A Case Model for Predictive Maintenance

By
Jiawei Li

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Manufacturing
at the Massachusetts Institute of Technology
Sept. ~~25~~, 2007 Feb. 2008
[February 2008]

Author ______________________________
Mechanical Engineering
Sept. 25, 2007

Certified by ______________________________
Duane S. Boning
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by ______________________________
David E. Hardt
Ralph E. and Eloise F. Cross Professor of Mechanical Engineering
Thesis Reader

Accepted by ______________________________
Professor Lallit Anand
Chairman, Committee on Graduate Students
Department of Mechanical Engineering

A Case Model for Predictive Maintenance

By
Jiawei Li

Submitted to the
Department of Mechanical Engineering
September 11, 2007
In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Manufacturing

ABSTRACT

This project is to respond to a need by Varian Semiconductor Equipment Associates, Inc. (VSEA) to help predict failure of ion implanters. Predictive maintenance would help to reduce the unscheduled downtime of ion implanters, whose throughput and uptime is highly important to customers.

Statistical analysis is performed on historical data to extract metadata that can reflect the machine health, and statistical process control (SPC) is applied to detect deviations from normal or in-control behavior. Methods for failure prevention are also investigated. Challenging points in this project are the noise in raw signal data and the difference in data signals of different robots. To address these challenges, we apply signal filtering to extract cycle motions from raw data, and develop different generic as well as specific metadata extraction techniques for different robots.

We test the extraction approaches and results using healthy data of ten machines, and find that the metadata on which we chose to perform SPC is suitable and can serve as a consistent indicator of a machine's health. We further develop an application using Visual Basic based on our study, and provide a user guide on how to generate the analysis reports on new data using our application.

Thesis Supervisor: Duane S. Boning
Title: Professor of Electrical Engineering and Computer Science

Table of Contents

Acknowledgements

First, I would like to thank my advisor, Professor Duane S. Boning, for his supervising my thesis and this project. This project couldn't have been done without his help, instruction and advice.

This work could not have taken place without the support of Varian Semiconductor Equipment Associates Inc (VSEA). I would like to thank Richard Sprenkle for his arrangement of my whole project in VSEA, and his guidance in the model development. I would also thank Paul for his help in PMACs programming, and Jamie McLane for his help in understanding machine performance.

Chapter 1. Introduction and Motivation for Research

This thesis proposes a way to generate models for unusual behavior detection and time-to-failure prediction for robot assemblies in the ion implanters used in the wafer transport subsystem at VSEA. Statistical analysis is performed on historical data to extract metadata that can reflect the machine health, and statistical process control (SPC) is applied to the extracted metadata.

The methods of failure prevention are also investigated. We extract a variety of metadata from real-time data streams recorded on the robots, and apply SPC on both the healthy and unhealthy data. The difference in result shows that the metadata are able to tolerant the fluctuations in healthy data, and at the same time, detect and signal the unhealthy data sharply whenever they appear. Thus, we prove that the metadata and SPC are capable to generate alarms when implanter assemblies go wrong.

We further developed an application using Visual Basic based on our study. The application enables the user to extract real-time data from a machine log, and get analysis reports in CSV format. A user guide is also provided on how to generate the analysis reports on new data using our application.

The rest of the thesis is organized as follows. Chapter 1 gives a background introduction of the transport subsystem in the Varian ion implanter, and discusses the problem explored in this thesis. Chapter 2 is devoted to metadata construction. In Chapter 3, predictive maintenance methods are investigated. Chapter 4 gives a user guide for the application. Conclusions and possible future work are provided in Chapter 5.

1.1 Background Introduction of Transport Subsystem in the Varian Ion Implanter

Varian Semiconductor Equipment Associates, Inc. (VSEA) designs, manufactures, and services semiconductor processing equipment used in the fabrication of integrated circuits. The company is dedicated to satisfying the ion implantation needs of IC manufacturers worldwide, and keeps leading in the industry. This section will give a brief introduction to the wafer transport subsystem, which is a key component of ion implanter.

Introduction to Mechanical Structure of Transport Subsystem

The wafer transport subsystem is responsible for the input, process, and output handling of wafers within the implanter. A cassette of wafers is loaded into the system at atmosphere, the cassette is placed under high vacuum and is mapped, the wafers are moved individually and oriented, and each wafer is placed on the platen for implant.

After implant, the wafer is returned to the wafer cassette. The structure of the wafer transport subsystem is shown in Fig. 1.

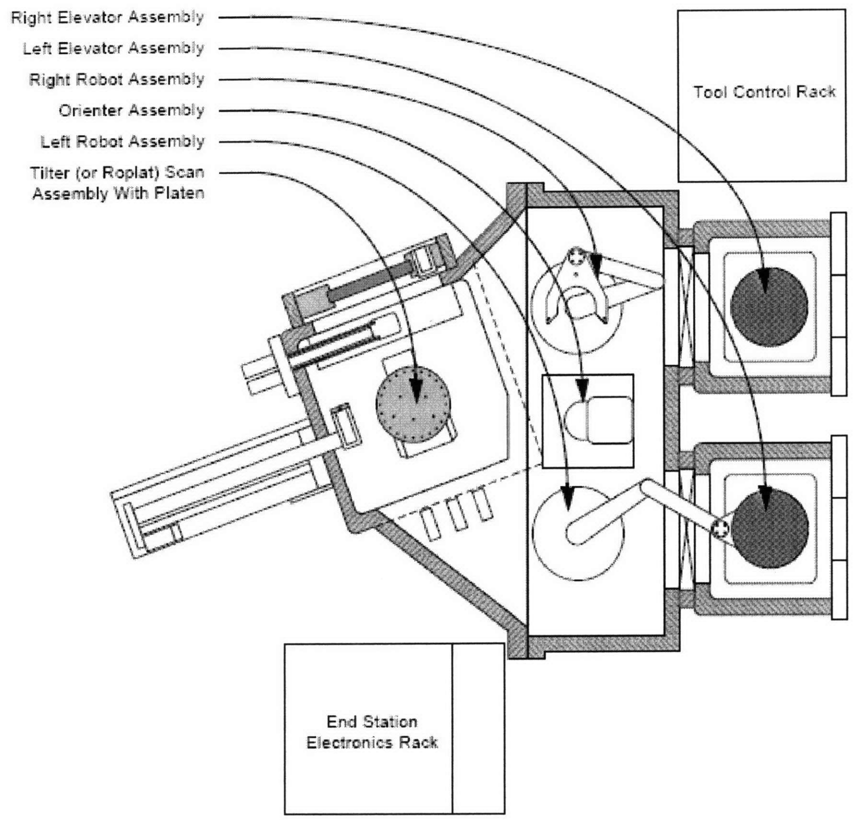


Fig. 1. End station wafer transport subsystem.

Normally, a wafer is first transferred by the left arm from the elevator to the orienter, then by the right arm to the platen which contains an xtilter and ytilter for scan. Finally, the wafer will be taken by the left arm again back to the elevator. Every robot is controlled electrically, and their positions are tracked.

Introduction to System Controller of Transport Subsystem

The machine is controlled by programmable multi-axis controllers (PMACs), which supplied by Delta Tau Data Systems, Inc., to control the motorized mechanical systems used to handle wafers. In this project, we will use the data retrieved from PMACs.

The PMAC is controlled by a proprietary, real-time, cooperative multitasking operating system. A task can be a PMAC motion or PLC(C) program. A PMAC motion program is an interpreted set of ASCII statements that control a set of motors, collectively referred to as a coordinate system. A PLC program is an interpreted set of ASCII statements that perform tasks "asynchronous" to axis motions. A PLC(C) program is a compiled version of a PLC program, the advantage being speed of execution.

Each robot in an ion implanter and its motors can represent and report its status; this data is captured by the PMAC in time. The robots and their key motors are summarized in Table 1.

Table 1. Robot and motor status variables

Robot	Motor
Left Elevator; Right Elevator	DAC Output, Position Error
Left Arm; Right Arm	Angular DAC Output, Angular Position Error, Radial DAC Output, Radial Position Error
Orienter	DAC Output, Position Error
Scan	DAC Output, Position Error
Twist	DAC Output, Implant Position Error, Position Error, Load PositionObserver, Position Degrees
Profiler Motor	DAC Output, Position Error
Platen	Platen Lift DAC Output, Position Error
Xtilt, Ytilt	DAC Output, Implant Position Error, Position Error, Load PositionObserver, Position Degrees

1.2 Statement of Problem

Ion implanters are complex systems. Utilization of these implanters in fabrication facilities places a great emphasis on throughput and system uptime. Unscheduled downtime can have significant adverse effects on the customer. To avoid unscheduled downtime, a means to predict failure is needed such that preventative maintenance can be scheduled during normal scheduled downtimes.

Many subcomponents of the ion implanter have already undergone analysis for predictive failure; however, a particular class of components has not yet been studied. Our purpose in this project is to examine the robot and motors, as summarized above, for fault detection and prediction opportunities.

In order to predict machine failure, we first need to detect the failure. In this project, we seek to retrieve data from PMACs, analyze their relationship with machine failure, and explore ways to build a model to detect, and ultimately predict potential machine failures.

