MotionCode™

Step-by-step solutions for motion applications

Galil MotionCode Solution:

Increased-Throughput Semiconductor Metrology Instrument

Contents **Page**

1. MACHINE DESCRIPTION 1
2. REQUIREMENTS 1
3. COMPONENTS SELECTED 1
4. IMPLEMENTATION 2

1. Machine Description

During semiconductor fabrication, a wafer goes through microlithography, implant, deposition, and etch. After deposition, the wafer is removed and brought to a metrology station to inspect the film thickness on the wafer. Each inspection may take measurements at hundreds of XY locations on the wafer; thus, maintaining high throughput is extremely important.

Galil's DMC-18x6 Accelera controller will be used to increase the throughput of a metrology instrument (*Figure 1*) originally developed around the DMC-18x0 Optima series, which was a limiting factor in the machine throughput. The DMC-18x6 provides increased software processing power as well as faster servo loop update rates.

There are three axes on this metrology instrument: two axes (XY) of linear motors with 1.0 µm resolution control the position of the wafer while a single Z axis linear motor also with 1.0 µm resolution raises and lowers the film thickness sensor.

Multiple measurements are taken in a raster pattern across the wafer surface. A single measurement cycle consists of the following:

(1) The XYZ measurement location is sent from the PC to the controller.
(2) The wafer and sensor move to their destination.
(3) At each measurement location, XYZ motion must settle to within +/-3.0 µm for 1 second before taking a measurement.
(4) Once the axes have settled, an analog film thickness sensor is sampled 500 times every 2 ms.
(5) An average film thickness is computed.
(6) The average film thickness along with actual XYZ locations are sent to the PC.

The above process is repeated for each measurement.

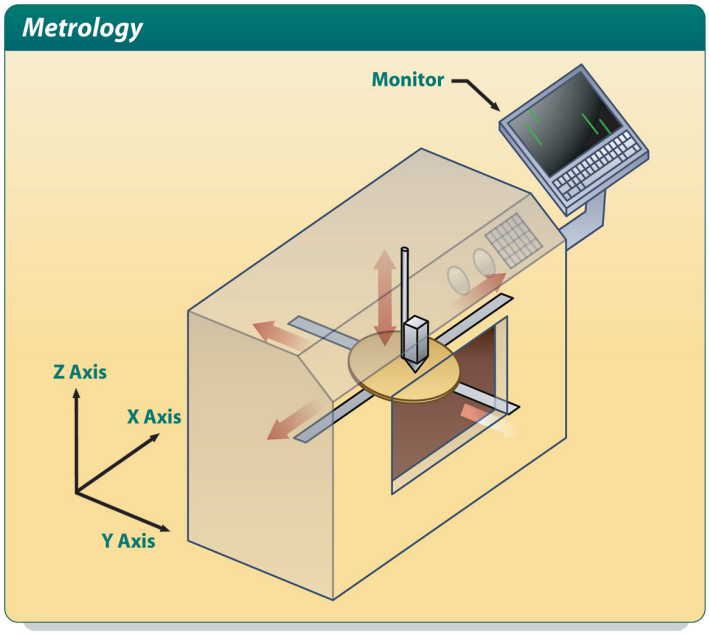

Figure 1. Semiconductor metrology machine

2. Requirements

This section summarizes the requirements for the machine described above:

(1) Two axes (XY) for wafer positioning
(2) One axis (Z) for sensor positioning
(3) Index pulse, forward and reverse limit switches for every axis
(4) Analog input to read film thickness sensor
(5) XYZ position error must not exceed 3 µm for a duration of 1 second before a measurement is taken.
(6) PC program to send XYZ target positions, acquire film thickness data, and present results to the user (not described in detail)

3. Components Selected

This section describes the Galil hardware and software products chosen to implement the machine's control system. Below is a complete bill of materials followed by a description of major components.

Table 1. Bill of Materials for Metrology Instrument Control System

Part Number	Description	Unit Price (U.S.) Qty 1 / qty 100
DMC-1836	Accelera generation PCI Bus 3-axis motion controller	$1895/$935
CABLE-100-1m	High density cable in 1 meter length	$125
ICM-2900-OPTO-FL	Opto-isolated interconnect module	$345/$245
WSDK Servo Tuning Software	Servo Tuning and Analysis Software	$195 (one time)

Controller: DMC-1836 Accelera controller

To achieve the highest machine performance, the DMC-1836 Accelera series motion controller is selected. This controller gives command processing as fast as 40 µsec per command as well as increased servo update frequencies.

