Advanced Applications
For
ASI's MS2000 DC Servo Motor
Stage Controller

Table of Contents

Introduction

The MS2000 servo motor controller is designed for automated control of ASI's motorized microscope stages. The needs of users, over the years, have led to the development of several special function firmware modules that can be included in the MS2000 controller. Some of this firmware has become an integral part of current shipping firmware, whereas other modules are more specialized and are made available only on request. This document will describe the majority of these advanced firmware features, describe how the user can determine the configuration of her controller, and describe the process for firmware or hardware upgrades that the user may desire.

Brief Description of Special Firmware Modules

Firmware modules are identified by the firmware compiler directive name. The following list has a short description module function. Those marked with (**†**) are discussed in more detail in this document.

ARRAY MODULE †	Firmware module for x,y moves in array pattern.
AUTOFOCUS †	Video autofocus scanning firmware module
CRIFF †	Firmware for the CRIFF focus system.
DAC OUT †	WRDAC command sends specified voltage to SV1-Pin 5.
LED DIMMER †	PWM dimmer on standard TTL out, used for ASI LED illuminator
MULTIAXIS MOVES	Supports circular and spiral moves directly from the controller
PEDALS	Support for foot pedals to control Z-axis and Zoom systems
PLANAR CORRECTION †	Supports 3-point focal plane correction
RING BUFFER †*	Internal 50 position Ring Buffer is supported
SCAN MODULE †	Supports 1-d and 2-d programmable scan patterns.
SEARCH INDEX *	Supports ability to search for the index on linear encoders
SEEK LIMITS	Optimized limit switch detection for coordinate reference
SEQUENCER †	Control several TTL I/O's, stages, delays, w/ programmable logic
TTL REPORT †	Supports synchronous position reporting.
TRACER †	G-Code interpreter system allows stage to follow CAD files.
TRACKING †	Firmware module to support PhotoTrack system
XY KNOBS	Support for dual knobs for XY manual input in addition to joystick

* These modules are usually included in standard builds.

The special firmware modules often required particular hardware resources. It is not be possible to include all combinations of modules without resource conflicts.

Determining which modules are in the Controller Firmware

When the compiler directives for the special firmware module are included in the firmware build, the directive name is included in the response to the **BUILD X** command. For example:

BUILD X	
STD_XYZ_SEQ	The official build name
Motor Axes: X Y Z	The names of the controlled axes
CMDS: XYZFTR	Names that may be used in commands
BootLdr V:1	Onboard re-programmer version
Hdwr REV.E	Main circuit board hardware revision
LL COMMANDS	The list of special modules…
RING BUFFER	
SEARCH INDEX	
ARRAY MODULE	
IN0_INT	
DAC OUT	
SEQUENCER	
PLANAR CORRECTION	

As you can see, some builds can have several special modules at the same time.

You can use **ASI CONSOLE,** available on our website, or any terminal program to communicate with the controller directly. **ASI CONSOLE** includes the output of the BUILD X command at the connection screen.

Manual Sections and Descriptions for the Special Modules

The rest of this document will describe in detail the operation of the various special firmware modules that are available. Parts of the appropriate manual sections will be included.

`ARRAY MODULE` for ARRAY Scanning

This manual section applies to the ARRAY MODULE firmware module. With the ARRAY MODULE included in the firmware, the controller will support functions associated with moving to specific locations in a 2-dimensional XY array of positions. The module includes support for commands to define the size and spacing of the array and the location of the first position in the array. There are also commands to address particular positions and sequence through the array either automatically, or with external TTL or serial command control.

Setting up the Array

The **ARRAY** command is used to define the number of array points and spacing between rows and columns. The X and Y parameters specify the number of columns and rows respectively, and the Z and F parameters specify the signed move distance in millimeters between columns and rows. The command to set up the array for a standard 96 well plate would be:

`ARRAY X=12 Y=8 Z=9.0 F=-9.0`

These happen to be the default settings. The Y-axis distance between rows is set as a negative number because the direction of motion to go from row A to row B on the well plate is in the negative direction of motion of the stage.

The location of the first position in the array, the 1,1 location, is specified by the **AHOME** command. The default value for the coordinates of the 1,1 position are 0.0, 0.0 in the controller's coordinate system. You need not even use the **AHOME** command if the stage is manually moved to the location of the 1,1 array position and then the ZERO button is used to set the coordinate zero to that location. The **AHOME** command, with out arguments, will establish the current stage position as the 1,1 array position. Hence, if you move the stage manually to the 1,1 location and issue "**AH**", the stage will maintain its original coordinate system yet have the correct offsets for the fixed array moves.

Controlling Moves to Array Positions

There are three ways to control the movement to array positions. They are random access serial moves with the **AIJ** command, automated self-scanning, and TTL or serial commanded moves to the next array position. As with any commanded move, TTL output pulses can be programmed to indicate the completion of the move to the array location using the **TTL** command.

Random Access Using Serial Command

The serial AIJ command will command the stage to the any row *j* and column *i* coordinate:

AIJ X=*i* Y=*j*

Self-Scanning

The entire array can be visited sequentially in either a serpentine or raster pattern. Scanning is initiated manually with a quick press and release of the @ button, or by issuing the **ARRAY** command without any arguments.   The sequence will start by moving to the 1,1 location. When the stage arrives on target, it will delay for a period of time set by the command **RT Z=*time_delay*** before continuing on to the next position.
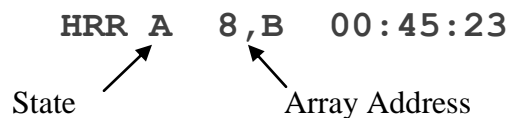
Commanded Next Position

The array can be traversed in order with moves made "on command" using either a serial command or TTL trigger pulse. Using the **TTL** command, set the TTL input function to the array trigger using "**TTL X=7**". Either a TTL pulse on the IN0 input, or the **RM** command without arguments, will send the stage to the next array position. Move to the 1,1 array position and gracefully start this scanning mode is the command "**RM X=0**". When started this way, the sequence will complete after the last position is processed and subsequent commands (TTL pulse of "**RM**") to move to the next position will be ignored.

LCD Display

The bottom line of the LCD display will show the state of the ARRAY MODULE state machine as well as the coordinate location of the current array cell. The LCD status line should look something like this:

HRR A 8,B 00:45:23

State   Array Address

For arrays with 4, 8, or 16 rows, the Y address is shown as the corresponding alphabet characters – as is the common practice for standard well plate formats. The meanings of the various state indicators are described below.

```
I      IDLE
S      START
M      MOVING
D      DELAY
A      ARRIVED
N      NEXT_POSITION
C      COMPLETE
E      ERROR
```

When the state characters are displayed in lower case on the LCD display, the TTL IN0 interrupt is disabled pending completion of the current move operation.

Stage Scanning Speed and Accuracy

The leadscrew pitch of the stage will largely determine where on the speed versus accuracy continuum the stage will operate. The table below shows typical performance for various pitches of leadscrew for a rotary encoded stage when no backlash correction is used, and the *Finish Error* is set to the resolution shown in the last column.

Lead Screw Pitch	Maximum Stage Speed (mm/s)	Rotary Encoder Resolution (nm)	Typical Performance			
			Time to traverse 96 well-plate (s)	Time / 9 mm move (s)	Time / 0.2 mm move (ms)	Fast Positioning Resolution (μm)
A – 1.58mm	1.7	5.5	644	6.7	235	0.08
B – 6.35mm	6.8	22	160	1.67	70	0.25
C – 12.7mm	13.5	44	79	0.82	49	0.5
D – 25.4mm	26	88	47	0.48	40	1.0

Opening up the Finish Error to about 10 rotary encoder counts allows the move to complete quickly rather than have to hunt for on-the-count precision.

