert Files
Clear TEMP Directory

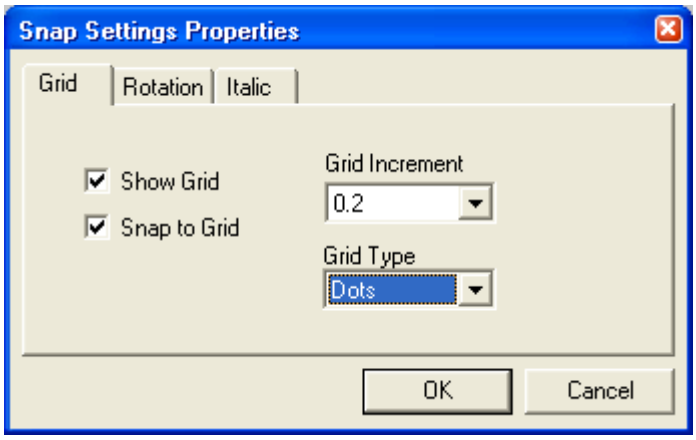
Object is LOCKED:

The Object Is LOCKED menu entry reflects the state of the currently selected object's Lock Properties flag. The Lock Properties flag is set via a checkbox found on the General page of the object's property pages. The menu entry reads as 'Object is LOCKED' when the lock is active and 'Object is NOT locked' when inactive. The menu item cannot be selected to perform the action. It must be set via the property pages. When locked, an object cannot be manipulated graphically. It settings can only be changes through the property pages.

Object is ACTIVE:

This menu entry indicates it the current object is active, meaning that it will be marked. Activation and inactivation of an object is performed using the Object Toolbar or under the General tab of the object's property pages. Inactivation indicates that the item should be used for alignment purposes.

Snap Settings:



The Snap Settings are used to overlay grid points, or snap-to points, on the marking field. These points allow for more accurate placement and orientation of the objects in a document. Each submenu item (Grid, Rotation, Italic, and General) brings up a tabbed dialog box that allows the user to

manipulate the snap settings.

Grid:

When Show Grid is checked, the grid will be displayed in the edit window, covering the mark field, when it is unchecked, the grid is not displayed. When Snap to Grid is checked, objects being moved will jump between grid points. The position handle of the object will snap to the grid points, allowing for precision placement. When it is unchecked, free movement of the objects in the drawing area is allowed.

The Grid Increment drop list allows for the selection of the grid spacing. These values are in the same units as the object placement and sizing values.

The Grid Type drop list toggles the grid between lines and points. When Lines is selected, the grid is displayed as a cross hatch across the field, when Points is selected, only the points to which objects snap are displayed.

Rotation:

Show Rotation and Snap to Rotation behave similarly to Show Grid and Snap to Grid. However, when the Rotation snap settings are shown, they appear as a ring of small red dots centered about the position handle of the object. Either one or both ends of the object will fall on this circle. As the object is rotated, it will snap to the angular settings indicated by the rotation snaps.

The Rotation Increment drop list allows for the selection of the angular increment. These values are in degrees.

Italic:

Show Italic and Snap to Italic behave similarly to Show Grid and Snap to Grid. However, when the Italic snap settings are shown, they appear as a ring of small green dots centered about the position handle of the object. When the italic handle is manipulated, the italic angle will snap to these settings.

The Italic Increment drop list allows for the selection of the angular increment. These values are in degrees.

General:

The General submenu item brings up the Snap Settings dialog with the most recently used tab selected.

Mirror:

The Mirror submenu allows for the toggling of X and Y mirror mode on the currently selected object. When Mirror X is selected, the object is mirrored about the vertical axis going through its position handle. When Mirror Y is selected, the object is mirrored about the horizontal axis going through its position handle.

Mirroring about both axes is equivalent to rotating the object 180 degrees. Mirroring about one axis will cause the object to be a ‘mirror image’ of the original. This can be useful for images on the back of transparent materials.

Justification:

Using the horizontal and vertical justification settings, any one of nine possible justification points can be selected. This point will serve as the position point of the object. All rotations are centered about this point.

Left
Center X
Right

These three menu items toggle the selected object's horizontal (X) justification. When a particular menu item is selected, the justification handle of the object will be on the left vertical line of the object, the center vertical, or the right vertical.

Top
Center Y
Bottom

These three menu items toggle the selected object's vertical (Y) justification. When a particular menu item is selected, the justification handle of the object will be on the top horizontal line of the object, the center horizontal, or the bottom horizontal.

Aspect Mode:

The Aspect submenu toggles the aspect mode of the currently selected object. The aspect settings control how an object behaves to stretching with the graphic handles.

Free:

In free aspect mode, the object's height and width can be freely manipulated with the graphic handles. Wherever the user drags a handle, it will set the height, length, or both values of the object. In this mode, height and length are independent values.

Natural:

In natural aspect mode, the object's height and width are set to the default values. Text characters will appear in their standard, or natural, aspect. They will not appear squeezed or stretched at any size. As either height or length is manipulated, the other value will be altered to maintain the natural aspect of the text. This mode is identical to Fixed Aspect with a value of 100% aspect ratio.

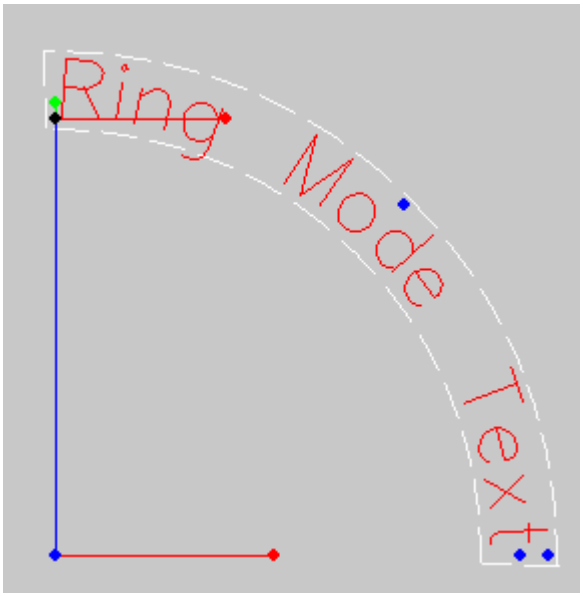
Fixed:

When fixed aspect mode is selected, the current aspect (height to length ratio of the characters) is maintained. From that point on, the characters will maintain the same shape, growing or shrinking as they are manipulated. This is similar to the behavior of natural aspect, with the exception that the aspect is now defined by the user.

Squeeze:

When squeeze mode is selected, the object's length and height control the size and shape of the characters. When the text string is small enough to fit within the object's length, it will appear in natural aspect. As the string becomes too large to fit in this area, the characters will be shrunk so that they will fit into the box defined by the height and length of the object. This mode is useful for marking text of unknown length (i.e., variable text) into a fixed size window.

Ring Mode:



This item toggles a text object between standard and ring mode text. This does not apply to graphic items. Ring mode text is drawn such that the baselines of the characters form a circle whose radius is defined by the user. The text can be drawn clockwise (with a positive radius) or counter-clockwise (with a negative radius). When the ring mode button is used to toggle the text back and forth between standard and ring mode, the default radius will generate clockwise text over an approximately 90 degree arc.

When text is in ring mode, two additional graphic handles will appear. A blue handle appears at the center of the circle formed by the text. This handle is used to set the radius of the text. It can be dragged from one side of the text to the other to change from clockwise to counter-clockwise.

The second handle is a red rotate handle. This will rotate the text about the center of the ring. Note that text is rotated using this handle, the object's position values are updated accordingly.

Center At Origin:

The Center At Origin menu item will set the current object's position to the center of the galvo field (X=0, Y = 0). Note that this will not change the object's justification settings.

Rotate:

The Rotate menu item will rotate the selected object by 30 degrees clockwise.

Italicize:

The Italicize menu item will step the text object through varying degrees of italic angles, both positive and negative.

Real-World Size:

The Real-World Size menu item is only available for vector graphic object whose source file contains sizing and unit information. When selected, the object is set to be draw at the size specified in the source graphic file.

Real-World Position:

The Real-World Position menu item performs the same function as the Real-World Size function except that is also sets the objects position to the location specified in the source graphic file. To get the proper alignment, the object is forced into bottom, left justification.

Fill Type:

Outline Only
Hatch Only
Outline & Fill

The Fill Type menu item applies only to objects that can be filled. This includes all closed-polygon vector graphics and fonts (e.g. True Type fonts). Such objects may be marked as simple outlines, just the hatch fill lines, or both.

Fill Direction:

Uni-Directional
Bi-Directional

The Fill Direction menu item is available only to objects with fills selected to be marked. When Uni-Directional is selected, all hatch lines are marked in the same direction. Bi-Directional will make alternate hatch line marking direction. Bi-Directional generally reduces overall mark time as the positioning between consecutive hatch lines is shorter and faster. Uni-Direcitional will have more uniform leading and trailing edge effects.

Show Tool Path:

The Show Tool Path menu item will cause ProLase to draw the entire path take by the galvos when drawing the selected object. When selected, the object will have the laser marked portions in red as usual, but it will also show the moves with the laser off in black. When not selected, the entire path is displayed in black. The display of the tool path is helpful in diagnosing some problems in marking. The value of Tool Path Optimization becomes visible when Show Tool Path is selected.

Tool Path Optimization:

The Tool Path Optimization menu item toggles the current graphic's Tool Path Optimization (TPO)

setting. When this feature is off, the object is marked normally, using the vector information as it appears in the graphic file. When on, the TPO engine operates on the graphic's vector list eliminating extraneous moves and minimizing the distance moved with the laser off. The result of this is generally faster, cleaner marks. Using Show Tool Path feature can show the dramatic effects of TPO.

Reload Object:

The Reload Object menu item will force the objects vector data information to be purged from the Graphic Library and reloaded from the source files. This can be useful when editing a graphic to improve the mark quality. If editing is required after a test mark, a newly saved graphic file's changes will not be reflected in ProLase. ProLase has the original vector information loaded into memory already. Selecting Reload Object will force such an update.

Reload All:

The Reload All menu item behaves exactly like Reload Object except that it applies to all objects, not just the current object.

Convert Files:

The Convert Files menu item can be used to store graphic files in the native ProLase format. A graphic that is commonly used and not subject to editing, such as a company logo, can be converted from its native file format (DXF, PLT, etc) to the ALG format. Using ALG files will allow the documents that use those graphics to load faster.

Clear TEMP directory:

The Clear TEMP directory menu item will cause the contents for the TEMP folder located in the ProLase installation folder to be deleted. This can be done to save hard drive space, minimize the ProLase footprint for backup purposes, or to purge older versions of converted graphics from the system. Once the temporary conversion files for fonts and graphics are eliminated, ProLase must re-import them when they are needed again in the future.

Main Menu:

File Edit View Insert Tools **Marking** Preferences Window Help

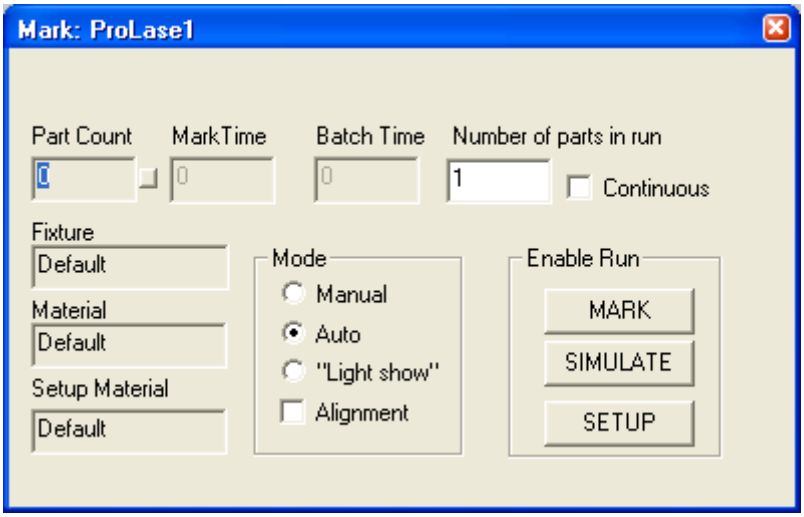
Marking Menu:

Mark Control
Queue
Fixture
Material
Config >
Field

Delays
Driver
Calibration
External Axes
New
Load
Save
Save As
Backup
Restore
Text Outline
I/O Monitor
Data Monitor
ProLase Plus Capable
Axis Monitor
Center Galvos

The *Marking* Menu contains functions that control how the system works with the operator and equipment. This menu includes the Mark Control for production marking, access to the material and fixture databases, system configuration, hardware monitor, and a center galvo function.

Mark Control:



The Mark Control selection is used to present the operator with the Mark Control window. This window is the main operator interface for production marking of parts.

The Mark Control allows the operator to select a mode for marking, the number of parts in a run, and provisions for simulating a mark process to the CRT screen. This windows also displays run status during the mark process, displays the cycle time for the mark, and displays the total number of parts marked so far in the run. The currently used fixture and material file names are also displayed.

Part Count:

The Part Count display tells the operator how many parts have been marked thus far during the current run. The Part Count is reset to zero whenever the run is stopped because the run was completed or because the mark mode was aborted. The small button to the right of the display can be used to reset

the count to zero at the user's discretion.

Mark Time:

The Mark Time display reports the elapsed marking time of the previous layer. The time is measured from the acceptance of the start signal until the end of the layer processing. When marking multi-layer documents, the value is updated between layers, but may not last long enough to view. Also, if the last layer is a background layer, the mark time will be zero (or near zero).

Batch Time:

The Batch Time display reports the elapsed time for an entire batch of parts. This time will include all time spent awaiting start signals and the entire time spend marking the parts. The display is updated when the batch is finished, whether completed of aborted.

Number of parts in run:

The user can instruct the system to run a specified number of parts in a run. Simply enter the number of desired parts to mark. If the number of parts to be marked is unknown, the operator can select Continuous, which instructs the system to mark until the run is aborted by the operator. In other words, this instructs the system to run an infinite number of parts. Most automatic assembly line markers run in this mode.

Mode:

The system can mark in one of three modes: Manual, Auto, and Setup. Alignment is used to mark only non-active, alignment object.

Manual mode a marks one part at a time and ignores any hardware start signal requirements for the first layer. Marking starts as soon as one of the Enable Run buttons is pushed. The Mark button on the Standard Toolbar performs the same function. This mode is used for testing and 'one off' type processing.

Auto mode is used for automated or batch processing. The first layer's Initial Output will be set with the pressing of an Enable Run button. However, all start signal requirements must be met before the marking starts. To run in Auto Mode, select AUTO, enter the number of parts in the run, Enable Run, and provide the required start signals.

Light Show mode is for testing. In this mode, marking starts as soon as an Enable Run button is pressed. Start I/O signal requirements are ignored. The marking will be automatically repeated as quickly as possible giving the effect of a laser light show. This is typically done with only a visible pointing laser to ensure alignment of the part in the lens field.

Alignment is checked when the user wishes to align a part correctly before actual marking. When checked, only Alignment objects are marked. An object can be defined as an alignment object by un-checking the ACTIVE check box on the General tab of the object's properties page. Usually Alignment is used in combination with Light Show mode, using Setup marking.

Most often customers use an outline of the part itself as an alignment object. Some customers use rectangles around the active objects, and other customers use registration points. Either method is available since alignment objects are created the same way regular objects (graphics, boxes, lines, etc.). Some software packages simply mark boxes around objects as the 'alignment' feature, but many

customers consider this too restrictive. The alignment feature in ProLase is expanded to mark anything. In the Mark Control window check *Alignment* and press the *Setup* button. Select *Light Show* to mark the alignment objects continuously, giving you time to physically align your parts.

Enable Run:

This selection is used to tell the system how to process the run, either as a live marking, a simulation, or setup marking.

Mark is used for actual laser marking of parts. The laser is set with the values specified in the object or in the Material File listed in the Mark Control.

Simulate will process the document in the same manner as Mark, however, no laser or galvo position controls are sent to hardware. The simulation marks to the CRT just like it would to the live marker, including processing of machine control I/O and variable text processing. This mode is used to test the function of the document and to train operators without using an actual laser marker. This is the only mode available in the DEMO program and can be used to mock-up a system.

Setup is an alternate marking mode. It behaves just like Mark, except that the laser settings are governed by the material file specified in the Setup Material display in the Mark Control. Usually named SETUP.MAT, this material file allows you to set parameters that do not mark but allow you to see where the mark will go. On many systems with visible pointing lasers this simple means that you set the power to zero in the Setup Material file. This is done for flexibility. Customers without pointing lasers can set the power very low and the frequency very high so that they can see the beam on the part without actually marking the part.

Don't forget that you can also define a laser as a 'background layer'. This is normally used to display an outline of the part that is calibrated to match the position of a part as it sits in the fixture. In this way, operators can place marks based on the position of the picture of the part in the background layer. If calibrated correctly, the marks will be in the same place on the real part as it is on the picture of the part. Background layers can also contain special instructions and annotations.

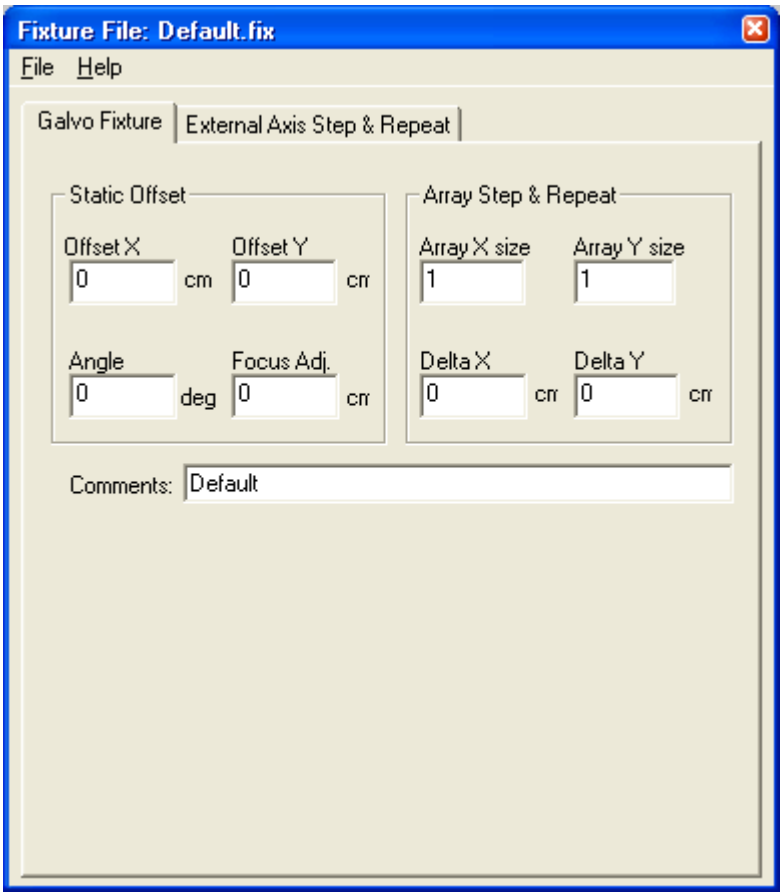
Fixture:

Each document (LAZ) file is linked to a fixture (FX2) file. The fixture file describes what type of offset and Step & Repeat (S&R) process is to occur during the execution of the program file. Most laser marking software stores these parameters in the program file, requiring modification of each program file if the fixture changes. The advantage of ProLase, with the fixture control parameters stored in a separate file, is that editing one fixture file can change the fixture setup for many program files. For example, assume the user has three different products that use the same 5x5 element, nested fixture. There is a program file for each of the products and each is linked to a fixture file called "5x5.FX2". If the fixture needs to be offset to the right 0.5 inches and the fixture data needs to be changed to compensate, only the "5x5.FX2" file is changed, automatically adjusting all document files linked to it. If a user has many document files, and only a few different fixtures, this feature becomes a very powerful way to manage the processes. The currently loaded fixture file is identified in the Fixture window, the I/O Settings & Process Links tab of the document properties, and in the Layer/Object Layout window under the Fixture entry. DEFAULT.FX2 is the default fixture file. The default data contained in this file can be edited by the user, but usually defines a fixture of 1 part with a zero value for all offsets.

The name of the current fixture file is saved in the document file, when the program is stored with a *File/Save* or *File/Save As* command. This becomes the linked fixture file for the document. If the fixture file is modified an saved under a new name, the current document must subsequently be saved to retain the link to the new fixture file. The linked fixture file is loaded automatically when the document file is loaded (by a document *File/Open* command or automatically by the job queue feature). At any time after loading a mark program, another fixture file can be loaded and used. If a mark program is saved after loading a new fixture file, then the new fixture file is linked to the program. In this way, same program to be easily marked on different fixture setups.

Mark Control/Simulate can be used to preview a program file with all fixture data incorporated, including S&R.

Galvo Fixture:



Offset X, Offset Y**:**

The *Offset X* and *Offset Y* values are used to offset the X/Y position of all objects in all documents linked to the fixture file. The position is offset from the absolute positions defined for each object. This feature is used to compensate for fixture errors. For example, consider a single part fixture that is designed to be centered at the mark field center. For some reason, the fixture is manufactured so that the fixture center is 0.1inch to the right and 0.2 inch below the mark field center. Instead of re-

manufacturing the fixture, the *Static Offset* values can set to X=+0.1 inch and Y=-0.2 inch, compensating for the error in the fixture. When S&R is used, the first part in the array is positioned at the programmed object absolute X/Y positions modified by the *Static Offsets*, subsequent array positions are 'stepped' from these points.

Angle:

The *Angle* function is used to offset the rotation a mark program. The entire mark field is rotated, allowing this feature to compensate for any fixture rotations including nested part fixtures.

Focus Adjust:

The *Focus Adjust* parameter is used to offset the height of the optical system to the part. This usually means the user must provide a mechanism that either moves the entire laser/optics system or moves the part. The *Focus Adjust* value is added to the Layer Z Table value to determine the total Z position. *Focus Adjust* is like the *Offset X/Y* parameters in that it is designed to compensate for fixture error.

Array X, Array Y**:**

The *Array X* and *Array Y* properties define the number of parts in a nested array. *Array X* describes the number of parts in a row and *Array Y* describes the number of parts in a column. This information along with the Delta parameters can be used to define any rectangular array. The total number of parts in the array is *Array X Size* multiplied by *Array Y Size*. Nested arrays are processed in 'Row major order' meaning that a row of parts is marked (stepping *Delta X* between parts) then Y is stepped and another row of parts is marked.

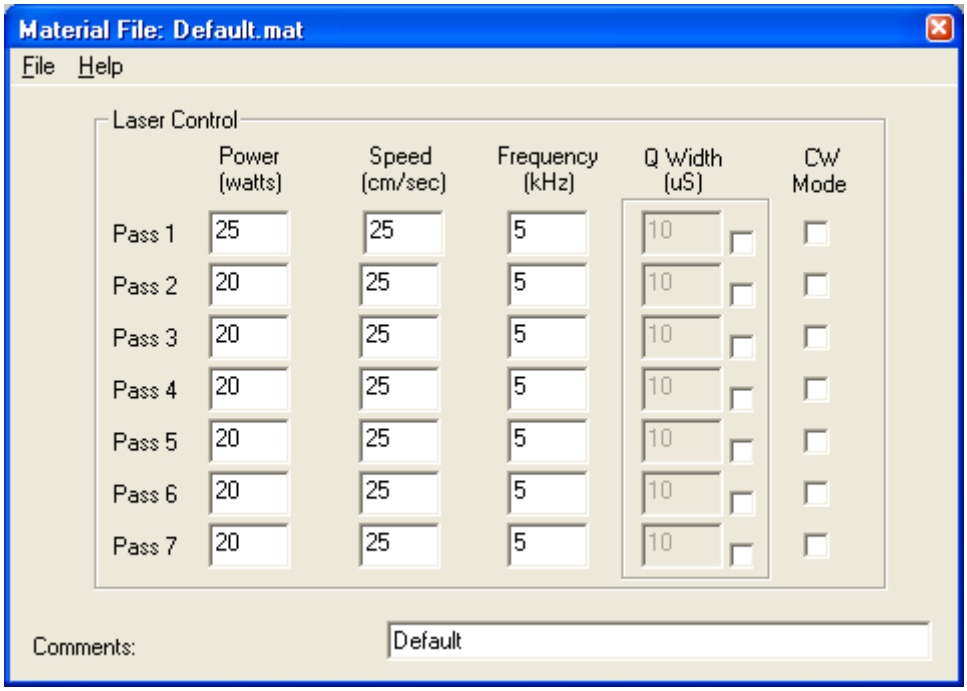
Delta X, Delta Y:

The *Delta X* and *Delta Y* properties define the amount of distance between parts in a nested array (S&R). *Delta X* defines the spacing between any two parts in a row. D*elta Y* defines the spacing between any two parts in a column.

External Axis Step & Repeat:

The External Axis Step & Repeat page is only available for ProLase PLUS versions of the software. For more information, please see the External Axis Step & Repeat entry in the ProLase PLUS chapter below.

Material:



The Material dialog is the built-in applications data base system. This system allows the user to develop material dependent processing parameters and store them. These processes can be recalled later and used in any mark program. The process files are referenced by material and can be loaded using the directory control provided. The data values apply generally to all objects in the document. However, the *Laser Control* tab of an object's property pages permits variations between objects.

