

## Rapid Array Scanning with the MS2000 Stage

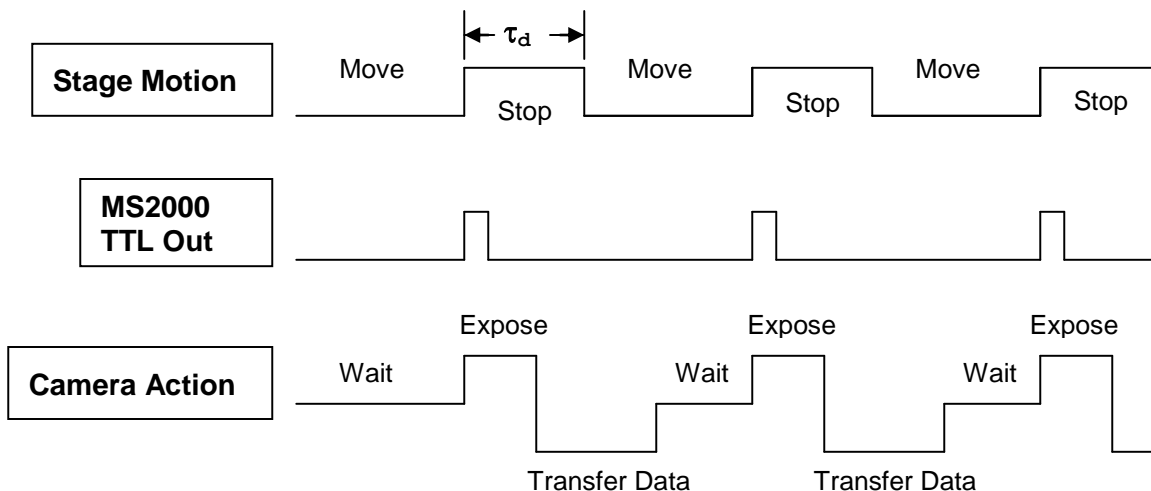
### Introduction

A common problem for automated microscopy is to quickly image an extended region that is larger than the camera field of view. This can be done by rapidly moving the stage between successive fields, stopping to snap an image at each location. Rapid stops and starts expose the inherent limitations in the mechanical system. These limitations include the inertia of the motor rotor and stage, current limitations of the motor, and how quickly the system can detect and correct motion errors.

The MS2000 stage and controller provides several advantages over conventional stepper stages for this type of application. The coreless DC motors used in the MS2000 stage have very lower rotor inertia with an overall mechanical time constant of only 10ms. Stepper motors with steel and copper armatures are inherently much more sluggish. The servo loop time of the MS2000WK controller can be as low as 1ms for a single axis system, allowing plenty of opportunity for control compared to the mechanical time constant of the system. In order to take full advantage of the hardware capabilities that are inherent in the MS2000 system, we offer specialized control firmware and provide for custom servo loop tuning. The rest of this note will discuss these specifics and provide some example applications.

### Imaging Strategies for Rapid Array Scanning

Best performance can be obtained if there is close synchronization between the camera and the stage. A camera that can be externally triggered with a TTL signal allows the stage controller to initiate the exposure command. The stage has the most temporal jitter and uncertainty because of its inherent mechanical nature. Using the stage to trigger image acquisition provides the tightest synchronization. If TTL camera triggering is not possible, then the camera control software will need to poll the stage "Busy" state over the serial interface, or act on the TTL signal from the stage controller to know when to trigger the camera.



## The ARRAY MODULE Firmware

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=8.0 F=-8.0**

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, without 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. To synchronize with a camera using this mode, use the TTL output pulse ("**TTL Y=2**") to trigger the camera.

### ***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.

<b>I</b>	<b>IDLE</b>
<b>S</b>	<b>START</b>
<b>M</b>	<b>MOVING</b>
<b>D</b>	<b>DELAY</b>
<b>A</b>	<b>ARRIVED</b>
<b>N</b>	<b>NEXT_POSITION</b>
<b>C</b>	<b>COMPLETE</b>
<b>E</b>	<b>ERROR</b>

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.

## **The Speed versus Accuracy Trade-off**

The ideal stage moves exactly to target in no time and stays put once it is there. In reality, motors have to push and pull the mechanics, sensors have to read position, and control electronics need to coordinate the motion. All of this takes time, and timing uncertainty leads to position uncertainty; rapid acceleration leads to jerky motion and further position uncertainty. To make matters worse, stages often don't like to stay put on the sub micron level.

### ***Linear versus Rotary Encoders***

For array scanning applications especially when generating montages, it may be important that the rows of scanned images be well aligned. When scanning in a serpentine pattern, mechanical backlash in the system can lead to systematic row-to-row errors because of the opposite direction of motion along the row as the images are obtained. This problem will be much more noticeable when using rotary encoders on the stage motors than with linear encoders on the stage plates. With rotary encoders, systematic errors of several microns from row to row are to be expected when scanning in a serpentine pattern. Despite the backlash errors, repeatability and relative spacing between frames can be quite good. Linear encoders can be used to largely eliminate backlash errors and provide excellent absolute positioning accuracy across the entire array, but at a significant added cost.

If one wishes to avoid the added cost of linear encoders, and still get good row-to-row frame alignment, then scanning in a *raster* pattern rather than a *serpentine* pattern will ensure that all of the frames are acquired when moving from the same direction.