AUTOFOCUS with the ASI video AUTO-FOCUS Electronics

The ASI auto-focus system uses an electronic circuit to detect the high spatial frequency information present in a video signal coming from an analog camera, and converts it to a *focus value* for each frame – the better the focus, the larger the *focus value*. Depending on the sample and the type and power of the objective, the conditions to achieve the best *focus value* may change. To compensate, the system incorporates an auto-calibration routine that automatically sets various internal parameters to get a good *focus value*. The system also allows you to highlight any subset of the video field and focus on an object in it.

Video IN from

Video OUT to Monitor

(a

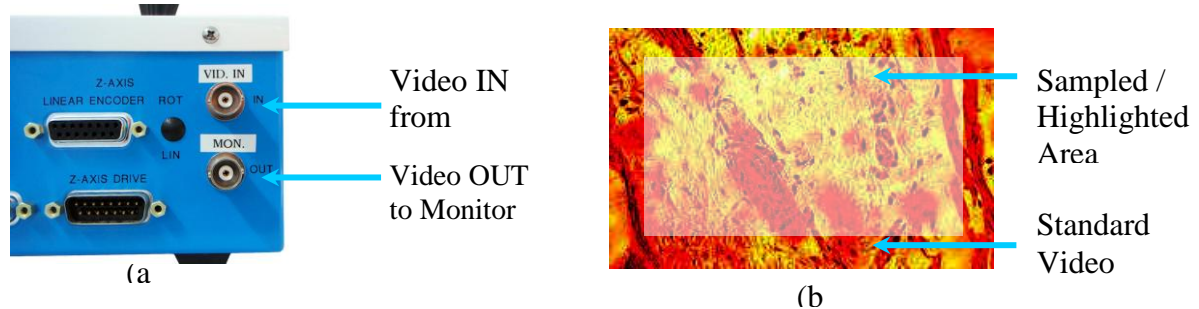Sampled / Highlighted Area

Standard Video

(b

Fig 1: (a) Relevant Auto-focus Connections (b) Screenshot of Video Monitor signal
from the Auto-Focus controller showing Sampled/Highlighted Area

When the controller is given an auto-focus command, it moves from a starting position to a stop position, constantly checking the focus signal. The focus signal goes through a filtering process to remove unwanted false focuses caused by noise and other disturbances. In "*Normal*" mode, the controller takes the filtered signal and watches for the maximum value. When a new maximum value is detected, the controller keeps track of the position where it was found. When the end of the scan is reached, the controller goes back to the position that had the highest value.
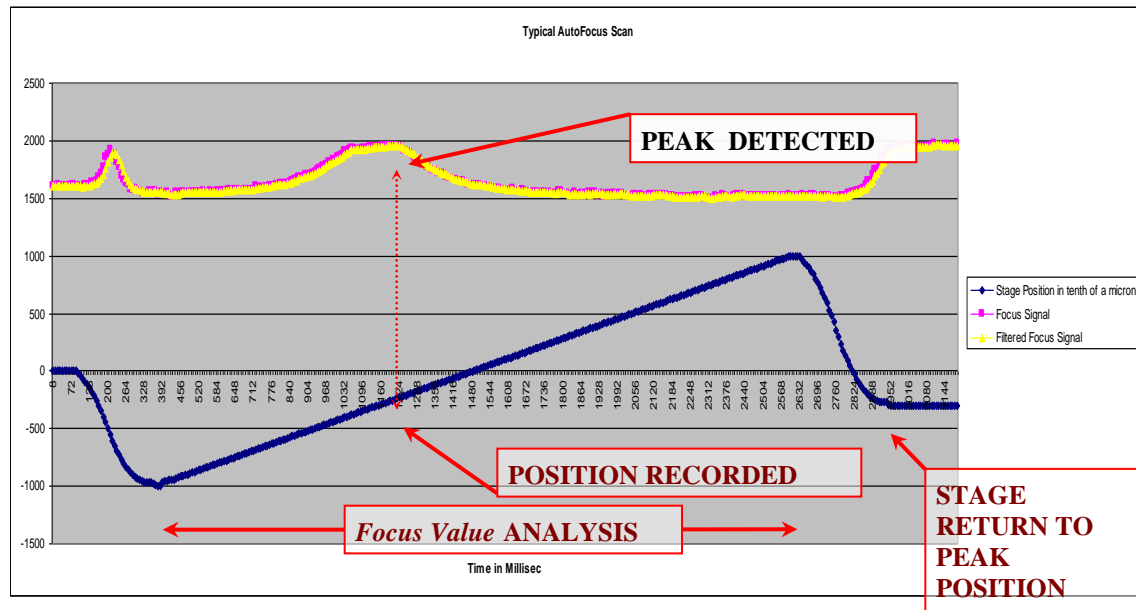
Fig 2: A "*Normal*" auto-focus scan showing a stage's focus-axis position in tenths of microns, and the focus signal and filtered focus signal *(focus-value)* extracted from input video signal.

There is also a "*Hill Detect*" mode available. In this mode, the auto-focus controller watches for the *focus value* to increase in value and then start to decrease, thus detecting a hill. At this point, the auto-focus controller stops the scan and goes back to the position that had the highest *focus value* found on the hill. This is somewhat subject to false focusing and should be used only on high contrast single-layer samples with sharp edges.

For a complete description of this hardware/firmware feature, please refer to the complete Video Autofocus Manual.

`CRISP` Continuous Autofocus System

The Continuous Reflection Interface Sampling and Positioning (CRISP) system provides for a very high level of focus stability, allowing a specimen to remain accurately focused for hours at a time with drift <0.1 μm. The system compensates for focus changes caused by temperature variations as well as mechanical drifts of the microscope mechanisms. The CRISP system promises to be a solution to focus drifts that plague time-lapse experiments at high magnification. The CRISP system uses a pupil obscuration method to determine focus from reflective surfaces. The control system allows adjustment of the focal lock position, relative to a nearby surface, once the system is locked. The unit is a C-mount device, which can be placed at the C-mount port. Usually it is used in conjunction with the dual C-mount Splitter (DCMS) so both the CRISP unit and a data recording camera can share the same microscope photoport.

System Overview

The CRISP system consists of optical, electronic, and mechanical components. The optical system injects IRLED light into the microscope, captures the beam reflected from the specimen slide or cover slip, and routes the reflected beam onto a position-sensitive detector (PSD). The signal from the PSD is conditioned by an amplifier circuit in the MS2000 controller and used as the feedback signal for Z-axis control. The MS-2000 Z-axis controller changes the focal position of the microscope either with a servomotor or with a PZ-2000 piezo Z-axis stage.
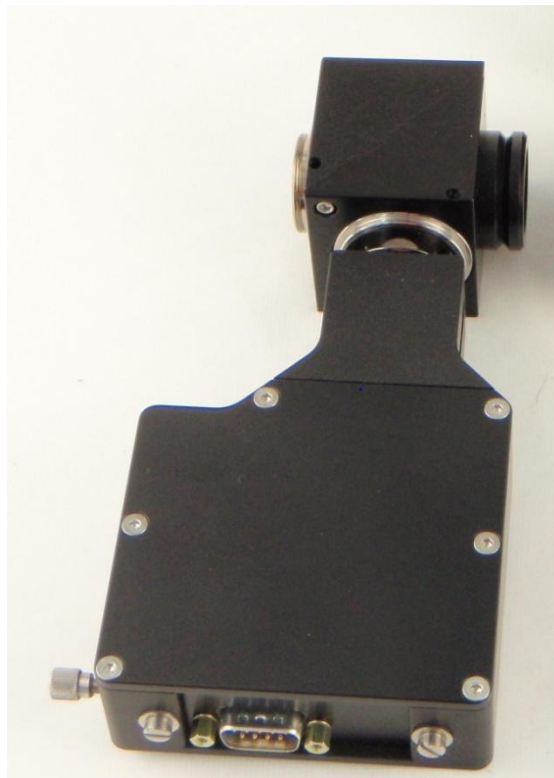

Figure 1: CRISP with DCMS photo-port splitter.