File/Open:

The *File/Open* selection provides a directory control box for selection of a material (MAT) file. The files are generally referenced by a material name. Each Document (LAZ) file is linked to a material file. This file describes what type of laser process is to occur during the execution of the document file. Control of laser power, Q-switch frequency, Q-Switch pulse width, and writing speed is defined by the material file. In most laser marking system software, laser control parameters are stored with the document file, requiring modification of each program file if the process changes. The advantage of ProLase, with the laser control parameters stored in a separate file, is that the laser process for many document files can be changed by editing one material file. For example, assume the user has three different products made of the same plastic material. There is a program file for each of the products (PRODUCT1.LAZ, PRODUCT2.LAZ, and PRODUCT3.LAZ) and all three are linked to a material file called "PLASTIC.MAT". If the composition of the plastic material changes and the user needs to adjust the laser process to compensate he simply changes the "PLASTIC.MAT" file, thus automatically adjusting all of the document (.LAZ) files linked to it. If a user has many document files, and only a few materials, this feature becomes a very powerful way to manage the laser processes. Whenever a *File/Save* or *File/Save As* is used to save a document, the current material file is automatically linked. To see which applications file is currently loaded view the document file properties.

Pass (1...7)

The applications system allows the user to specify up to a seven pass process, with separate laser parameters for each pass. Each object in a document can be programmed to mark one or more or

theses passes.

Power:

Power is the desired laser power level, usually calibrated in either watts or amps. On Nd:YAG systems, equipped with external power control this parameter specifies the lamp or diode current. A voltage DAC or digital power control port is used to send the power command to the laser electronics. On CO_2 systems, this parameter controls the pulse width modulation (PWM) to the laser tube. A PC counter/timer board can be used to generate the PWM signal. A DAC can also be used if the laser includes pulse width modulation electronics that accept an analog power command.

This power level applies to the programmed pass for the entire mark program. Individual objects can be marked at a power level that is a specified percentage of this value.

Frequency:

On Nd:YAG lasers, this controls the Q-switching frequency in kHz. The interface can be a voltage DAC for lasers accepting voltages for external frequency control. A PC counter/timer can also be used as a MOD IN signal for lasers accepting TTL pulsing commands. Pulse width is normally a constant, around 10 microseconds. A frequency of zero indicates an unmodulated CW marking, a TTL gating signal is provided by the ProLase system.

On CO_2 lasers, this function sets frequency of the PWM signal. Usually it is set to the fundamental control frequency recommended by the laser manufacturer, typically between 3 and 20 kHz and varying with laser size. The pulse width modulation of this frequency controls power. The interface is either a PC counter/timer board, or a DAC (see Power control).

ProLase outputs Q-Switch MOD signals and PWM signals with 1 μs resolution. Thus, as the frequency increases (and period drops), the resolution in PWM power levels and allowed range for Q-Switch pulse widths decrease. At 5 kHz, the PWM signal can have 200 power settings. However, at 50 kHz, only 20 unique power levels are attainable.

Speed:

Speed controls the rate at which the laser beam is moved across the mark plane. On DAC output systems, an internal timer is used to control the DAC update rate. This, along with the *Field* parameters, allows the system to control mark speed (distance units/second).

File Save:

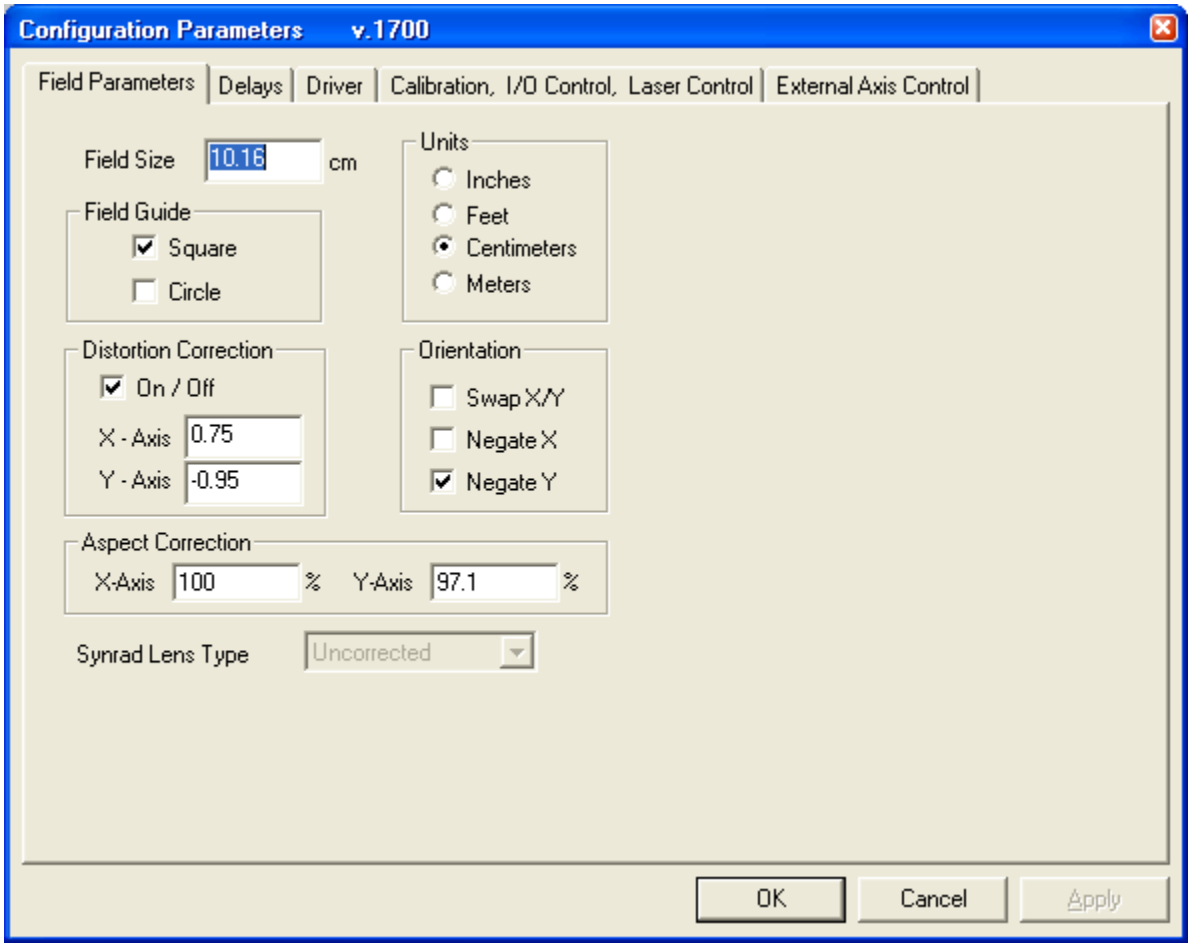
File Save is used to save the programmed laser parameters as a material (MAT) file. The user will be prompted to either overwrite the current file or enter a new file name. It is recommended that the name refer to the material being marked.

The name of the current material file is saved in the document file when the program is stored with a *File/Save* or *File/Save As* command. This becomes the default applications file for the document. The default applications file is loaded automatically when the document is loaded. At any time after loading a mark document, another material file can be loaded and used. If a document is saved after loading a new material file, then that material file becomes the documents new link. This feature allows the same document to be easily marked on different materials.

Config:

The Configuration ('Config') values convey to ProLase the physical parameters of the hardware used in the marking system. This includes the marking field size, PC driver cards, co-ordinate system orientation, laser power supply and external stepper motor axes (if applicable). Normally these parameters will be set at the factory or during system installation and should not be modified unless there is a change to the system hardware such as from a lens replacement. The user should not try to adjust any of these parameters without factory advice. When selected, the following property pages will be displayed:

Field Parameters:



This configuration tab presents controls that affect the field size, shape and orientation. These parameters help tell ProLase the actual size and shape of the marking field available with the chosen hardware.

Field Size:

The *Field Size* parameter is used to describe to ProLase the maximum extent of the laser marking system's galvos and lens. All ProLase documents (LAZ files) use real world units (inches, cm, etc.) to describe objects, so changing the field (by changing lenses for example) does not require any changes

to the documents. In other words, all LAZ files are automatically scaled correctly no matter what lens is used. One inch is always one inch. However, this does mean that the field size must be set properly. Changing lenses requires calibration of the *Field Size* parameter. To calibrate the system's field, create a square object in ProLase of some size (e.g. 2 inches). Enter a guess for the *Field Size* parameter. Mark the square and then measure it with a ruler. If the marked square does not match the size programmed, change the *Field Size* and repeat the mark. The *Field Size* and object size are inversely related. Increasing the *Field Size* will shrink the object and vice versa. Repeat this process until the marked square is the same size as programmed for the square object.

The maximum ProLase *Field Size* is 32 by 32 of whatever units are being used.

Field Guide:

The *Field Guide* controls toggle the display of an outline of the marking field in the working area. ProLase allows both the typical square field and the larger circular field that circumscribes it. The circular field yields about 40% more marking area. The *Field Guides* are controlled separately. Thus, only one, both, or no guide may be displayed.

NOTE: The circle option is disabled when the Synrad FLCC driver is selected.

Units:

This property tells ProLase what dimensional *Units* the system is calibrated to use. Regardless of the *Units* selection, the maximum *Field Size* has a scalar value of 32. If inches are selected, the maximum *Field Size* is 32 inches; if meters are selected, the maximum *Field Size* is 32 meters. The positioning resolution of the system is always 1/10,000 (0.0001) units, regardless of which *Units* are selected. If inches, the resolution is 0.0001 inch (0.1 mils); if meters are selected, the resolution is 0.0001 meters (0.01 centimeters). When larger units are used ProLase can support a bigger field, but there is a trade-off in lower resolution. Most ProLase marking systems use either inches or centimeters.

Orientation:

This property can be used to adjust the orientation of the field. This compensates for the various combinations of galvo mounting orientations and wiring order.

NOTE: Swap XY will also swap the *Aspect Correction* when the Synrad FLCC driver is selected

Distortion Correction:

This property is used to correct the field for barrel and pincushion distortions caused by X/Y galvo systems using F-Theta, flat field lenses. To adjust the correction, *Insert* a square object. Mark the square and take a straight edge to see if the sides are flat. If they are not flat, either they bow in towards the center of the square (pincushion distortion) or they bow out (barrel distortion). Minus numbers correct for barrel and plus numbers correct for pincushion. Change the numbers and repeat marking until the sides of the square become flat.

Aspect Correction:

This property can be used to square up a rectangular field. For various reasons, including servo tuning and hardware distortion correction, a marking field may not be exactly square. Proper distortion correction will ensure that all the lines are straight, but a marked square will still be rectangular. ProLase can compress a long axis to make it equal to the shorter axis. ProLase cannot lengthen the short axis because the DACs are already swinging full scale. In other words, ProLase can't make a +/- 10VDC DAC do more than that range. ProLase can however reduce the range on the long axis. It

should be remembered that using this feature to correct for galvo amplitude sacrifices resolution.

NOTE: It is ALWAYS best to square up the field by adjusting the amplitudes in the servo hardware. This preserves the original galvo accuracy. The software amplitude adjustment is provided solely as a convenient substitute for when adjusting the hardware is impractical.

To square the field with the software, the shorter axis should get a value of 100%. The longer one is simply the factor by which it must be multiplied to have the same length as the shorter axis, given as a percentage. The valid range of the factors is 10%-100%, although typical values should rarely go below 75%.

NOTE: *Axis Correction* may be set automatically when the Synrad FLCC driver is selected.

Lens Type:

This parameter is only available when the Synrad FLCC driver is selected. When using the Synrad FLCC driver in conjunction with a Synrad DH, FH, or Fenix head, the appropriate lens type should be selected from the list. The list associates ProLase configuration parameters with the physical performance of the lens.

Selecting an FH/Fenix lens that supports internal distortion correction has the following effects on the other field page configuration parameters:

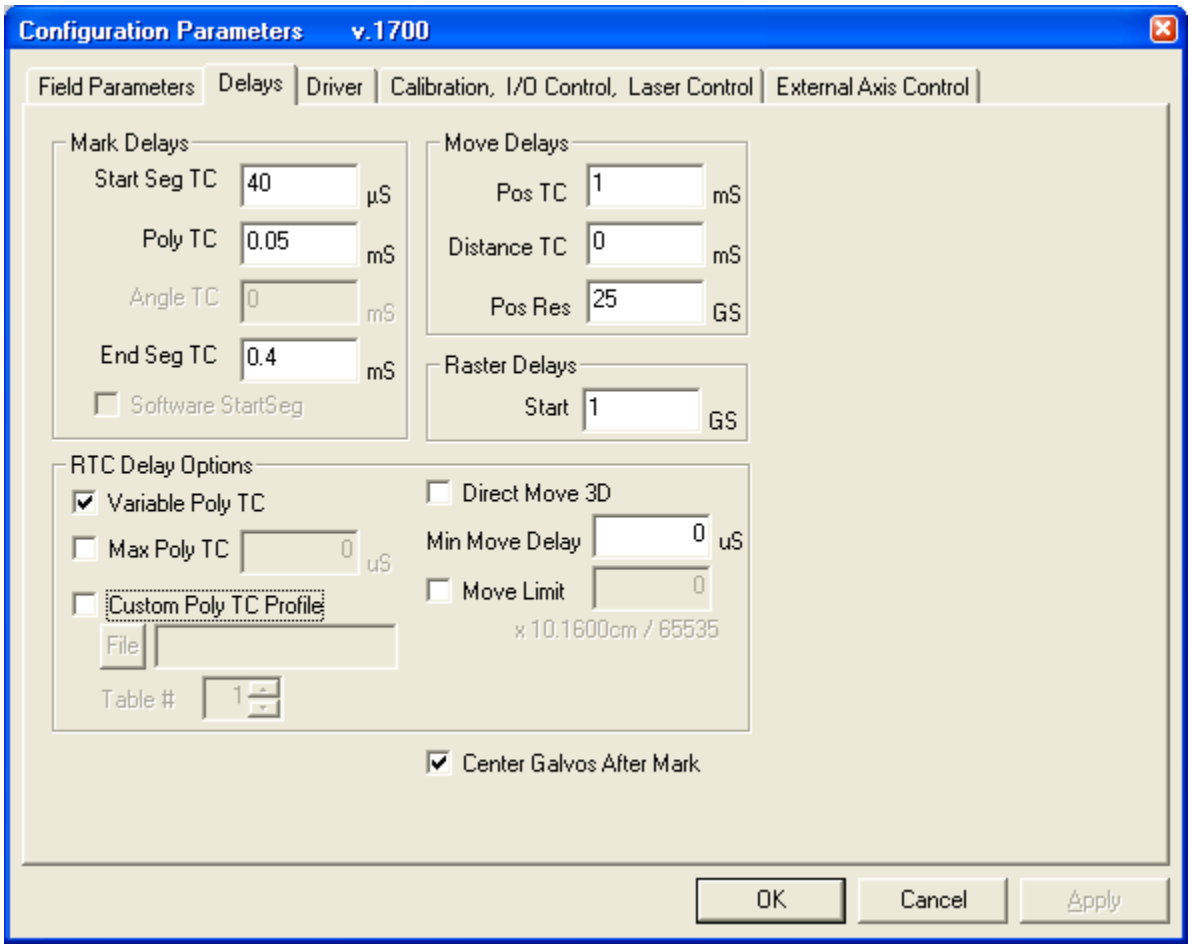
1. ProLase *Distortion Correction* is deactivated. The axis values remain the same, but they are not used.
2. The built in Synrad distortion correction affects the field size asymmetrically. ProLase 'squares' the resulting mark field by adjusting the *Aspect Correction* values. One axis will be compressed to about 80%. Which axis is compressed depends on the *Swap XY* setting.

Additionally, selecting any lens with a known focal length will set the ProLase *Field Size* parameter accordingly. The 'Uncorrected' lens type indicated that no special action is taken by ProLase.

NOTE: The settings made to the configuration parameters by selecting a lens are only *suggested* values. Any of these values may be changed freely following the lens selections. This provides for fine-tuning by the user if these default values aren't perfect. Switching lens types will only modify those parameters that are known for the selected lens.

Switching to a different driver will leave the current configuration values the same. Choosing the Synrad FLCC driver will impose the values associated with the last lens selection.

Delays:



Tuning delays are used by the system to compensate for servo system lag times. The delays are required due to the lag time between the software/DAC position and the actual hardware/mirror position resulting from physical effects such as mirror inertia. These delays are set at installation by the integrator and user adjustment is not recommended. The software is capable of marking at speeds far beyond the ability of ANY galvo to respond; marking speeds are limited by mirror inertia, servo electronics, and laser power not the software. To see the software speeds that can be achieved, set all delay timers to 0, and use an X/Y configured oscilloscope connected to the DAC outputs to view the marking. (The oscilloscope simulates a perfect galvo/servo system with zero lag time.)

NOTE: The value or use of any of the delays may be augmented, modified or ignored as required by the hardware used in the marking system. This is particularly evident with RTC based systems. Please see *RTC Delay Options* below and the appropriate hardware documentation for range limits and use for the delay values.

Start Seg TC:

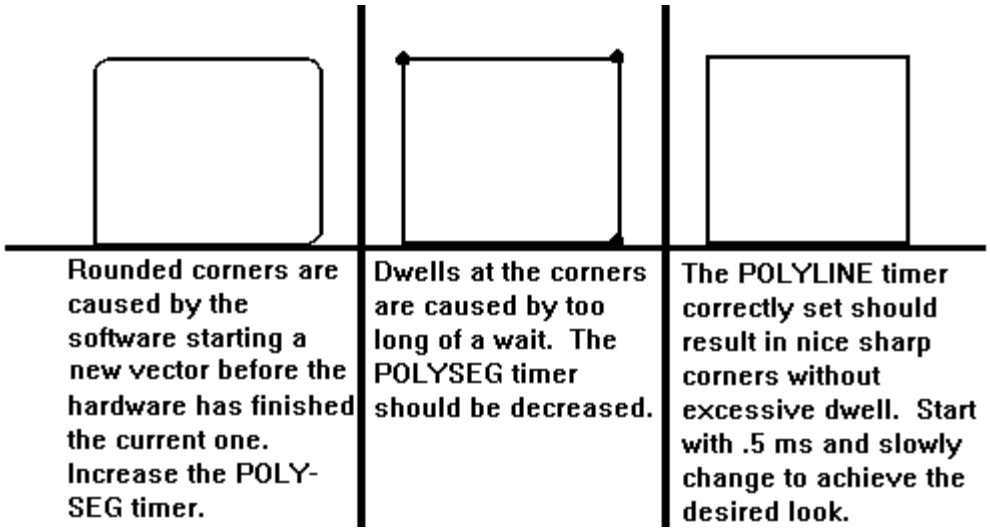
The *Start Seg TC* delay is used to minimize the acceleration effects at the beginning of vectors. Increase this timer to reduce hot spots at the beginning of marks. If too large a number is used the vector will be short or shapes will not close. This timer affects only the hot spots that occur at the beginning of vectors or the beginning of a series of polyline vectors. *End Seg TC* is used to control hot spots that occur at the end of vectors. *Poly TC* is used to control hot spots at corners.

The *Start Seg TC* value can be negative. This will cause the laser to be activated shortly before the galvos are commanded to move. This process emphasizes the dwell at the start of vectors and can be used to 'pre-heat' the substrate before the marking of the vector begins.

Start Seg TC is a hardware dependent feature. The optional counter-timer board (used to control either the CO_2 PWM directly or the Nd:YAG Q-switch pulsing directly) or if an RTC card is required to implement the *Start Seg TC*. On non-RTC systems without the counter-timer card, *Software Start Seg* should be used (see *Software Start Seg* below).

Poly TC:

The *Poly TC* parameter is used to control how long the software will wait at vector connection points. This timer applies to all vectors whose endpoint is also the start point of the next vector (polyline connection points). In other words, this timer applies to end of all vectors in a series of connected vectors, except for the last one (the end of last one is controlled by the *End Seg TC* parameter). The three connected points in a square or the intermediate connection points in a polyline circle are examples of points the *Poly TC* parameter can effect. The starting point of the square is controlled by *Start Seg TC* parameter. The last corner of the square is controlled with the *End Seg TC* timer.



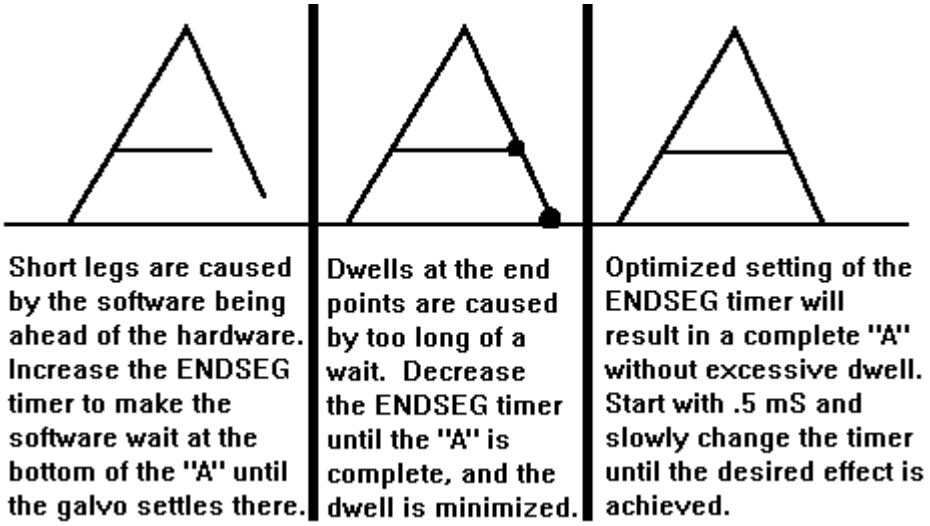
Angle TC:

The *Angle TC* parameter operates only on polyline connection points just like the *Poly* TC delay. However, the *Poly TC* causes a fixed delay and the *Angle TC* causes a variable delay to occur at corners. The *Angle TC* delay is calculated during the mark based on the angle between the two vectors. As the angle between the vectors increases, the delay increases up to the full *Angle TC* value. Thus, the corners on a large polylined circle shape will be get small delays and have a more rounded look. A square will still have sharp corners.

The full delay time at a polyline connection point is the sum of the fixed *Poly TC* delay and the variable *Angle TC* delay.

End Seg TC:

The *End Seg TC* parameter is used to control how long the software waits at the end of a series of vectors. The wait is required because the software is always 'ahead' of the hardware and must wait for the hardware to catch up before continuing with the positioning to the next vector. This delay applies to the end of all vectors in which the laser is to be turned off after execution. Considering the letter 'A', this timer only applies to the right hand leg and the middle bar. The *Poly TC* parameter applies to the top of the 'A'. Another example of and *Endseg* vector is the end of the last line that completes a HPGL polyline circle.



Software StartSeg:

This toggle enables a software-based implementation of the laser signals at the start of a vector. Normally, this is accomplished by programming the 9513 chip on the CTR05 card with the appropriate delays and activating the counters with the free running 1 MHz clock. In the software mode, the same laser signals are set/cleared during the motion control. As a result, the accuracy of this delay varies with the chosen laser velocity. As marking speed increases, the accuracy of the *Start Seg TC* delay generally decreases. This is simply an alternate method to produce the same signals and is recommended for users that are experiencing problems with wiring or CTR05 gating.

NOTE: It is ALWAYS preferable to implement the *Start Seg TC* in hardware. Only use *Software Start Seg* is a hardware solution is unavailable.

Pos TC & Distance TC:

The delays required for laser off position moves are calculated based on a formula that includes a static offset and a distance proportional component. *Pos TC* is the static offset portion and is added to all position moves. The *Distance TC* parameter is multiplied by the distance moved and normalized. These two components are added together to determine the total delay required:

Total delay after a move = (length of move X *Distance TC*) + *Pos TC*

Pos TC is set to eliminate tails on small moves and *Distance TC* is used to eliminate the remaining long move tails, such as at the start of a sentence. If these timers are set too high the marks will look good

but the marking throughput will be adversely affected. It is very important to find the smallest values that eliminate the tails.

	Equal "tail" lengths mean that the software is marking a vector before the galvo has settled at the vector start position. Increase POS_TC until most tails are gone.
	Tails at the start of a sentence means that the DISTANCE_TC delay needs to be increased.
	Correct settings for POS_TC and DISTANCE_TC will result in no tails. These timers should be minimized for best performance.

Pos Res:

Pos Res controls the slew rate limit for galvo movements when the laser is off. It is measured in galvo steps (GS). Fast settling galvos can use high *Pos Res* in the range of 20 GS or more. 5 GS is a good average limit. Older, slower galvos may require lower limits in the range of 1 to 2 GS. Old galvos that accept step inputs to the destinations can achieve this by setting *Pos Res* to 0. Step inputs means the position command is instantly set to the new location without ramping. A *Pos Res* setting of 1 GS is the slowest movement possible whereas a *Pos Res* of 0 is the fastest possible. Too fast of a *Pos Res* setting will make the galvos 'ring' for long periods at the destination point.

WARNING: Some Galvos can not accept step inputs (PosRes = 0) and may even be damaged by doing so. Before using step inputs contact the galvo manufacturer to see if this would cause damage.

General Tuning Procedure:

To tune performance start with all delays at zero, and *Pos Res* at 25. Adjust *Pos TC* and *Distance TC* until the marking 'tails' (marks leading to the beginning of characters) are eliminated. Very high values of these parameters will eliminate the tails but will unnecessarily slow the marking. If tails exist on long moves then increase *Distance TC*. If tails are equal for long and short moves increase *Pos TC*.

Adjust *Poly TC* and *Angle TC* to increase the 'sharpness' of points where polyline segments connect.

Values that are too high will create unsightly dwell marks. Values that are too low will make the connection points rounded.
Adjust *End Seg TC* to increase the wait at the end of a line or a series of polylines. This insures that the bottom right leg of the letter 'A', for example, is complete before starting the next letter. Too high a value will create dwells at the end of letters; too low a value will create 'short legs'.

Increase the *Start Seg TC* to eliminate dwells at the start of a line or polyline. Too high a value will create 'short legs' at the start; too high a value will leave a dwell.