1.3 Brief Literature Review

According to "The Predictive Maintenance Technology Conference and Expo" held in 2007, much of predictive maintenance in industry still focuses on practical experience [14] [15] and mechanical analysis [12] [13]. Theoretical methods like data mining and pattern recognition have been proposed for complex data analysis [4] [5] [11], yet to date is not used widely for practical real time alarm systems.

An effective method in process and equipment diagnosis is statistical process control (SPC) [10]. The use of SPC has been growing fast in recent years and has been successfully applied in various industries, including semiconductor manufacturing [1] [2] [3], chemical manufacturing [7], and manufacturing for medical equipments [8].

SPC has traditionally been applied using a single sample to measure one or more characteristic of the product produced in a piece of equipment, or to a single parameter representative of the operation of the equipment, for each process execution. But in this project, we have many real time measurements during each "run" of the equipment. Thus, proper metadata are required to be selected from that pool to serve as the indicators for machine health, using SPC detection and trend analysis approaches.

Chapter 2. Metadata Construction

The construction of proper metadata which have good discrimination between healthy and unhealthy data is the central part of this project. An ideal metadata sequence should be able to tolerant the fluctuation in healthy data, and at the same time, capture the unhealthy data sharply whenever they appear.

Since we do not have sufficient unhealthy data, we focus on looking for highly repeatable patterns for machines when healthy, and propose a model to compare the metadata performance on unhealthy data. Two robots, the orienter and tilter, are selected to explore and demonstrate the extraction of metadata from real time data streams.

We use a three step approach to find potential metadata streams. They are: to overlay cycles, to find median example of a cycle, and to find other generic as well as specific metadata in the cycles. Each of these steps is explained in more detail in the following sections.

2.1 Overlay cycles for Orienter

The purpose behind overlaying many cycles (repeated wafer moves or operations) is to see whether it is possible to find repeatable patterns of the cycles. Here, we define a cycle as the shortest repeatable pattern detectable in a variable data stream. The time and length of a cycle only depends on the features of that variable data stream, and is independent from other variables. Thus, the cycles from two different variables (position and DAC) do not necessarily line up in time or length. That makes sense, because it is possible to have two cycles of DAC to drive two position changes which can be recognized as one cycle of movements.

We take the raw data of the orienter, and plot its position and DAC (detected motor current required to drive the robot to the desired position) signal in Fig. 2. The raw data is captured in every recordable machine cycle, and is plotted in time series. Each discrete time point corresponds to 10 milliseconds.

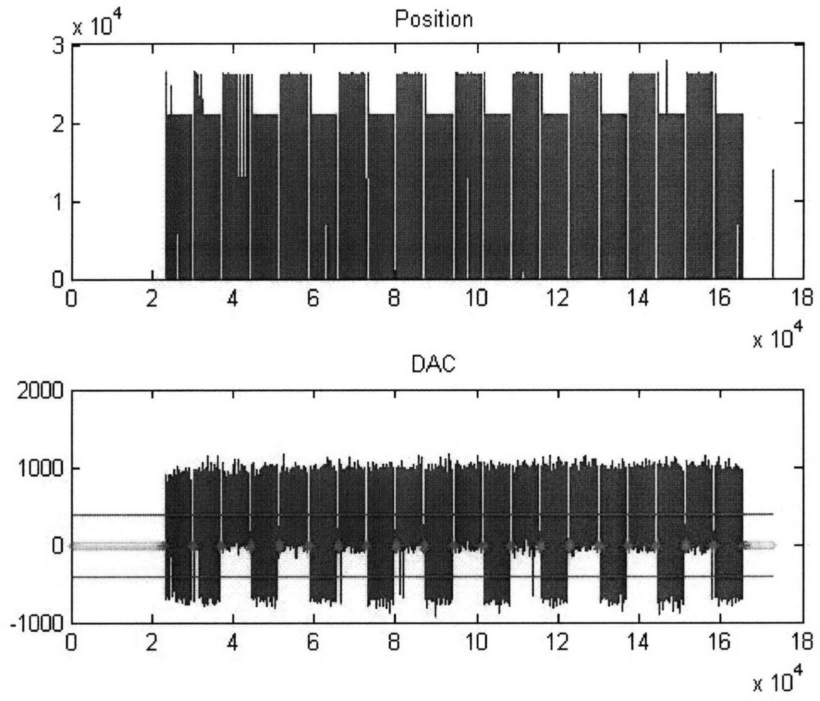


Fig. 2. Overview of Position and DAC

We can see from Fig. 2 that there are two "families" of motions in both. We define Family 1 data as whose DAC only have positive values, and define Family 2 data as those whose DAC have both positive and negative values. Thus, we decide to overlay cycles in each family as the next step, to explore the possibility of getting independent repeatable patterns for each family.

2.1.1 Cycle extraction method

In this section, we will explain how we capture cycles, and explore the possibility to find the repeatable pattern for both position and DAC signals of the orienter.

2.1.1.1 Filtering

We first perform data filtering on the raw data to make the cycle smoother and make the cycle that we want to extract stand out. To be specific, we set a threshold (red line in Fig. 2) that can set the magnitude of most of the data points to zero, but at the same time keep the peaks, which is the set of features that defines the very cycle that we want. The selection of threshold value is set at 400 according our observation. We also set the value to zero if there are more than 5 continuous points at the same value. The result of these two filtering steps is shown in a zoomed-in set of data in Fig. 3, where the data before and after filtering for the position data is shown.

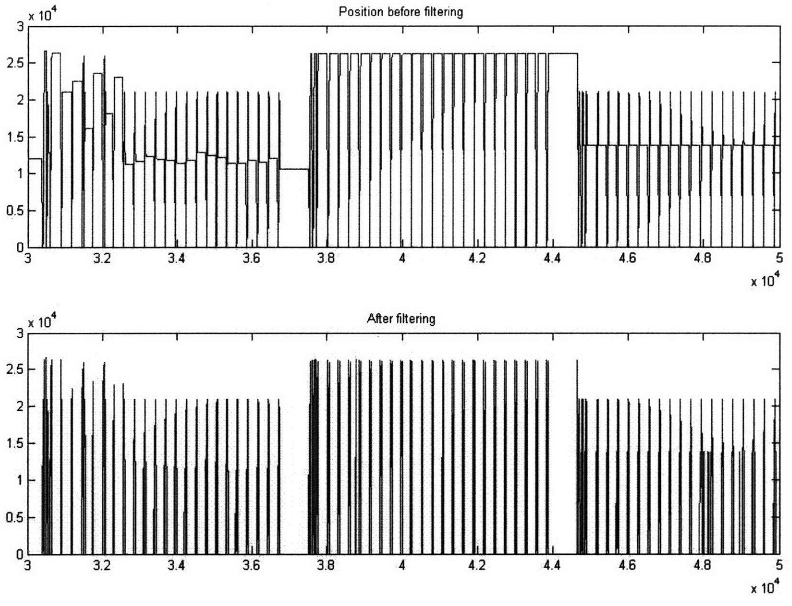
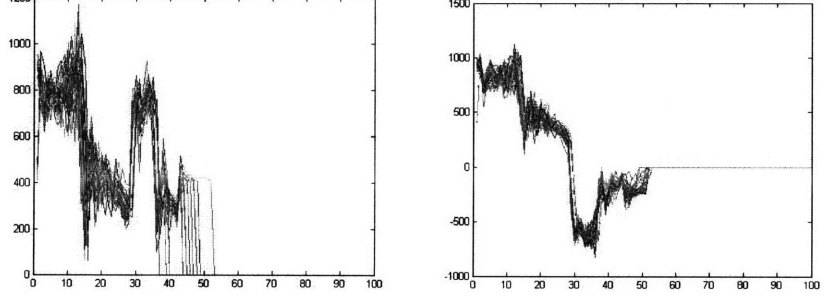


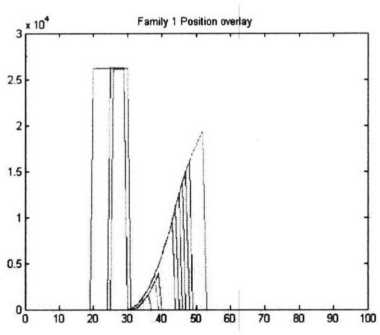
Fig. 3. Filtering on Raw Data

Then, we locate a cycle by identifying the zero data points at both sides of the cycle (the common pattern of a cycle) and then extract the non-zero part of the data which lies in the middle of the sequence.

2.1.2 Cycle overlay

Thirty cycles in one family of motion for orienter DAC signal and position signal are plotted and shown in Fig. 4. Cycles are captured according to their DAC value. That is, the first point after one hundred zeros have been counted is the start point of each cycle, and the last point before one hundred zeros have been counted is defined the end point of each cycle. Position cycles are captured according to the DAC period, with thirty points before and after.