As shown in Figure 2, a dichroic beam splitter that reflects light from the IR LED and passes visible light to the camera is used to couple the CRISP unit to the system at the C-mount photo-port.
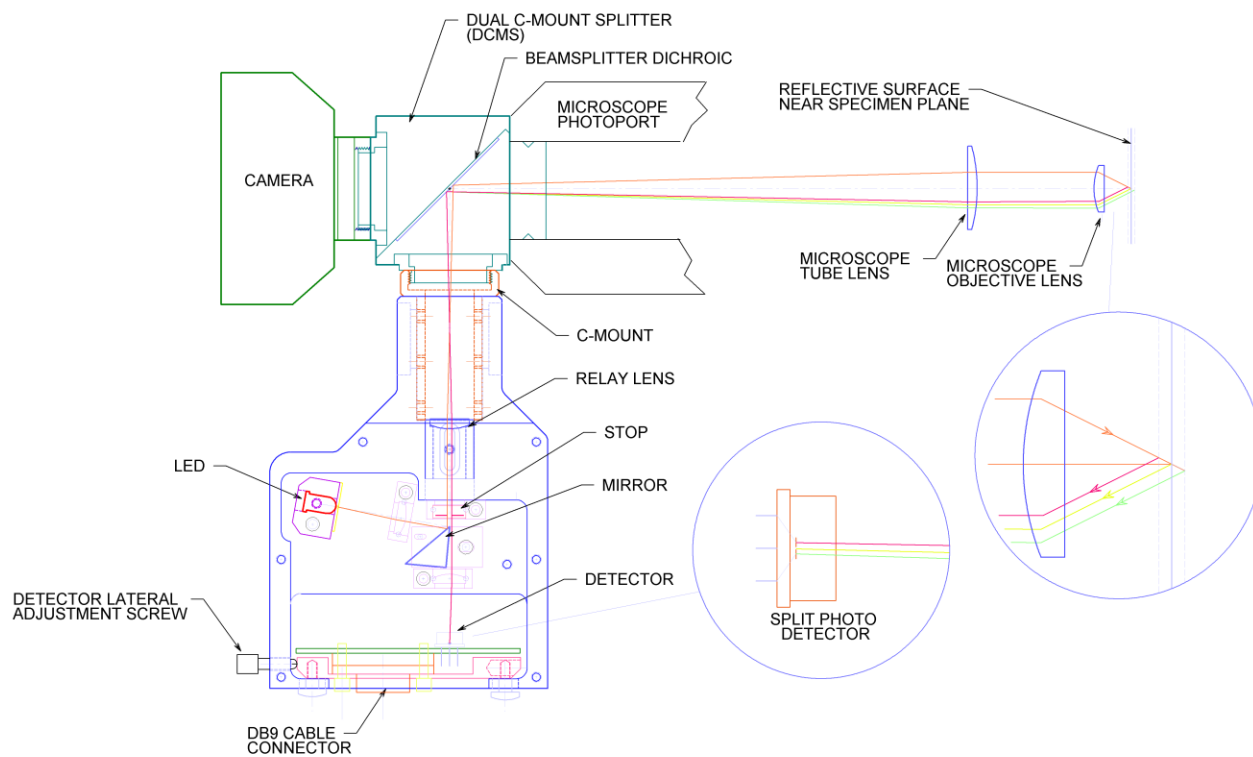
Figure 2: Schematic diagram of CRISP optical system

Please see the complete CRISP Instruction Manual for more details.

`DAC OUT` 16 Bit Ten Volt DAC channel

This firmware directive allows use of the on-board 16 bit DAC for general use. The DAC is reserved for driving external 0-10V piezo z-axis devices when required. However, if the stage system does not need the DAC for that purpose, then this directive allows it to be put to general use. The serial command for this purpose is the WRDAC command.

WRDAC (firmware 8.4f+)   DAC OUT firmware required

Format: WRDAC X=n

Function: Lets the user set the voltage on header pin SV1-5 on WK2000 board. The voltage can be varied between 0 and 10 Volts, with an accuracy of 0.1V. Maximum Output drive current is 35mA. Input value in volts. Does not work with Piezo units.

Reply: If there are no errors, a positive response of " *:A*" will be returned.

Example   **WRDAC X=1.1**

       :A   (Voltage on PIN SV1-5 is 1.1Volts)

       WRDAC X=20 OR -1

       :N-4   (Parameter out of range)

`LED DIMMER` TTL Pulse Width Modulation for LED Control

The LED DIMMER module provides for a user settable Pulse Width Modulation output on the TTL OUT port of the MS2000 control. This output can be used in conjunction with a getable LED such as the ASI MIM-LED-LAMP to provide an automated brightness control function. The serial command for this purpose is the LED command.

LED   **LED DIMMER** Firmware Required

Format: LED [X= *0 to 99*]

            LED X?

Function: Sets the brightness of ASIs LED illuminator by generating PWM on TTL OUT. TTL out mode should be set to '9' (i.e. `TTL Y=9`). The Enable input on the LED illuminator should be connected to TTL OUT on controller. This setting can be saved in non-volatile memory using the **SAVESET** command.

`PLANAR CORRECTION` Firmware Module

This manual section applies to the PLANAR CORRECTION firmware module. With this module included in the firmware, the controller will support functions to establish a tilted focal plane by selecting three sample points. The controller can then be enabled to make a Z-axis correction such that commanded Z positions are specified with reference to the established plane. Joystick moves also follow the adjusted plane.

LCD Display

The status line of the LCD display (DIP SW 2 UP) will show the status of the Planar Correction three-point acquisition and enable state.   A single character on the bottom line contains the state information. Usually it will look something like this:

<center>

`HRR1000:000 00:45:23`

</center>

State

or this, for systems with SEQUENCER and ARRAY MODULE:

<center>

`1        IIIIII00:45:23`

</center>

State

The meanings of the state indicators are described below.

1	Awaiting selection of first point
2	Awaiting selection of second point
3	Awaiting selection of third point
Z	Z correction enabled
G	Z correction not enabled (points selected)

When the state characters are displayed in lower case on the LCD display, the TTL IN0 interrupt is disabled pending completion of the current move operation.

Establishing the Correction Plane

To establish the location of the correct focal plane, three linearly independent points need to be selected. The simplest procedure is to use the joystick to move to the three points near the edges of the sample region of interest, focus the sample, and select the points with the joystick button. The LCD display can aid in guiding the user through the process.

Using the Joystick Button to Select the Focal Plane

The button on the JOYSTICK, held down for three seconds, is used to select a correctly focused XYZ location. The HOME button, held down for three seconds, will turn off the planar correction and initialize the firmware for subsequent point selection. These are the steps:

1) Check LCD. State should be '1' indicating controller ready to accept points.
2) Move stage to first XY point; focus on specimen surface.
3) (Optional) Press ZERO button on controller. This will enforce Z=0 is on focal plane.
4) Hold down Joystick button for three seconds. Verify that the state indicator on the LCD changes to '2'.
5) Move stage to second XY point; focus on specimen surface.
6) Hold down Joystick button for three seconds. Verify that the state indicator on the LCD changes to '3'.
7) Move stage to third XY point; focus on specimen surface.
8) Hold down Joystick button for three seconds. Verify that the state indicator on the LCD changes to 'Z'. The focus plane has been defined and the correction is being used for all Joystick and Commanded moves.