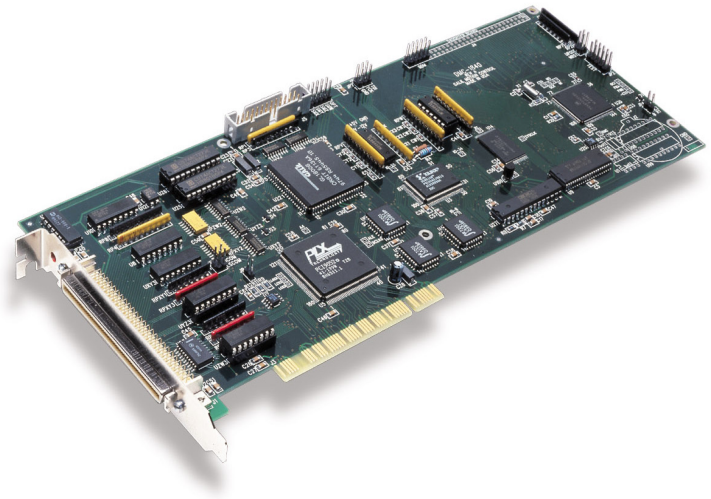

Figure 2. DMC-1836

Interconnect module: ICM-2900-OPTO-FL

The ICM-2900 is an interconnect module that takes the 100-pin high density cable from the DMC-1836 and breaks it out into removable screw terminals.This unit also provides 24 V opto-isolation for all the general purpose outputs of the DMC-1836. Inputs are also opto-isolated on the DMC-1836 card.

Figure 3. ICM-2900-Opto

WSDK software

WSDK (Windows Servo Design Kit) is a development tool which aids in setup and troubleshooting of the Galil motion control system. With the high-resolution linear motors used in this metrology application, the WSDK tuning methods and storage scopes are extremely helpful in selecting the proper PID parameters and motion profiles. In addition to simplifying the tuning and motion programming, WSDK also offers basic setup tools, spreadsheet parameter entry, and system diagnostics.

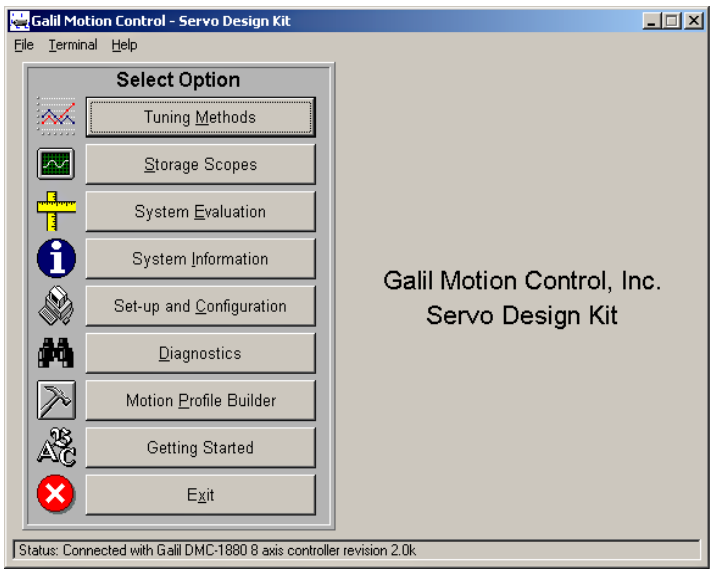

Figure 4. WSDK (Windows Servo Design Kit)

4. Implementation

This section details how the components selected above were used to implement the control system.