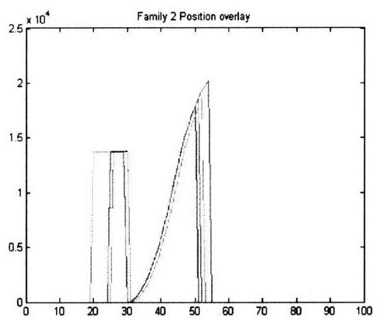


Fig. 4. Overlay of DAC and position

From the above result, we can see that cycles in the orienter DAC and position signals are quite similar and repeatable, with some difference in cycle time. That convinces us that it is worthwhile to investigate repeatable patterns for orienter DAC and position signals in Section 2.2.

2.2 Find median example of a cycle for the orienter

Based on the study in the overlay plots of cycles for the orienter position and DAC, we are convinced that it is possible to find repeatable patterns of the cycles. Thus, we try to find those repeatable patterns in this section by looking for a median or representative example of a cycle.

2.2.1 Mean and media cycle extraction

First, a "usual behavior" trace or data signal is created using the mean performance, on a discrete time point by time point basis, of 30 cycles. The total squared deviation of any given data cycle from this "usual behavior" template can then be calculated as the sum of squared differences, between each cycle and the "usual behavior". Then, by choosing the cycle which has the minimum deviation, we can identify a "median example" or template cycle. In subsequent conparisons of cycle data, we can make the comparisons either against the mean or "average" cycle data or against the "median" cycle. Fig. 5 is an example on DAC Family 1 data.

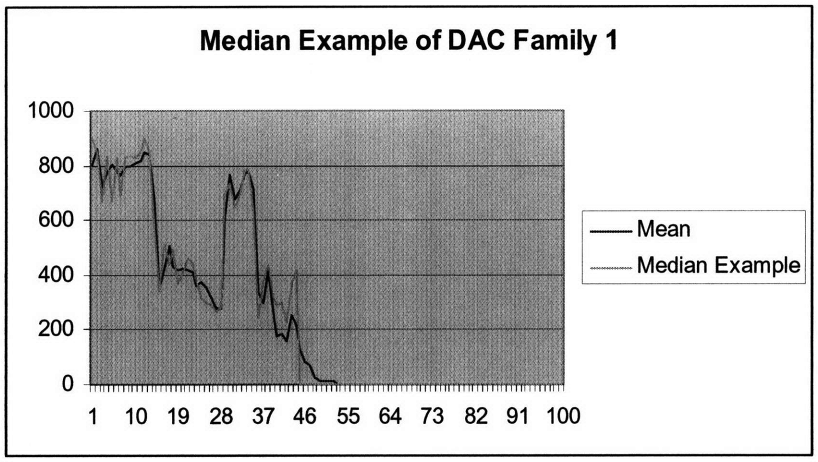


Fig. 5. Mean and Median extracted example cycles.

We can see from Fig. 5 that the median examples are close to usual behavior, which means that it is reasonable to regard the "median example" as the repeatable pattern.

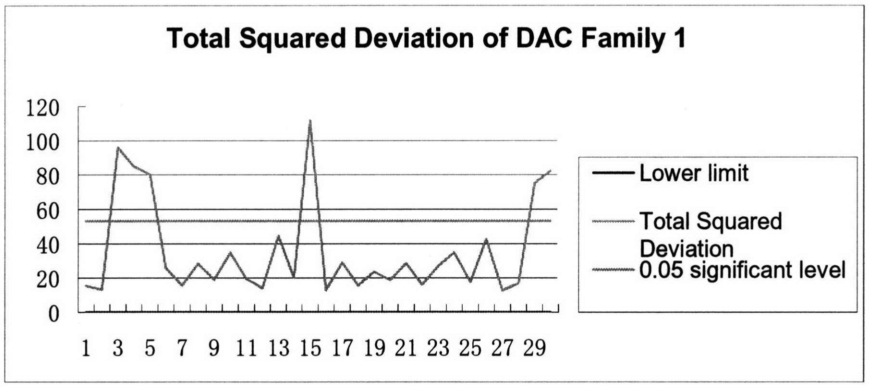
2.2.2 Deviation as metadata and SPC over it

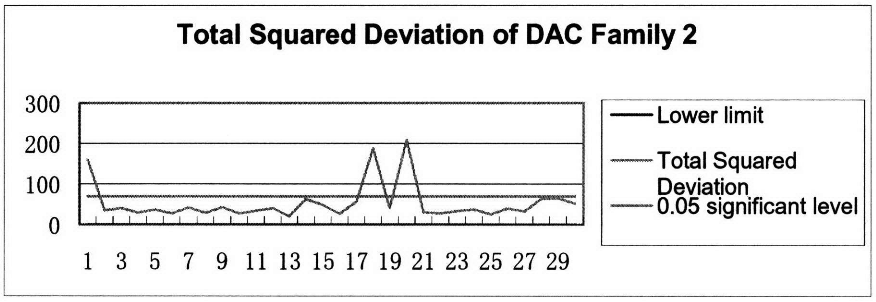
We still have a whole data sequence for each cycle or "run" of the equipment. To do SPC, we need to condense this down to a single representative value that we can plot on an SPC chart to indicate, run by run, the status of equipment health. We will apply SPC on several metadata. If the metadata for all healthy data are nearly all within the 3-sigma control lines, it suggests that these metadata may be good to indicate machine health.

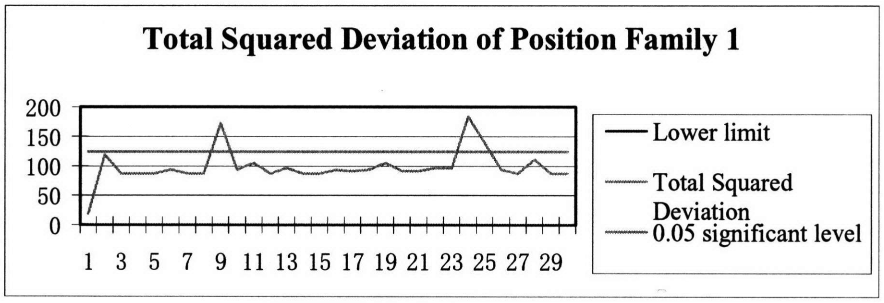
The first metadata we try is the "total squared deviation" defined as the sum of squared deviations of the cycle from the extracted median cycle. The SPC figures of the two families of DAC and position data are shown in Fig6.

Because it is Gaussian for the deviation of each data point within cycle, so their total squared deviation is expected to follow a Chi square distribution. That is: $x_i \sim N(\mu, \sigma^2)$,

then $y_i = \sum_{i=1}^{n} (x_i - \overline{x})^2 \sim X_{n-1}^2$. We take 0.05 significant level as the control limit for that Chi square distribution







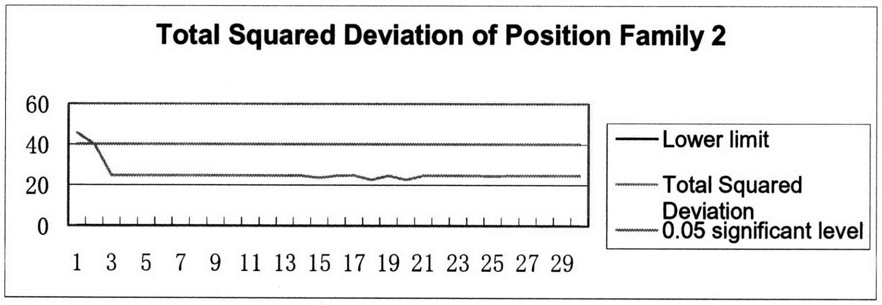


Fig. 6. Total Squared Deviations.

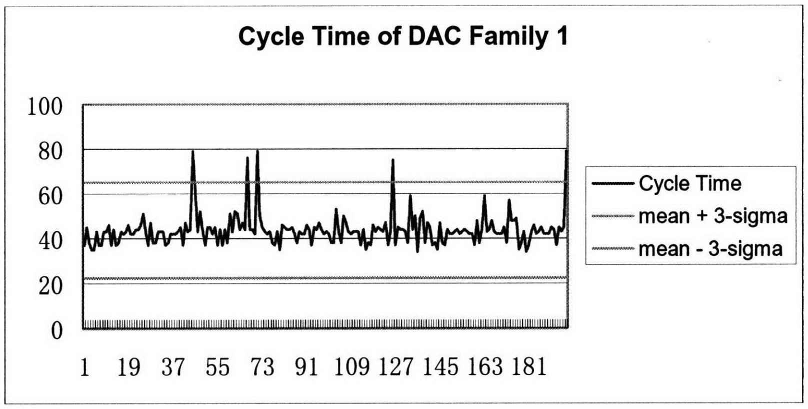
From the above figures, we can see that most of the data for each cycle fall within the control limits. Thus, the total squared deviation is a good candidate type of metadata for detection of unusual or unhealthy equipment operation.