Using the HOME Button to Disable and Reset the Focal Plane Correction

The HOME button, held down for three seconds (or six seconds if RING BUFFER is included in the firmware as well) will turn off the focal plane correction if it is currently enabled, and will reset the place selection processes if the correction is already disabled. When the planar correction is disabled, but the three points have previously been defined ('G' state), you can reenable the correction with a three second press of the Joystick button.

Using Serial Commands to Select the Focal Plane

The CCB command can be used to automate the focal plane selection process. In general, some method of autofocus would be required to find the Z focus locations for the three point correction. Software autofocus or ASI's hardware video autofocus could be used. Please see the programming manual for details of the CCB command.

Stage Control with Planar Correction

Once you have established planar correction, you can pretty much forget that you have anything special going on. The stage will respond to Joystick moves, and manual focus moves with the controller's focus knob as expected. Commanded moves will be with respect to the corrected surface, again more or less as you would expect. The main LCD display shows the corrected Z-position and the WHERE command responds with the corrected current position. If you are curious to see how much correction is actually being made at any particular point, the two-column LCD display mode (DIP SW 1 UP) will show the true encoder target position on the right side of the Z-axis line. For example:

```
Z 0.00005> 0.041295E
```
Corrected Position   Actual Stage Encoder Target

You will notice that as you move around in XY, the corrected position will remain constant, but the actual stage encoder target will change as the correction demands.

Known Issues

Whenever the coordinate system of the stage is changed, as is done with the ZERO or HERE commands, or with the ZERO button, the height of the correction plane will change. For the ZERO command and ZERO button, the focus offset is also set to zero at this location. Hence, if the slide is in focus, pressing the ZERO button will not change anything. The HERE command for the X or Y axes will change the height of the focus offset, although the planar tilt will not be affected. After a HERE command, you should expect to have to find focus again.

Saving a Planar Correction to Flash

You can use the planar correction to "level" a sample holder once, and then save the correction to flash and it can be used again on start up. The save settings command "SS Z" saves the current correction tilt variables along with the Enable status to flash memory. When the controller is power cycled or reset, the saved variables are used when the controller initializes.

`RING BUFFER` Multi-Point Save/Move

Often there is a need to mark several positions to later revisit. The `RING BUFFER` module enables a buffer with up to 50 locations that the user can load with position information and then visit sequentially.

Using controller buttons to manipulate the RING BUFFER

The easiest way to use the ring buffer is to save positions and revisit them using the buttons on the controller. The default button functions are listed below. In some cases, additional firmware modules may preempt these button functions. Contact ASI if you have questions.

Save locations using the Joystick Button

The current stage position can be saved to the buffer by depressing the button on top of the joystick and holding it for one to three seconds. (A short tap of the button toggles the joystick speed) You can move to the next position of interest and again save the position in the buffer by holding down the joystick button. Continue this procedure to save all positions of interest.

Revisit locations using the @ Button

Save locations can be revisited by pressing the @ button briefly. Each press of the @ button causes the stage to advances to the next position. When you reach the last position, the next press of the @ button will take you back to the first position.

Clear the buffer with the HOME Button

Holding the HOME button down for longer than one second will clear all the stored positions in the ring buffer.

Using serial commands to load the buffer and control automated moves

The ring buffer may be preloaded with values via the serial command LOAD. Serial commands can also be used to advance to the next position as well as to control which axes will be affected by the move commands. See LOAD, RBMODE, and TTL commands in the Programming Manual for details. Contact ASI for details.

`SCAN MODULE` MS-2000 Encoder Sync and Scanning Addendum

Raster and Serpentine scanning with the MS-2000 controller has been made simple and flexible with some special firmware and hardware options added to our standard controller. Raster scanning is often used in conjunction with other equipment which collects data during the scan, the objective being the generation of data that can be mapped accurately in the two spatial scan dimensions. To facilitate this data collection, the MS-2000 controller with the Encoder Sync option provides two TTL signal channels that are clocked to positional references. These signals are wired from the axis chosen by the customer when the controller is manufactured. The first channel, SYNC, provides a "start-of-line" reference signal that occurs as the stage scans across a predetermined positioned for every raster-scanned line. The second channel, ENCODER, provides direct encoder pulses from the active encoder (rotary, or linear if installed) of the raster-scanned axis. Alternatively, by switching a toggle switch on the back panel, the ENCODER output gives a pulse every N'th encoder pulse, where N is programmable with the SCANR command. In this way, the ENCODER signal can effectively be the pixel clock of the data acquisition system, where the user can program the distance between pixels.

Raster and Serpentine Scanning

The user can select either *Raster* or *Serpentine* scanning modes using the SCAN F=mode command. During a *Raster* scan, the active scan begins at the same edge for each line, and is followed by a high speed retrace movement at the end of the scan line. For a *Serpentine* scan, the scan proceeds first in one direction, then in the opposite direction, advancing vertically one line each time the direction changes (there is no retrace movement). The "start-of-line" reference position is moved from one side of the scan area to the other depending upon the direction of the scan.

Raster scanning has the advantage that system backlash will be negligible, since the motion is always in the same direction. Serpentine scanning has the advantage of faster coverage (since there is no retrace operation) when very high accuracy is not required.

Encoder SYNC Hardware

The encoder sync pulse is obtained directly from the encoder counting chip inside the controller. This encoder chip can be loaded with a "compare" position, so that whenever the current encoder count is equal to the "compare" count, a flag pin on the chip becomes active. We add additional circuitry to latch the sync pulse during the active part of the scan.

A toggle switch on the back panel selects the pulse source for the ENCODER output pixel clock. Pulses can be derived either directly from the encoder (DIRECT), or from a processed signal coming from the controller's microprocessor (DIVIDE). In the latter case, the pulses from the encoder are sent to an interrupt line on the controller's microprocessor. The interrupt causes a programmable counter to count down. When the counter reaches zero, the microprocessor sends a pulse to the ENCODER output connector. These pulses are only enabled during the active scan line.
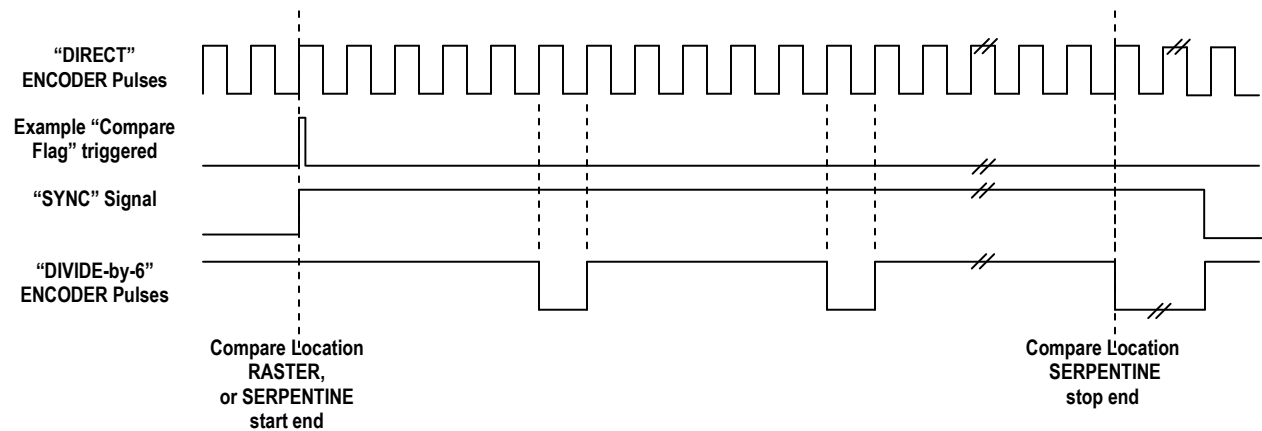
Figure 3: Timing Example

WK2000 Board Jumpers

Either the X or Y-axis can be used as the fast raster axis for scanning. Jumpers on the main controller card are used to select the appropriate wiring. The firmware defaults with the X-axis as the fast axis. When configured at the factory, the jumpers are set for a fast X-axis. The jumper configuration is shown in the table below.