Program Organization

The controller application program is structured to "handshake" with a PC program. The application program starts automatically (#AUTO) and then waits for the PC to send the appropriate move data. Each measurement position is sent from the PC to the controller using an array command (QD), which includes distance increments, speeds, and settling windows. An example array download transaction is below and *Table 2* explains each field in the array:

```
QD Data[]
1, 1000, 0, 0, 1000, 1000, 1000, 3\
```

Table 2. Data sent from the PC to the controller for each measurement

Array Element	Variable Name	Description	Example Value	Units
Data[0]	begin	Begin bit. PC sets to 1 to initiate a new measurement.	1	
Data[1]	xpos	X relative distance	1000	counts
Data[2]	ypos	Y relative distance	0	counts
Data[3]	zpos	Z relative distance	0	counts
Data[4]	vspd	XY vector speed	1000	counts/s
Data[5]	zspd	Z speed	1000	counts/s
Data[6]	delay	To consider the machine settled, XY and Z position error must be less than encerr counts for at least delay ms	1000	ms
Data[7]	encerr	To consider the machine settled, XY and Z position error must be less than encerr counts for at least delay ms	3	counts

After the Data[] array is sent, the controller reads the begin bit, performs the specified motion (`#Measure`), verifies settling on all three axes (`#Settle`), samples the analog input, stores the data in an array, and computes the average (`#Analog`). Finally, the controller uploads an array to the PC with the actual XYZ positions and the average value of the analog input. This constitutes one complete measurement cycle and is repeated for each measurement on the wafer.

Linear interpolation mode (LM, LI)

The linear interpolation mode (LM) is used to position the X and Y axes. In this mode, the motion of up to 8 axes is coordinated to maintain a prescribed vector speed (VS), acceleration (VA) and deceleration (VD) along a specified path. Move profiles are specified with LI as incremental distances from the previous point and the end of the path is specified with LE. Motion begins with BGS and AMS waits for profiled motion to complete.

Motion settled routine

A special routine `#Settle` replaces the use of the AM (or MC) command and verifies that all three axes have settled on their target destinations before allowing the program to proceed. After profiled motion has completed, `#Settle` does the following:

(1) waits until the position error (TE) on all three axes is less than `encerr` counts

(2) makes sure that the position error on all three axes is less then `encerr` counts for `delay` milliseconds.

(3) if the error exceeds `encerr` during 2, go back to 1

Throughput Increase

Machine throughput is affected by the average time it takes to perform a single measurement: the less time per measurement, the faster the machine throughput.The new DMC-18x6 Accelera series controller offers a speed advantage by processing each Galil-language command in about 40 microseconds, which is 10 times faster than the DMC-18x0 Optima controller. Some routines (especially math routines such as taking the average of 500 array elements) benefit greatly from this performance increase while other routines such as a point-to-point move take the same amount of time on both controllers (the move time is only a function of the motion parameters: distance, slew speed, acceleration, and deceleration).

Figure 5 shows how the time is spent in each stage of a single measurement for a 1000 count move. The stages are moving, settling, sampling the analog input, and computing the average. The first three stages all take about one second on both controllers since these operations are independent of the controller's command processing speed. The last operation, however, is significantly sped up from 584 ms on the DMC-1830 to 32 ms on the DMC-1836: almost a twenty-fold increase! Thus, the DMC-1836 performs one hundred measurements about a minute faster than the DMC-1830!

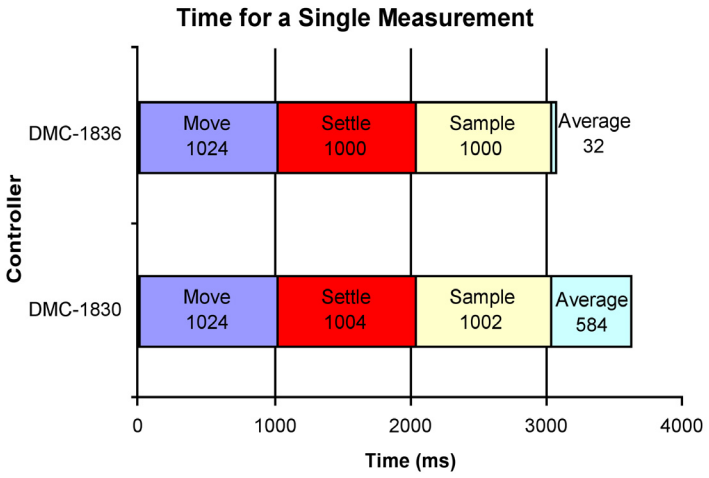

Figure 5. Performance comparison between DMC-1830 and DMC-1836 for QD Data[] 1, 1000, 0, 0, 1000, 1000, 1000, 3\ with AC = DC = VA = VD = 256000