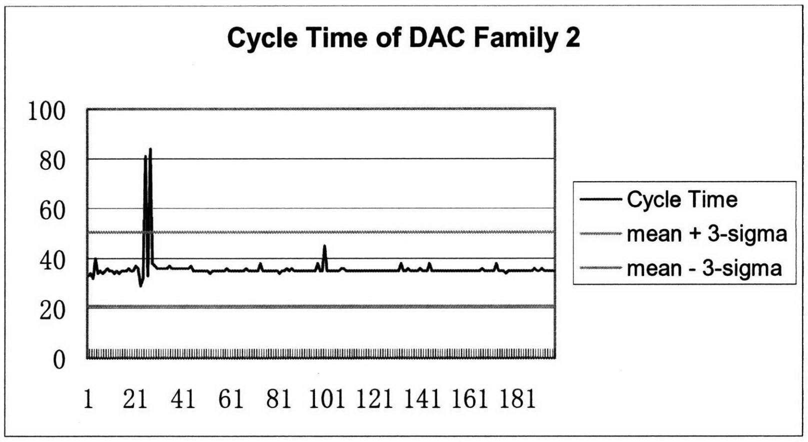
2.3 Other generic and specific metadata in orienter cycles

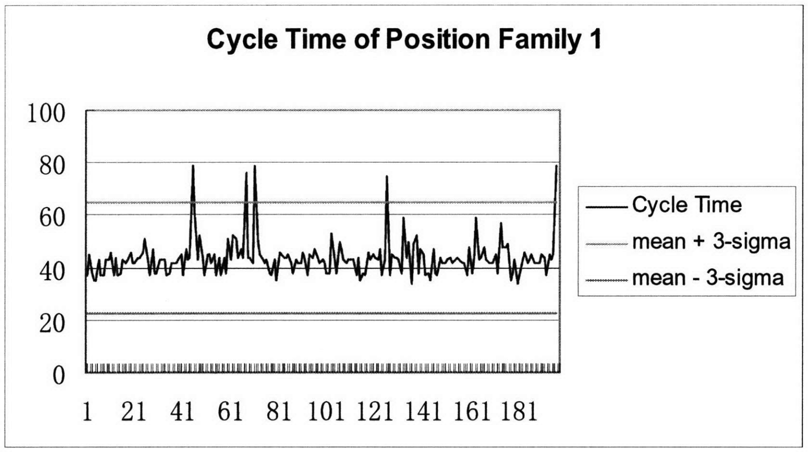
In this section, we further explore the candidate metadata extracted in both "generic" and "specific" ways. We will first try generic metadata such as cycle time, max value, and std/mean. They are "generic" features in all signal analysis, meaning that these extractions can generally be performed on all types of cycle data without attention to specific features of the data. Then, we will explore specific metadata based on particular repeatable features within DAC or position data.

2.3.1 Cycle Time (Generic Metadata)

Cycle time is simply the length of the cycle data extracted using the very method specified in Section 2.1. Fig. 7 is a plot of the cycle time for each of 190 cycles. It is a generic metadata variable, which means that it is not limited by special features of the orienter data.


Cycle Time of DAC Family 1
100
80
60
40
20
0
1 19 37 55 73 91 109 127 145 163 181
Cycle Time
mean + 3-sigma
mean - 3-sigma


Cycle Time of DAC Family 2
100
80
60
40
20
0
1 21 41 61 81 101 121 141 161 181
Cycle Time
mean + 3-sigma
mean - 3-sigma



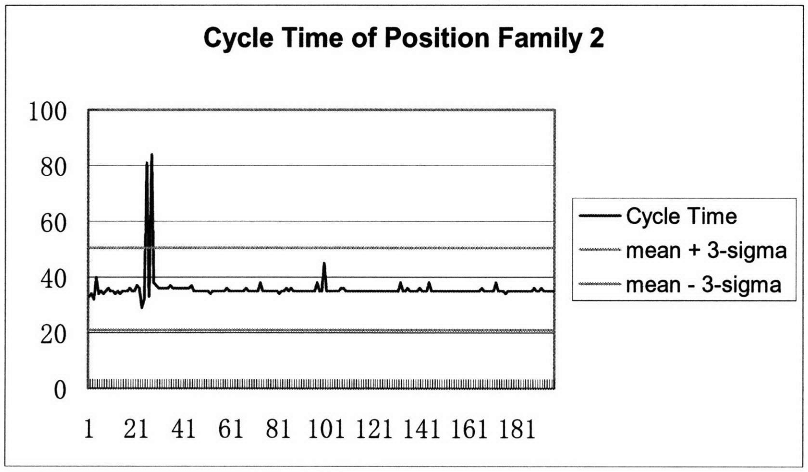


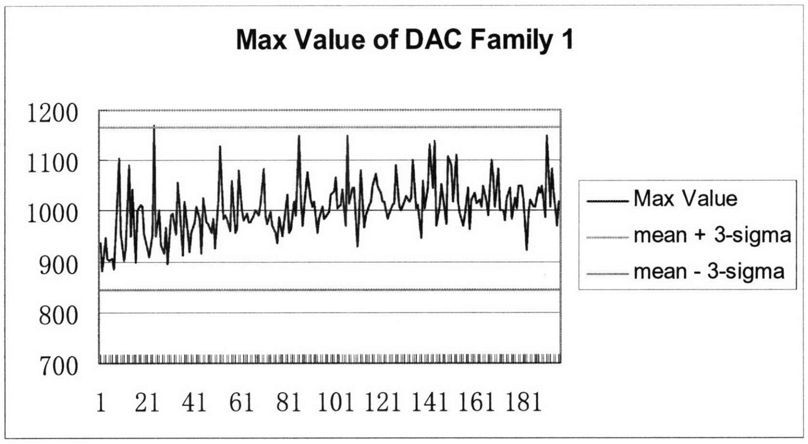
Fig. 7. Cycle Time

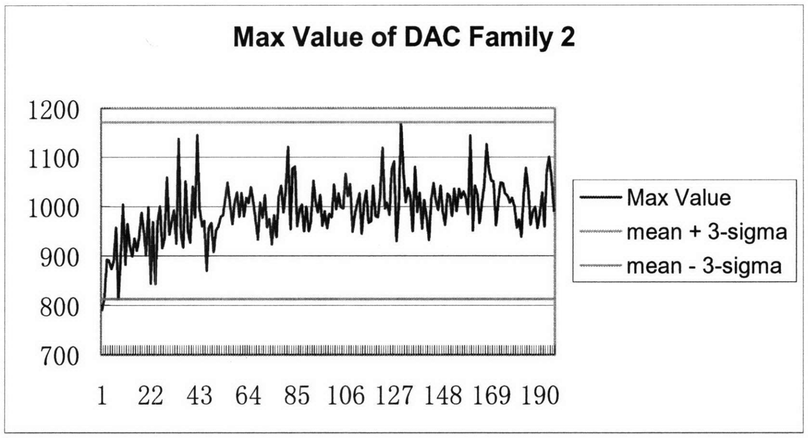
In Fig. 7, cycle time of DAC and position Family 1 have five points out of the three sigma scope; and cycle time of Family 2 has several continuous points far away from the three sigma range.

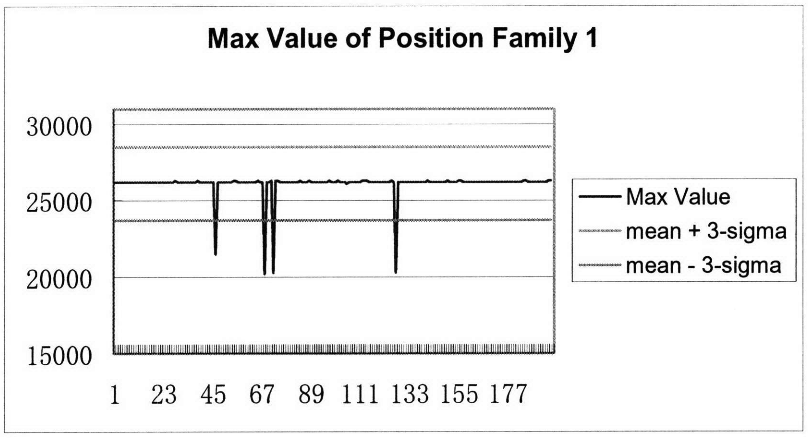
That convinces us that there may be a challenge or problem in attempts to use cycle time as a health indicator. If we use it as the indicator, the data assumed to be "healthy" can give substantial numbers of "false alarms." In future work, some investigation of these cycles may be beneficial; perhaps these points actually indicate some kind of unusual or problem behavior.

2.3.2 Max Value (Generic Metadata)

Max value is defined as the maximum value of each cycle. These are plotted for 190 cycles of position and DAC in Fig. 8. The result tells us that max value can serve as a good metadata for DAC, but not for position, because of the potential for large numbers of false alarms in the position max value data.