Function	Jumper	Fast Axis	
		X	Y
Sync Flag	JP1	1-2	2-3
Encoder Pulses	JP2	1-2	2-3

The output signals are found on header SV1. SV1 Pin 2 is the Encoder Divide by N output and SV1 Pin 7 is the Line Sync output. These pins should be connected to BNC connectors on the rear of the controller. (Controllers not specifically set up for the SCAN firmware will usually have SV1 Pin 2 connected to the TTL OUT BNC and SV1 Pin1 connected to the TTL IN BNC. In that case, switch the wire from Pin1 of the connector over to Pin 7 so that the Line Sync will appear on the TTL IN BNC.)

SCAN firmware versions that have the IN0_INT module rather than the ENC_INT module should leave JP2 unconnected since encoder pulse counting is not enabled in these versions. The ENC_INT module provides the interrupt service handling so the encoder pulses can be counted. The internal interrupt service must be enabled using the TTL command, **`TTL X=1`**.

Using the SCANR, SCANV, and SCAN Commands

To perform a scan, you must first define the scan region, the pixel clock resolution, and the number of scan lines. The serial commands SCANR, SCANV, and SCAN let the user define the

scan region, the scan mode, and initiate the scan process. See the programming section for a full description of these commands.

Let's go through an example where we wish to scan a 1 mm × 1 mm region at a resolution of about 0.5 μm on an XY stage. We will consider the X axis to be the 'horizontal' direction, and Y to be the 'vertical' direction. (Although any axis is eligible to be 'horizontal', only one will have been wired at the factory for encoder output signals – see the **SCAN** command's parameters for alternative configurations.) For a standard 6.35 mm pitch lead-screw stage, the encoder resolution is 45396 counts/mm or 0.022 μm/count. However, this resolution includes all of the quadrature edges of the two encoder channels. Our counting circuit is only looking at the positive edges of one encoder signal and hence sees a count spacing four times larger. The maximum count rate is every other pulse – which translates to 8 quadrature counts per output pulse. (If you are not sure of the encoder resolution of the scanned axis, issue the command **INFO X**, where **X** is the axis in question, to query the controller about the parameters that are used.)

If we generate a clock pulse every 24 encoder counts (6 full periods of a single encoder signal), then our pixel clock resolution will be 0.022 × 24 = 0.528 μm. To set up the 'horizontal' raster, we issue the command:

SCANR X=0.0 Y=1.0 Z=24   (X= start, Y= stop, Z=encoder_divide)

The command above will generate a scan line with 1891 pixels, determined by multiplying the encoder resolution by the length of the scan and dividing by the encoder_divide parameter. If you wish to specify exactly the number of pixels you want to have in a scan row, you may issue the command using the F = # of pixels, instead of Y = stop_positon.

For the 'vertical' direction, if we want square pixels, then we will need 1891 scan lines as well. To set up the 'vertical' scan, we issue the command:

SCANV X=0.0 Y=1.0 Z=1891 (X=start, Y=stop, Z=number_of_lines)

In both commands, parameters "start" and "stop" are measured in millimeters from the current origin ("Home" position), and negative values are allowed.

Next, we need to determine the correct scan speed based upon the processing time for each pixel. If we assume a 1ms time for each pixel, then we would want to scan at a rate of 0.528 μm/ms or 0.528 mm/s. To set this up for the X-axis we issue the command:

SPEED X=0.528

We can specify that the scan mode will be a raster scan using the command:

SCAN F=0

Finally, we need to turn on the encoder interrupt with:

TTL X=1

Once all of the scan parameters are set up the way we want them, it is possible to save them to non-volatile memory so that we do not need to re-enter them each time we power up.

The command to save settings is:
SS Z

When ready to initiate the scan, issue the command:
SCAN

to start the process. Alternatively, you can use the @ button on the controller to start and stop the scan. To start a scan, hold the @ button down for more than 1 second. When you release the button the scan will start. To abort a scan in process, momentarily depress the @ button.

Limitations and Internal Operation of the SCAN module

During set-up for a scan, the number of horizontal pulses (pixels) is calculated. During the scan, pulses are counted until the requisite number has gone by. Further interrupts are disabled and the current encoder position is read in. If the current position is not consistent with the expected position based upon the number of pulses and the encoder counts between pulses, an error code is placed in the error log.

The internal data size for certain variables limits the acceptable range some parameters. The *encoder-divide* parameter and the total number of pixels per scan line must be less than 32768.

With the MS2000WK controller, internal jumpers JP1 and JP2 are used to select either the X-axis (jumper positions 1-2) or the Y-axis (jumper positions 2-3) as the fast scanned axis. Controllers are shipped with the X-axis as the default fast raster axis. Please see the SCAN command to change the default axes for the scanning algorithms. Only the X (default) or Y axis may be used when generating the hardware encoder synchronization TTL pulses. The firmware controlled scan patterns can be used with any pair of axes, however internal errors (code 85 & 86) will be generated if the proper internal signals are not present or jumped correctly.

SEQUENCER – MS2000 Programmed Sequence Control

Overview

The SEQUENCER module consists of several hardware signal outputs, both TTL signals (TTL#) and Analog Voltage Outputs (AVO#). Incremental stage motion can be directly controlled in a sequence as well (STG#). A TTL trigger INPUT and various RS232 serial and manual control inputs are also supported. Each hardware output is configurable using a serial command. Logic connecting the inputs and outputs is implemented using a programmable BLOCK structure. BLOCKs, by themselves, do not produce output. Instead they provide the structure for building the event sequence required for a task. BLOCKs and OUTPUTs can have START, REPEAT, STEP, and RESET *condition* codes. They also can have DELAYS and a number of REPETIONS associated with them, as well as completion *action* codes.

Conditions

Condition codes connect events with logical progressions of the BLOCKS and OUTPUTS.   Not all condition codes are applicable in every situation. The table below lists the valid **CONDITION CODES** with an asterisk indicating which codes are available for the logic commands.

CONDITION Description	CODE	BLK START	BLK REPEAT	TTL START	TTL STOP	AVO, STG, or LST STEP	AVO, or STG RESET
NEVER	**0**	*	*	*	*	*	*
TTL_TRIGGER_RECEIVED	**1**	*	*	*	*	*	*
ARM_COMMAND_RCVD	**2**	*	*	*	*	*	*
AT_BUTTON_PRESSED	**3**	*	*	*	*	*	*
XYZ_STAGE_NOT_BUSY	**4**	*	*	*	*	*	*
BLOCK_#n_DELAY_COMPLETE	**5**	*	*	*	*	*	*
BLOCK_#n_COMPLETE	**6**	*	*	*	*	*	*
BLOCK_#n_REPEAT	**7**	*	*	*	*	*	*
BLOCK_#n_REPEAT_or_START	**8**	*	*	*	*	*	*
BLOCK_#n_DELAY_COMPLETE_or_START	**9**	*	*	*	*	*	*
BLOCK_#n_REPEAT_or_COMPLETE	**10**	*	*	*	NA	NA	NA
BLOCK_#n_REPETION_#m	**11**	*	NA	*	NA	NA	NA
ALWAYS	**12**	*	*	NA	NA	NA	NA
ARRAY_MOVE_DONE (**ARRAY_MODULE**)	**13**	*	*	*	*	*	*

Unused BLOCKS and OUTPUTS are set up by default with all codes set to NEVER .