Raster Start Delay:

Raster Start Delay controls start of vector effects similar to *Start Seg TC* but applies only to objects marked in raster format. This includes rastered versions of bitmap graphics (B/W or Greyscale mode, not Connect-the-Dots), rasterized 2-D barcode formats, and dot matrix fonts. This control is measured in galvo steps (GS) and indicates the amount of space added to the beginning of a raster to allow the galvos to reach a uniform velocity before the marking of a single pixel.

RTC Delay Options:

The RTC Delay Options controls expose additional galvo delay tuning parameters available only to uses with the SCANLAB RTC hardware. Only those controls available based on the driver selection are enabled.

Variable Poly TC works similar to *the Angle TC* described above. However, the RTC card does not combine the *Poly TC* and *Angle TC* values. Only the *Poly TC* value is used, but it varied with the angle between the vectors. This is effectively the same as setting *Poly TC* to zero and only using the *Angle TC* as described above. The formula for the delay time as a function of the angle between the vectors is specified by default by SCANLAB (see SCANLAB documentation) or can be set by the user in some configurations (see *Custom Poly TC Profile* below). The *Poly TC* value is used when calculating the delays.

Max Poly TC specifies a cutoff value the maximum delay desired for any angle. This may be necessary due to the profile of the *Variable Poly TC* or the reaction of the substrate. Please see the SCANLAB documentation for more details.

Custom Poly Profile allows the user to customize the function used by the RTC to determine the *Variable Poly TC* to suit their specific requirements. The procedure for creating the custom profile is described in the SCANLAB documentation. Use the controls here to select the correct file and table. Failure to properly load the specified table from the specified file will result in the default profile to be used.

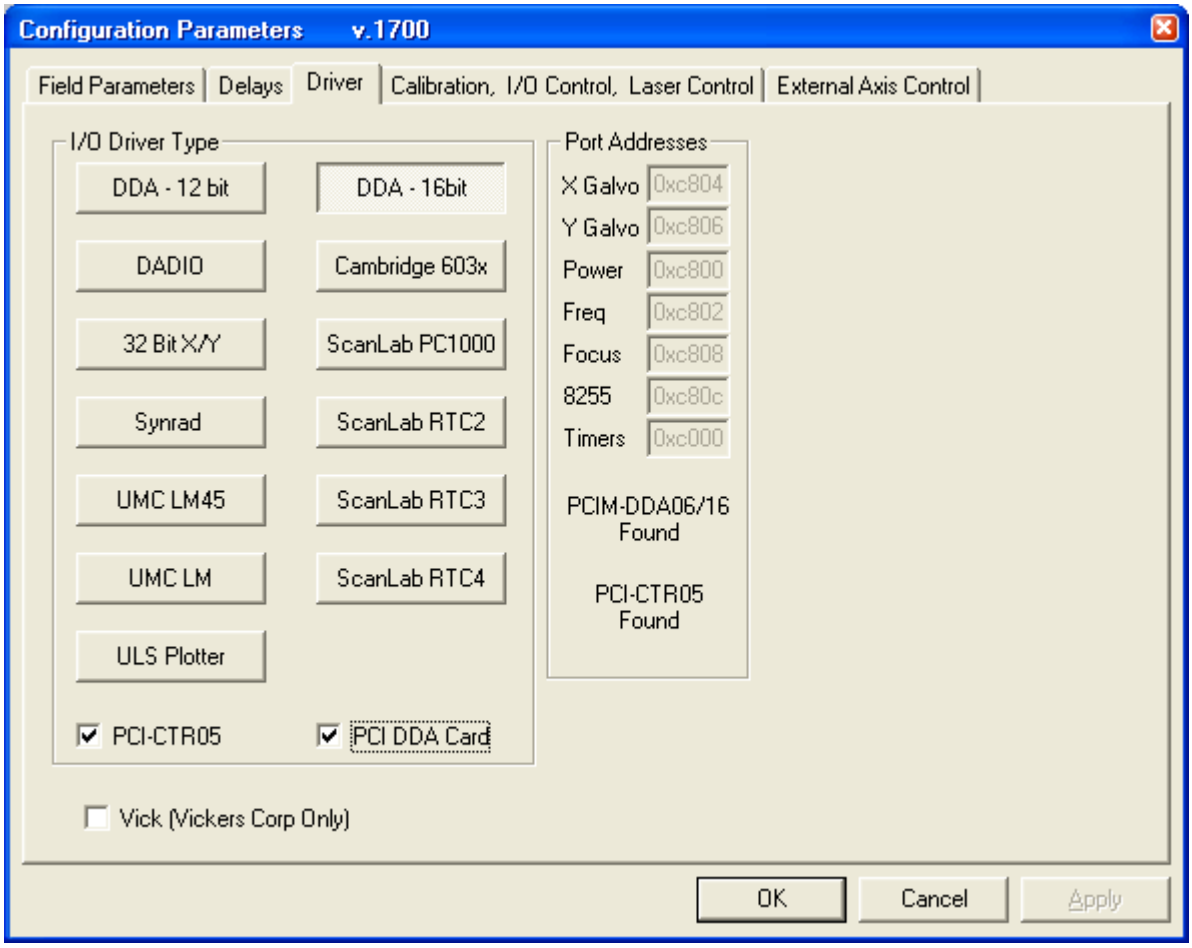
Direct Move 3D replaces the previous *Direction Variable TCs* control. It applies only to systems equipped with a Z-axis galvo and its functionality is documented by SCANLAB.

Min Move Delay and *Move Limit* can be used to adjust the delays associated with positioning the galvos with the laser off. They combine to set a linear delay profile with a maximum value set by the *Pos TC* value. The *Move Limit* distance is specified in fractions of the full field size. See the SCANLAB documentation for full details.

Center Galvos After Mark:

When ProLase completes the marking of a part, the galvos are left pointing at the end point of the last vector. This is done to increase part throughput. Moving the galvos to a home position following every mark results in two unnecessary movements and their associated move delays : from the end point to the home position and from the home position to the start point of the next part. Espescially on small quickly marked parts, this extra overhead is significant and is omitted by default. The *Center Galvos After Mark* control will cause ProLase to override this default behavior and set the galvos to the center of the marking field (0,0) after every part is completed.

Driver:



Driver selection configures the software for one of several output devices. When this tab is selected, a list of possible devices is displayed with the currently selected device will be highlighted. The hardware wiring for a particular driver is documented in Chapter 2 of this manual.

ProLase supports a large variety of systems hardware formats including off-the-shelf DAC cards, helper cards, and even proprietary systems. The capabilities of each system configuration and the responsibilities of ProLase vary with each configuration. See Chapter 2 above for more details about configuring any hardware.

I/O Driver Type:

This control must be set to the type of hardware used in the marking system. The selection of a particular driver will activate or deactivate other controls, such as the choice of a PCI-CTR card.

DDA-12bit, **DDA-16 bit**, and **DADIO** are all standards DAC interfaces using off-the-shelf cards. The X and Y galvo positions are controlled by DAC. Additional channels for laser power, laser frequency and laser focus are available on some cards. These cards all support digital I/O as well.

Cambridge 603X, **32 bit XY**, **Synrad**, and **SCANLAB PC 1000** all use various other ways to control the X/Y galvo position but still require a CIO-DDA06/12 card for the laser control DACs and the digital I/O.

UMC LM45 and **UMC LM** both use proprietary hardware from Unitek Miyachi Corp. to control galvo position, laser control, and digital I/O. A laser management program, “LM45 Manager” or “LM Manager” must run concurrently with ProLase to ensure proper laser control.

SCANLAB RTC2, RTC3, RTC4 activates the interface to the appropriate SCANLAB RTC control software and displays the additional Configuration Settings window for that driver.

ULS Plotter allows ProLase to control a Universal Laser rectilinear plotter type marker. The Universal Laser print driver is required. A CIO-DDA06/12 card can be used in conjunction to provide digital I/O signals.

PCI-CTR05:

This control is used to indicate that the CTR05 card being used is located on the PCI bus (NOT the ISA bus) and that it is Plug & Play compatible.

PCI-DDA:

This control is used to indicate that the DDA card (DDA02, DDA04, DDA06, etc.) being used is located on the PCI bus (NOT the ISA bus) and that it is Plug & Play compatible.

Port Addresses:

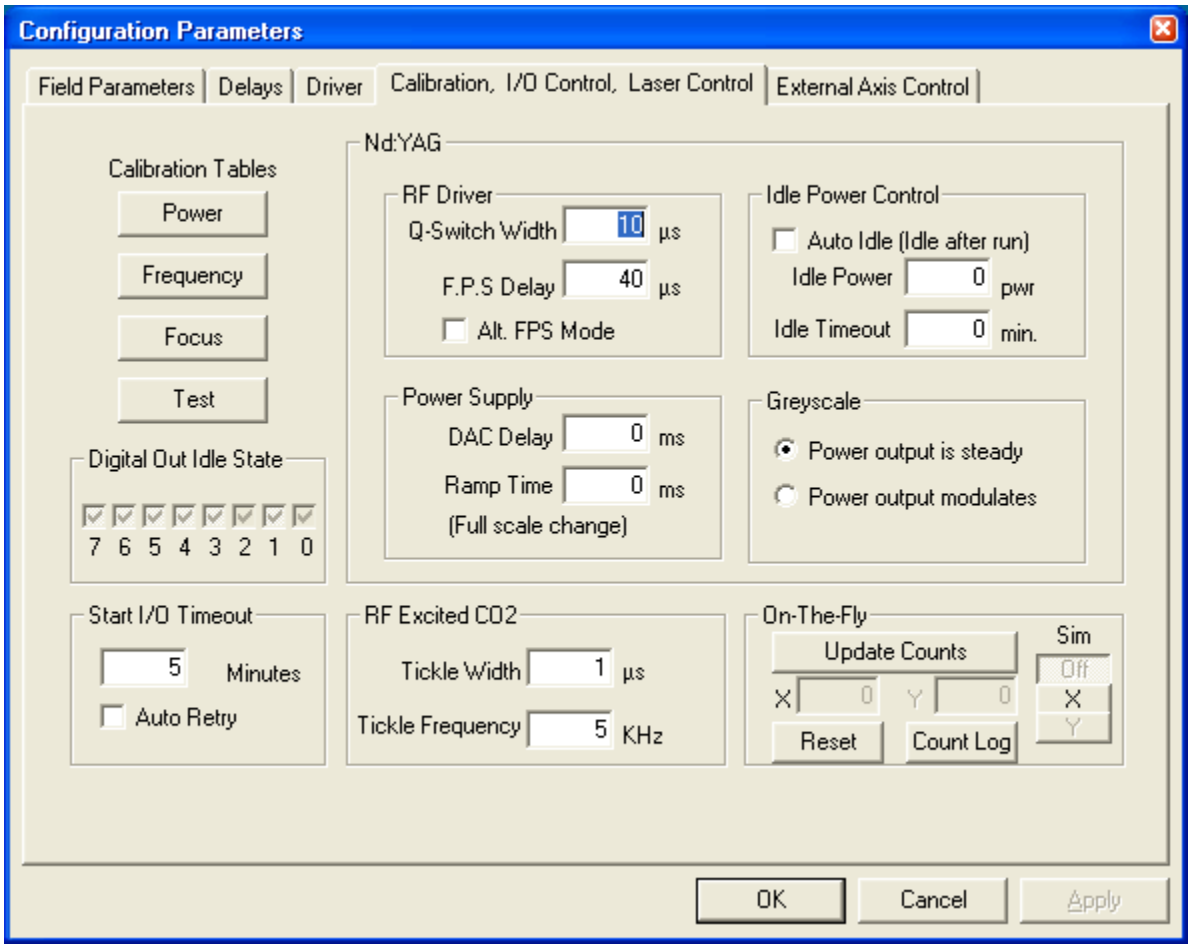
This section provides feedback as to the address assignments expected by ProLase for the driver selected. Addresses are reported for the X and Y galvos, power control DAC, frequency control DAC, focus control DAC, 8255 digital I/O chip and the CTR05 counter timer card.

Below the addresses, ProLase indicates if any PCI cards specified in the driver selection were located by the system. If a card is found, the type of card is displayed along with the message ‘Found’. If the driver does not require a PCI card or ProLase cannot locate it, the card type and ‘Assumed’ is displayed.

Vick (Vickers Corp Only):

This control is used EXCLUSIVELY to activate undocumented custom features created for Vickers Corporation. Other customers should not enable this control.

Calibration, I/O Control, Laser Control:



This configuration tab presents controls that allow the user to specify laser control parameters including power calibration tables, power supply settings and, if available, On-The-Fly settings.

The *Calibrate* function is used to calibrate the DAC control outputs that are normally used for power/current, focus and frequency control. A menu is displayed that shows a list of voltages (0-5 VDC). At the far right of the menu is a list of power or frequency values. When one of these values is selected, the DAC will be activated to the corresponding voltage. The user measures the value of the power/current, focus or frequency (a power/current meter or frequency counter is required to measure the parameter) and inputs the results. This procedure is repeated for each value in the table. This table is used as calibration data so that when a desired power/current or frequency is programmed in the material database, the DAC output will produce the correct value. Applications developed on calibrated systems can be moved to other calibrated systems without adjusting parameters. Re-calibrating a system can be used to compensate for time dependent degradations (like old flash lamps, or old CO_2 gas), thus avoiding periodic changes to the process database.

Alt. FPS Mode:

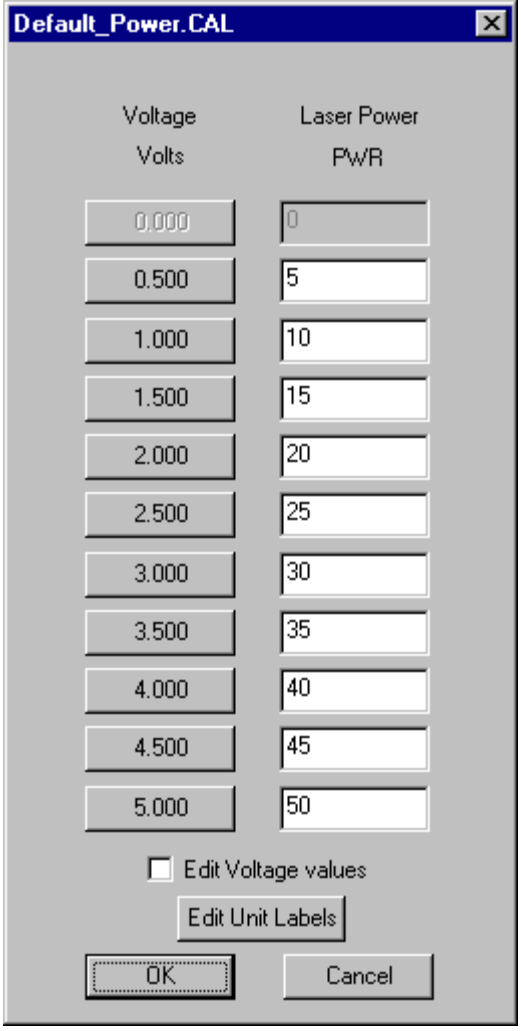
Activating this feature will make ProLase set use the Alternate First Pulse Suppression mode for use with Q-switch YAG lasers. Normally, the Q-switch modulation signal will begin pulsing only after the FPS delay has expired. When the Alt. FPS Mode is selected, the Q-switch pulsing begins at the end of the StartSeg delay and coincidentally with the beginning of the FPS signal. See Figures 4 and 5 in Chapter 2 for more detailed timing diagrams. This selection should be made in accordance with the laser manufacturer's specifications and is available only for those drivers options using the CTR05 card.

Load Config:

This selection will load the current configuration data from the disk Config file. This can be used to restore a configuration to its original state after Config values have been edited.

Save Config:

This selection saves the current configuration values to the configuration disk file.



Main Menu:

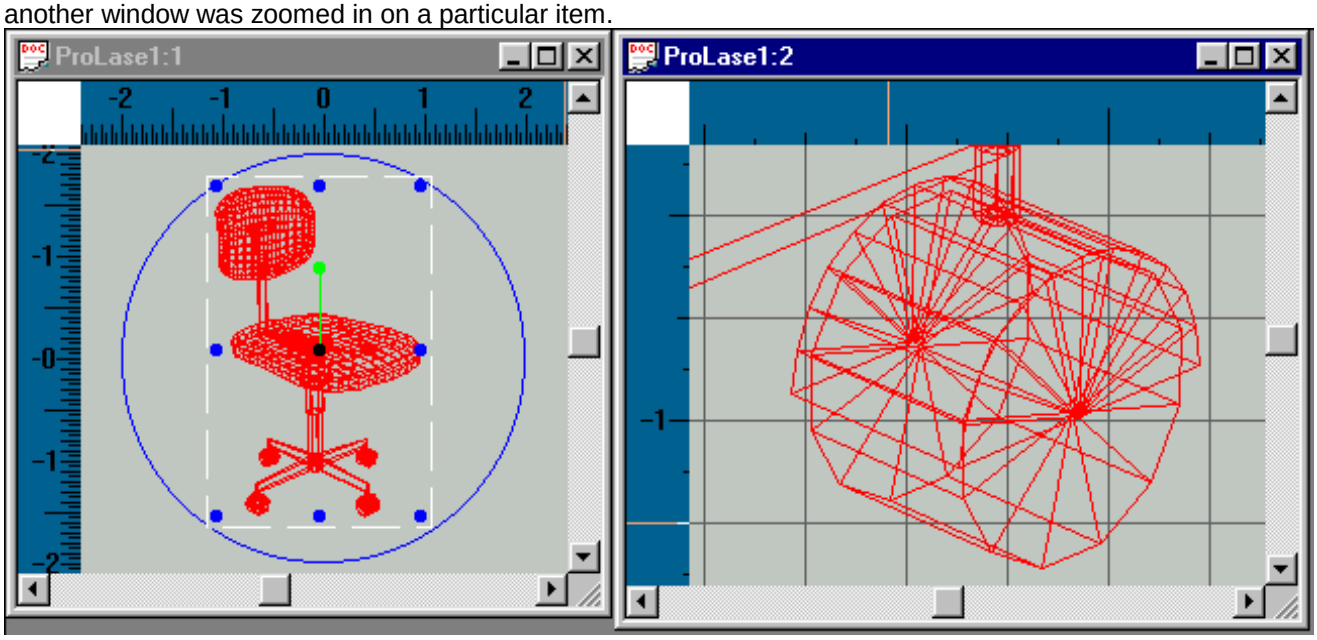
File Edit View Insert Tools Marking Preferences **Window** Help

Window Menu:

New Window
Cascade
Tile
Arrange Icons

New Window:

The New Window menu item will create a new window of the current document. This is not the same as a new document, it is a new view of the same document. Changes made in one window will be reflected in the other. However, window position, sizing and zoom level, grid settings, etc. are independent. For example, in one window the zoom level could display the entire marking field while another window was zoomed in on a particular item.



Cascade:

Selecting this menu item will organize all of the current windows so that they appear on top of each other, each consecutive window offset from the ones above it slightly.

Tile:

Selecting this menu item will arrange all the windows so that they do not overlap at all. The sizes and shapes of each window will be roughly equal and will take up the entire program window.

Arrange Icons:

Selecting this menu item will align any minimized window icons that are currently in the program window.

Execute Main Menu:

File Config **QUEUE** Run Display
Process I/O Fixture Edit

QUE:

Add
Insert
Delete

Add:

The *Add* function allows the user to add a program file to the Job Queue. First a file is selected (*File/Open*), and variable information is provided. When *Que/Add* or *Que/Insert* is selected the user will be prompted for the quantity of parts to marked. The file, its variable information and the quantity are saved at the bottom of the queue.

Insert:

The *Insert* command works like the *Add* command except the user can determine where in the queue to add the file. The user will be prompted for the *Insert* position, the file is placed at this position and all files that were at or below this position will be moved down the queue.

Delete:

Delete is used to remove an entry from the Queue. The system will prompt for a Delete position, the entry at that position will be deleted and all entries below it will be moved up the queue.

Execute Main Menu:

File Config Queue **Run** Display
Process I/O Fixture Edit

RUN:

Program
Que
Simulate
Simulate Que

In any of the Run modes a status area at the bottom left of the screen will display the following information:

Status - empty, Awaiting Start, or Marking Layer number.
.ALI file name.
Number of parts marked.
Time of last mark sequence.
Variable information name and data.

A marking sequence is initiated by either the F1 key or a valid digital I/O start signal. After mark completion the system will return to the "Awaiting...Start" mode and marking can again be initiated. A "Mark Complete" signal is provided to facilitate parts handler interfacing. This procedure is repeated until all parts have been marked. If programmed for multiple part fixtures, all parts on a "tray" are marked with one start initiation. If the quantity entered is less than a full tray then a partial tray is marked. The tray must filled according to the step and repeat programming. Usually X steps are positive and Y steps are negative requiring trays to be filled left to right and top to bottom. If no quantity is entered then all trays are assumed to be full, if the entered quantity is larger than a full tray, then full trays are assumed until the number of parts remaining to be marked becomes less than a full tray. The job queuing feature can be used to pre-enter part counts (and/or other variable information) for a sequence of trays. Using <Escape> at any time while in the *Run* mode will cause the system to abort the marking and exit the *Run* mode. The system actually completes the current object and then aborts executing the rest of the part. Using <Escape> when the system is *"Awaiting...Start"* insures that there are no partially marked parts. When returned to the *Run* main menu, the operator can either re-run the same file with a new quantity, *File/Open* the same file to change variable information, or File/Open a completely different file.

Program:

Run/Program will execute the current mark program (ALI) file. Motion and laser process control are in full effect during this mode. The variable information obtained by the system when the mark program was loaded is used during the marking process. *Var_Text* objects defined as *Code_Reader* will prompt for information after marking each part. *Var_Text* objects defined as *File* will read information from the disk file after each part. The selection will prompt for the quantity of parts to be marked. If a number is entered, the system will mark that number of parts, and return to the main *Execution* menu when finished. If <Enter> is used without entering a number then the system will mark parts until <Escape> is used to exit the mode. In other words entering no quantity is used to mark an undetermined quantity of parts. The *Config/Menu Options/Auto Quant* can be used to disable the quantity prompting. If this option is selected a quantity of 0 (infinite) is assumed. The <Escape> is used to abort the run after the desired quantity is marked.

Que:

The *Run/Que* selection is used to execute the sequence of programs currently stored in the job Queue. When this mode is selected the program at the top of the queue is loaded and prepared for execution. The system will wait for a start signal as in the *Run/Program* mode. When the quantity has been marked, the program is removed from the queue and the next program is loaded. This sequence is repeated until the queue is empty. If <Escape> is used to abort processing, the number of parts remaining for the current queue entry is placed in the queue as the new quantity. This allows queue processing to be aborted and continued at any time. Queue processing can be suspended, a non-queued file loaded and run, and then the queue continued. In this way routine work can be interrupted for special runs. The current queue entries are stored to disk if the operator returns to the DOS system. This allows the queue to be memorized between invocations of ProLase. This feature is especially useful in a process where many different types of marking go on the same part.

Job Queue Example:

lets assume we have a product that requires an ID plate containing specific information such as color. We have orders for 15 red, 10 blue and 7 black versions of the product and we need marked ID plates for each. All ID plates are the same when blank. A parts handler system is used to remove marked plates and present the blank plates to the mark field. A file called Example.ALI is programmed with a

Programmed_Text/Keyboard object called color. The operator executes a *File/Open,* selecting Example.ALI and entering "red" at the "Color" prompt. *Que/Add* is selected and 15 is entered as the quantity. This procedure is repeated for blue (10) and black (7). Now *Run/Que* can be selected, all 32 plates will be marked resulting with the correct number of red, blue and black plates. At any time the processing can be suspended, more plate orders added to the queue, and processing continued.

Simulate and Simulate_Que:

Simulation modes are used for testing of programs and interfaces without actually marking parts. The graphics area of the CRT screen is used to simulate the laser marking field. Serialization, and reading records from disk are active just as in real marking. Simulation is an good way to test program files, experiment with system configurations, and train operators without using an actual laser system.

Chapter 5 Supplemental User Instructions ProLase PLUS

This section of manual applies only to the ProLase7 Plus software package.

ProLase7 Plus is ProLase7 with additional capability to control up to four external stepper motor axes (X, Y, Z, A). The X/Y axis control allows for marking of large parts in sections. Each Layer within a document (*.LAZ file) can specify a table X/Y offset position. The step and repeat (Fixture page) has been expanded to allow array processing using the X/Y positioning capability. Alternately, ProLase can automatically re-center the marking field over each logo or character. The focus axis (Z) can be set for each layer allowing for the processing of 'stepped' parts (parts with marks at more than one elevation). A rotary axis (A) is used to do cylindrical marking. ProLase indexes this axis automatically during the marking process, providing the capability to mark cylinder shaped objects.

ProLase controls the external axes through a user stepper motor controller program. ProLase sends step and directions signals to the controller and may receive feedback from it. ProLase can communicate directly only with Indexer LPT. Many other stepper motor controller programs can be used through the use of simple batch files written by the user. Of course the user must also provide tables, amplifiers and stepper motors.

Indexer LPT

Indexer LPT is a multi-axis stepper motor controller program provided by Ability Systems Corporation (215-657-4338, www.abilitysystems.com). Indexer LPT sends step, direction, and limit switch control signals conveniently via the printer port. Up to two (2) axes may be controlled by each printer port. ProLase systems that use printer port dongles (software locks) cannot use the same port for the ProLase dongle and to control an external axis. One or more additional printer ports must be installed. Any Ability Systems dongle is designed to be 'pass through' for stepper motor control.

Indexer LPT should be installed, tested, and used as instructed by Ability Systems documentation. Computers running in the Windows 7 operating system require additional set up steps as described in the note below.