### ***Backlash Correction***

The MS2000 controller default is to use a backlash correction algorithm with rotary encoded stages. This ensures that the move as the stage reaches target is always from the same direction. This works well for moves to arbitrary locations, but is not recommended for array scanning, and should be turned off with the serial command “**B X=0 Y=0**”. The added little movement jog of the backlash correction takes time and stresses the motors more than just using a *raster* scan pattern, which can accomplish the same goal as the backlash correction algorithm.

### ***Lead Screw Pitch***

Faster speed lead screws can lead to quicker moves, but the speed advantages decrease when high landing precision is required or when the move distances shrink.

### ***Acceleration***

Stage acceleration is set by specifying the time for the stage to ramp up to maximum velocity. Typical default value for this parameter is 100ms. When making lots of small moves, reducing the ramp time can improve throughput. Very short ramp times (<20ms) could potentially damage the motors due to excessive motor commutator current. Ramp times >20ms are generally safe for microscope stage motors.

### ***Post-Move Control: Choosing the Finish Error and Drift Error Parameters***

The speed/accuracy tradeoff becomes clear when setting the motion error tolerances. As a rule of thumb, the stage will arrive on target with no hunting when the finish tolerance is 5 to 10 encoder counts. For rapid scanning you want no hunting, so in general specify the finish error at about 10 encoder counts and the drift error slightly higher. The INFO command will show the value of the Finish Error (PC command), both in units of millimeters and as the number of encoder counts.

Upon reaching the target position to within the Finish Error tolerance, the move is considered complete. The BUSY flag is released for subsequent serial STATUS requests, and if programmed, a TTL pulse will be provided to indicate the end of the move. Default behavior is for the motors to be turned off at this point, and only turned on again if the encoder error becomes larger than the Drift Error parameter setting. Alternatively, the motors and servo loop can remain active by using the command “MA x=2 Y=2”. This allows the servos to continue to nudge the stage toward target even after the BUSY is released. See the MAINTAIN command in the Programming Manual.

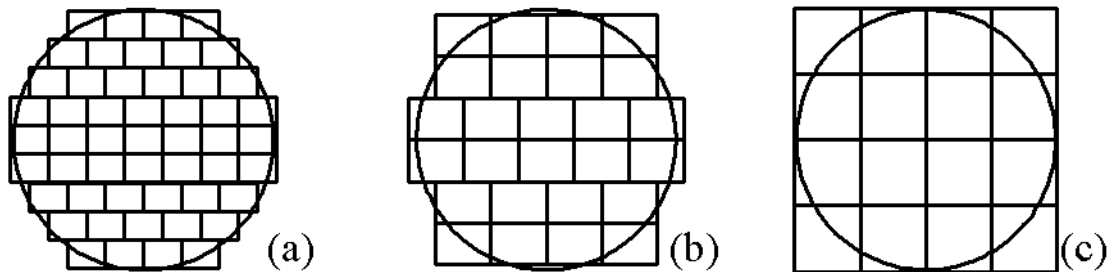
**Table 1: Scanning performance for various lead screws and move distances**

Lead Screw Pitch (mm)	Maximum Stage Speed (mm/s)	Rotary Encoder Resolution (nm)	Typical Move Accuracy ( $\mu\text{m}$ )	Typical Move Times: PC=10cnts; AC=30ms; Backlash Correction Off		
				0.1 mm (s)	1.0 mm (s)	10.0 mm (s)
1.59	1.6	5.5	0.7	0.106	0.668	6.40
6.35	6.4	22	1	0.051	0.195	1.60
12.7	12.5	44	2	0.038	0.120	0.82
24.5	24	88	4	0.035	0.090	0.50

The table above shows typical performance for a stage with rotary encoders making moves of various distances. Notice that for very short moves, typical for one field of view with a high power objective, faster lead screws provide very little overall speed improvement and significantly degrade the stage accuracy performance. For longer move distances, faster screws make a marked difference in the time per move and may be worth the loss of accuracy. The highlighted row shows the common configuration for microscopy scanning applications that represents an optimal compromise between speed and accuracy.

## Camera Sensor Size and Throughput

The few steps and exposures that are needed to characterize the sample; the faster will be the process. The camera format can make a big difference on the system throughput. As a simple example we will consider the number of exposures necessary to cover a single well of a 96 well plate. Figure 1 illustrates possible tiling with three different camera formats when using 10X magnification.



**Figure 1: Tiling a well at 10X with camera formats a) 2/3", b) 1", c) 15.2mm square.**

The 2/3" camera requires about 51 images optimally tiled or 63 images if tiled in a square array. The 1" camera requires 26 to 30 images, and the 15.2mm square sensor needs 14 to 16 images to cover the well. With the ARRAY MODULE firmware we can easily set up a rectangular array to cover the wells, corresponding to the 63, 30, or 16 images for the three formats. Using a rotary stage with 12.7mm pitch lead screw, we can measure the time to make these moves with the ARRAY MODULE. The table below shows the results using stage settings as in the highlighted row in table 1.

**Table 2: Time to scan a well plate with various camera sensors**

Camera Sensor	Number of Images per well	FOV Size (mm)	Scan time for one well (s)	Total move time for full well plate (s)	Total scan time w/ 15ms exposures for full well plate (s)
2/3" (8.8 mm x 6.6 mm)	63	0.88 x 0.66	6.75	725	816
1" (12.8 mm x 9.6mm)	30	1.28 x 0.96	4.0	461	504
1.2" (15.2 mm x 15.2mm)	16	1.52 x 1.52	2.5	317	340

Scan time for the full well-plate includes the time to move from well to well in addition to the scans of each individual well. Short image exposure times are included in the last column. Again, few images means few exposures – so higher throughput.