The condition codes number 5 through 11 refer to the state change of a particular block. When using these codes you must also supply the block number in question, even if it is the same as the block you are defining.

Actions

Action codes connect BLOCK *completion* with MS-2000 controller actions.

Description	Firmware Module Used	Code	BLK END
ACTION_IDLE	none	0	*
ACTION_RING_BUFFER_NEXT_POSITION	RING_BUFFER	1	*
ACTION_INITIATE_AUTOFOCUS	AFOCUS	2	*
ACTION_ARRAY_NEXT_POSITION	ARRAY_MODULE	3	*
ACTION_ARRAY_START	ARRAY_MODULE	4	*
ACTION_SEND _WHERE	none	5	*
ACTION_SEND_TIMESTAMP	none	6	*
ACTION_SEND_STATESTAMP	none	7	*

Presently not Implemented

Actions let the sequencer direct certain MS2000 controller functions. For example, RING_BUFFER or ARRAY_MODULE firmware can be used to set up a series of XY target locations. These positions can be traversed using ACTION_MOVE_NEXT_POSITION or ACTION_ARRAY_NEXT_POSITION codes at the end of a BLOCK.

BLOCKS

The BLOCK state logic structure is shown in the illustration below.

Figure 4: BLOCK logic structure.

Both delay times and repetitions are handled with the block logic. Delays are processed and repetitions are incremented. BLOCKS start when an event matching the START condition code occurs. If repetitions are programmed, the block waits for a REPEAT condition to continue. If delays are programmed, the delays time out before the block continues to check for the next REPEAT condition. Once all repetitions are complete, the block is COMPLETE and returns to the IDLE state. If no repetitions are programmed, any programmed delay is timed out before the block COMPLETEs. In Figure 1, transient conditions that can correspond to the CONDITION CODES are shown in **RED**.

Repetitions are processed before the delays are timed out.

BLOCKS without repetitions programmed can be used as delay generators.

BLOCKS without delays programmed can be used looping counters.

BLOCKS with both delays and repetitions can be used as timed sequence generators.

BLOCK SEQUENCING

The transitions points in the BLOCK sequence progression can be used as conditions to start, repeat, stop, step, or reset other sequencer components. Blocks spend most of their time either IDLE, waiting for a REPEAT condition, or timing out a programmed DELAY.
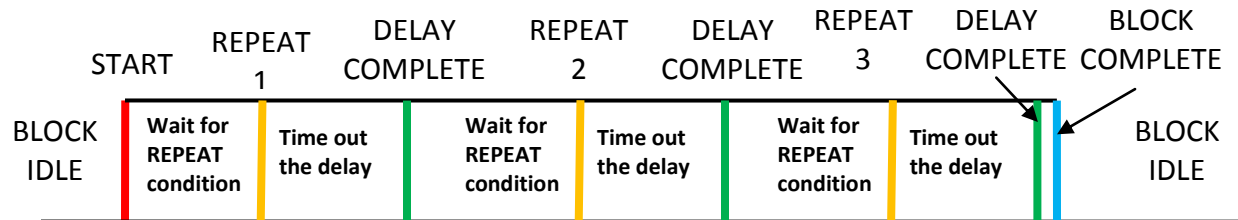

Figure 5: Basic BLOCK sequence execution.

Transition states in one block can be used to place another block in a transition state. There is the potential that several transitions states must be quickly processed in one time step. Up to six simultaneous transition states may be processed before a recursion error is flagged.

INTERACTIVE CONTROL

The SEQUENCER is programmed via the serial RS232 or USB port on the controller using the commands described in the sections below. There are several methods to start and/or stop a programmed sequence.   For interactive control, the controller **@** button provides a convenient master start button. You can use the condition code AT_BUTTON_PRESSED to start a "master" block. Once the sequencer is actively running pressing the **@** button will stop the sequencer and abort the running sequence. All stage motion is halted, and blocks with ALWAYS start conditions are place in the IDLE state.

Systems that include the ARRAY_MODULE for XY motion can use the @ button "long press" to start the XY array scan.

The SEQUENCER option is supported in the ASI-CONSOLE program to facilitate programming, saving and restoring sequence scripts.

See the complete SEQUENCER Manual for more details.

`TTL REPORT` Synchronous Encoder Reporting

This manual section applies to the TTL_REPORT firmware module. Including this module in the firmware allows for precisely synchronized encoder position reporting.   The user's hardware supplies a TTL trigger to the MS2000WK that causes a processor interrupt which performs an encoder-read operation on all of the controller's axes. The TTL IN0 interrupt must be turned on using the serial command "TTL X=6" prior to sending the read command pulses to the external TTL input.

Timing Specifications
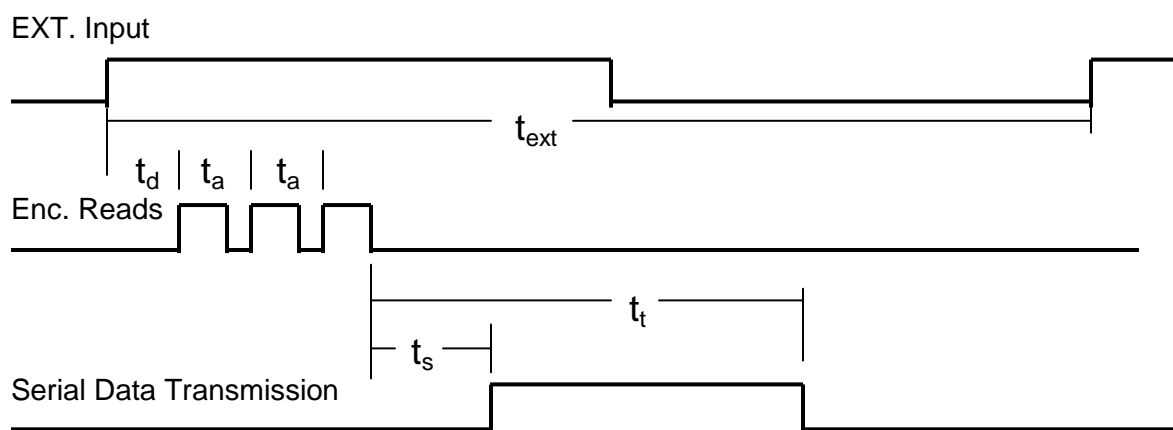
Item		Symbol	Spec. Values		Unit
			Min	**Max**	
Initial Read Delay		t_d	3.0	10.0	μs
Successive Axis Read Delay		t_a	11	13	μs
Serial Transmit Delay		t_s	110	500	μs
Transmit Time	115200 baud, three axes	t_t	1.2	1.75	ms
External re-trigger time	115200 baud, three axes	t_{ext}	1.7		ms

The encoder read operations follows the external trigger input as indicated in the timing diagram. The largest jitter component comes from the delay in executing the interrupt service routine that performs the read operations. Most of the time the interrupt proceeds with the minimum delay time specified. Occasionally, the reads are delayed until current data buss operations complete.

Data Transmission
The encoder position data is transmitted using the auxiliary serial port on the MS2000WK controller. The data baud rate for this port is set to be identical to the main serial communications port and is selected with DIP Switches 4 and 5. The Serial Out port on the

MS2000 controller should be configured with a straight-through female-to female RS-232 cable to communicate with a PC.