> ***Indexer LPT on Windows 7***
>
> *Indexer LPT running on a computer with the Windows 7 operating system behaves differently that previous versions of Indexer LPT for Windows 95/98/ME. To provide backward compatibility with existing ProLase systems, ProLase7 PLUS comes with several small 'helper' programs.* ***TableServer.EXE****,* ***Table.EXE*** *and* ***TableBATConverter.EXE*** *are installed in and must be present in the ProLase7 PLUS installation folder. Upgrading to the Window 7 environment may also require the modifications of the InitXYZA.BAT file, see Initialization Proceedure below.* ***Please see Appendix C for more information about using ProLase7 PLUS with Indexer LPT on Windows 7.***

Indexer LPT on Windows 95/98/ME operating systems works using a driver accessed through the `motor` file. Axes can be controlled by a set of commands from a batch file such as

```
echo set_home:a > motor
```

Here, the DOS command `echo` is used to display the command to the screen for the benefit of the

user. The text `set_home:a` is then redirected (the DOS '>' command) into the `motor` file, where it is interpreted by the Indexer LPT driver.

When using the Windows 7 version of Indexer LPT, the commands should have the form:

```
table set_home:a > motor
```

which invokes the Table.EXE program to appropriately pass the `set_home:a` command to the Indexer LPT driver and place the response in the `motor` file.

'Other' Stepper Motor Controllers

Customer using non-Indexer LPT stepper motor controller programs must provide an interface to relay commands from ProLase to the controller program. Due to their ease of use, this is most often done using batch files (*.BAT), although any executable file format is acceptable. Regardless of file type, it must accept 8 command line arguments of the form:

```
move.bat a, A, b, B, c, C, d, D
```

where `a,b,c,d` are the axis IDs for the X,Y,Z,A ProLase axes and `A,B,C,D` are the positions to set the each axis, respectively. The positions are given as absolute or relative values as specified by the ProLase configuration, External Axis page. The batch file must convey this data to the stepper motor controller as necessary. An example of one such `move.bat` batch file is included with the ProLasePLUS installation.

In addition to a 'move' batch file, the user must supply a 'status' and an 'initialize' file. The status file should retrieve the current positions/status from the controller and write this data to the Status Read File text file. The initialization file should configure the axes as needed for proper use as described below. The names and locations of each of these files must be specified in the External Axis Control configuration page window.

Initialization Procedure

Before ProLase can send meaningful position commands the motion profile parameters must be set and each enabled axis must be homed. ProLase initializes the axes upon program start up and whenever the configuration is modified. To perform this initialization, ProLase calls a simple batch file created by the user. A batch file called INITXYZA.BAT is provided in the Table sub-directory. The user should modify this simple batch file. The correct axis IDs and motion parameters must be selected. This batch file should contain the following commands as a minimum:

```
echo set_accel:a,15000  >  motor
echo set_lowspeed:a,2000 > motor
echo set_highspeed:a,20000 > motor
echo set_home:a > motor
```

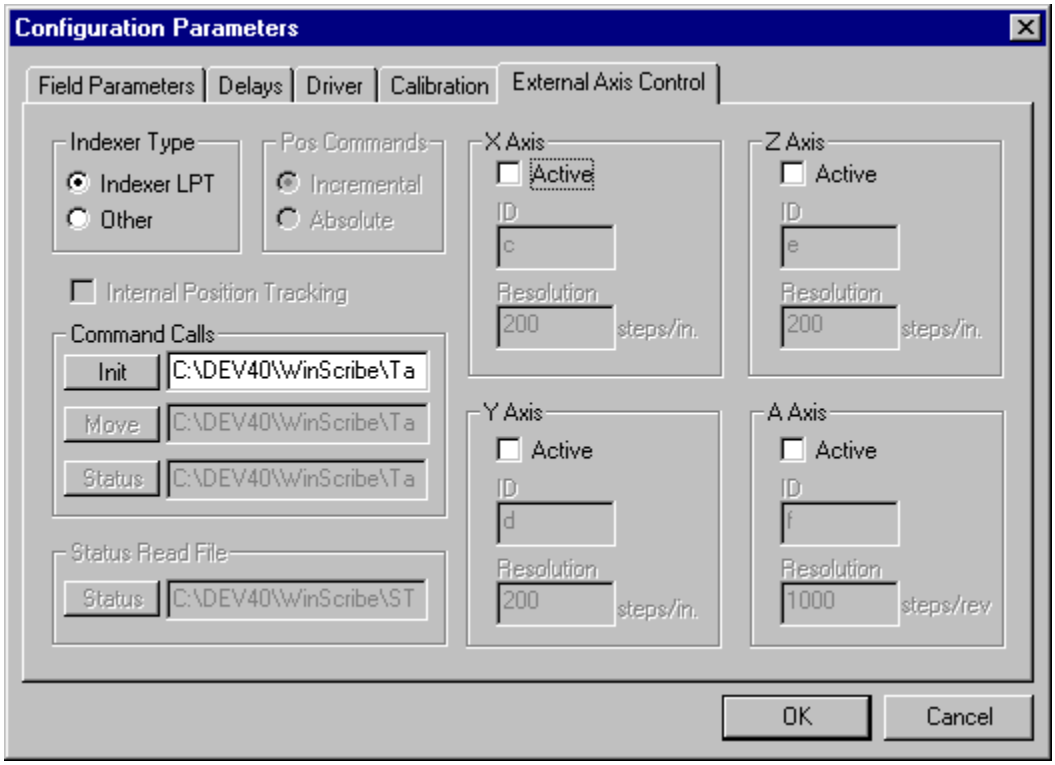
NOTE: All batch files used by ProLase must have their CLOSE ON EXIT property set. To set this property in Windows use Explorer or My Computer to find the icon for the *.BAT file. Right click on the icon, and select Properties from the menu. Go to the Program tab. Make sure "Close on exit" is checked.

The batch file shown above is for use with Indexer LPT as the stepper motor controller and running on

a Windows 95/98/ME operating system and the numeric values shown are just examples. The correct values for speed and acceleration depend on motor power, table mass, gearing, etc. and should be determined by experimentation. The numbers should be selected so that the table positions in minimum time without missing steps. The Indexer LPT axis ('a' in the above example) is determined by the address of the printer port being used (See the Indexer LPT manual for complete details). Of course each axis (four maximum) to be used by ProLase must be setup in this manner.

The `set_home` command sets the internal position keeping logic of Indexer LPT to zero (0). Failure to execute a `set_home` prior to running ProLase will generate a table positioning error. If at any time an axis hits a limit switch, the initialization batch file (InitXYZA.BAT) should be re-executed. Otherwise ProLase will generate a table positioning error. Use the *Marking / External Axis Control* menu selection to re-initialize the table system. The initialization batch file should perform a procedure that moves the table to a consistent position prior to executing the `set_home` command. One way to do this would be to command the table to move in one direction until it hits a limit switch (and then back to the zero position if not at that edge). If the switch is hit at a low speed the positioning will be more consistent than if the switch is hit at a high speed.

Before using the external axes from ProLase, several configuration parameters within ProLase must be set. These parameters are in the *Config/External Axis* menu. Basically this involves giving ProLase enough information to build the command string required to position the tables.



ProLase PLUS Capable

ProLase PLUS provides general feedback as to the status of the initialization of the external axes with the *Marking* | *ProLase PLUS Capable* menu item. A button on the Standard Toolbar mirrors this state. If not selected, no axis motion will occur. The item is only enabled when a PLUS version of ProLase is

running. For ProLase PLUS configured to use the 'Other' Indexer Type and those using Indexer LPT on Windows 95/98/ME systems, the *ProLase PLUS Capable* item is selected only when at least one axis is enabled.

When using the Windows 7 version of Indexer LPT, this item is selected only when one or more axes are enabled AND TableServer is running and completely initialized. ProLase PLUS continuously polls for the presence of TableServer. If TableServer is terminated for some reason, *ProLase PLUS Capable* will be deselected. Manually selecting it will reinitialize the ProLase axes, restarting TableServer in the process.

X Axis ID is the unique identifier used to describe which axis is the 'X' axis in ProLase. Iwhen using Indexer LPT, this is a letter between 'a' and 'f' depending on the printer port address as described in the Indexer LPT manual.

Y, Z, A Axis ID identify their respective axis in the same manner as ***X Axis ID***.

XRes, YRes, ZRes are the parameters that describe the number of steps per unit (inch, feet, m or cm as selected in the Configuration Field page) of travel for each respective axis. This number is based on the number of stepper motor counts per motor revolution times the gearing ratio of the system. For example on a table system with a 5 pitch lead screw (5 revolutions per inch) and stepper motors set up for 200 steps per revolution will have a ***Res*** factor of 1000 (1000 total steps per one inch movement of the table).

These values are best determined experimentally. Using the External Axis control window, move an axis a number of steps and measure the resulting travel. Use this as the ***Res*** number for that axis. Test this in ProLase and adjust until the required resolution is obtained.

A negative ***Res*** may be used to move the tables in the opposite direction. However, the value must be an integer.

ARes is similar except that it describes the number of steps required to rotate a cylinder 360 degrees (one revolution). Remember this may also involve gearing.

This completes the set-up, making the table control ready to accept positioning commands from ProLase.

X/Y/Z/A Axis by Layer:

External axis motion is programmed on a layer by layer basis. First, right-click on the desired layer and select the Properties menu option. Next select the External Axis tab to modify the parameters which govern external axis motion.

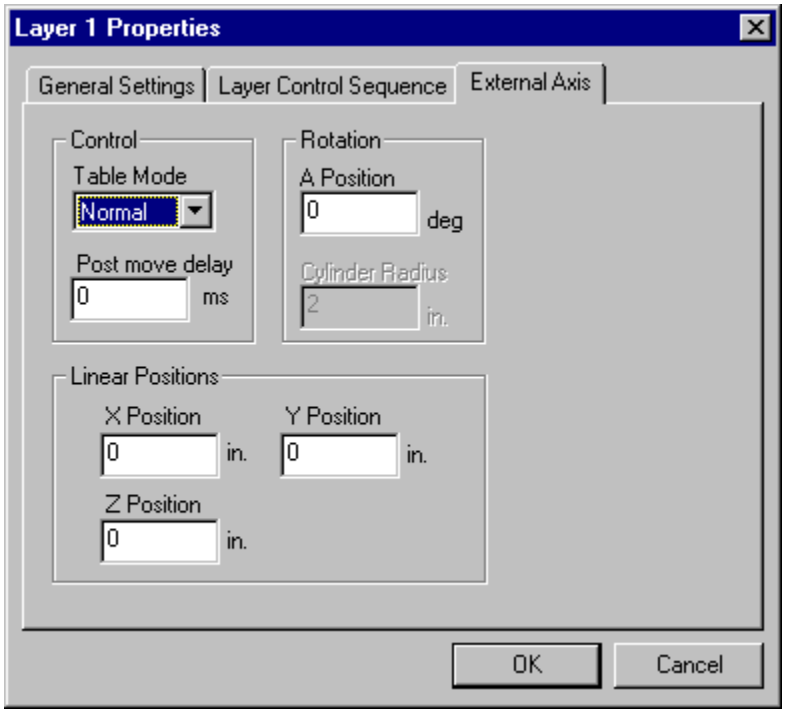
The *Post Move Delay* is provided to allow the tables to completely settle following motion commands. To maximize throughput, this value should always be set to the minimum value that provides acceptable mark quality.

The *Table Mode* selection determines how and when the Layer will move the tables. **Normal** mode allows the user to specify the four co-ordinates (X,Y,Z,A) for the layer, it's 'starting point'. At the start of any layer, the external axes move to re-center the marking field over the layer's position. Several layers within a program can be used to create marking programs that mark over an area much larger

than the scanning lens field. The only limitation is that each object within a layer must be marked within the lens field.

ProLase will not tile a graphic (break a large graphic up into a number of smaller graphic). To perform this operation properly, very precise knowledge about the hardware of the marking system is required. Since American Laserware, Inc. does not provide this hardware, no assurance can be made regarding the agreement of external axis stepper motor motion and the position response and optical distortion of the galvanometer marking head. Even tiny errors can result in unacceptable alignment between the ends of a single vector that spans multiple tiles. Hence, ProLase does not support this feature.

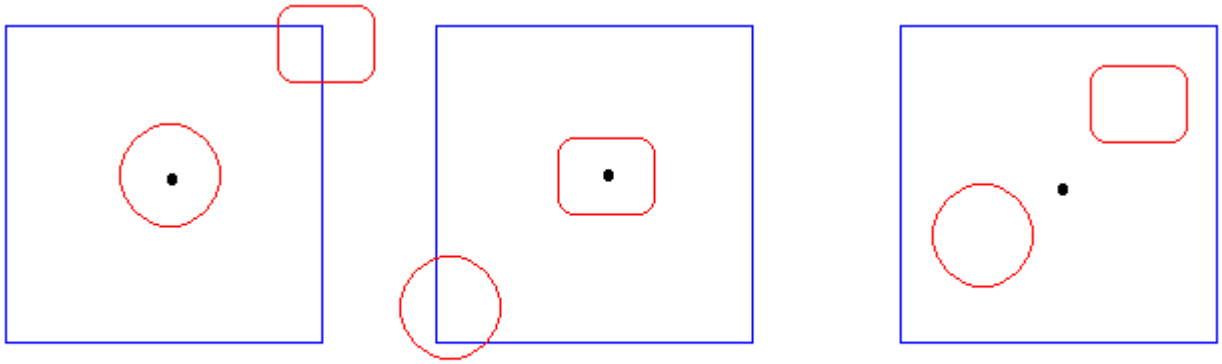
During part design, the blue Marking Field box (or circle) will be redrawn to represent the new location for the current layer in the ProLase world. All objects for that layer must be found completely within this



boundary or the laser will not be physically able to mark them.

AutoXY mode will have ProLase automatically move the tables as needed to mark all the objects on a layer. As ProLase marks the layer, the extents of each object are examined. If any part of the object is outside of the current lens field, ProLase repositions the tables to center the current object under to lens field.

AutoXY mode is useful and greatly simplifies the part design process. However, it may not be the most efficient. Since table motion can represent a significant portion of the cycle time, minimizing it by manual programming with Normal mode may be optimal. Consider the example below.

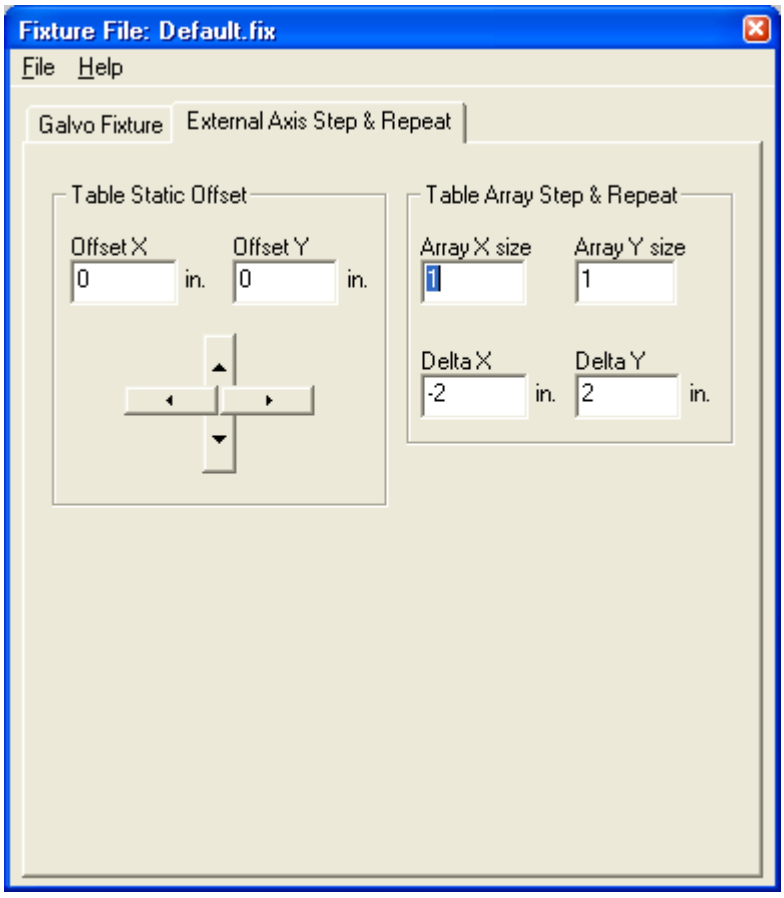


The diagram shows the same part (of 2 graphics) marked in both modes. The 2 left hand images show the mark process in AutoXY mode. In this mode, the table oscillates between the center of the 2 objects. The table moves twice for each part. The right hand image shows a Normal mode configuration with a judiciously chosen layer position. Both objects are completely within the lens field and no more part motion is required.

Cylindrical marking (A axis):

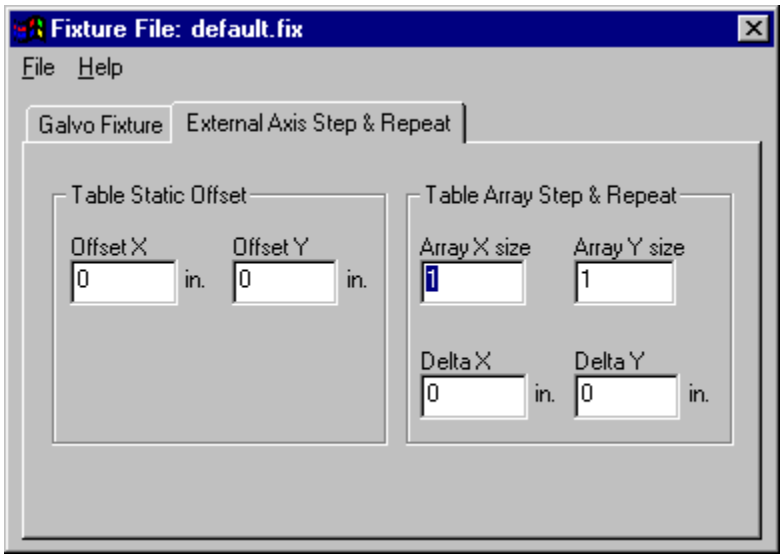
Cylindrical marking can be done by setting the *Table Mode* to **A as X** or **A as Y**. ProLase marks around a cylinder by positioning a rotational axis so that the tangent point is the center of a character or logo to be marked. ProLase can be configured to mark cylinders that rotate along either the X or Y axis. Graphically ProLase represents the cylinder as if it has been sliced along is rotational axis and laid out flat. The 0 degree position is at the zero axis position and the ruler represents arc length. The total arc length around the cylinder (circumference) is $2\pi R$, where R is the cylinder radius value specified. If a string of text is marked that is longer than the circumference then that sentence will end up writing over itself. When **A as X** is selected, text should be rotated to zero or 180 degrees. When **A as Y** is selected, text should be rotated to 90 or 270 degrees. For each object in the layer, ProLase calculates the appropriate angle based on position and mark accordingly. For text marking this means the motor will be rotated for each character. If a Z-axis is used it must be set to insure that the tangent point of the cylinder is at the focal point of the lens.

External Axis Step & Repeat:



ProLase can also move the X/Y table in a Step & Repeat (S&R) manner allowing the user to mark large, nested arrays of parts. The controls for this are found in the Fixture window on *the External Axis Step & Repeat* tab. This function works the same as the normal S&R except that the table is moved to a new location for each part (or nest of parts if Galvo S&R is used at the same time). If both Galvo and External Axis S&R are done in the same process the Galvo S&R is executed first. For example a 5x5 Galvo S&R (25 parts) can be placed within an external axis S&R of 2x5 allowing for marking of 250 (25 x 2 x 5) parts per run.

The *Table Static Offset* Offset X and Offset Y values and the *Table Array Step & Repeat* values for array sizes and spacing operate in the same manner as the corresponding value for the galvo fixture. There is no support for uniform rotation of the external table as the Angle offset provides for galvos. The four arrow buttons can be used to jog the Table to the correct static offset position.


Fixture File: default.fix
File
Help
Galvo Fixture
External Axis Step & Repeat
Table Static Offset
Offset X
0
in.
Offset Y
0
in.
Table Array Step & Repeat
Array X size
1
Array Y size
1
Delta X
0
in.
Delta Y
0
in.

Chapter 6 Supplemental Information ToolKit

Files and Directories:

This information is for the ProLase ToolKit product only. All information for ProLase7 for Windows or ProLase7 Plus is contained in other sections of this manual.

All the files needed for the dll are included in the ProlaseDLL directory <DevDir>. This directroy includes the .dll, .lib and .h files that are needed to build and execute Visual Studio C++ projects using ProLase.dll.

In addition to the files needed for compilation, there are a set of ProLase support files that the DLL needs access to at run time. These should be copied to the applications execution directory <InstallDir>. See appendix A for a complete list of support files ProLase (or any program using ProLase.dll) requires. Look at the end-user ProLase sub-directory structure after installation for a complete picture of the support files and sub-directories required for fully functional ProLase. To set everything up for development, install the included ProLase setup. This will install the end-user version of ProLase. UnZip the ProLase DLL project. Replace the end-user version of ProLase.exe with ProLase.exe from the DLL project. Now the end-user ProLase can be used to create “template” documents (LAZ files) that can be modified by the DLL.

Note the the DLLs themselves (ProLase.dll and Laser.dll, found in the ProlaseDLL subdirectory) must be present either in the application's execution directroy <InstallDir> or the Windows System directory <WinSys>. In reality there are two DLLs that are used with the toolkit. First is ProLase.dll which is an MFC extension DLL, requiring the MFC runtime library DLLs. Secondly is Laser.dll which contains the actual motion and laser control logic. ProLase.DLL is designed to be implicitly loaded and is built using the ProLase.lib export library. Laser.DLL is explicitly loaded by ProLase.DLL when certain ProLase.DLL member functions are executed. Laser.DLL does not require an import library. The software security system does not take affect until Laser.DLL is loaded. That means it is possible to have an application that includes ProLase.DLL that will load without the security locking device. However none of the member functions in ProLase.DLL can be called without the device. Any attempt to call those functions without the device will result in a “Security device not found” error message, and the application will be shut down.

The TESTDLL directory contains the files required to build and execute the sample application that uses the DLL. There is a 4.0 Developer Studio WorkSpace Project file. We currently use version 4.0 (Visual C++), but the project can loaded and converted by 5.0 or 6.0.

The TestDLL directory also contains ProLase for Windows 95, along with it's support files, fonts, import filters, etc. This can be used to create LAZ file templates that can be manipulated with the DLL.

Member variable naming convention:

American Laserware Inc. uses the following naming conventions for all member properties:

m_	member property
m_l	long (32 bit integer)
m_i*	int (16 bit integer)
m_b	BOOL (TRUE or FALSE)
m_f	float (normal floating point)
m_s	CString (C++ String)
m_n	UINT (unsigned integer)
m_u	unsigned char (unsigned 8 bit character)

*Note: int is really a 32 bit number in Windows. We use it to represent 16 bit numbers that may overflow. The range of values for data we specify as "int" is +/- 32767.

Units:

Several types of units are used in the software. Distance Units can be Inches, feet, Centimeters or meters. The selection is made by the operator on the Config/Field property page. Internally ProLase does not care what units are being used, except to print the correct units on various dialog boxes and property sheets. Distance units are stored internally as 16 bit numbers, scaled at .0001 distance unit increments. So if the system is configured to be in inches, then distance unit properties like X position would be 10000 counts per inch. Angular units are .1 degrees internally. To set an angle to 30 degrees, you would set the property to 300. Time units are labeled as used, either microseconds or millisecond depending on the property.

The coordinate system is a standard Cartesian coordinate system where 0,0 is the center of the field. Positive X is to the right and Positive Y is up (from the CRT frame of reference). A standard laser marker is oriented with the mark plane parallel to the ground. If oriented so that the X axis is east-west then west is negative X, east is positive X, north is positive Y and south is negative Y. Up and down is the focus axis (Z axis). Up is positive Z and down is negative Z.

Exported ProLase.dll Classes:

These are the basic classes exported by the DLL. These exports are true classes, which means the developer can derive classes from them. All supported member functions work across the EXE/DLL boundary except Cstring. Cstring objects seem to only work if the Cstring being changed is originally longer than the new value. This seems to be a bug in the Windows Operating system. There are two ways around this, the first is to make sure the Template LAZ file contains very long strings and always modify them with shorter strings. The second method is to use the American Laserware Inc. supplied SetItemText function to change the object text.

CWSData Class:

This class loads and manipulates a .laz (ProLase) document. In addition to the data corresponding to the layers and items, it also contains the material and fixture information that is attached to the .laz file.

There should be 1 instance of this class in each document of the application. (Assuming each document corresponds to a LAZ file as it does in ProLase7 for Windows).

Supported Member functions:

BOOL Load(LPCTSTR sPathNamePtr);

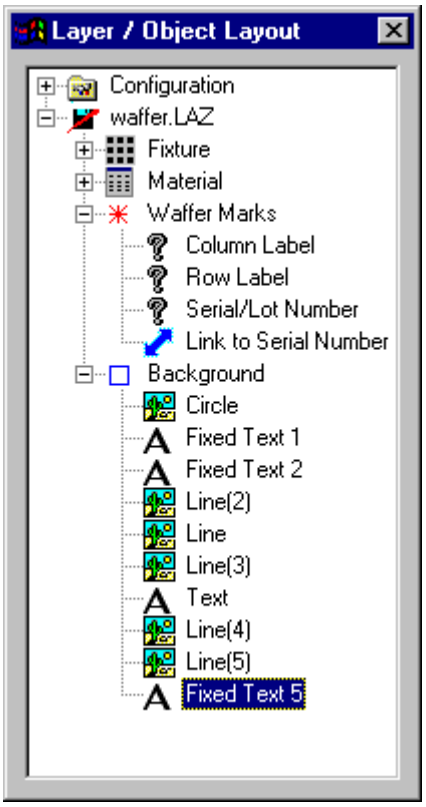
Loads a .laz (Prolase7) file. SPathNamePtr is a string that contains the complete path specifying the location of the LAZ document file to load. Load returns TRUE if the load was successful, otherwise a return of FALSE indicates an error condition

Example:

```
CWSData WSData;                 // declare an instance of WSData;
WSData.Load("Demo.laz");        // Load WSData object with Demo.laz  document file
```

CItem* operator ()(int nLayer, int nItem);

Returns a CItem pointer to point at the Item specified by nLayer and nItem. Each WSData object contains a zero based, two-dimensional array of objects (Items) and layers. The first layer shown in the Document is Layer 0, the first Item in Layer 0 is Item 0. In this way any Item in a document can be uniquely identified by two numbers, i.e. nLayer, nItem. The returned Item pointer can be used to access any of the Items methods or properties. A NULL pointer is returned if the specified object does not exist.