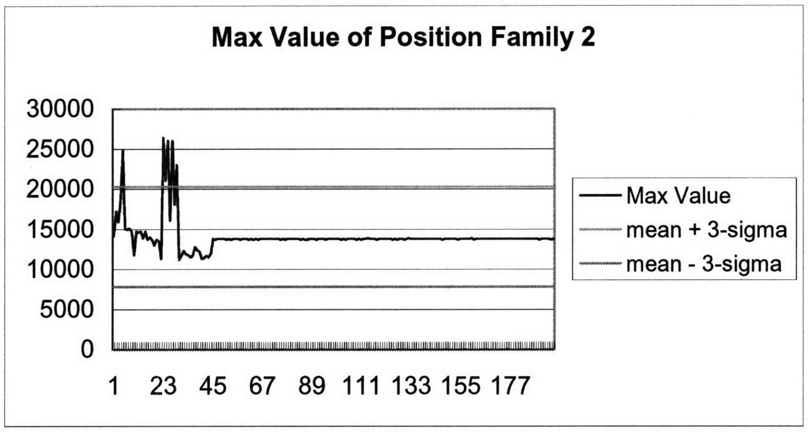


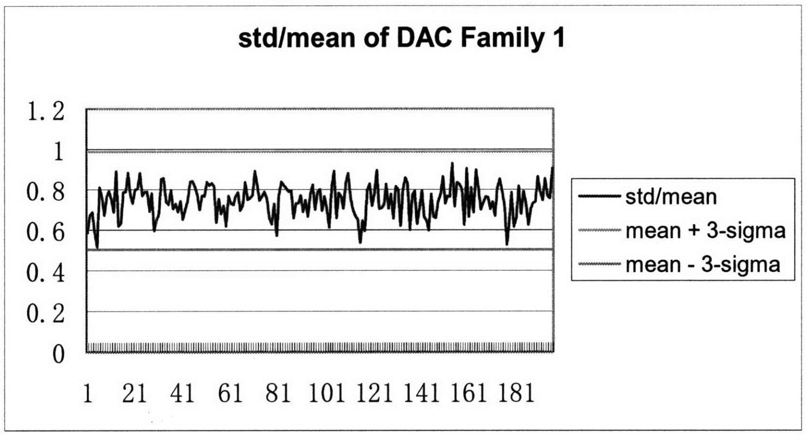
Fig. 8. Max Value

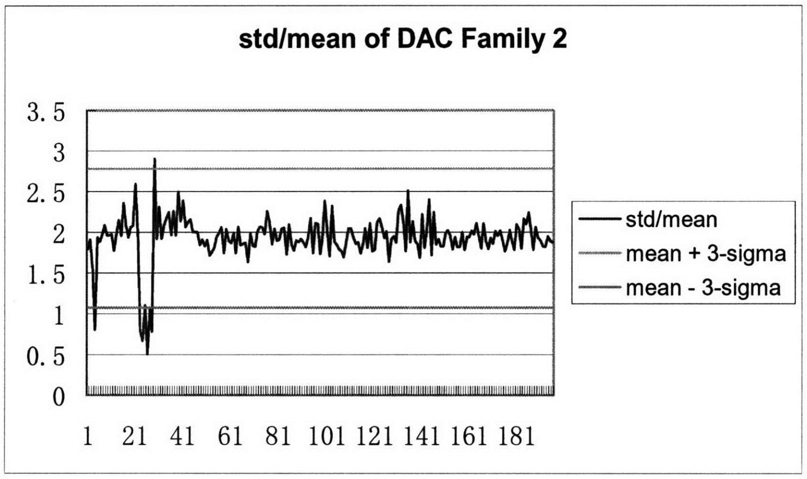
These plots are also very interesting from the perspective of preventative problem detection. In particular, the plots for DAC max value show a trend in addition to "random" fluctuations. If there is a trend in time, that indicates that something may be changing over long periods of time in the equipment. One could project or extrapolate forward in time, for example, potentially estimating a time in the future when this indicator might exceed some specification limit. Thus we could suggest how many more tool operations might be allowable before a future preventative maintenance operation is required.

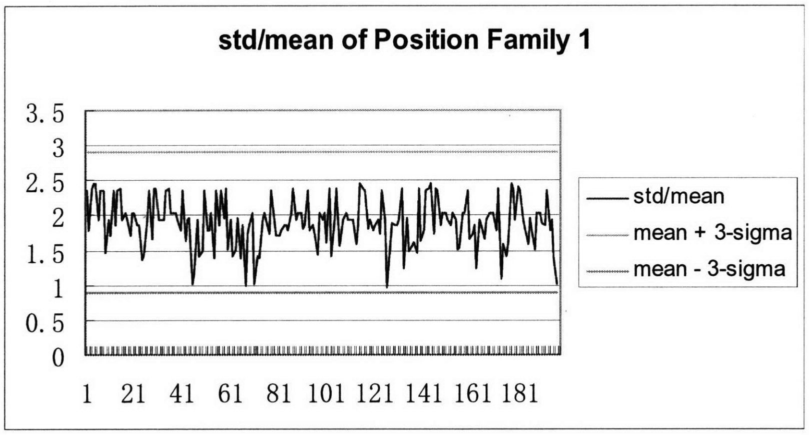
2.3.3 std/mean (Generic Metadata)

Std/mean is defined as the standard deviation of the data in a cycle divided by their mean. For cycle i, $\overline{y_i}$ is the mean of cycle i. Then

$\overline{y_i} = \sum_{j=1}^{T_i} y_{ij} / T_i$, where *i* is the *i*th cycle, T_i is the cycle time for cycle *i*, and *j* indicates the jth data point within cycle i. The standard deviation $\overline{s_{yi}} = (\frac{1}{T_i - 1}\sum_{j=1}^{T_i}(y_{ij} - \overline{y_i})^2)^{1/2}$. So, std/mean is $s_{yi} / \overline{y_i}$. These are plotted for multiple cycles of orienter DAC and position data in Fig. 9. We can see from the result that std/mean is a good metadata, especially for Family 1 data.







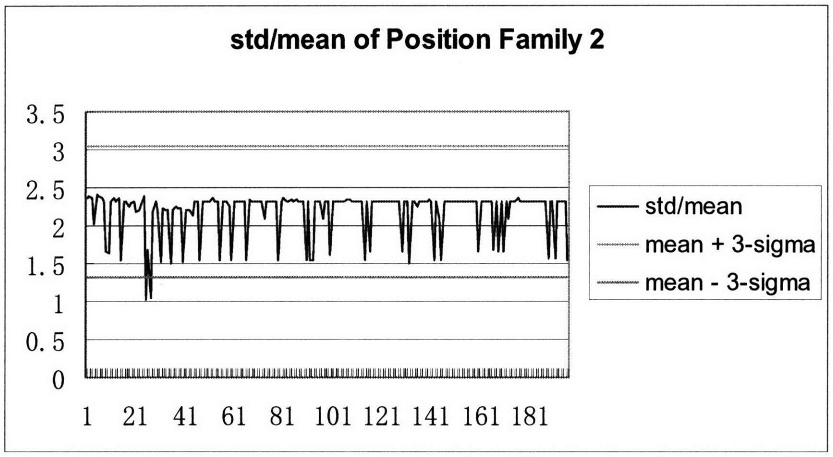


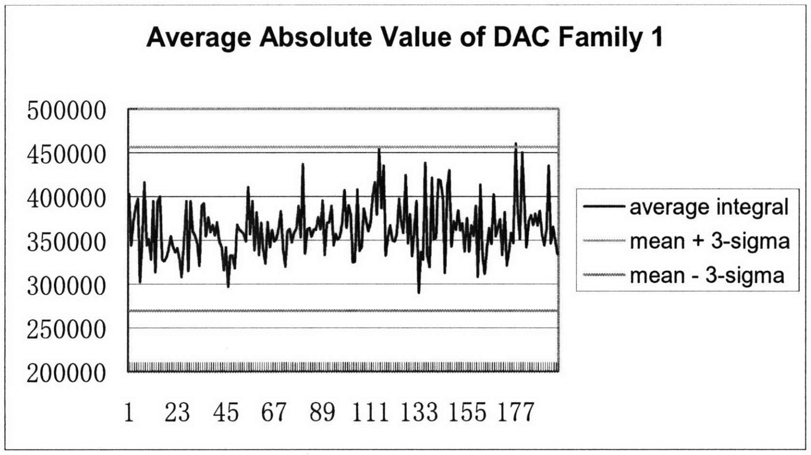
Fig. 9. std/mean for DAC and position

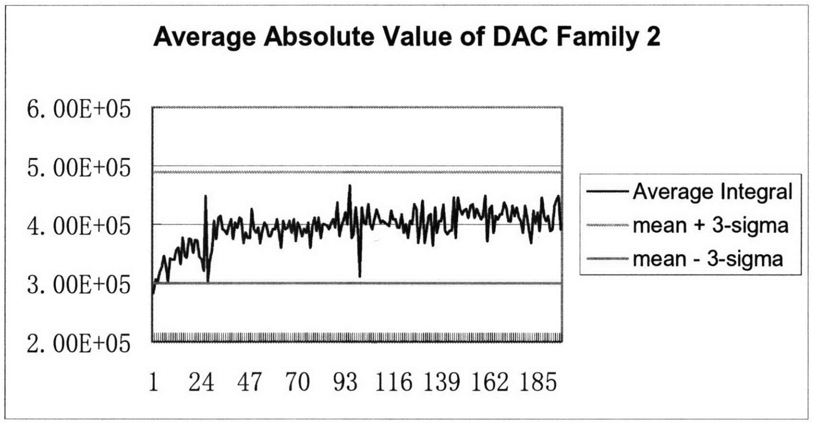
2.3.4 Average absolute value of each cycle (Generic Metadata)