Binary data is sent in the following form (here illustrated for a three-axis system):

AID	LSB	MB	MSB	HB	AID	LSB	MB	MSB	HB	AID	LSB	MB	MSB	HB	CR

The axis identifier **AID** is the Low Level Format axis identifier string. The next four bytes represent the 32-bit encoder position value for the axis. This is a signed long integer value, low-byte first, high-byte last. This pattern is repeated for all of the axes that are present in the controller. A final terminating <CR> character (hex 0x0D) completes the data stream.

The usual axis identifiers are as follows:

Axis	Decimal value	Hex
X	24	0x18
Y	25	0x19
Z	26	0x1A
F	27	0x1B

Other output configurations are possible. Data can be sent to the usual Serial IN port, and/or the data can be sent in ASCII form. Contact ASI for details.

Error Conditions

An error condition will exist if the TTL triggers occur more quickly than the data can be sent out the serial port. When this condition is detected the error

TTL_REPORT_BUFFER_OVERRUN   87

is appended to the internal error log buffer. If TTL triggers are present as the controller is powering up, some of these errors are to be expected.

TRACER CAM G-code Interpreter

Tracer is the project name for a set of firmware and software enhancements that enable an ASI stage to be moved along user-defined paths and assert triggering signals at user-defined places. (Without Tracer, ASI stages can move to user-specified locations, but they cannot follow any specified path to get there.) Tracer interprets G code output from computer-aided manufacturing (CAM) applications to implement these functions.

Tracer includes the following components:

- WK-2000 stage controller firmware modules with build names that include the suffix *_TRACER*, and

- a PC application, *Tracer Console (TC),* with which interprets G code files and controls the WK-2000.

Supporting Documents

Tracer's G code interpreter supports a subset of G code as described in the document *FANUC Series oi-MC OPERATOR'S MANUAL,* numbered *B-64124EN/01*.

The WK2000 with Tracer firmware supports the commands described in the ASI document *MS-2000 Operation and Programming Manual,* available for downloading at http://www.asiimaging.com/pdfs/Operations_and_Programming_Manual.pdf.

Document Conventions

In this document, non-readable data characters are represented as hexadecimal values in the formats 0xFF and **#FF**, may be delimited by a following space when part of a character string. The space character is represented by its byte value: **#20**. Readable characters appear as themselves or enclosed in single quotes, e.g., 'A'. For example, the following string includes the data characters 0xFF, '1', 'A', and 0x0D:
#FF 1A#0D or **#FF1A#0D**

This string may also be represented this way:
#FF #d1 #41 #0D or **#FF#d1#41#0D**

A variable field in a message is enclosed between angle brackets:

#07<status byte>#06#0d

Operation

When using Tracer to control the MS-2000, always set the serial baud rate to 115200 baud. You do that by setting DIP switches 4 and 5 on the back panel to the DOWN position and then restarting the MS-2000.

Tool functions include movement in the X, Y, and Z axes, and trigger on/off commands. In normal use, G code files generated by CAM software define all the movements.

Since microscope stage controllers with lasers are not supported by CAM software, ASI expects that users may need to edit, or at least be familiar with the contents of, the G code files. G code files are text files which can be edited by text editing software such as Windows Notepad.

All WK2000 controllers that support Tracer functions contain Tracer firmware and as such reply to the **BUILD** command (see *MS2000 Programming Guide*) with a word ending in **_TRACER**. These WK2000 controllers also function as standard units supporting the High Level Command set, plus joystick, knob, and button functions.

Tracer Console (TC) Operation

In order for the WK2000 to be controlled by G code, it is necessary to use the Tracer Console (TC) application. The application window is pictured in Figure 6.
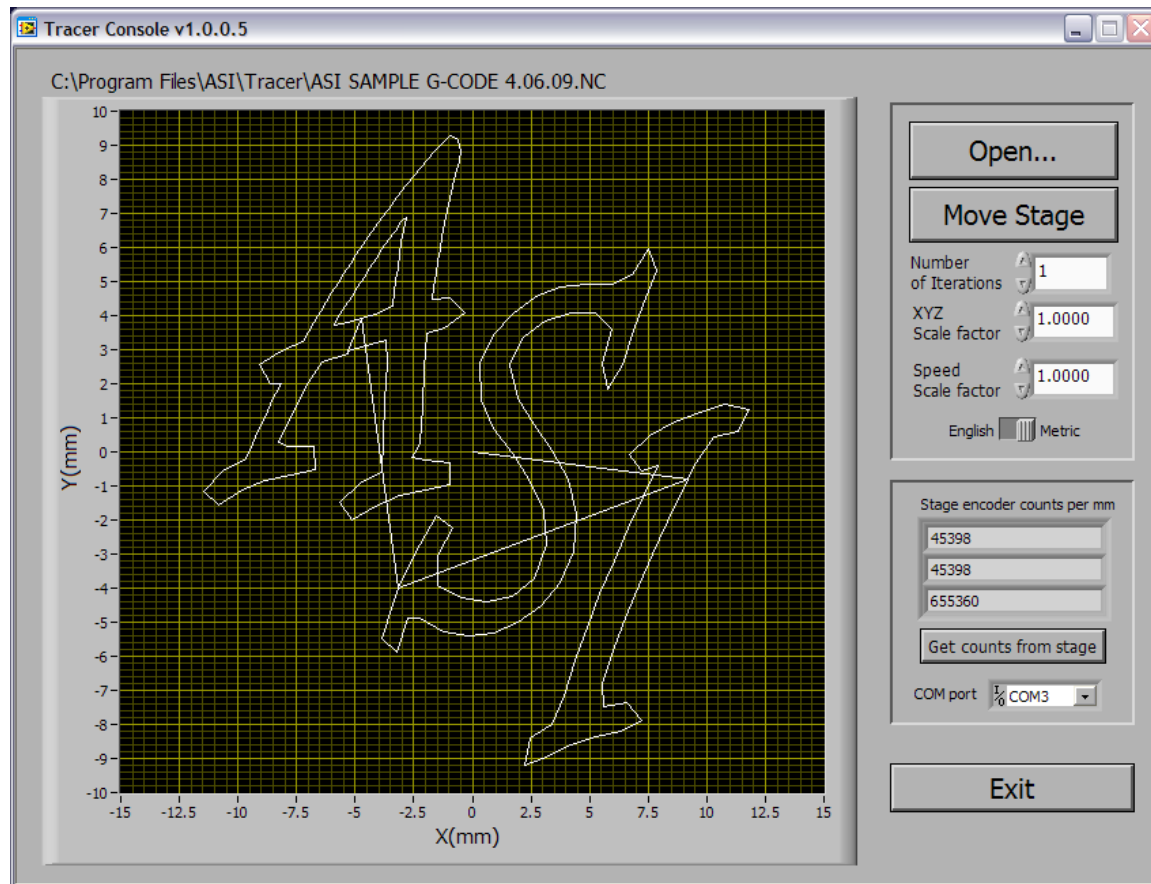

Figure 6 Tracer Console Application Window

After opening a file by clicking the **Open...** button, the shape to be traced by this file is displayed in the graph pane. Shown above is the example G code file included with Tracer Console.

The **Stage encoder counts per mm** panel is for information only. If the counts all zeroes, then communication via the serial port or USB is not working. After checking for the correct **COM port** setting, verifying cable connections, and correcting any contention for port usage by other applications, click the **Get counts from stage** button. This should correct the encoder counts per mm numbers. The most common values for X, Y, and Z are 45398, 45398, and 20000 or 655360 for piezo units.