"Wafer Marks" is Layer 0, "Column Label" is Item 0 of Layer 0. "Background" is Layer 1 and "Circle" is Item 0 of Layer 1.

Example:

```
CItem *pItem;                   // declare Item pointer
pItem = WSData(0,0);            // set pointer for the first layer, first item in the document
```

```
pItem->m_lXPos = 0;                 // Sets Items X position to zero
WSData(0,0)->m_lXPos = 0;           // Does the same exact thing
```

CLayer* GetAt(int nLayer);

Returns a CLayer pointer for access to the specified Layer. nLayer is the zero based layer array ordinal. A NULL pointer is returned if the Layer does not exist.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first layer in Document
```

SetItemText(int nLayer, int nItem , LPCTSTR sTextString);

Sets the Text in a Fixed or Variable Text object.

Example:

```
WSData.SetItemText(0,0, “This is some text to be marked”);   // Set Layer 0, Item 0 Text property
```

SetItemFont(int nLayer, int nItem, LPCTSTR sFontName);

Sets the font file name for Fixed or Variable Text objects. This function works for *.ALF and *.FNT file types.

Example:

```
WSData.SetItemFont(0,0, “C:\ProLase\Complex.fnt”);      // Set Layer 0, Item 0 font file name.
```

WSData Properties:

m_Fixture

The m_Fixture object can be used to access the fixture database file linked to the WSData Document.

Example:

```
WSData.m_Fixture.m_ldOffsetAngle = 300;          // Set Fixture angular offset.  Angle Units are .1 degrees.
WSData.m_Fixture.m_lOffsetX = 20000;        // Set Fixture X Offset to +2 “.  Distance Units .0001 “.
WSData.m_Fixture.m_lOffsetY = -20000;            // Set Fixture Y Offset to –2”.
WSData.m_Fixture.m_lFixFocusAdjust = 10000; // Adjust Focus Height by +1”;
WSData.m_Fixture.m_nArrayX = 3;                  // Set up 3 x 4 array of parts, 3 parts wide
WSData.m_Fixture.m_nArrayY = 4;                  // 4 parts high.
WSData.m_Fixture.m_lDeltaX = 5000;               // .5 “ spacing between columns
WSData.m_Fixture.m_lDeltaY = -7500;              // .75 “ spacing between rows
```

m_Material

The m_Material object can be used to access the material database file linked to the WSData Document.

Example

```
WSData.m_Material.m_adFreq[0] = 5.0;          // Sets the Pass 1 frequency to 5 kHz.
WSData.m_Material.m_adFreq[1] = 6.0;          // Sets the Pass 2 frequency to 6 kHz.
        .
        .
        .
WSData.m_Material.m_adFreq[6] = 25.0;         // Sets the Pass 7 frequency to 25 kHz.


WSData.m_Material.m_adSpeed[0] = 10.0;        // Sets the Pass 1 speed to 10 “/sec
        .
        .
        .
WSData.m_Material.m_adSpeed[6] = 32.0;        // Sets the Pass 7 speed to 32”/sec


WSData.m_Material.m_adPower[0] = 5.0;         // Sets the Pass 1 Power to 5.0
        .
        .
        .
WSData.m_Material.m_adPower[6] = 22.0;        // Sets the Pass 7 Power to 22.0
```

m_sApsFile

The m_sApsFile is the CString that contains the name and path of the Material. The name is stored on disk as a relative Path, but is always converted on a <InstallDir> based absolute path during the Serialize in process. M_sApsFile will always be an Absolute path.

Example:

```
CString MaterialFileName = WSData.m_sApsFile;          // Get The Material File Name
```

m_sFixFile

The m_sFixFile is the CString that contains the name and path of the Material. The name is stored on disk as a relative Path, but is always converted on a <InstallDir> based absolute path during the Serialize in process. M_sApsFile will always be an Absolute path.

Example:

```
CString FixtureFileName = WSData.m_sFixFile;
```

CLayer Class:

This class provides access to the Layer properties. A Layer is a member of the WSData class. A pointer used to access the Layer properties can be obtained from the WSData class using the WSData::GetAt(int nLayer) function.

CItem* GetAt(int nItem);

Returns a CItem pointer for access to the specified Item. nItem is the zero based Item array ordinal. A NULL pointer is returned if the Item does not exist.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first layer in Document
CItem *pItem = pLayer->GetAt(0);           // Get pointer to first Item in first layer
```

int GetNumberItems(void);

Returns the total number of Items in the Layer.

Example:

```
Int TotalNumberOfObjects = *pLayer->GetNumberItems();
```

CLayer Properties:

m_bActive

Holds the Active/Background state of the Layer. TRUE means the layer is Active and will mark. FALSE means the Layer is background only, and will not be marked.

Example:

```
CLayer *pLayer0 = WSData.GetAt(0);          // Get pointer to first Layer
CLayer *pLayer1 = WSData.GetAt(1);
pLayer0->m_bActive = TRUE;                  // Set first Layer to Active
pLayer1->m_bActive = FALSE;                 // Set Second Layer to Background
```

m_sLayerName

Holds the CString Layer name. This name corresponds to the name displayed in the ProLase Layer/Item Tree control.

Example

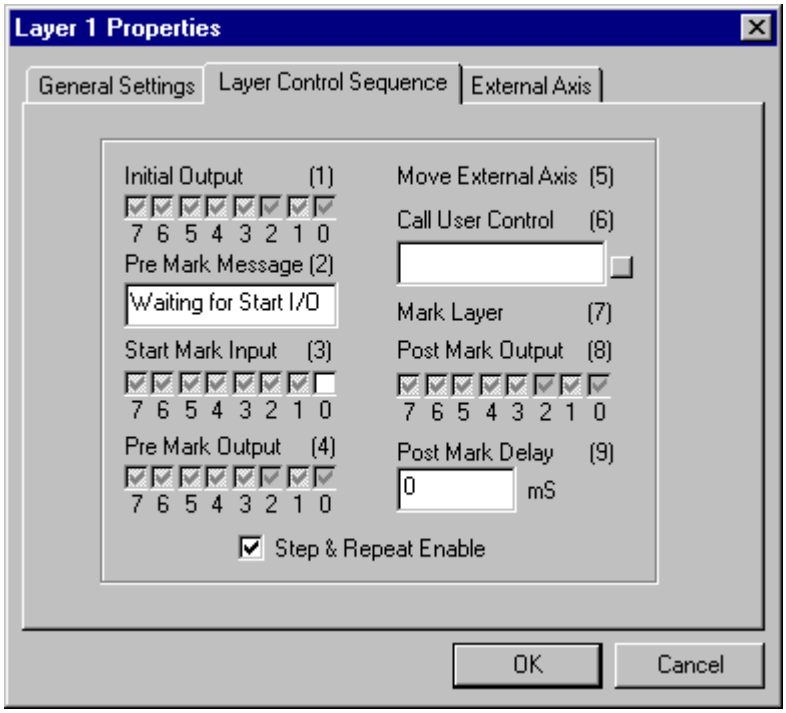
```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to the first Layer
CString LayerName = pLayer->m_sLayerName;// Get the Layer Name
```

m_uInitOut, m_uInitOutMask

Holds the tri-state programming of the Initial Output I/O control for the Layer. The Initial Output is set on the digital I/O port at the start of the Layer mark processing. This property is displayed as a Layer property on the Layer Control Sequence property sheet. The eight bits of the Mask property correspond to the eight output bits available on the I/O card. A bit value of 1 means that bit is to be set, a value of zero means that bit should be left in its current state. The 8 bits of the m_uInitOut property are the actual values to be set. For example if bit 7 of the m_uInitOutMask property is a one, and bit 7 of the m_uInitOut property is a zero, then the hardware bit 7 output will be set to zero. If on the other hand bit 7 of the Mask is set to zero, then the hardware bit 7 will be left unchanged. In the ProLase control, grayed check marks means the Mask bit is zero (leave Unchanged), black checks indicate the Mask bit is set to one, and the corresponding m_uInitOut bit is set to one (force the output bit to 1). A black box without any check means the Mask bit is set to one, and the corresponding m_uInitOut bit is set to zero (force the output bit to 0). Bits 0 and 2 are used internally by ProLase for the GATE and NOT GATE signals, and should not be changed (their Mask bits should always be set to zero).

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to the first Layer
pLayer->m_uInitOutMask = 0x80;             // Set bit 7 mask bit set
pLayer->m_uInitOut = 0xff;                 // Set bit 7 to ON, the other bits are left // //unchanged even though they are set in the //m_bInitOut instruction. This is because the //Mask value for these other bits is zero, and //therefor they are left unchanged.
```



m_sPreMarkMessage

Holds the CString that contains the Pre Mark Message displayed by ProLase during mark processing.

Example:

```
CLayer *pLayer = WSData.GetAt(0);                          // Get pointer to the first Layer
pLayer->m_sPreMarkMessage = “Press Mark Button to Begin”;  // Inform Operator how to begin
```

m_uPreIn, m_uPreInMask

Holds the tri-state programming of the Start Mark Input. This describes the Input port condition required to start marking the Layer. A Mask bit value of zero means that the input bit will be ignored. In other words, if the Mask bit is zero, any state of the corresponding input bit will meet the start requirements. In ProLase grayed checks mean “don’t care”, black checks mean the corresponding bit must be a one, and a blank means the corresponding bit must be a zero.

Example:

```
 CLayer *pLayer = WSData.GetAt(0);          // Get pointer to the first Layer
pLayer->m_uPreInMask = 0x07;                // Set bits 1,2,3 mask
pLayer->m_uInitOut = 0x03;                  // Set bits 1,2 to ON, and bit 3 to off.  The other
                                            //bits are ignored.  This Layer will not mark until
                                            //the Input I/O port has bits 1 and 2 set to a logic
                                            //high and bit 3 to a logic low.  The state of the
                                            //other bits does not matter since the Mask is set
                                            //to zero for those bits.
```

m_uPreOut, m_uPreOutMask

Holds the tri-state programming of the Pre Mark Output I/O control for the Layer. The Pre Mark Output is set on the digital I/O port after the start conditions are met and just before actual marking begins. The eight bits of the Mask property correspond to the eight output bits available on the I/O card. A bit value of 1 means that bit is to be set, a value of zero means that bit should be left in its current state. The 8 bits of the m_uPreOut property are the actual values to be set. For example if bit 7 of the m_uPreOutMask property is a one, and bit 7 of the m_uPreOut property is a zero, then the hardware bit 7 output will be set to zero. If on the other hand bit 7 of the Mask is set to zero, then the hardware bit 7 will be left unchanged. In the ProLase control, grayed check marks means the Mask bit is zero (leave Unchanged), black checks indicate the Mask bit is set to one, and the corresponding m_uPreOut bit is set to one (force the output bit to 1). A black box without any check means the Mask bit is set to one, and the corresponding m_uPreOut bit is set to zero (force the output bit to 0). Bits 0 and 2 are used internally by ProLase for the GATE and NOT GATE signals, and should not be changed (their Mask bits should always be set to zero).

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to the first Layer
pLayer->m_uPreOutMask = 0x80;              // Set bit 7 mask bit set
pLayer->m_uPreOut = 0xff;                  // Set bit 7 to ON, the other bits are left // //
                                           //unchanged even though they are set in the
                                           //m_uPreOut instruction. This is because the
                                           //Mask value for these other bits is zero, and
                                           //therefor they are left unchanged.
```

m_uPostOut, m_uPostOutMask

Holds the tri-state programming of the Post Mark Output I/O control for the Layer. The Post Mark Output is set on the digital I/O port after the completion of the marking for that layer. The eight bits of the Mask property correspond to the eight output bits available on the I/O card. A bit value of 1 means that bit is to be set, a value of zero means that bit should be left in its current state. The 8 bits of the m_uPostOut property are the actual values to be set. For example if bit 7 of the m_uPostOutMask property is a one, and bit 7 of the m_uPostOut property is a zero, then the hardware bit 7 output will be set to zero. If on the other hand bit 7 of the Mask is set to zero, then the hardware bit 7 will be left unchanged. In the ProLase control, grayed check marks means the Mask bit is zero (leave Unchanged), black checks indicate the Mask bit is set to one, and the corresponding m_uPostOut bit is set to one (force the output bit to 1). A black box without any check means the Mask bit is set to one, and the corresponding m_uPostOut bit is set to zero (force the output bit to 0). Bits 0 and 2 are used internally by ProLase for the GATE and NOT GATE signals, and should not be changed (their Mask bits should always be set to zero).

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to the first Layer
pLayer->m_uPostOutMask = 0x80;             // Set bit 7 mask bit set
pLayer->m_uPostOut = 0xff;                 // Set bit 7 to ON, the other bits are left // //
                                           //unchanged even though they are set in the
```

```
                                        //m_uPostOut instruction.  This is because the
                                        //Mask value for these other bits is zero, and
                                        //therefor they are left unchanged.
```

m_iPostMarkDelay

Holds the value for the post mark delay. This can be used to wait a fixed time period before going to Process the next Layer. This delay is in milliseconds, i.e. 1000 = 1 second.

Example

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to the first Layer
pLayer->m_iPostMarkDelay = 2000;           // Set Post Mark Delay to 2 seconds
```

CItem Class:

This class provides access to the Item properties. An Item is a member of the CLayer class. A pointer used to access the Item properties can be obtained from the CLayer class using the CLayer::GetAt(int nItem) function. Items have no useful external methods, but have lots of properties.

CItem Properties:

m_sItemName

Contains the CString name of the Item.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
CString ItemName = pItem->m_sItemName;     // Get the Item Name.
```

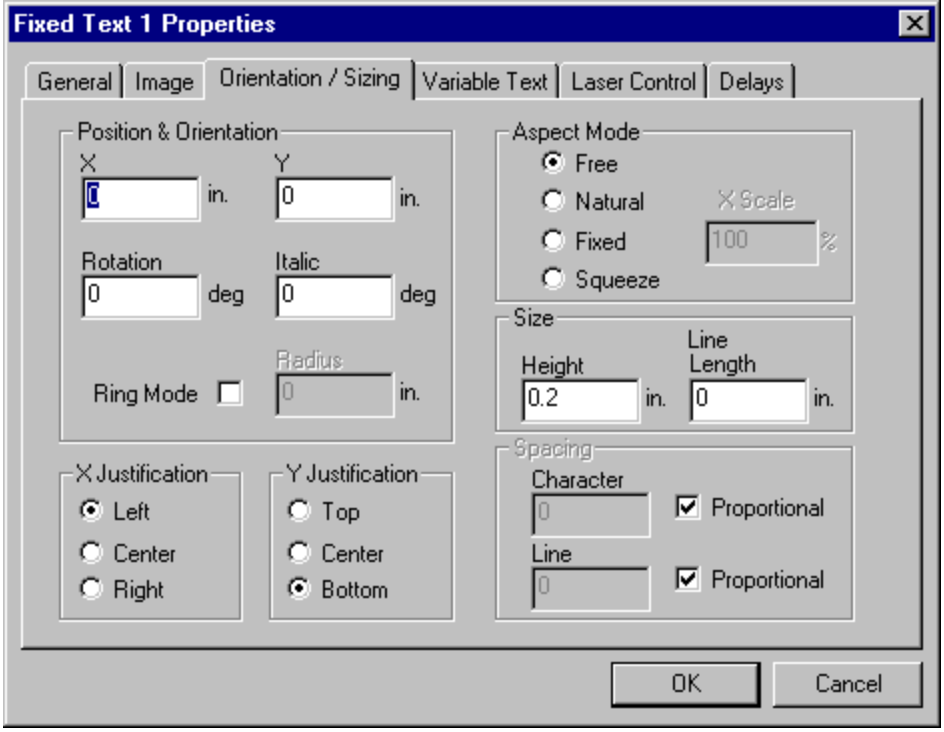
m_sFileName

Contains the CString graphic file name for the Item. The graphic filet name is an absolute path and name.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
CString GraphicFileName = pItem->m_sFileName;  // Get the Graphic file Name.
```

m_lXPos



Contains the X position for the Item. 0 is the center of the field on the X axis. The units are .0001 of the user units. If the user units is inches then 10000 = 1 inch.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
pItem->m_lXPos = 25000;                    // Sets the X Position to 2.5 user distance units
```

m_lYPos

Contains the Y position for the Item. 0 is the center of the field on the Y axis. The units are .0001 of the user units. If the user units is inches then 10000 = 1 inch.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
pItem->m_lYPos = -20000;                   // Sets the Y Position to -2.0 user distance units
```

m_iAngle

Contains the rotation angle for the Item. 0 is the angle where a perpendicular to the sentence axis is pointed north. Positive angles rotate clockwise (like a compass). The units are .1 degrees.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
pItem->m_iAngle = 900;                     // Sets the Angle to +90 degrees
```

m_iItalic

Contains the Italic or tilt angle for the Item. 0 is the angle where a perpendicular to the sentence axis is pointed north. This results in no tilt. Positive angles rotate clockwise (like a compass). The units are .1 degrees.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
pItem->m_iItalic = 50;                     // Tilts the Item by 5 degrees
```

m_lRadius

Contains the Ring Radius for marking on the circumference of a ring. 0 indicates the object is to be drawn flat (default). The units are .0001 user distance units, same as the position units. A positive radius will mark clockwise on the ring (bottoms of letters towards the center of the ring). A negative radius will mark counter clockwise on the ring.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
pItem->m_lRadius = 20000;                  // Mark on ring of radius 2.0 user distance units.
```

m_nXJust, m_nYJust

The justification properties control how an object is drawn about the m_lXPos and m_lYPos coordinates. If both X just and Y just are set to center then the object will drawn such that the center of the object will be at m_lXPos, m_lYPos.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
pItem->m_nXJust =     JX_LEFT;
                      JX_CENTER;
                      JX_RIGHT;            // Sets X Justification (1 if 3 values)
pItem->m_nYJust =     JY_BOTTOM;
                      JY_CENTER;
                      JY_TOP;              // Sets Y Justification (1 of 3 values)
```

m_lHeight

Contains the Height for the Item. For text objects this is the Height of a standard capital letter, like the letter “A”. For Graphics it is the total height of the extents of the graphic. The units are .0001 of the user distance units. If the user units is inches then 10000 = 1 inch.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
```

```
pItem->m_lHeight = 20000;                    // Sets the Height 2.0 user distance units
```

m_lLength

Contains the Length for the Item. For Natural aspect mode objects with proportional character spacing, m_lLength is zero, indicating the length should correspond to the natural length of the object based on the Height. For Free aspect mode with proportional character spacing, Length is the Length of a sentence for text object, for graphic objects Length is the total length of the graphic. For fixed space Text objects, Length refers to the character length (or character width). The units are .0001 of the user distance units. If the user units is inches then 10000 = 1 inch.

Example:

```
CLayer *pLayer = WSData.GetAt(0);            // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);            // Get pointer to first Item of first Layer
pItem->m_lHeight = 20000;                    // Sets the Height 2.0 user distance units
```

m_lCharSpace

Contains the character spacing property. If 0, then the character will be marked with the natural spacing built into the font. Fixed spacing (m_lCharSpace > 0) is used for dials and rulers. Characters will be drawn such that their centers will be equally spaced by the m_lCharSpace value.

Example:

```
CLayer *pLayer = WSData.GetAt(0);            // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);            // Get pointer to first Item of first Layer
pItem->m_lCharSpace = 5000;                  // Sets the Char Space to .5 user distance units
```

m_lLineSpace

Contains the sentence line spacing property. If 0, then the lines of text will be marked with the natural spacing built into the font. Fixed spacing (m_lLineSpace > 0) is used for dials and rulers. Sentences will be drawn such that their centers will be equally spaced by the m_lLineSpace value.

Example:

```
CLayer *pLayer = WSData.GetAt(0);            // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);            // Get pointer to first Item of first Layer
pItem->m_lLineSpace = 2000;                  // Sets the Char Space to .2 user distance units
```

m_bMirrorX, m_bMirrorY

Contains the mirror flags for the object. MirrorX = TRUE means the object will be drawn mirrored about the X axis. MirrorY = TRUE will cause the object to be mirrored about the Y axis. Setting one of the Mirrors to TRUE will cause the object to be drawn “inside out”.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
pItem->m_bMirrorX = TRUE;                  // Mirror about the X axis
pItem->m_bMirrorY = TRUE;                  // Mirror about the Y axis
```

m_bUseMaterialFile

The laser process parameters (writing speed, laser power, pulsing frequency) can be calculated based on a material file or specified by object. The m_bUseMaterialFile property determines which set of numbers will be used. If m_bUseMaterialFile = FALSE, then the Object values will be used, else values will be calculated based on the Material file entries and the object percent modifiers.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
pItem->m_bUseMaterialFile = TRUE;          // Specify use of Material File
```

m_fFreq2

The Object laser pulsing frequency in kHz. This value will be used to mark the object if m_bUseMaterialFile is set to FALSE.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
PItem->m_bUseMaterialFile = FALSE;
pItem->m_fFreq2 = 5.0;                     // Specify use 5 kHz laser pulsing
```

m_fPower2

The Object laser power in user selected units (usually either AMPS or WATTS). This value will be used to mark the object if m_bUseMaterialFile is set to FALSE.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
PItem->m_bUseMaterialFile = FALSE;
pItem->m_fPower2 = 50.0;                   // Specify use 50 watts
```

m_fSpeed2

The Object writing speed in user distance units per second. This value will be used to mark the object if m_bUseMaterialFile is set to FALSE.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
PItem->m_bUseMaterialFile = FALSE;
pItem->m_fSpeed2 = 50.0;                   // Specify use 50 user distance units per second
```

m_iFreqFac

The Percentage of the Material File specified Frequency to use during marking. The resultant value will be used to mark the object if m_bUseMaterialFile is set to TRUE. Units are in .1 % increments.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
PItem->m_bUseMaterialFile = TRUE;
pItem->m_iFreqFac = 500;                   // Specify 50% of Material File frequency value
```

m_iPowerFac

The Percentage of the Material File specified Power to use during marking. The resultant value will be used to mark the object if m_bUseMaterialFile is set to TRUE. Units are in .1% increments.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
PItem->m_bUseMaterialFile = TRUE;
pItem->m_fPower2 = 750;                    // Specify use 75% of Material File Power value
```

m_fSpeedFac

The Percentage of the Material File specified Speed to use during marking. The resultant value will be used to mark the object if m_bUseMaterialFile is set to TRUE. Units are in .1% increments.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
PItem->m_bUseMaterialFile = FALSE;
pItem->m_fSpeed2 = 1500;                   // Specify 150.0% if Material File Speed Value
```

m_uCondMark, m_uCondMarkMask

Holds the tri-state programming of the Conditional Mark Input. This describes the Input port condition required to mark the object. If the condition is not met the object is skipped.. A Mask bit value of zero means that the input bit will be ignored. In other words, if the Mask bit is zero, any state of the corresponding input bit will meet the mark

condition. In ProLase grayed checks mean “don’t care”, black checks mean the corresponding bit must be a one, and a blank means the corresponding bit must be a zero.

Example:

```
CLayer *pLayer = WSData.GetAt(0);          // Get pointer to the first Layer
CItem * pItem = pLayer->GetAt(0);          // Get pointer to first Item of first Layer
pItem->m_uCondMarkMask = 0x07;             // Set bits 1,2,3 mask
pItem->m_uCondMark = 0x03;                 // Set bits 1,2 to ON, and bit 3 to off. The other
                                           //bits are ignored. This Object will be skipped
                                           //unless the Input I/O port has bits 1 and 2 set to
                                           //a logic high and bit 3 to a logic low. The state
                                           //of the other bits does not matter since the
                                           //Mask is set to zero for those bits.
```

CMark Class:

This class contains the vector information and the laser marking engine. It also contains the configuration data. There should be only 1 instance of this class in the application. This should be placed either in the Mainframe or CApp class. The Cmark object should be created by initializing a pointer as follows:

```
CMark m_pMarkEngine;

m_pMarkEngine = CMark::NewMark();
```

The pointer should be used to access all member functions of CMark.

The CMark object will automatically initialize the interface cards according to the driver selected in the Configuration. The simplest thing to do is set up and test the configuration using the ProLase7 program. You can directly read and write to the DIO ports if you wish. You MUST NOT change bit 0 and bit 2 of the output port, these are used internally as the GATE and NOT GATE signals. The chip used for the DIO allows you to read the outputs, so read them and use ANDs and ORs to only change the bits you want to change and leave bits 0 and 2 unchanged. For the DDA06 card the input port address is 0x33c,and the output port is 0x33d. These are 8 bit ports so use the _inp and _outp commands. If you decide to control the I/O yourself make sure your "template" has all Layer I/O settings of "don't care" (XXXXXXXX). Also make the ABORT OUTPUT property of the parts program are all "don't care".

CConfig* GetConfig();

Returns a pointer to the ProLase Configuration object.

Example:

```
CMark *m_pMarkEngine = CMark::NewMark();                        // Create MarkEngine Object
M_pMarkEngine->GetConfig()->Load((LPCTSTR) “Default.cfg”);      // Load the Config File
```

SimpleMarkRun(CWSData* pData, int iNumberParts);

Marks all layers and all objects in the passed WSData object. All I/O will be in effect and behave as programmed.

Example:

```
CMark *m_pMarkEngine = Cmark::NewMark();                        // Create MarkEngine Object
m_pMarkEngine->GetConfig()->Load((LPCTSTR) “Default.cfg”);      // Load the Config File
CWSData WSData;                                                 // declare an instance of WSData;
WSData.Load(“Demo.laz”);                                        //Load WSData object with
                                                                //Demo.laz document file
m_pMarkEngine->SimpleMarkRun(&WSData,5);                        // Mark a run of 5 parts.
```

void SetDLLLayerMode(int iMode);

int GetDLLLayerMode(void);

The layer mode setting affects the way the toolkit processes layers during a call to SimpleMarkRun().The layer mode is set to MARK_DLLLAYERMODE_NORMAL upon creation and remains constant unless changed by using SetDLLLayerMode() or re-initialization. GetDLLLayerMode() is provided for access to the current setting. The argument for SetDLLLayerMode() must be one of the following constants defined in the Mark.h file:

MARK_DLLLAYERMODE_NORMAL
- Layer processing halts only for layer with specified start conditions (NOT all ‘don’t cares’).
- No visual indication is provided.
- Layer start signals are accepted via the general purpose digital I/O or the <ENTER> key.

MARK_DLLLAYERMODE_MODAL_ALL
- Layer processing halts for ALL layers regardless of specified start conditions.
- Message box displayed with Pre-Mark Message.
- Message box OK/CANCEL is ONLY layer control input. NO digital I/O start signal.

MARK_DLLLAYERMODE_MODAL_SOME
- Layer processing halts only for layer with specified start conditions (NOT all ‘don’t cares’).
- Message box displayed with Pre-Mark Message.
- Message box OK/CANCEL is ONLY layer control input. NO digital I/O start signal.

Example:

```
CMark *m_pMarkEngine = Cmark::NewMark();                        // Create MarkEngine Object
m_pMarkEngine->GetConfig()->Load((LPCTSTR) “Default.cfg”);      // Load the Config File
CWSData WSData;                                                 // declare an instance of WSData;
WSData.Load(“Demo.laz”);                                        //Load WSData object with
                                                                //Demo.laz document file
m_pMarkEngine->SetDLLLayerMode(MARK_DLLLAYERMODE_MODAL_SOME);
m_pMarkEngine->SimpleMarkRun(&WSData,5);                        // Mark a run of 5 parts.
```

CDisplay

This class creates and displays the screen images.

There should be 1 instance of this class for each view of the application.

Drill Objects:

The ProLase DLL contains additional features for drilling applications. A set of functions is available for creating and marking various drill objects. An example of the use of these drilling features can be found in the DrillDlg.h and .cpp files in the TestDLL project. See below for more details. For all of the functions, the position values are given in 1 / 10,000 units and the angle values are in 1 / 10 degrees. For instance a position of (0.5 in, 1.25 in) would be given as (5000 , 125000) and an angle of 45 degrees is given as 450. nRepRate is in pulses per second (pps).

The following functions are used to assign drill object properties to existing CItem instances.