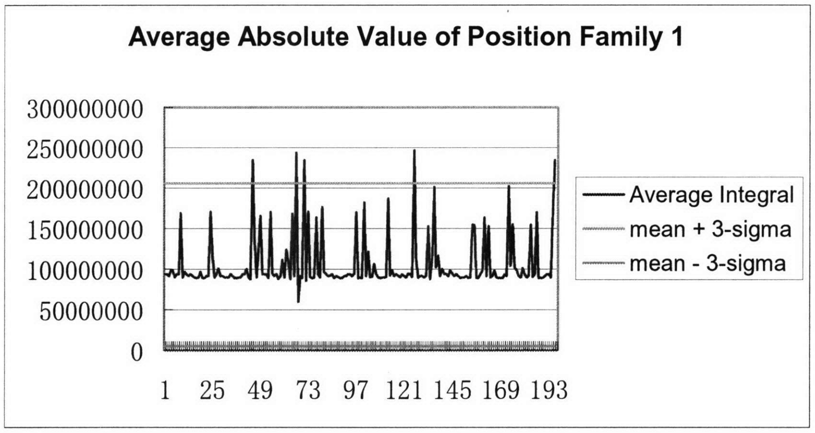
The average absolute value is defined as the normalized average of the absolute value of all data within a given cycle. It is defined as total deviation over zero of that cycle. Because of the difference in cycle time as shown in 2.3.1, we normalize by the corresponding cycle time, to form the average absolute value.

For each cycle *i*, the average absolute value is counted as $a_i = \frac{1}{T_i}\sum_{j=1}^{T_i} | y_{ij} |$ where *i* is the *i*th cycle, T_i is the cycle time for cycle *i*, and *j* indicates the *j*th data point within cycle *i*. The resulting extractions are shown in Fig. 10. While the absolute value will not be

Gaussian distributed, we show the three-sigma lines as a guide for an assumed Gaussian distribution.







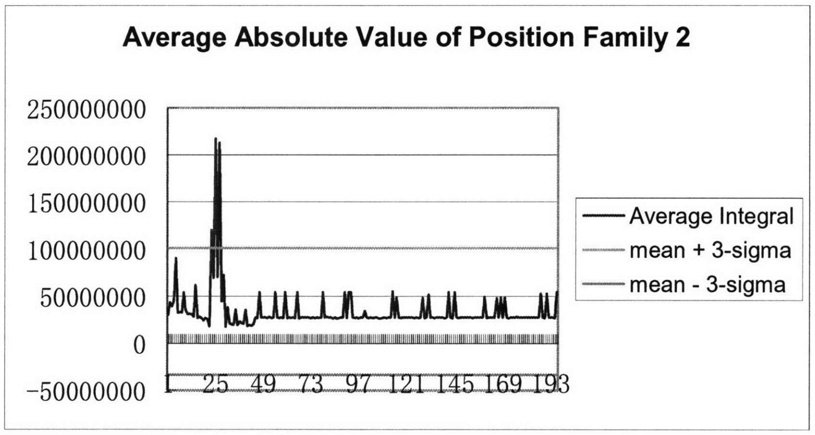


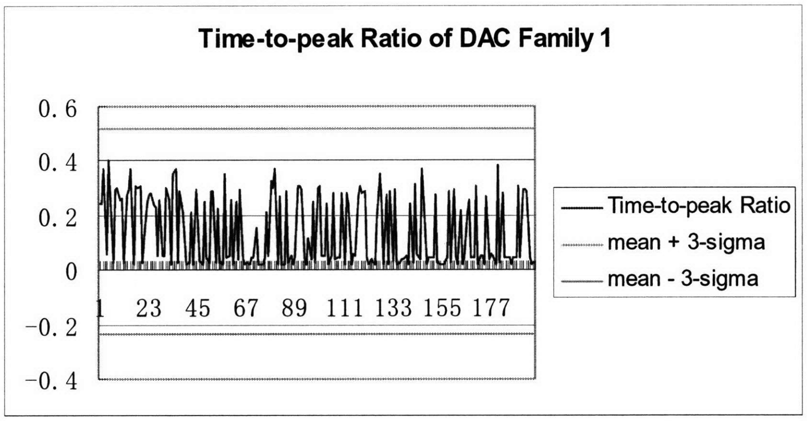
Fig. 10. Average absolute value for orienter DAC and position data, for many data cycles

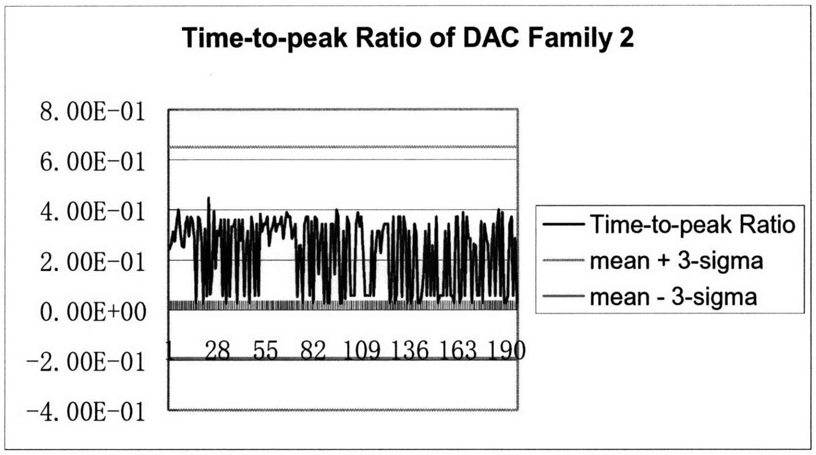
The result tells us that average absolute value works well on DAC data. It does not work on position data because, referring to the overlay plot (Fig. 4), the position integral is different from cycle to cycle.

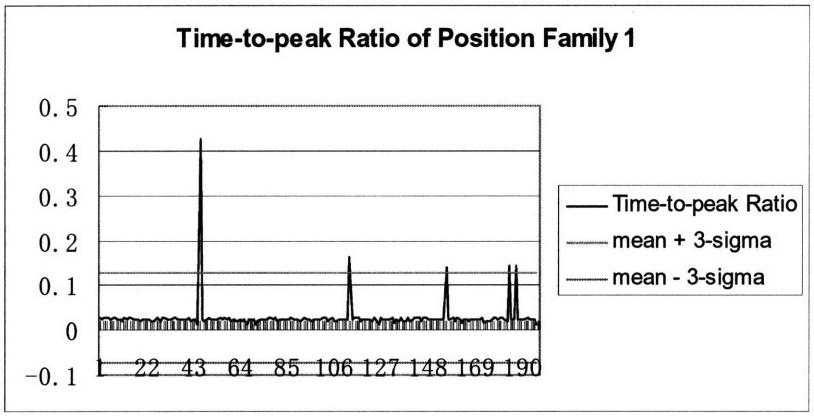
Besides that, the average absolute value of DAC Family 2 is very interesting from the perspective of preventative problem detection. In particular, that plot shows a trend in addition to "random" fluctuations. If there is a trend in time, that indicates that something may be changing over long periods of time in the equipment, again providing an opportunity for preventative maintenance recommendations.

2.3.5 Time-to-peak Ratio (Special Metadata)

Time-to-peak ratio is defined as the time between the first data point in a cycle and the maximum value point, divided by the width of that cycle. The reason we divide by cycle time is to offset the effect of variation in the cycle length. As shown in Section 2.3.1, we find that the cycle time is not a constant value. Time-to-peak ratio is a special metadata variable because it is extracted according to the special features of the orienter data, and involves the detection of one or more specific features in time within each cycle. Fig. 11 plots the extracted time-to-Peak ratio for multiple orienter DAC and position data.