When TC's **Move Stage** button is clicked after opening a G code, TC switches the WK2000 into Tracer mode, translates the G code into a stream of binary commands, transmits those commands in real time to the WK2000. The LCD display on the WK2000 shows even the smallest changes in position when the stage is moving.

When the moves are completed or stopped by clicking the **Stop** button, TC switches the WK2000 back to normal operating mode.

XYZ Scale factor sets a constant multiplier of distances moved. **Speed Scale factor** sets a constant multiplier of stage speed.

Supported G codes include the following, plus any and all associated commands, e.g., I, J, K, R, and F associated with G2 and G3:

- G00, G01, G02, G03
- G04 with P_ only, IS-B (time unit is ms)
- G15, G16
- G17, G18, G19
- G20, G21
- G90, G91
- M codes (See Table 1.)

	Name	**Function and Usage**
M03	Assert TTL output	**Function**: Causes a TTL level output to be asserted high at the BNC *OUT* connector on the MS-2000 back panel **Usage**: M03 This signal may be used to trigger a shutter controller such as the ASI SC-2000 or SH-2.
M05	De-assert TTL output	**Function:** De-asserts TTL output. See M03.

Table 1 M Codes

The file referred to in Figure 6, *ASI Sample G-code 4.06.09.NC*, was generated by CAM software for a computer numerical control (CNC). If a CNC were operated under the control of this file, it would move a tool tracing an image of the letters *ASI*.

Figure 6 illustrates the path followed by a machine under control of this file. The straight lines that are not part of the letters *ASI* show where the cutting tool been lifted away from the metal by the command **Z.1**.

The setup is calibrated so that Z=0, the cutting tool is minimally in contact with the metal surface. Each line with content **Z-.1** causes the cutting tool to be lowered by .1 length units into the metal. The subsequent sequence of X and Y moves causes the cutter to trace the desired path in the metal.

To illustrate a possible application based on this example, ASI has hand-entered the lines M03. A users's system may be configured so that M03 turns on a laser to an intensity corresponding to the last Z value setting, which would be -.1 in each case. Table 1 describes

There are two BNC connectors on the WK-2000 back panel. The M codes in Table 1 cause the ASI controller to assert a digital I/O external trigger signal at one BNC connector.

WK-2000 stage controllers can be configured to generate an analog voltage at the other BNC connector on the back panel in response to Z commands. These signals can be wired to a laser controller. In this configuration, the output at the connector ranges between 0 and 10 volts. Using the MS-2000 High Level commands SETUP and SETLOW, a range *min <= Z <= max* may be established. This range maps to the voltage range *10 >= Vout >= 0*. Suppose, for example, that *min*, the lower Z limit, is set to -50 microns using the command **SETLOW Z=-.05**; and the upper Z limit is set to 50 microns using the command **SETUP Z=.05**. Note that as *Z* increases from *min* to *max*, *V* decreases from *10* to *0*, and vice-versa.

A standard WK-2000 firmware build uses the analog BNC connector for piezo-driven focus control. It outputs a fixed range of 0-10V. The standard piezo system has a movement range of 100 microns. The Z limits of movement are expressed as -50 to +50 microns. Using this standard setting, the conversion of a Z value in microns to a V value in volts $V = f(Z)$ is as follows:

$V = 0.1Z + 5$.

Equation 1 Laser Voltage Output as f(Z)

Conversely,

$Z = 10V - 50$.

Equation 2 Z Position as f(V)

Error! Reference source not found.Error! Reference source not found.Error! Reference source not found.Error! Reference source not found.

`TRACKING` ASI PhotoTrack System

The PhotoTrack quadrant photomultiplier (PMT) is a C-mount device where the photo-cathode of the PMT is located at the C-mount image plane. The PMT may be attached to any microscope C-mount and be correctly positioned, however it is more common to interpose a photo-port beam splitter so that the instrument can be used with the data-recording camera. The ASI Photoport Beam Splitter works with most inverted microscopes to provide two direct C-mount ports at the microscopes design location. The splitter uses a 36.0 x 25.5 x 1.1 mm plate beam-splitter, which is the common size for most Zeiss filter cubes. Either partially silvered mirrors or dichroic beam splitters can be used in the splitter depending upon the application.
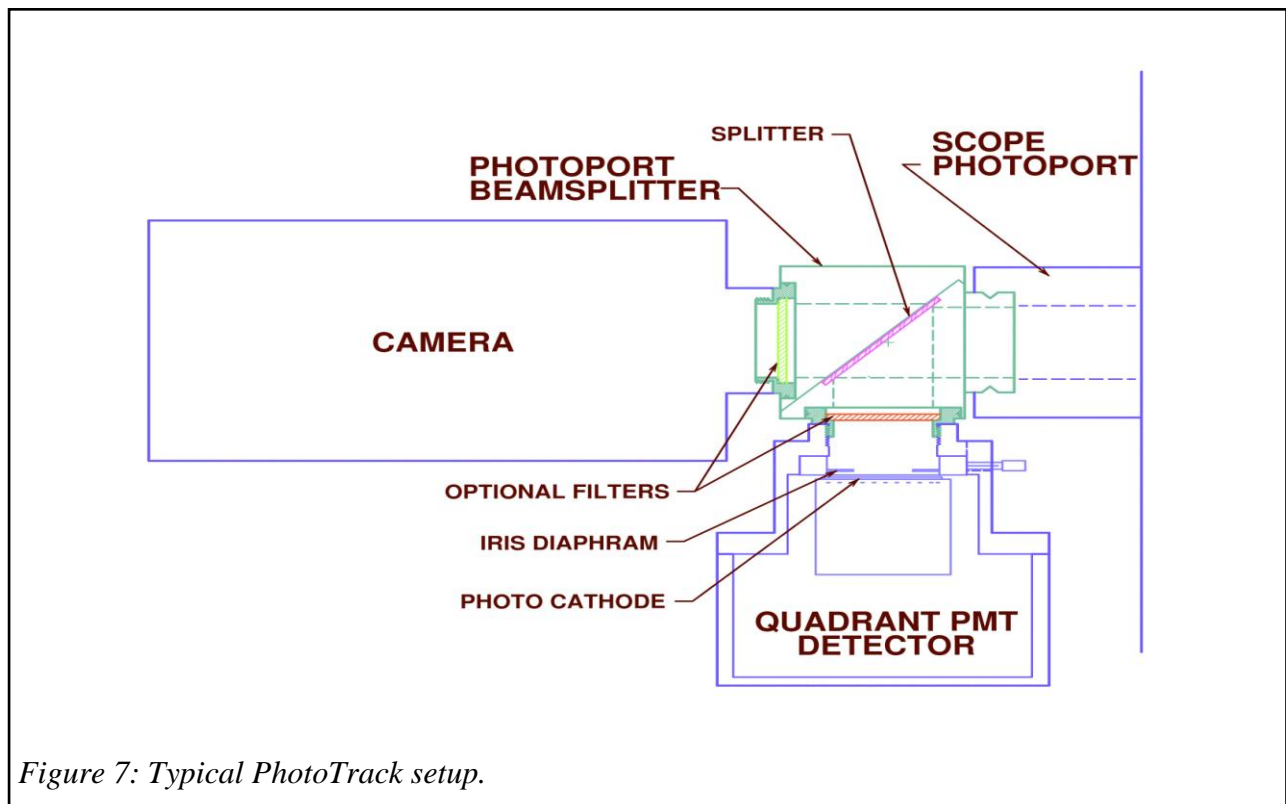

Figure 7: Typical PhotoTrack setup.

Please see the PhotoTrack Manual for details.