```
void CItem::CreateDrillSpot(long lX, long lY, UINT nRepRate, UINT nPasses);
```

Positions the laser spot at (lX, lY) and fires nPasses laser pulses at the frequency corresponding to nRepRate pps.

```
void CItem::CreateDrillLine(long lX1, long lY1, long lX2, long lY2, long
            lVel, UINT nRepRate, UINT nPasses);
```

Draw nPasses lines from (lX1, lY1) to (lX2, lY2). When on, the laser spot moves at lVel velocity and is operating at the frequency corresponding to nRepRate pps.

```
void CItem::CreateDrillCircle(long lCenX, long lCenY, long lX, long lY,
            long lVel, UINT nRepRate, UINT nPasses);
```

Draws nPasses circles centered at (lCenX, lCenY). (lX, lY) is a point on the circle and marks the start/end point of the mark. When on, the laser spot moves at lVel velocity and is operating at the frequency corresponding to nRepRate pps.

```
void CItem::CreateDrillTrepan(long lX, long lY, long lAngle, long
             lDiameter, long lVel, UINT nRepRate, UINT nPasses, long
             lDeltaAngle = 0);
```

Draws nPasses trepans centered at (lX, lY) with a lDiameter OUTER diameter size. When on, the laser spot moves at lVel velocity and is operating at the frequency corresponding to nRepRate pps. lAngle specifies the angle offset of the tangent point of the inner and outer circles from the center horizontal. lDeltaAngle is an incremental change in the angle that is applied for each additional pass.

```
void CItem::CreateDrillSpiral(long lX, long lY, long lAngle, long
            lInnerDia, long lOuterDia, long lPitch, long lVel, UINT
            nRepRate, UINT nPasses, long lDeltaAngle = 0);
```

Draws nPasses spirals centered at (lX, lY) with a lInnerDia inner diameter, lOuterDia outer diameter and lPitch spacing between marks along the same radial. When on, the laser spot moves at lVel velocity and is operating at the frequency corresponding to nRepRate pps. lAngle specifies the angle offset of the tangent point of the inner and

outer circles from the center horizontal. lDeltaAngle is an incremental change in the angle that is applied for each additional pass. The spiral generated will always begin and end at the specified angle. This may extend the spiral past the specified outer diameter at the specified pitch until the current revolution is complete and has a diameter greater than the outer diameter specified.

```
void CItem::CreateDrillSpiral2(long lX, long lY, long lAngle, long
            lInnerDia, long lOuterDia, long lPitch, long lVel, UINT
            nRepRate, UINT nPasses, long lDeltaAngle = 0);
```

Spiral2 is the same as Spiral except that where a normal spiral would terminate, a sprial2 will continue to mark at the outer diameter until it intersects itself.

```
void CItem::CreateDrillMove(long lX1, long lY1, long lX2, long lY2, long
            lVel, UINT nPasses);
```

Move is the same as a line except that the laser is NEVER turned on while marking it. It can be used for controlled movement of the galvos at precise speeds rather than the high-speed slewing normally used during moves between marks.

The following CWSData member functions will create objects in the same manner as the CItem functions listed above. Rather than modifying and existing object however, these functions will return a reference to newly created CItem. The calling program is responsible for freeing the memory associated with the referenced CItem.

```
CItem *CWSData::CreateDrillSpot(long lX, long lY, UINT nRepRate, UINT nPasses);

CItem *CWSData::CreateDrillLine(long lX1, long lY1, long lX2, long lY2, long lVel,
            UINT nRepRate, UINT nPasses);

CItem *CWSData::CreateDrillCircle(long lCenX, long lCenY, long lX, long lY, long
             lVel, UINT nRepRate, UINT nPasses);

CItem *CWSData::CreateDrillTrepan(long lX, long lY, long lAngle, long lDiameter,
            long lVel, UINT nRepRate, UINT nPasses, long lDeltaAngle = 0);

CItem *CWSData::CreateDrillSpiral(long lX, long lY, long lAngle, long lInnerDia,
            long lOuterDia, long lPitch, long lVel, UINT nRepRate, UINT nPasses,
            long lDeltaAngle = 0);

CItem *CWSData::CreateDrillSpiral2(long lX, long lY, long lAngle, long lInnerDia,
            long lOuterDia, long lPitch, long lVel, UINT nRepRate, UINT nPasses,
            long lDeltaAngle = 0);

CItem *CWSData::CreateDrillMove(long lX1, long lY1, long lX2, long lY2, long lVel,
            UINT nPasses);
```

The CItems created or modified by the previous functions are expected to be marked using the CMark::MarkSingleItem member function. The CMark class provides additional functions that encapsulate this functionality. These functions create and mark the drill object in one simple step using the same arguments as the previous CItem and CWSData functions. All these functions simply return the result of the internal MarkSingleItem call.

```
UINT CMark::MarkDrillSpot(long lX, long lY, UINT nRepRate, UINT nPasses);

UINT CMark::MarkDrillLine(long lX1, long lY1, long lX2, long lY2, long lVel, UINT
           nRepRate, UINT nPasses);

UINT CMark::MarkDrillCircle(long lCenX, long lCenY, long lX, long lY, long lVel,
           UINT nRepRate, UINT nPasses);

UINT CMark::MarkDrillTrepan(long lX, long lY, long lAngle, long lDiameter, long
           lVel, UINT nRepRate, UINT nPasses, long lDeltaAngle = 0);

UINT CMark::MarkDrillSpiral(long lX, long lY, long lAngle, long lInnerDia, long
           lOuterDia, long lPitch, long lVel, UINT nRepRate, UINT nPasses, long
           lDeltaAngle = 0);

UINT CMark::MarkDrillSpiral2(long lX, long lY, long lAngle, long lInnerDia, long
           lOuterDia, long lPitch, long lVel, UINT nRepRate, UINT nPasses, long
           lDeltaAngle = 0);