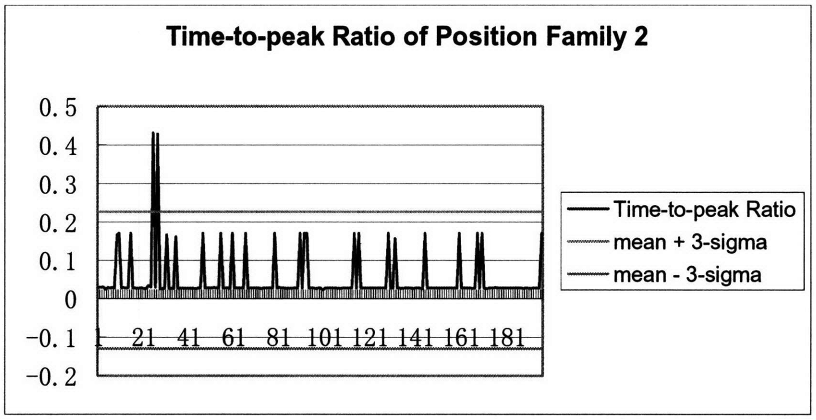
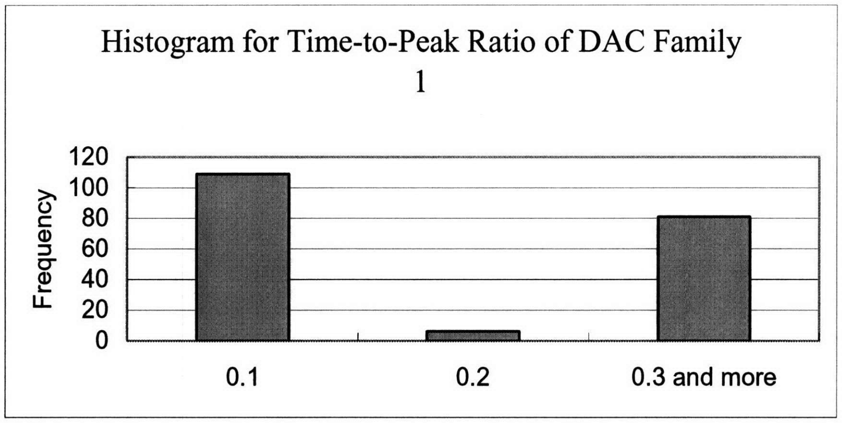
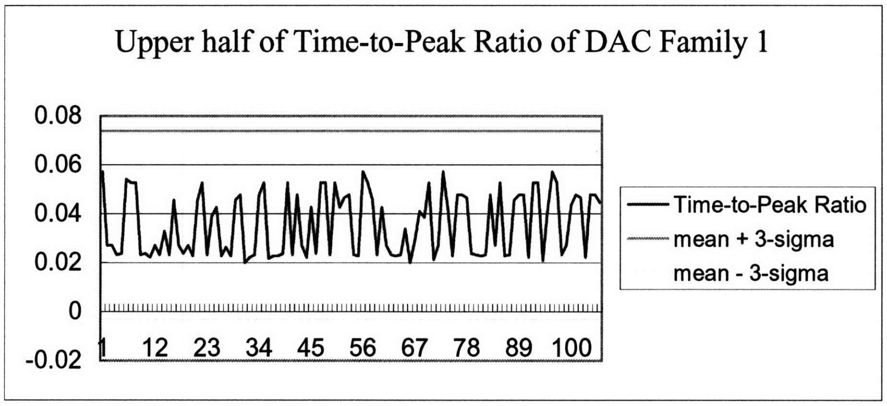


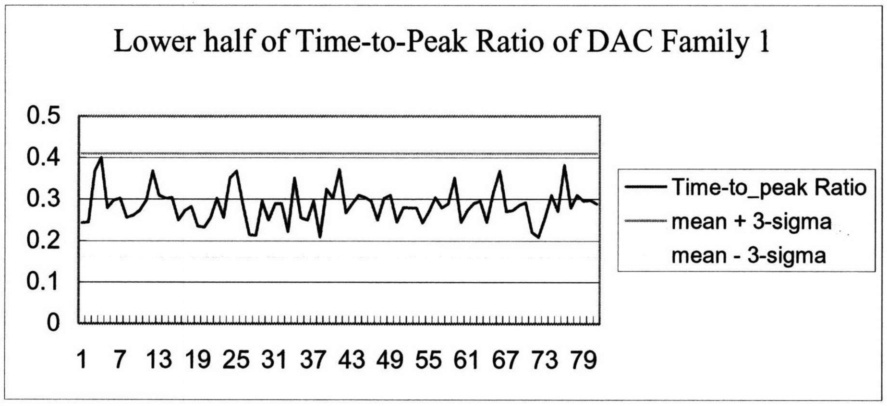
Fig. 11. Time-to-Peak Ratio

The result tells us that time-to-peak ratio is a good metadata variable for the DAC, but is not a good metadata variable for position. That makes sense, because position has a very sharp peak as we can see from the overlay figure (Fig. 4). A small amount of change in the rising time may change a lot in the ratio over cycle time.

From Fig. 11 DAC plots, we can see that the values are gathered around either the lower or upper limits of the data. Thus, we plot the frequency of the data in histogram to investigate that feature in Fig. 12. The bimodal distribution indicates the inappropriateness of a single set of control limits. That triggers our idea that one could separate points into one of two different distributions, and could then set up separate control charts for the corresponding behavior from each of the two cases. We apply SPC limits separately on the upper and lower half of the data in Fig. 12. We also overlay two cycles in Fig. 12. One is from upper half, and the other is from lower half, of the observed bimodal distribution.


Histogram for Time-to-Peak Ratio of DAC Family 1
Frequency
120
100
80
60
40
20
0
0.1
0.2
0.3 and more


Upper half of Time-to-Peak Ratio of DAC Family 1
0.08
0.06
0.04
0.02
0
-0.02
1 12 23 34 45 56 67 78 89 100
Time-to-Peak Ratio
mean + 3-sigma
mean - 3-sigma


Lower half of Time-to-Peak Ratio of DAC Family 1
0.5
0.4
0.3
0.2
0.1
0
1 7 13 19 25 31 37 43 49 55 61 67 73 79
Time-to_peak Ratio
mean + 3-sigma
mean - 3-sigma

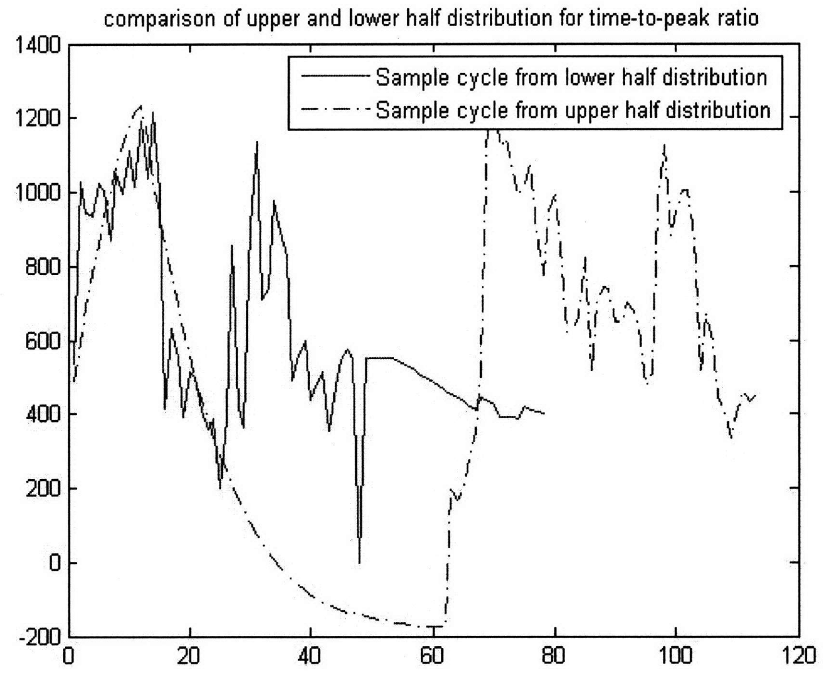


Fig. 12. Histogram and separate parts of Time-to-Peak Ratio of DAC Family 1

2.3.6 Peak-to-peak Ratio (Special Metadata)

From the overlay plot (Fig. 4), we can see that there are two peaks in DAC and position data. Thus, we define peak-to-peak ratio as the time between the main and minor peak, divided by its corresponding cycle time.

Because there are many fluctuations in the peaks as is shown in Fig. 13, it is difficult to capture the exact position of the "peaks". Thus, we take the time between ends of peaks as the peak-to-peak time. As shown in Fig. 13, we apply a threshold, which is 500, on the cycle, then capture the first and second fall across the threshold value. The time between the two falls is taken as the peak-to-peak time.

We can tell from Fig. 14 that Peak-to-Peak ratio works well on Family 1 data, with most of the points within control limits.