Program listing

The complete program listing used for this article is included here. The comments explain the code in detail:

```
#AUTO                          ;'Start program automatically on power up
  DM Data[8],Read[4]           ;'Dimension two arrays.
                                'Data comes from PC and Read goes to PC
  Data[0] = 0                  ;'Initialize begin bit

  'Axes are homed to a the reverse limit switch, index pulse, and offset
  'Find limit switches
  JG*=-10000                   ;'Jog towards reverse limit switch
  BGXYZ                        ;'Begin motion towards limit
  AMXYZ                        ;'Wait until we hit the limit

  'Find index pulses
  JG*=500                      ;'Move slowly towards the index pulse
  FIXYZ                        ;'Find index
  BGXYZ                        ;'Begin motion towards index
  AMXYZ                        ;'Wait until we hit the index.  Position is set to 0.
  DP -825, -3783, -1581        ;'Define offset where zero is
  PA*=0                        ;'Move to zero
  BG                           ;'Begin motion
  AM                           ;'Wait for motion complete

  #Data_WT                     ;'Wait for PC to set begin bit (Data[0])
    #Loop
      begin = Data[0]          ;'Get begin bit from PC
    JP#Loop, begin=0           ;'Loop while waiting for begin bit

    JS#Measure                 ;'Jump to measurement subroutine

    Data[0] = 0                ;'Reinitialize begin bit
  JP#Data_WT                   ;'Loop back to Data_WT to await next measurement
EN
```

```
'Moves xyz to destination, waits for settling,
'samples analog input, takes average, and sends data to PC
#Measure                       ;'Measurement routine
  'Place array data in variables
  xpos = Data[1]               ;'Relative X distance
  ypos = Data[2]               ;'Relative Y distance
  zpos = Data[3]               ;'Relative Z distance
  vspd = Data[4]               ;'XY vector speed
  zspd = Data[5]               ;'Z speed
  delay = Data[6]              ;'Time XYZ error must be within encerr to proceed
  encerr = Data[7]             ;'Maximum position error allowable during settling

  SPZ=zspd                     ;'Set Z speed
  PRZ=zpos                     ;'Set Z move distance

  VS vspd                      ;'Set XY vector speed
  LMXY                         ;'Enter linear interpolation mode for X and Y
  LI xpos,ypos                 ;'Set XY vector move distance
  LE
  BGSZ                         ;'Begin vector move and Z move
```

(Continued next page)

```
  JS#Settle                    ;'Jump to routine to verify motion settled

  JS#Analog                    ;'Sample analog input and average

  'Send data to PC
  Read[0] = Average            ;'Read analog input 1 for measurement
  Read[1] = _TPX               ;'Read X axis encoder position
  Read[2] = _TPY               ;'Read Y axis encoder position
  Read[3] = _TPZ               ;'Read Z axis encoder position
  QU Read[],0,3                ;'Upload array to PC
EN
```

```
'Waits until all three axes have errors less than encerr for at lease delay ms
#Settle
  AMSZ                         ;'Wait until profile is complete

  'Wait until error is within the Window
  #Wait1
  JP#Wait1, (@ABS[_TEX] > encerr)|(@ABS[_TEY] > encerr)|(@ABS[_TEZ] > encerr)
  InitTime = TIME              ;'store the time at which we entered the window

  #Wait2
    'Start over if we exit the window within the window time
    JP#Wait1, (@ABS[_TEX] > encerr)|(@ABS[_TEY] > encerr)|(@ABS[_TEZ] > encerr)
  JP#Wait2, TIME - InitTime < delay
EN
```

```
'Samples analog input every 2 ms for 1 sec, then computes the average
#Analog
  N=500                        ;'Number of analog input samples
  DM A[N]                      ;'Array of analog input values

  'Sample analog input 1 for 1 sec every 2 ms
  I=0                          ;'Initialize counter
  AT0                          ;'Initialize time reference
  #Loop1
    A[I] = @AN[1]
    I = I + 1
    AT-2
  JP#Loop1, I < N

  'Find the average value
  I = 0                        ;'Initialize counter
  Sum = 0                      ;'Initialize sum
  #Loop2
    Sum = Sum + A[I]
    I = I + 1
  JP#Loop2, I < N

  Average = Sum / N            ;'Store average
EN
```