UINT CMark::MarkDrillMove(long lX1, long lY1, long lX2, long lY2, long lVel, UINT
           nPasses);
```

TestDLL Sample Application:

In the TestDLL demonstration application, all the code needed to implement the Prolase.DLL is racketed by:

/*** PROLASE DLL ***/
/******************/

to make it easy to find. This application demonstrates the easiest way to use the DLL, which is to create a Template LAZ file using the standard ProLase front end program. The Template should contain the desired number and types of objects. As many properties as possible should be set in the Template so the DLL will only have to set those properties that will change during the processing. The ProLase front end should also be used to create and test Fixture, Material, and the Configuration files. Again this is to avoid setting a lot of properties in the DLL that will never change. It is much faster, easier and simpler to test these properties using the ProLase front end. The frame work for the sample application was created using the Visual Studio Apps Wizard. Only a small number of additions were required to implement laser marking using ProLase.DLL. The Sample Application can load any LAZ Document file and mark it. The “Test” menu selection creates a WSData Object, a MarkEngine Object, loads CONFIG.SYS, loads DEMO.LAZ, changes several object properties and marks the resultant WSData Object.

6 files have been altered:

MainFrm.cpp	MainFrm.h
TestDLLDoc.cpp	TestDLLDoc.h

TestDLLView.cpp TestDLLView.h

MainFrm.h:
Included ProlaseDLL\Prolase.h
Declared CMark as a member of the mainframe

MainFrm.cpp
In the constructor (CMainFrm::CMainFrm), first
load the config file, then create the mark engine.

TestDLLDoc.h
Included ProlaseDLL\Prolase.h
Declared CWSData as a member of the Doc
Declared a flag to tell us if the WSData class
is valid (has been loaded)

TestDLLDoc.cpp
Included mainfrm.h --- Document needs to access the mainfrm,
or where the CMark class is a member of
In the constructor, set the valid flag to false
In OnFileOpen, allow the user to select a file,
then load the file.
In OnFileMark, if the data is valid, get a
pointer to the mainframe and call the mark engine's
marking routine

TestDLLView.h
Included ProlaseDLL\Prolase.h
Declared an instance of the display class as a member.

TestDLLView.cpp
In OnInitialUpdate, create the display
In OnDraw:
Zoom to the field size
Draw the field box
Draw all the items
Update the screen
In OnSize:
First two times, don't do anything

After that, call the display's onsize function
with the same parameters

The simplest way to use this is to create a ‘template’ of items and layers, using the regular

ProLase7 for Windows editor, and modifying the properties using the examples above. You can find the complete list of member properties in the ITEM.H file.

AXLase Active X Sample Application:

The AXLase sample is an Active X control project using the ProLase7 DLL to perform its functions. American LaserWare has included a basic set of interfaces that, when possible, parallels the COM automation interface documented in Chapter 7 of this manual. Users are invited to add functionality to this Control, all source code is included. A VB sample project called “TestActiveX” is included which uses the LaseAX Active X control. A very similar VB project called “TestVB” uses the COM AUTOMATION capability described in Chapter 7. One of the differences between the Active X control and the COM AUTOMATION is the function “sCreateMarkEngine(ProLasePath)”. This function creates a LAZ object in the Active X control, and must be executed before any loading of LAZ files. The parameter “ProLasePath” is a path to a ProLase7 installation folder. This is where the Active X control can find all the support files (fonts, .CFG, box, circle, etc.) needed to run. The default installation creates “C:\Program Files\American LaserWare\ProLase7\” as the installation folder. So an example of the sCreateMarkEngine function would be “sCreateMarkEngine(“C:\Program Files\American LaserWare\ProLase7\”.

Chapter 7 Supplemental Information ProLase7 Server

Command Line interface:

ProLase7 command line interface is a method for calling Prolase7 from another program or a DOS command line, passing parameters such as LAZ file name, mode, and quantity to mark.

PROLASE7.EXE lazfilename, mode, quantity

Modes:

AUTO	Goes to mark dialog, mark button pushed. Remains in GUI when done.
AUTOEXIT	Same as AUTO except exits when done.
SIM	Same as AUTO except SIMULATE button pushed for testing.
SIMEXIT	Same as SIM except exits when done, also used for testing.

For example the command line: PROLASE7.EXE SAMPLE.LAZ AUTOEXIT 12, will command ProLase to come up with the file SAMPLE.LAZ in the mark mode, waiting for start signals. As soon as ProLase has marked 12 parts it will quit back to the command line.

Automation Server interface:

ProLase Server is a COM based automation interface to the core marking technology within the ProLase marking software. When this interface is used ProLase runs in the background providing services to a customer created user interface. Since ProLase Server is based on COM software technology its services can be accessed from any language that supports COM. This includes Visual C++, Visual Basic, and many others. The interface is defined through a Type Library (.TLB) file, called “Prolase.TLB). In Visual C++, an interface class can be automatically created through Class Wizard. Simply go to the AUTOMATION tab and press the ADD CLASS button. Select FROM A TYPE LIBRARY. Use the file browser to select “ProLase.TLB”. In Visual Basic use the PROJECT/REFERENCES menu selection and use the BROWSE button to select “ProLase.TLB”. The OBJECT BROWSER tool can be used to view the interface.

There are several groups of functionality including Document functions, Layer functions, Item functions, and marking control functions. American LaserWare uses a function naming convention as follows: Document functions begin with “d”, Layer functions begin with “l”, Item functions begin with “I”, and marking control functions begin with “m”. Some controller development enviroments (like Visual Basic for example) uses the term Properties for some functions, even though these are functions from the servers point of view (GET, SET functions). These Properties still follow the naming conventions. Any GET functions without a SET function listed means the property is meant to be read only, for example the marking status functions. The interface includes the SET functions but they will not do anything if

called.

Document Functions

BOOL **dLoadLazDocument**(**LPCTSTR** *lazfilename*)

lazfilename is the complete name and path to a ProLase created .LAZ document file. This function loads the document into the ProLase Server. Returns TRUE if the document successfully loaded.

SHORT **dSaveLazDocument**(**LPCTSTR** *lazfilename*)

lazfilename is the complete name and path for the .LAZ document file. This function save the current document to the specified file. Returns 1 if successful.

BSTR *Get***dFreq (short** *pass* **)**
This function returns the laser frequency for the specified *pass* in the current Material file.

VOID *Set***dFreq (short** *pass***, float** *newValue* **)**
This function sets the laser frequency for the specified *pass* of the current Material file to *newValue*.

BOOL **dIsDocumentLoaded ( )**
This function returns TRUE if the previous call to *dLoadLAZDocument(…)* was successful; FALSE otherwise.

BOOL **dIsValidItem (short** *LayerIndex*, **short** *ItemIndex* **)**
This function returns TRUE if *LayerIndex* and *ItemIndex* specify an existing item on an existing layer in the currently loaded document; FALSE otherwise.

BOOL **dIsValidLayer (short** *LayerIndex* **)**
This function returns TRUE if *LayerIndex* specifies an existing layer in the currently loaded document; FALSE otherwise.

BOOL **dLoadLazDocument (LPCTSTR** *LazFileName* **)**
LazFileName is the complete name and path to a ProLase created LAZ document file. This function loads the document into the ProLase Server. It returns TRUE is the document is successfully loaded; FALSE otherwise.

BSTR *Get***dMaterialFileName ( )**
This function returns the name and path of the material file to which the currently loaded document is linked.

VOID *Set***dMaterialFileName (LPCTSTR** *lpsznewValue* **)**
This function sets the material file to which the currently loaded document is linked to *lpsznewValue*, which must be a complete path to a ProLase material file. If the specified file exists, it is loaded into memory, otherwise a default material file is used.

SHORT **dSaveMaterialFile (LPCTSTR** *Name* ***)***

Saves the current material files settings to the file specified by *Name*.

float *Get***dMaterialPower (short** *Pass* **)**
Returns the laser power setting in the material file associated with pass *Pass*.
VOID *Set***dMaterialPower (short** *Pass*, **float** *newValue* **)**
This function sets the material file's laser power setting for pass *Pass* to *newValue*.

float *Get***dMaterialFreq (short** *Pass* **)**
Returns the laser frequency setting (in kHz) in the material file associated with pass *Pass*.
VOID *Set***dMaterialFreq (short** *Pass*, **float** *newValue* **)**
This function sets the material file's laser frequency setting for pass *Pass* to *newValue*. *newValue* = 0.0 cause CW lasing.

float *Get***dMaterialSpeed (short** *Pass* **)**
Returns the laser speed setting (in units/second) in the material file associated with pass *Pass*.
VOID *Set***dMaterialSpeed (short** *Pass*, **float** *newValue* **)**
This function sets the material file's laser speed setting for pass *Pass* to *newValue*.

SHORT *Get***dMaterialQWidth (short** *Pass* **)**
Returns the laser Q-Switch pulse width setting in the material file associated with pass *Pass*.
VOID *Set***dMaterialQWidth (short** *Pass*, **SHORT** *nNewValue* **)**
This function sets the material file's Q-Switch pulse width setting for pass *Pass* to *nNewValue*.

SHORT *Get***dMaterialQWidthDefault (short** *Pass* **)**
Returns the laser Q-Switch pulse width default setting flag in the material file associated with pass *Pass*. If set, the Q-Switch pulse width value specified in the configuration file is used. When not set, the material file Q-Switch pulse width for this pass is used.
VOID *Set***dMaterialQWidthDefault (short** *Pass*, **SHORT** *nNewValue* **)**
If *nNewValue* = 1, this function sets the material file's Q-Switch pulse width default flag setting for pass *Pass*; otherwise, the flag is cleared and the configuration file setting is used.

SHORT *Get***dMaterialCW (short** *Pass* **)**
Returns the laser CW flag state in the material file associated with pass *Pass*. If set, the frequency is set to 0 kHz and CW mode is indicated. When not set, the laser is pulsed at the material file frequency for this pass.
VOID *Set***dMaterialCW(short** *Pass*, **SHORT** *nNewValue* **)**
If *nNewValue* = 1, this function sets the material file's CW flag for pass *Pass*; otherwise, the flag is cleared and the laser is pulsed at the frequency specified for this pass.

BSTR *Get***dPower (short** *pass* **)**
This function returns the laser power for the specified *pass* in the current Material file.
VOID *Set***dPower (short** *pass***, float** *newValue* **)**
This function sets the laser power for the specified *pass* of the current Material file to *newValue*.

BSTR *Get***dSpeed (short** *pass* **)**
This function returns the laser speed for the specified *pass* in the current Material file.
VOID *Set***dSpeed (short** *pass***, float** *newValue* **)**
This function sets the laser speed for the specified *pass* of the current Material file to

newValue.

FLOAT *Get***fOffsetAngle ( )**
Returns the value of the field Angle offset of the current fixture file in degrees.
VOID *Set***fOffsetAngle (float** *newValue* **)**
Sets the field Angle offset to *newValue*.

FLOAT *Get***fOffsetX ( )**
Returns the value of the field X offset of the current fixture file in the current configuration units.
VOID *Set***fOffsetX (float** *position* **)**
Sets the field X Offset to *newValue*.

FLOAT *Get***fOffsetY ( )**
Returns the value of the field Y offset of the current fixture file in the current configuration units.
VOID *Set***fOffsetY (float** *position* **)**
Sets the field Y Offset to *newValue*.

Layer Functions

Most Layer functions access the Layer through use of a zero based index(**short** *layerindex …)*. Most Layer functions are actually Layer Properties in the ProLase UI, and consist of GET, SET functions.

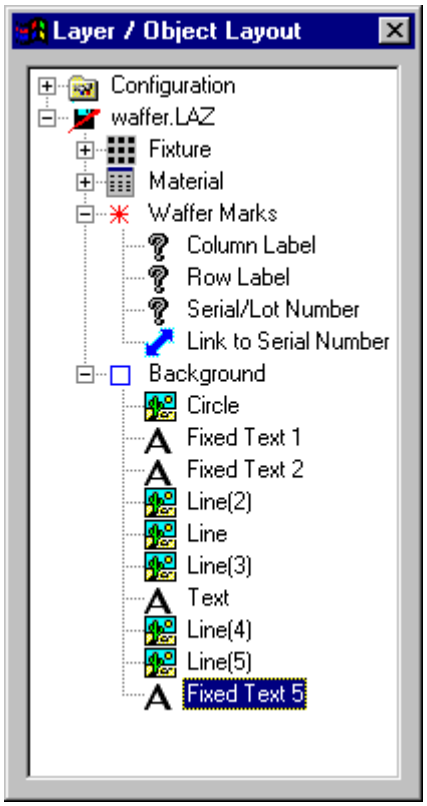
BSTR *Get***lLayerName (short** *LayeIindex* **)**
Returns the current value of the layer's name.
VOID *Set***lItemName (short** *LayerIndex***, LPCTSTR** *newValue* **)**
This function sets the layer's name.

Item Functions

For the purposes of the server, ProLase marked objects are called 'Items'. Hopefully this will be less confusing since the word 'object' refers to so many things when programming. A ProLase Item consists of a paragraph of fixed text, a line of variable text, a graphic, an item link, a line, or a drill object.

Most Item functions access the Item through use of two indices (**short** *layerindex*, **short** *itemindex*…): The zero based index to the Layer that contains the Item, and the zero based index (within the Layer) to the Item itself. In the example the Layer named 'Wafer Marks' is *layerindex* = 0, 'Backgound' is *layerindex* = 1, etc. The Item named 'Column Label' is *layerindex* = 0, *itemindex* = 0. 'Link to Serial Number' is *layerindex* = 0, *itemindex* = 3. 'Circle' is *layerindex* = 1, *itemindex* = 0. As with Layers, most Item functions are actually 'Object Properties' in the ProLase UI and consist of GET, SET functions.

Dimensional information (XY position, height, length, etc.) is in whatever units are currently set in the configuration. For example if ProLase is in inch mode, SetiXPosition(0,0,1.003) will set the items X Position to 1.003 inches. If ProLase were in centimeter mode then the item would be set to 1.003 cm. X=0, Y=0 is the center of the mark area, with positive values being to the right and up. Negative values are left and down from center.



SHORT *Get***iActive (short** *LayerIndex*, **short** *ItemIndex* **)**

This function returns whether the specified item is Active or used for alignement. A return value of 1 indicates Active = TRUE, 0 indicates Active = FALSE.

VOID *Set***iFixedText (short** *Layelindex*, **short** *ItemIndex*, **short** *nNewValue* **)**

This function sets the specified item to be Active during normal marks or to be used for alignment purposed only. Set nNewValue = 1 for Active, nNewValue = 0 to specify an alignement object.

FLOAT *Get***iAngle (short** *Layelindex*, **short** *ItemIndex* **)**

Returns the current value of the item's rotation angle.

VOID *Set***iAngle(short** *LayerIndex*, **short** *ItemIndex, **float** newValue* **)**

This function sets the item's rotation angle. Units are in degrees. The convention is like a compass: 0 degrees is North and angles increase clockwise. The range is 0 - 360.

SHORT *Get***iAspectMode (short** *LayerIndex***, short** *ItemIndex* **)**
Returns the current Aspect Mode setting.

VOID *Set***iAspectMode (short** *LayerIndex***, short** *ItemIndex, **short** nNewValue***)**
Sets the value of the items Aspect Mode to *nNewValue*, where

0 = Free
1 = Natural
2 = Fixed
3 = Squeeze

If Squeeze is selected (3), then the SetiFixedFactor(…) function should be used to set the fixed aspect factor.

FLOAT *Get***iCharSpacing (short** *Layelindex***, short** *ItemIndex* **)**
Returns the current value of the item's character spacing, where 0 indicates proportional character spacing.

VOID *Set***iCharSpacing (short** *LayerIndex***, short** *ItemIndex, **float** newValue* **)**
This function sets the item's character spacing in the same units as the item's height parameter. The character spacing parameter is the center-to-center distance of adjacent characters in fixed spacing mode. Use *newValue* = 0 for proportional character spacing.

FLOAT *Get***iFixedFactor (short** *LayerIndex***, short** *ItemIndex* **)**
Returns the current iterm's Fixed Factor value.

VOID *Set***iFixedFactor (short** *LayerIndex***, short** *ItemIndex, **float** newValue* **)**
Sets the value of the item's Fixed Factor. This value sets the aspect of an item as a percentage of Natural Aspect. *newValue* is given as a percentage. For example *newValue* = 50 would compress the item length to one half of its natural length at the current height. *newValue* = 200 would expand the length to twice the natural length at the current height.

BSTR *Get***iFixedText (short** *LayerIndex***, short** *ItemIndex***, short** *TextLine* **)**
This function returns the specified line of the item's fixed text. *TextLine* is the zero based line of text to be returned.

VOID *Set***iFixedText (short** *LayerIndex***, short** *itemindex***, short** *TextLine***, LPCTSTR** *lpszNewValue***)**
This function sets the specified line of the item's fixed text. *TextLine* is the zero based line of text to be set (Fixed text objects can have up to 10 lines of text), and *lpszNewValue* is the string of text to set.

BSTR **iGetFontName (short** *LayerIndex***, short** *ItemIndex* **)**
This function returns the name of the item's This string may be the complete path and file name of a Laser Engraving font, a bar code symbology name, the dot matrix format, or the face name of a TrueType (or OpenType) font.

SHORT **iGetFontTTFBold (short** *LayerIndex***, short** *ItemIndex* **)**
This function returns whether or not the bold flag is set for the item's font. If the item is set to

use a TrueType (or OpenType) font and the bold option is selected, the function return value is 1; 0 otherwise.

SHORT **iGetFontTTFCharSet (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns the character set for the item's font. If the item is set to use a TrueType (or OpenType) font, the function return value is the character set value of the font; 0 otherwise.

SHORT **iGetFontTTFHKern (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns the height base kerning factor for the current font. If the item is set to use a TrueType (or OpenType) font, the function return value is the item's HKern value; 0 otherwise.

SHORT **iGetFontTTFItalic (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns whether or not the italic flag is set for the item's font. If the item is set to use a TrueType (or OpenType) font and the italic option is selected, the function return value is 1; 0 otherwise.

SHORT **iGetFontTTFKernMode (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns the kerning mode for the current font. If the item is set to use a TrueType (or OpenType) font and the item is set to use Window's kerning pairs, the function return value is 1; 0 otherwise.

SHORT **iGetFontTTFLKern (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns the length base kerning factor for the current font. If the item is set to use a TrueType (or OpenType) font, the function return value is the item's LKern value; 0 otherwise.

SHORT **iGetFontTTFStrike (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns whether or not the strikethrough flag is set for the item's font. If the item is set to use a TrueType (or OpenType) font and the strike through option is selected, the function return value is 1; 0 otherwise.

SHORT **iGetFontTTFUnder (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns whether or not the underline flag is set for the item's font. If the item is set to use a TrueType (or OpenType) font and the underline option is selected, the function return value is 1; 0 otherwise.

SHORT **iGetFontType (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns a value based on the type of font the item has selected. The return values may be as follows:

0 = Laser Enagraving
1 = TrueType (or OpenType)
2 = Bar Code

3 = Dot Matrix

BSTR *Get***iGraphicFileName (short** *LayerIndex***, short** *ItemIndex* **)**
This function returns the name and path for the graphic file. If this name was set by ProLase (original name in the LAZ document) then it will always be an absolute path.

VOID *Set***iGraphicFileName (short** *LayerIndex***, short** *ItemIndex,* **LPCTSTR** *lpszNewValue* **)**
This function sets the name of the desired graphic file. The path must be relative to where ProLase Server is located. Of course if a complete absolute path is provided there is no question where the file is.

SHORT *Get***iGraphicType (short** *LayerIndex***, short** *ItemIndex* **)**
This function returns the graphic type of the object. The return values may be as follows:

0 = Vector graphic
1 = Raster graphic

VOID *Set***iGraphicType (short** *LayerIndex***, short** *ItemIndex,* **SHORT** *nNewValue* **)**
This function sets the type of the specified graphic. See *Get***iGraphicType** for allowed values of *nNewValue*.

FLOAT *Get***iHeight (short** *LayerIndex***, short** *ItemIndex* **)**
Returns the current value of the item's height.

VOID *Set***iHeight (short** *LayerIndex***, short** *ItemIndex,* **float** *newValue* **)**
This function sets the item's height. For Graphic items, this is the total height of the graphic. For text items, it is the character height. If the item is in Natural Aspect mode then the length will change with height to keep the item at natural aspect. Height controls the item size, *SetiLength(…)* commands will have no effect in Natural Aspect mode.

In Free Aspect mode, height and length are controlled separately.

FLOAT *Get***iLength (short** *LayerIndex***, short** *ItemIndex* **)**
Returns the current value of the item's total length.

VOID *Set***iLength (short** *LayerIndex***, short** *ItemIndex,* **float** *newValue* **)**
This function sets the item's length. Setting *newValue* = 0 will force the object to Natural Aspect, no matter which Aspect Mode is set. This command will have no effect in Natural Aspect mode. In Free and Free Aspect modes, height and length are controlled separately. In Squeeze mode, this will set the maximum length of the text. The text will be compressed from Natural Aspect as needed to limit the text at the specified height to the length specified here.

FLOAT *Get***iLineSpacing (short** *LayeIindex***, short** *ItemIndex* **)**
Returns the current value of the item's line spacing, where 0 indicates proportional line spacing.

VOID *Set***iLineSpacing (short** *LayerIndex***, short** *ItemIndex,* **float** *newValue* **)**
This function sets the item's line spacing in the same units as the item's height parameter. The line spacing parameter is the baseline-to-baseline distance of adjacent text lines fixed line spacing mode. Use *newValue* = 0 for proportional line spacing which will space the lines at 1.5 times the height value.

BOOL **iReloadGraphic(short** *LayerIndex***, short** *ItemIndex***)**

This function reloads the items graphic from disk to graphic memory. Use this function if using a generic graphic name in the document, but the user wants to copy different graphic files to that name. For example, if the document has an item linked to TEMP.DXF as the graphic name and the user wants to copy different DXF files to TEMP.DXF before marking. *iReloadGraphic(…)* needs to be called after the copy and before the mark so that the new TEMP.DXF is copied into memory instead of the old one. The old version of TEMP.DXF is purged from ProLase's graphic memory, releasing the memory space. This is one way to implement a variable graphic.

Another way would be to simply change the name of the item's graphic using the *SetiGraphicName(…)* function. This technique will keep all of the previous graphics in memory, so if the name is changed to something that has been loaded before, it will exist in graphic memory without loading from disk.

This function always returns TRUE.

FLOAT *Get***iRingModeRadius (short** *LayeIindex***, short** *ItemIndex* **)**

Returns the current value of the item's radius for ring mode marking, where 0 indicates normal, linear text.

VOID *Set***iRingModeRadius (short** *LayerIndex***, short** *ItemIndex,* **float** *newValue* **)**

This function sets the item's ring mode radius in the same units as the item's height parameter. Use *newValue* = 0 for normal, linear text. Positive values result in clockwise text, negative values yield counterclockwise text.

BSTR *Get***iSerialNumber (short** *LayeIindex***, short** *ItemIndex* **)**

Returns the current value of the item's serial number string.

VOID *Set***iSerialNumber (short** *LayerIndex***, short** *ItemIndex,,* **LPCTSTR** *newValue* **)**

This function sets the item's serial number strings. newValue must be a singe line, but may contain any character. Number and letters in newValue will be incremented by marking (as specified elsewhere in the object parameters) and any other character will be static placeholders.

BSTR **iSetFontLaserEngraving (short** *LayeIindex***, short** *ItemIndex* **,LPCTSTR** *Name* **)**

This function sets the item's font type to Laser Enagraving and the item's font to the file name specified in *Name*, which should be a complete path and file name to the desired font. The return value is the item's new font, which may be an empty string if the item specified is invalid.

BSTR **iSetFontLaserEngraving (short** *LayeIindex***, short** *ItemIndex* **,LPCTSTR** *Name,* **short** *Bold***, short** *Italic,* **short** *Under,* **short** *Strike,* **short** *CharSet,* **short** *KernMode,* **short** *HKern,* **short** *LKern* **)**

This function sets the item's font type to True Type and the item's font to the file name specified in *Name*, which must be the complete face name of the desired font. Each of the font effects (*Bold*, *Italic*, *Under*, and *Strike*) are individually set to 1 to apply that effect or 0 to not apply it. *CharSet* must be the specified character set parameter for the selected font. *KernMode* = 1 indicates that the standard Windows kerning pairs will be used to render the font. This is the 'Kerning Pairs' option in the Object property pages from the ProLase GUI. *KenrMode* = 0 applies the faster ProLase kerning method which uses the *HKern* and *LKern* values to add kerning based on the individual character's height and length values,

respectively. *HKern* and *LKern* are given in percentage values with the range of 0 – 99. The return value is the item's new font, which may be an empty string if the item specified is invalid.

BOOL **iUseRealPosSize (short** *LayerIndex***, short** *ItemIndex* **)**
This function applies only to vector graphic items that are linked to source files created by CAD packages. This function locates the item to the real-world position and size specified in the source graphic file. The values will be scaled to match the units chosen in the configuration. The item will be bottom-left justified and set to natural aspect mode. This function will fail if the graphic position or size exceeds the world size. This function returns TRUE if successful, otherwise FALSE.

BOOL **iUseRealSize (short** *LayerIndex***, short** *ItemIndex* **)**
This function applies only to vector graphic items that are linked to source files created by CAD packages. This function scales the height and length of the item to the real-world dimensions specified in the source graphic file. The values will be scaled to match the units chosen in the configuration. This function will fail if the graphic size exceeds the world size. This function returns TRUE if successful, otherwise FALSE.

SHORT *Get***iXJustify (short** *Layerindex***, short** *Itemindex* **)**
This function returns the X justification (along its length) for the specified object.

0	=	Left
1	=	Center
2	=	Right

VOID *Set***iXJustify (short** *Layerindex***, short** *Itemindex*, **short** *nNewValue* **)**
This function will set the X justification (along its length) of the specified object to the value given by *nNewValue*. See GetiXJustify() above for valid arguments.

BOOL *Get***iXMirror (short** *Layerindex***, short** *Itemindex* **)**
This function returns TRUE is the object is mirrored in the X axis (along its length), FALSE otherwise.

VOID *Set***iXMirror (short** *Layerindex***, short** *Itemindex*, **BOOL** *bNewValue* **)**
This function will mirror the in the X axis (along its length) if bNewValue is TRUE or turn off mirroring if bNewValue is FALSE.
Note: Mirroring occurs about the object's location and will change the extents of the object if it is not center justified.

FLOAT *Get***iXPosition (short** *LayerIndex***, short** *ItemIndex* **)**
Returns the current value of the item's X Position.

VOID *Set***iXPosition (short** *LayerIndex***, short** *ItemIndex, **float** newValue* **)**
This function sets the X Position. Units are the same as set in the ProLase configuration file. For example, SetiXPosition(0,0,1.003) sets the first items X position to 1.003 inches if ProLase is configured for inches, or 1.003 cm if ProLase is configured for centimeters.

SHORT *Get***iYJustify (short** *Layerindex***, short** *Itemindex* **)**
This function returns the Y justification (along its height) for the specified object.

0 = Bottom
1 = Center
2 = Top

VOID *Set***iYJustify (short** *Layerindex***, short** *Itemindex*, **short** *nNewValue* **)**
This function will set the Y justification (along its height) of the specified object to the value given by *nNewValue*. See GetiYJustify() above for valid arguments.

BOOL *Get***iYMirror (short** *Layerindex***, short** *Itemindex* **)**
This function returns TRUE is the object is mirrored in the Y axis (along its height), FALSE otherwise.

VOID *Set***iYMirror (short** *Layerindex***, short** *Itemindex*, **BOOL** *bNewValue* **)**
This function will mirror the in the Y axis (along its height) if bNewValue is TRUE or turn off mirroring if bNewValue is FALSE.
Note: Mirroring occurs about the object's location and will change the extents of the object if it is not center justified.

FLOAT *Get***iYPosition (short** *LayerIndex***, short** *ItemIndex* **)**
Returns the current value of the items Y Position.

VOID *Set***iYPosition (short** *LayerIndex***, short** *ItemIndex, **float** newValue* **)**
This function sets the Y Position.

SHORT *Get***iItemType (short** *LayerIndex***, short** *ItemIndex* **)**
This function returns the type of the specified object.

1 = Fixed Text
2 = Variable Text
3 = Graphic
4 = EMPTY
5 = Data Link
6 = Line
7 = RESERVED
8 = Drill Tool

-1 if the *LayerIndex* and *ItemIndex* do not specify a valid object.

VOID *Set***iItemType (short** *Layerindex***, short** *Itemindex*, **short** *nNewValue* **)**
This function will set the type of the specified object to the value given by *nNewValue*. See GetiItemType() above for valid arguments.

BSTR *Get***iItemName (short** *LayeIindex***, short** *ItemIndex* **)**
Returns the current value of the item's name.

VOID *Set***iItemName (short** *LayerIndex***, short** *ItemIndex,,* **LPCTSTR** *newValue* **)**
This function sets the item's name.

BSTR *Get***iVarTextPrompt (short** *LayeIindex***, short** *ItemIndex* **)**
Returns the current value of the item's variable text prompt string.

VOID *Set***iVarTextPrompt (short** *LayerIndex***, short** *ItemIndex,,* **LPCTSTR** *newValue* **)**
This function sets the item's variable text prompt string. newValue must be a singe line.

SHORT *Get***iVarTextPromptRate (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns the prompting rate of the specified object.

0 = Every part
1 = Every batch
2 = Never
-1 if the *LayerIndex* and *ItemIndex* do not specify a valid object.

VOID *Set***iVarTextPromptRate (short** *Layerindex***, short** *Itemindex*, **short** *nNewValue* **)**

This function will set the variable text prompting rate of the specified object to the value given by *nNewValue*. See GetiVarTextPromptRate() above for valid arguments.

BSTR *Get***iVarTextString (short** *LayeIindex***, short** *ItemIndex* **)**

Returns the current value of the item's variable text string.

VOID *Set***iVarTextString (short** *LayerIndex***, short** *ItemIndex,,* **LPCTSTR** *newValue* **)**

This function sets the item's variable text string. newValue must be a singe line.

SHORT *Get***iVarTextType (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns the variable text type of the specified item. Note that this parameter exists and can be set for ANY item, but will only have an effect if the item type is set to variable text.

0 = Keyboard
1 = Datecode
2 = Dynamic File
3 = Serial Number
4 = List File
-1 if the *LayerIndex* and *ItemIndex* do not specify a valid object.

VOID *Set***iVarTextType (short** *Layerindex***, short** *Itemindex*, **short** *nNewValue* **)**

This function will set the variable text type of the specified item to the value given by *nNewValue*. See GetiVarTextType() above for valid arguments.

SHORT *Get***iRasterRenderMode (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns the render mode for the specified raster graphic object.

0 = Black / White
1 = Connect the dots
2 = Greyscale
3 = Natural
-1 if the *LayerIndex* and *ItemIndex* do not specify a valid object.

SHORT *Get***iRasterRenderMode (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns the dither method for the specified raster graphic object.

0 = NONE
1 = Bayer
2 = Burkes
3 = Floyd-Steinberg
-1 if the *LayerIndex* and *ItemIndex* do not specify a valid object.

SHORT *Get***iRasterNegative (short** *LayerIndex***, short** *ItemIndex* **)**

This function returns whether the specified raster graphic object is rendered as a photographic negative.

0 = Normal

1 = Photo negative
-1 if the *LayerIndex* and *ItemIndex* do not specify a valid object.

SHORT *Get***iRasterResolution (short** *LayerIndex***, short** *ItemIndex* **)**
This function returns the rendered resolution (dot density) for the specified raster graphic object. The return value is in 'dots per unit', where unit is the unit specified by the current configuration, eg. 'dots per inch' or 'dots per cm'. The return value is -1 if the *LayerIndex* and *ItemIndex* do not specify a valid object.

SHORT *Get***iRasterBrightness (short** *LayerIndex***, short** *ItemIndex* **)**
This function returns the brightness adjustment value for the specified raster graphic object. The return value is in the range -255…+255.

SHORT *Get***iRasterBiDir (short** *LayerIndex***, short** *ItemIndex* **)**
This function returns whether the specified raster graphic object is marked by-directionally. This value only applies to graphics rendered in 'Connect the dots' mode.
0 = Uni-directional marking
1 = Bi-Directional marking
-1 if the *LayerIndex* and *ItemIndex* do not specify a valid object.

SHORT **iSetRasterValues (short** *Layelindex***, short** *ItemIndex* ,**short** *RenderMode,* **short** *Dither,* **short** *Negative,* **short** *Resolution,* **short** *Brightness,* **short** *BiDir* **)**
This function sets the item's raster parameters. See *Get***iRasterRenderMode**, *Get***tRasterDither**, *Get***iRasterNegative**, *Get***iRasterResolution**, *Get***iRasterBrightness**, and *Get***iRasterBiDir** for allowed values. This function always returns 0.

SHORT *Get***iUseMaterialFileSettings (short** *Layelindex***, short** *ItemIndex* **)**
Returns 1 if the specified Item uses the material file lasers parameters. Returns 0 if item specific lasers parameters are used. Returns -1 if *LayerIndex* and *ItemIndex* do not specify a valid item.

VOID *Set***iUseMaterialFileSettings (short** *LayerIndex***, short** *ItemIndex,,* **short** *nNewValue* **)**
If *nNewValue* = 1, the specified item will use the material file settings. Item specific settings will be used otherwise.

SHORT *Get***iPassFlags (short** *Layelindex***, short** *ItemIndex* **)**
The set bits in the least significant byte of the return value indicate which material file passes will be used to mark the specified item. Bit 1-7 indicate Pass 1-7, respectively. Bit 0 has no meaning. Returns 0 if *LayerIndex* and *ItemIndex* do not specify a valid item.

VOID *Set***iPassFlags (short** *LayerIndex***, short** *ItemIndex,,* **short** *nNewValue* **)**
Sets the specified item to use the passes indicated by the set bits of the least significant byte of *nNewValue*.

SHORT *Get***iUseMaterialFileSettings (short** *Layelindex***, short** *ItemIndex* **)**
Returns 1 if the specified Item uses the material file lasers parameters. Returns 0 if item specific lasers parameters are used. Returns -1 if *LayerIndex* and *ItemIndex* do not specify a valid item.

VOID *Set***iUseMaterialFileSettings (short** *LayerIndex***, short** *ItemIndex,,* **short** *nNewValue* **)**
If *nNewValue* = 1, the specified item will use the material file settings. Item specific settings will be used otherwise.

SHORT *Get***iPasses (short** *LayeIindex***, short** *ItemIndex* **)**
Returns the number of passes used to mark the item when using item specific laser settings. Returns 0 if *LayerIndex* and *ItemIndex* do not specify a valid item.
VOID *Set***iPasses(short** *LayerIndex***, short** *ItemIndex,,* **short** *nNewValue* **)**
Sets the item specific laser setting number of passes to N*newValue*.

FLOAT *Get***iPowerFactor (short** *LayeIindex***, short** *ItemIndex* **)**
Returns the item's percentage power adjustment to the material file setting. Returns -1.0 if *LayerIndex* and *ItemIndex* do not specify a valid item.
VOID *Set***iPowerFactor(short** *LayerIndex***, short** *ItemIndex,,* **float** *newValue* **)**
Sets the item's percentage power adjustment to *newValue*. 0.0 < *newValue* < 200.0

FLOAT *Get***iFrewFactor (short** *LayeIindex***, short** *ItemIndex* **)**
Returns the item's percentage frequency adjustment to the material file setting. Returns -1.0 if *LayerIndex* and *ItemIndex* do not specify a valid item.
VOID *Set***iFreqFactor(short** *LayerIndex***, short** *ItemIndex,,* **float** *newValue* **)**
Sets the item's percentage frequency adjustment to *newValue*. 0.0 < *newValue* < 200.0

FLOAT *Get***iSpeedFactor (short** *LayeIindex***, short** *ItemIndex* **)**
Returns the item's percentage speed adjustment to the material file setting. Returns -1.0 if *LayerIndex* and *ItemIndex* do not specify a valid item.
VOID *Set***iSpeedFactor(short** *LayerIndex***, short** *ItemIndex,,* **float** *newValue* **)**
Sets the item's percentage speed adjustment to *newValue*. 0.1 < *newValue* < 200.0

LONG *Get***iConditionalMarkFlags (short** *LayeIindex***, short** *ItemIndex* **)**
Returns the item's conditional mark byte and mask. The least significant byte holds the mark bit flags. The second least significant byte is the mask. The item is marked when the digital input lines match the bit settings in the mark byte. Any bit that is 0 in the mask byte is ignored. A mask byte = 0 will ALWAYS mark. Returns 0 if *LayerIndex* and *ItemIndex* do not specify a valid item.
VOID *Set***iConditionalMarkFlags (short** *LayerIndex***, short** *ItemIndex,,* **long** *nNewValue* **)**
Sets the item's conditional mark byte and mask byte to *nNewValue* as indicated in *Get***iContitionalMarkFlags** above.

FLOAT *Get***iWobbleWidth (short** *LayeIindex***, short** *ItemIndex* **)**
Returns the item's wobble width setting. Returns -1.0 if *LayerIndex* and *ItemIndex* do not specify a valid item. 0 indicates no wobble.
VOID *Set***iWobbleWidth (short** *LayerIndex***, short** *ItemIndex,,* **float** *newValue* **)**
Sets the item's wobble width to *newValue*. 0.0 < *newValue* < 150.0

FLOAT *Get***iWobbleFreq (short** *LayeIindex***, short** *ItemIndex* **)**
Returns the item's wobble frequency. Returns -1.0 if *LayerIndex* and *ItemIndex* do not specify a valid item. Wobble frequency only applies to RTC interfaces.
VOID *Set***iWobbleFreq (short** *LayerIndex***, short** *ItemIndex,,* **float** *newValue* **)**
Sets the item's wobble frequency to *newValue*.

SHORT *Get***iWobbleType (short** *LayeIindex***, short** *ItemIndex* **)**
Returns the item's wobble type setting:

0	**=**	**Line**
1	**=**	**Square**
2	**=**	**Circle**

-1 if the *LayerIndex* and *ItemIndex* do not specify a valid item.

VOID *Set***iWobbleType (short** *LayerIndex***, short** *ItemIndex,,* **float** *nNewValue* **)**
Sets the item's wobble type to *nNewValue*. 0 <= *nNewValue* <= 2

SHORT *Get***iWobbleDensity (short** *Layelindex***, short** *ItemIndex* **)**
Returns the item's wobble density setting:

0 = High
1 = Medium
2 = Low

-1 if the *LayerIndex* and *ItemIndex* do not specify a valid item.

VOID *Set***iWobbleDensity (short** *LayerIndex***, short** *ItemIndex,,* **float** *nNewValue* **)**
Sets the item's wobble density to *nNewValue*. 0 <= *nNewValue* <= 2

Marking Functions

BOOL **mAbortBatch ( )**
Aborts the current run. Returns TRUE if the mark is successfully aborted.

SHORT *Get***mDigInput ( )**
Returns the state of the 8 digital input ports. The return value should be masked with 0x00ff.

VOID *Set***mDigInput (short** *nNewValue* **)**
The digital input port is read and cannot be set. This function does nothing.

SHORT *Get***mDigOutput ( )**
Returns the state of the 8 digital output ports. The return value should be masked with 0x00ff.

VOID *Set***mDigOutput (short** *nNewValue* **)**
Sets the state of the 8 digital output ports. *nNewValue* will be masked with 0x0ff and only the lower 8 bits will be set. Note: bits 0 and 2 are reserved for Gate/Not Gate laser control signals and will not be modified by this command.

BOOL **mForceStart ( )**
Overrides the ProLase start conditions and forces a valid start. The controller program should use the mMarkStatus to make sure that ProLase is waiting for a start signal before issuing this command. This command is latched, meaning a new ForceStart command is required for each part or tray of parts.

SHORT *Get***mItemStatus ( )**
Returns the *ItemIndex* of the current Item being marked. This can be used to monitor the progress of each part.

VOID *Set***mItemStatus (short** *nNewValue* **)**
This function does nothing.

SHORT *Get***mLayerStatus ( )**
Returns the *LayerIndex* of the current Layer being marked. This can be used to monitor the progress of each part.

VOID *Set***mLayerStatus (short** *nNewValue* **)**
This function does nothing.

LONG *Get***mMarkStatus ( )**
Returns the status of the current batch run. The status is either normal operation (codes 0-10) or an abort code. Abort codes are generated whenever the batch was stopped for any reason before completion of the *BatchQuantity*.

0 = Not Running
1 = Initializing Mark
2 = Waiting for Start Condition
3 = Marking Layer
4 = Post Mark Delay
10 = Batch Complete

States 2 through 4 are repeated for each layer of each part in the run. State 10 is reset to zero when it is read once.

Abort Codes

50 = Abort Source Unknown
90 = Automation Controller Abort
100 = User Abort (Escape key)
101 = Digital I/O Abort
102 = List File Error
103 = "Waiting for Start" Timeout

VOID *Set***mMarkStatus (long** *markstatus* **)**
Sets the MarkStatus. This can be used to clear the status after reading an abort code for example.

BSTR *Get***mMarkerDocName ( )**
Returns the name of the document previously loaded by dLoadLazDocument or 'Nothing Loaded' if no document is currently loaded.

VOID *Set***mMarkerDocName (LPCTSTR** *lpszNewValue* **)**
This function does nothing.

FLOAT *Get***mOffsetAngle ( )**
Returns the current value of the external angle offset in degrees.

VOID *Set***mOffsetAngle (float** *newValue* **)**
Sets the external angle offset to *newValue*.

FLOAT *Get***mOffsetX ( )**
Returns the current value of the external X offset in the current configuration units. The X offset changes the X position of the entire field including all marked items in the document currently active in the marker (not necessarily the controller's document).

This property along with *mOffsetY* and *mOffsetAngle* can be used to externally control the field offsets based on information from a device that measures part offsets. A vision system for example can determine the current X, Y, ANGLE of a part and give that information to ProLase

through the controller interface so that ProLase can correctly transform the marks to hit the part in the right place and orientation. These properties can be used to control the transformation of any document being currently marked including the ProLase UI, or another controller's document.

VOID *Set***fOffsetX (float** *newValue* **)**
Sets the external X offset to *newValue*.

FLOAT *Get***fOffsetY ( )**
Returns the current value of the external Y offset in the current configuration units. The Y offset changes the Y position of the entire field including all marked items in the document currently active in the marker (not necessarily the controller's document).

VOID *Set***fOffsetY (float** *position* **)**
Sets the external Y offset to *newValue*.

SHORT *Get***mPartCount()**
Returns the number of completed parts in the batch. This can be used to monitor the progress of the batch.

VOID *Set***mPartCount (long** *nNewValue* **)**
This function does nothing.

BOOL **mRunBatch (double** B*atchQuantity* **)**
BatchQuantity is the number of parts to be marked in the run. This function marks all of the parts and then will return TRUE if the batch marked to completion. If the mark was aborted, this function will return FALSE. The GetMarkStatus function can be used to find out how the mark was aborted. This function is equivalent to pressing the MARK button (with AUTO selected) in the ProLase UI, MarkControl Dialog.

BOOL **mSpawnBatch (double** *BatchQuantity* **)**
Similar to mRunBatch(…), but the marking is done is a separate thread. This function returns immediately. The state of the marker can be monitored using various status functions.

BSTR *Get***mUnits()**
Returns the current unit size used by the configuration. Values can be:

in. = inches
cm = centimeters
ft. = feet
m = meters

VOID *Set***mUnits (LPCTSTR** *lpszNewValue* **)**
This function does nothing.

SHORT **mDefaultMarkMode(short** *iMode***)**
Returns the current (not previous) default mark mode flag state. If *iMode* = 1, then the default mark mode is enabled, otherwise it is disabled and the Mark Option flags (see below) will apply. While enabled, the Mark Option flags will have no effect.

SHORT **mMarkOptionAlignment(short** *iMode***)**
This function has the same effect as the 'Alignment' checkbox in the Mark Control window of

the ProLase GUI. If *iMode* = 1, then the alignment option is enabled and only objects with the 'Active' property *disabled* (alignment objects) will be marked. Otherwise, only 'Active' object are marked. Returns the current (not previous) state of the alignment flag.

SHORT **mMarkOptionMode(short** *iMode***)**

This function has the same effect as the Mode buttons in the Mark Control window of the ProLase GUI. Allowed values for *iMode* are listed below. Once set, marking will continue in that mode until changed (or mDefaultMarkMode(1) is called).

0 = Manual
1 = Auto
2 = Lightshow

SHORT **mMarkOptionSetup(short** *iMode***)**

This function has the same effect as the 'Setup' button in the Mark Control window of the ProLase GUI. If *iMode* = 1, then the setup option is enabled and the setup material file is used to mark the document. Otherwise, the regular material file that the document links to is used. Returns the current (not previous) state of the setup flag.

Appendix A

ProLase support Files

File	Description	Directory	Required? ProLase	+	DLL	Dev
ProLase.exe	ProLaseFront End	<InstallDir>	yes	yes	no	no
ProLase.dll	ProLase DLL	<InstallDir> or <WinSys>	no	no	yes	yes
Laser.dll	Laser Control DLL	<InstallDir> or <WinSys>	yes	yes	yes	yes
Ssiact.386	AEGIS Driver	<WinSys>	yes	no	no	no
Imgman32.dll	Raster Graphic DLL	<InstallDir> or <WinSys>	yes	yes	yes	yes
Im31bmp.dil	Raster Importer	<InstallDir> or <WinSys>	yes	yes	yes	yes
Ezbar32.dll	Bar Code DLL	<InstallDir> or <WinSys>	yes	yes	yes	yes
ImsType.dll	Vector Graphic DLL	<InstallDir> or <WinSys>	no*	no*	no*	no*
Isgdi32.dll	Vector Graphic Importer	<InstallDir> or <WinSys>	no*	no*	no*	no*
Mfc40.dll	MicroSoft MFC DLL	<WinSys>	yes	yes	yes	yes
Mfc42.dll	MicroSoft MFC DLL	<WinSys>	yes	yes	yes	yes
Msvcrt.dll	MicroSoft DLL	<WinSys>	yes	yes	yes	yes
.flt	Vector Filters	<InstallDir>\Filters	no*	no*	no*	no*
Pmpbt9k.exe	Vector Processor	<InstallDir>	no*	no*	no*	no*
Default.cfg	ProLase Configuration	<InstallDir>	yes	yes	yes	yes
Simplex.alf	ProLase Default Font	<InstallDir>	yes	yes	yes	yes
Bcali.alf	ProLase BarCode Font	<InstallDir>	no*	no*	no*	no*
Idmali.alf	ProLase ID Matrix Font	<InstallDir>	no*	no*	no*	no*
Bmp.alg	ProLase Raster Graphic	<InstallDir>	no*	no*	no*	no*
Box.alg	ProLase Box graphic	<InstallDir>	yes	yes	yes	yes
Circle.alg	ProLase Circle Graphic	<InstallDir>	yes	yes	yes	yes
Laz.New	“New” Laz file	<InstallDir>	no*	no*	no*	no*
Default.Pre	Preferences File	<InstallDir>	no**	no**	no**	no**
Default.Mat	Default Material File	<InstallDir>	no**	no**	no**	no**
Default.Fx2	Default Fixture File	<InstallDir>	no**	no**	no**	no**
Default_Power.Cal	Power Calibration File	<InstallDir>	no**	no**	no**	no**
Default_Freq.Cal	Frequency Calibration	<InstallDir>	no**	no**	no**	no**
Default_Focus.Cal	Focus Calibration File	<InstallDir>	no**	no**	no**	no**
ProLase.lib	ProLase DLL Interface	<DevDir>	no	no	no	yes
*.h	DLL Class Interfaces	<DevDir>	no	no	no	yes
Indexer LPT	Stepper Motor Indexer	Autoexec.bat, Config.sys	no	yes	no	no
Initxyza.bat	Stepper Initialize	<InstallDir>\Table	no	yes	no	no
Movexyza.bat	Axis Move (non LPT)	<InstallDir>\Table	no	no*	no	no
GetStatus.bat	Axis Status (non LPT)	<InstallDir>\Table	no	no*	no	no
Status.txt	Axis Feedback (non LPT)	<InstallDir>\Table	no	no*	no	no

NOTE: NT systems like ProLaseRTC2 and ProLaseRTC2 DLL require the following in addition to those defined above:

Installation of GIVEIO driver using IOINST.EXE. Example: IOINST.EXE GIVEIO GIVEIO.SYS
Installation of Sentinal driver.
Installation of SCANLAB RTC2 driver using RTCSetup.exe from SCANLAB.
RTC24NT4.DLL, RTC24NT4.HEX, and Cor_1to1.CTB or their equivalence are requried and are supplied by SCANLAB. The location of these files are parameters in the ProLase configuration when the RTC2 driver is selected. The default address for the RTC2 card is 240 hex. This is a parameter in the ProLase configuration and a parameter of RTCSetup.exe, the SCANLAB driver installer.

<WinSys> = Windows System directory, usually C:\Windows\System on most PCs. . As indicated in the above table, some items can be installed in <WinSys> as an alternative to the <InstallDir> sub-directory.

<InstallDir> = Installation Sub-directory. This is the sub-directory that the EXE file is installed in. The EXE file is either ProLase.exe or any program using ProLase.DLL. For example lets assume LaserMark.exe is a customer created program using the ProLase DLL. If LaserMark.exe is installed in C:\Program Files\CustProg, then <InstallDir> = C:\Program Files\CustProg. To run this program all items marked <InstallDir> should be installed to C:\Program Files\CustProg.

<DevDir> = Development sub-directory. This can be any directory since you specifically include the files into Visual Studio projects. All files marked <InstallDir> need to be copied to the Projects main directory, in order to run from Visual Studio.

No* = The file is not required to execute ProLase, but is required for some major feature to operate. For example ProLase will run without the vector graphic DLLs, however the graphic import capabilities will not operate correctly. Do not try to use capabilities that require missing components.

No** = The file is not required to execute ProLase. If the file is not present, ProLase will create default versions of the file give it a name of DEFAULT and save it to disk.

A minimum installation requires all files marked YES. This will allow the system to process SIMPLEX text objects only.

A complete and fully functional installation of ProLase requires all files marked YES, NO*, or NO**.

Appendix B

This appendix lists the details of the publicly defined file formats for file types created and used by American Laserware, Inc. and ProLase software.

*.fnt

The *.fnt format is a vector font file format used by ProLase. These fonts become available to text objects in ProLase when the 'Laser Engraving Fonts' font type is selected. Typically these are single or multiple stroke fonts similar to those used by mechanical engraving systems. The *.fnt file specifications are as follows:

1. ASCII text file (almost any text editor will do).
2. File must have the *.fnt extension.
3. Numeric data entered as text.
4. ONE number per line.
5. Whitespace preceding the numeric data is ignored.
6. Carriage return/Linefeed (CRLF) terminates each line.
7. Data range is –32768 to +32767 (signed short integer).
8. Characters must correspond to non-extended ASCII values, 0-127 (However, the ProLase editor will only allow you to enter viewable characters, ASCII 32-127).
9. Characters may be omitted. Missing characters are not output by ProLase.
10. Character definitions may occur in any order and characters may be redefined. ProLase will use the data corresponding to the last instance of the character in the file.
11. Characters are listed consecutively in the file; no blank lines to separate them.
12. Character definition:
    a. ASCII value
    b. Header
        i. Reserved
        ii. Min. X*
        iii. Min. Y*
        iv. Max X*
        v. Max Y*
        vi. Total X
        vii. Total Y
        viii. Character Height**
    c. Vector List
        i. Relative vector data
            1. Delta X = new X value - current X value
            2. Delta Y = new Y value - current Y value
            3. Delta I = X value of center of circle - current X value
            4. Delta J = Y value of center of circle - current Y value
        ii. 4 Types
            1. Move
                a. Code = 1
                b. Delta X
                c. Delta Y
            2. Line
                a. Code = 2

   b. Delta X
   c. Delta Y
3. Circular Arc***
   a. Code = 3
   b. Delta X
   c. Delta Y
   d. Delta I
   e. Delta J
4. End Of Character
   a. Code = 255

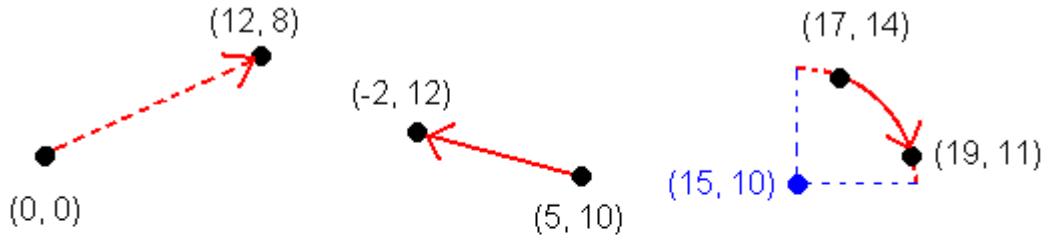
* The min/max extents of the character are given in absolute coordinate values. The vector data is relative to the current position. The initial position is always (0,0).

** Character Height is the height of a standard full height character drawn in the same scale as this character. For example, a lowercase ‘a’ that is half as tall as a capital ‘A’ might have Total Y = 100 and Character Height = 200.

*** Arcs MUST NOT cross a cardinal axis. Arcs that extend to both sides of a cardinal (horizontal or vertical line) must be broken into 2 separate arcs, first to the intersection with the cardinal and then one to the end point. Thus, a full circle can be made from 4 arcs.

Vectors

The vector information for the characters is given in a relative coordinates. The all characters are assumed to begin at (0,0) absolute position. The relative coordinates are simple the delta X and delta Y values to get to the next position. For arcs, the delta I and delta J values indicate the relative position of the center of the circle from the current, starting position. The coordinates specified by the arc command must actually lie on a circle. If not, the results are uncertain. Examples of the vector types are shown below. The list of numbers below the vector type represent the 3 or 5 values that would be found in the font file for that vector.



Move	Line	Arc
1	2	3
12	-7	2
8	2	-3
		-2
		-4

Vectors may extend in any direction from the origin. Characters that have vectors below the baseline (any absolute X value less than zero) will have decenders, such as ‘g ’or ‘p’. If any absolute Y value is greater than the Character Height value in the header, that character will have ascenders, such as ‘Å’.

Because ProLase uses dynamic scaling of every character, it is NOT required that all characters be drawn to the same scale relative to each other (although it is usually easier for the designer that way). ProLase will scale every character relative to its own Character Height value. This value represents the full size height (usually the height of most capital letters like ‘A’ or ‘T’ draw at the same scale. If the vector data for ‘B’ and ‘C’ are take from 2 separate graphics and ‘B’ has Total Y = 100 and ‘C’ has Total Y = 2000, they will both have the same height provided that Character Height = 100 and 2000 respectively.

Example
Following is a sample font made from parts of the Simplex.fnt file that comes with ProLase. It includes three characters, the space character (ASCII 32), ‘A’ (ASCII 65) and ‘a’ (ASCII 97).

```
 32
 1
 0
 0
 300
 0
 300
 0
 250
 1
 300
 0
 255
 65
 255
 0
 0
 254
 250
 254
 250
 250
 1
 32
 0
 2
 95
 250
 2
 95
-250
 1
-35
 83
 2
-119
 0
 1
-36
-83
 1
 222
 0
 255
 97
 255
```

```
 0
 0
 206
 166
 206
 166
 250
 1
 174
 166
 2
 0
-166
 1
-24
 12
 2
 24
 24
 1
-24
-24
 2
-23
-12
 2
-36
 0
 2
-24
 12
 2
-23
 24
 2
-12
 35
 2
 0
 24
 2
 12
 35
 2
 23
 24
 2
 24
 12
 2
 36
 0
 2
 23
-12
 2
 24
-24
```

```
 1
-142
-130
 1
 174
 0
 255
```

Appendix C

This appendix applies only to ProLase7 PLUS with Indexer LPT on the Windows 7 operating system. The names, locations and uses of all relevant 'helper' files are fully described.

ProLase7 PLUS can be run on any Windows operating system and it can use either Indexer LPT or any 'Other' stepper motor controller to driver the external axes. If Indexer LPT from Ability Systems Corporation (215-657-4338 or www.abilitysystems.com) is selected, the version must be matched to the operating system used. The Indexer LPT driver compatible with Windows 95/98/ME uses a driver that is accessed directly by ProLase as described above in Chapter 5. The Indexer LPT driver that runs in Windows 7 is different and, as a result, helper files are required by ProLase to provide backwards compatibility with existing ProLase PLUS systems.

ProLase7 PLUS requires that the **TableServer** and **Table** applications be present in the ProLase installation folder to manipulate the external axes. Of course, the Indexer LPT driver must previously be installed and working properly. Also included in the ProLase folder is the **TableBATConverter** application. This is a useful tool for updating the batch files by ProLase PLUS. Finally, The **TableClient** project folder is included as a tutorial for advanced users that wish to write a program that interacts with the stepper motors and runs concurrently with ProLase.

TableServer
TableServer is a small application that must be running for ProLase to interact with the Indexer LPT driver. The *TableServer.EXE* file must be located in the ProLase installation folder. Only a single instance of TableServer can exist on a computer at a single time; attempts to run multiple copies will not succeed. *When running, TableServer is the one and only application that can access the Indexer LPT library and driver. The Ability Systems Hardware Assist Module must be present and the driver must load and run properly or TableServer will terminate.* Even the Ability Systems IXDiag.EXE utility cannot run concurrently with TableServer. Because TableServer links to the Indexer LPT library, the Indexer LPT Hardware Assist Module must be present or TableServer will terminate. ProLase will automatically launch TableServer during its initialization of the external axes if it does not detect that it is already running.

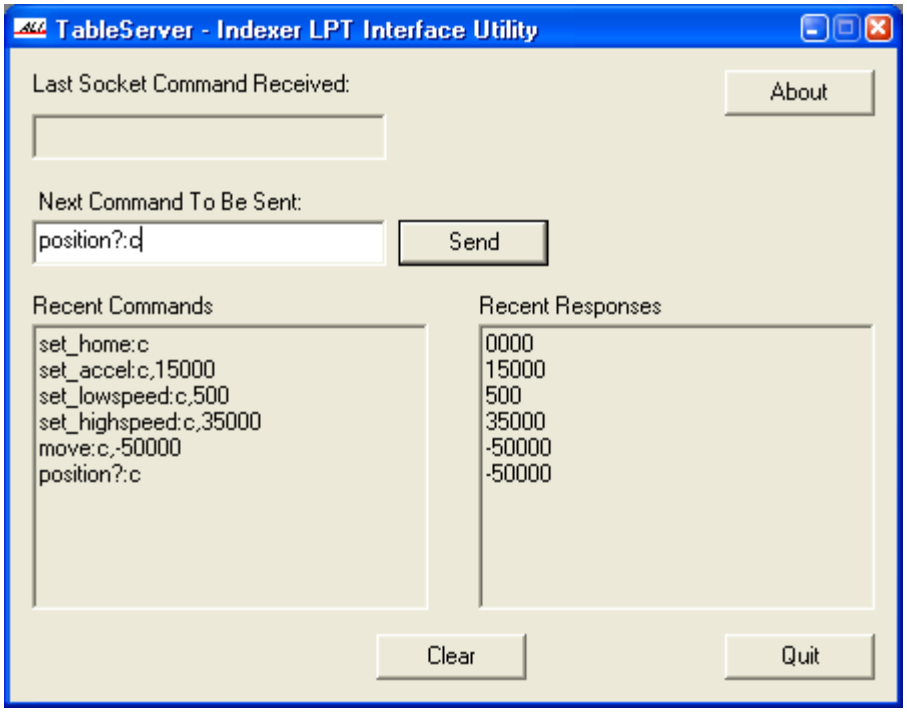
TableServer has a single window interface as seen in the figure below.

TableServer relays commands to the Indexer LPT library and displays the responses. The *Next Command To Be Sent* window allows the user to enter a command for Indexer LPT to process directly. The command is issued with the *Send* button. The *Recent Commands* and *Recent Responses* windows list the 10 most recent interactions with the Indexer LPT library. The *Clear* button will erase the data in all of the windows. The *About* button displays a window with version, build date, and interface data.

In addition to direct user commands, TableServer can relay remote command to Indexer LPT. These commands are received over a windows socket connection. TableServer listens on the port specified in the About window for a command. When it receives a command, TableServer shows it as the *Last Socket Command Received*, logs it in the *Recent Commands* list, and issues the command to Indexer LPT. The response from Indexer LPT is first displayed in the *Recent Responses* list, then send back over the socket that sent the command.

The primary use for TableServer is in this background relay capacity. The main visual interface is mainly used only during initial setup or to debug an error in the table motion system. As such, TableServer begins, and should generally remain, in a minimized state on the 'System Tray'. This is the small area near the clock on the Windows Task Bar. Double-clicking on this TableServer icon will open the main TableServer window.

Alternately, right-clicking will display a pop-up menu that allows opening the main window or exiting the application.



Table

Table is a command line utility application designed to provide full interaction with Indexer LPT through the TableServer application (see above). The *Table.EXE* file MUST be located in the ProLase installation folder. The complete Table command line must have the following form:

```
table command [/a address]
```

where *command* is the required command to send to Indexer and *address* is an optional IP address for the computer hosting an active instance of TableServer. Any additional parameters are ignored. Table will output the response from TableServer / Indexer LPT.

TableBATConverter

TableBATConverter is a small, utility application designed to help convert existing ProLase PLUS batch files to a form that can be used by ProLase7 PLUS running on Windows7 operating system. While Table is called directly from ProLase7 PLUS to move the external axis between or during layers, initialization of the system is accomplished by the InitXYZA.BAT batch file which issues a list of Table command lines to locate the axis unambiguously.

Previous versions of ProLase PLUS would call use an InitXYZA.BAT file of the following format:

```
echo set_accel:c > motor
echo set_lowspeed:c > motor
echo move:c, -50000 > motor
echo set_home:c > motor
```

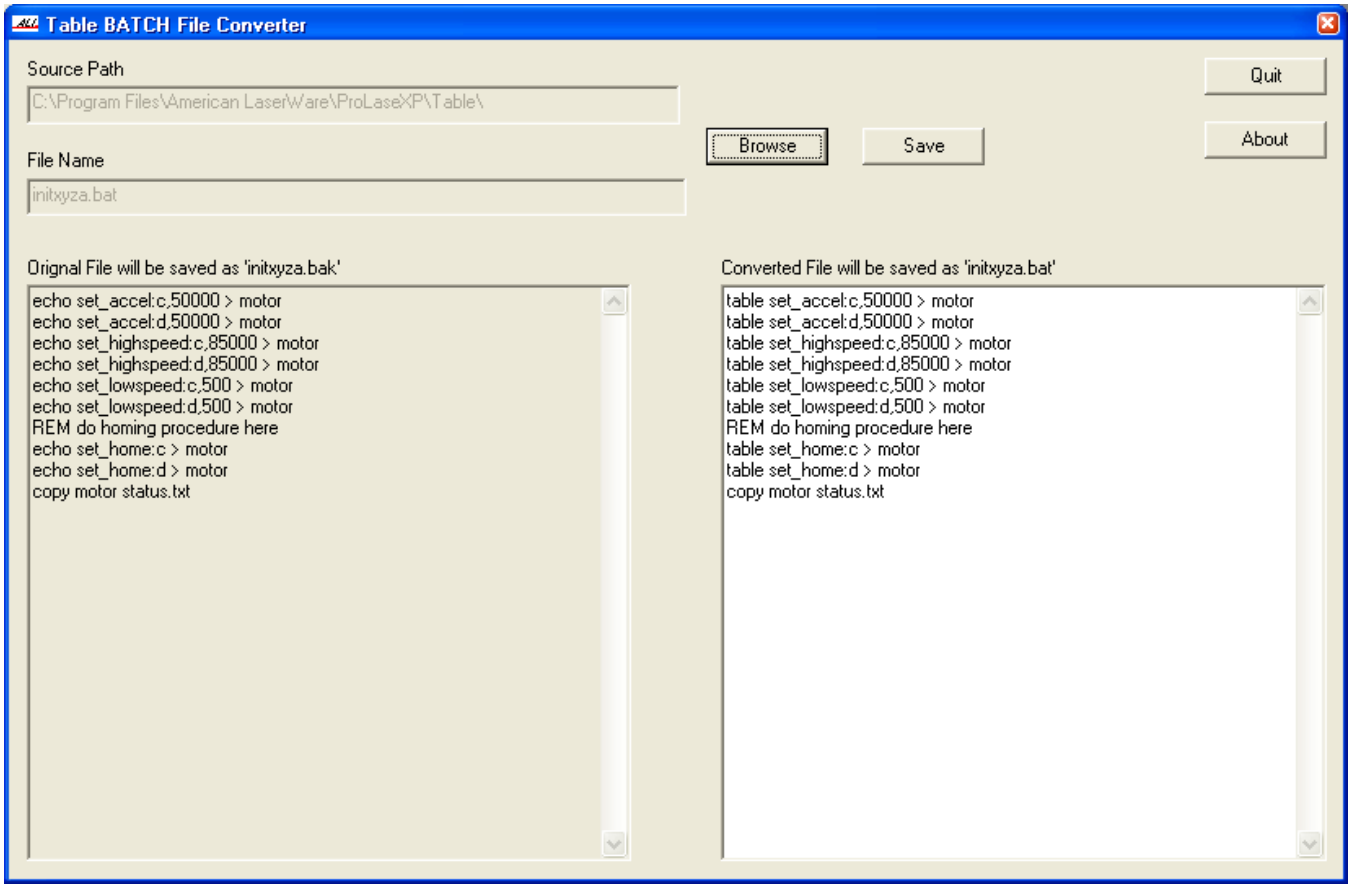
where `echo` is the DOS command to display the message and `>` is the DOS instruction to redirect the output stream. `motor` is the file name used to access the Indexer LPT driver.

To implement this initialization with the Windows7 system, the Table program must be used instead of the `echo` command. The previous example must then become

```
table set_accel:c > motor
table set_lowspeed:c > motor
table move:c, -50000 > motor
```

```
table set_home:c > motor
```

TableBATConverter facilitates this conversion. The figure below shows an example of this.



An InitXYZA.BAT file is selected using the *Browse* button. Once a valid file is chosen, the contents of that file are loaded, converted, and displayed in the listing on the right. The original file contents are displayed on the left side for reference. The converted file may be edited to add additional comments and to correct automated conversion errors. The *Save* button will store the converted file. The original file contents are backed up in a file with the same title but the 'BAK' extension.

TableClient

TableClient is a small, GUI application that demonstrates communication with TableServer application (see above) via Windows Socket technology. Commands for Indexer LPT are accepted, redisplayed, and sent to TableServer for processing. The responses are then received and logged as well.

Due to the design of the Indexer LPT driver for Windows 7, only a single application is allowed access. With ProLase7 PLUS, this application is TableServer. ProLase does NOT access the Indexer LPT driver directly specifically to allow other programs to operate. This allows for complex installations where stepper motors may be used for more than just ProLase control. Consider the fairly common case of a 'job shop' with a fixture for Step & Repeat processing of various parts. All of the parts have the same footprint and are easily accommodated by the same fixture. However, their heights are varied. Since the mark applied to them is the same for all parts, say the

company logo, only a single ProLase document (LAZ file) and a single fixture file are required. However, every tray of parts must be adjusted to in the Z axis to focus the laser at the top. This is quickly and easily preformed manually using a second Z axis control program. The user jogs the fixture's elevation to bring find the appropriate elevation. Then ProLase marks normally. Ignorant of the change in the Z axis.

TableServer allows both ProLase and the Z Axis control program to manipulate the appropriate axes using the same Indexer LPT stepper motor controller. Rather than directly accessing the Indexer LPT driver, the Z Axis control program must, just as ProLase, be written to use TableServer to interact with the axes. TableClient is a simple demostration of just how to accomplish this.

The entire TableClient project is provided with the ProLase7 Plus installation. This project contains all the source code for the TableClient application. ProLase customers who require such additional motor control as well advised to use the TableClient application source as a guide to writing their own application.