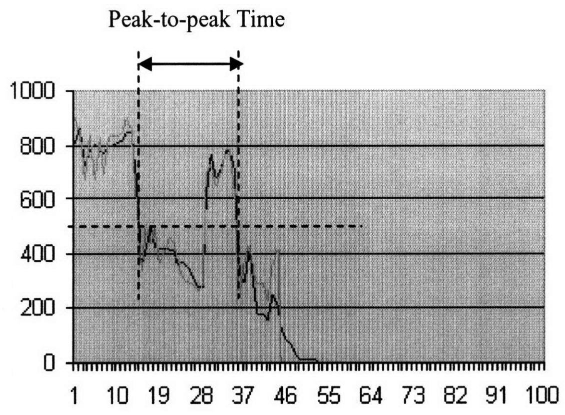
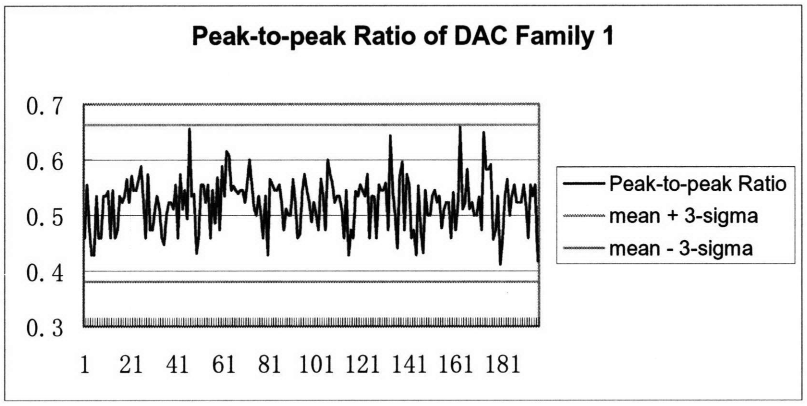
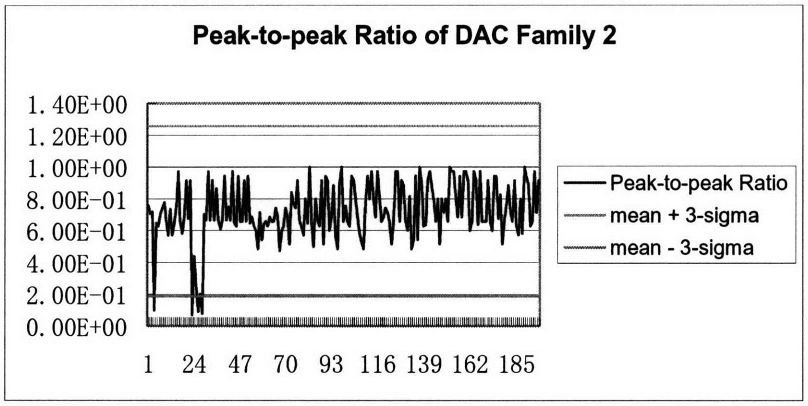
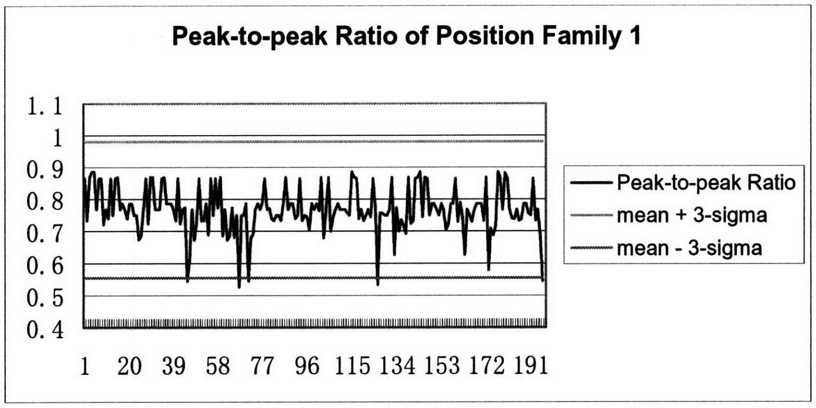


Fig. 13. The method to extract the peak-to-peak time.







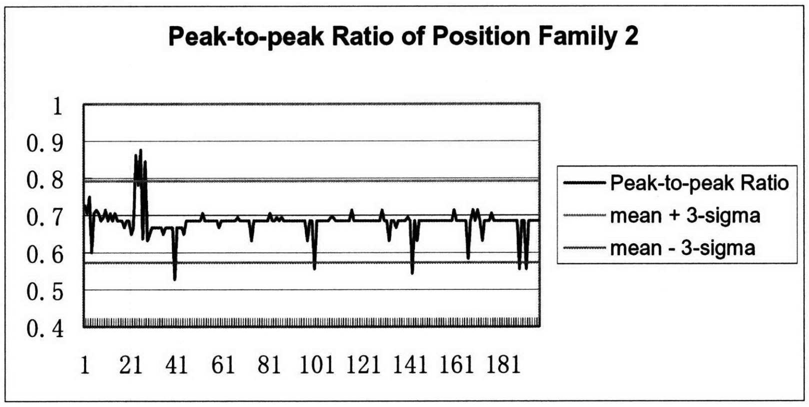


Fig. 14. Peak-to-Peak Ratio

2.4 Test Metadata on 10 Machines for Orienter

An ideal metadata should be able to tolerant the fluctuation in healthy data, both across many cycles in a given machine, and across different machines. In this section, we will test our candidate extracted metadata on 10 machines to select qualified metadata.

According to the analysis in Section 3.3, we choose the following metadata to test on 10 machines: total square deviation, std/mean, peak-to-peak ratio. A box plot is applied to perform the test. The box plot produces a box and whisker plot for each machine data. The box has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. Outliers are data with values beyond the ends of the whiskers.

2.4.1 Total Deviation

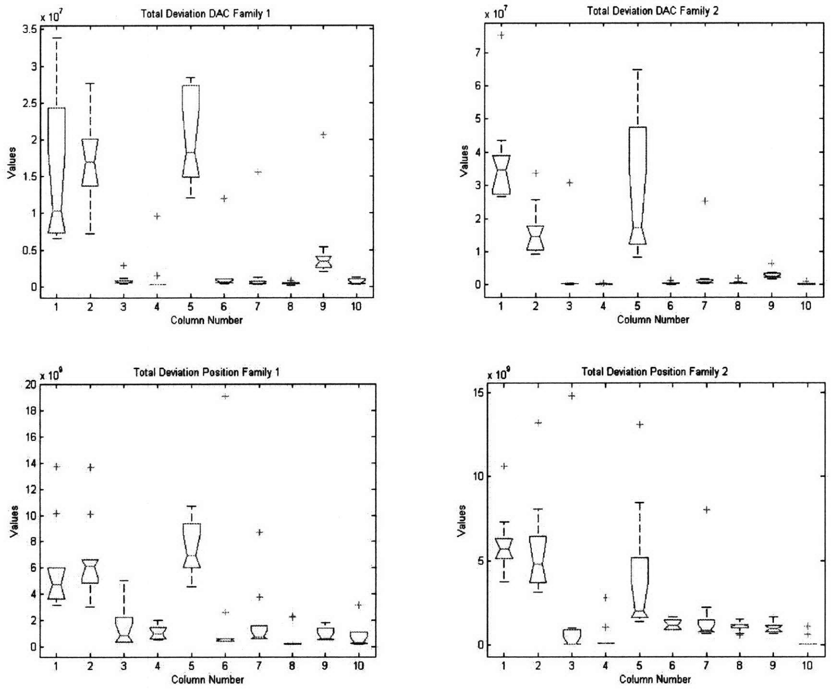
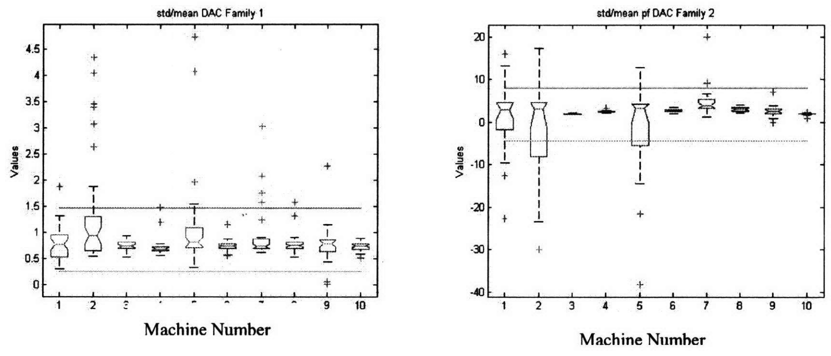


Fig. 14. Boxplot of total squared deviation, for orienter data on 10 machines

The above result tells us that whether that metadata can be applied to differentiate healthy and unhealthy data depends on machines. Some machines like Machine 1, 2, and 5 stay far away from the range of the other machines. This suggests that either those machines are not completely healthy or normal, or that a machine-by-machine calibration of the "healthy" range of metadata may be required.

2.4.2 std/mean



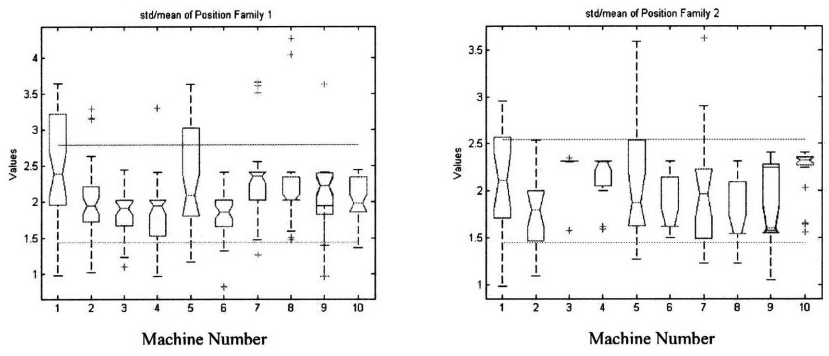


Fig. 15. Boxplot of std/mean

The results tell that std/mean is a fairly good metadata. Most of the boxes are within the 3-sigma control limits calculated using the means across all 10 machines. Yet there are outliers staying far away from the limits. This indicates that in most cases, machine data are within the normal behavior, but there are cases once in a while where that the data will be out of the scope, yet those outliers do not suggest that the machine is out of the healthy status.

2.4.3 Peak-to-Peak Ratio

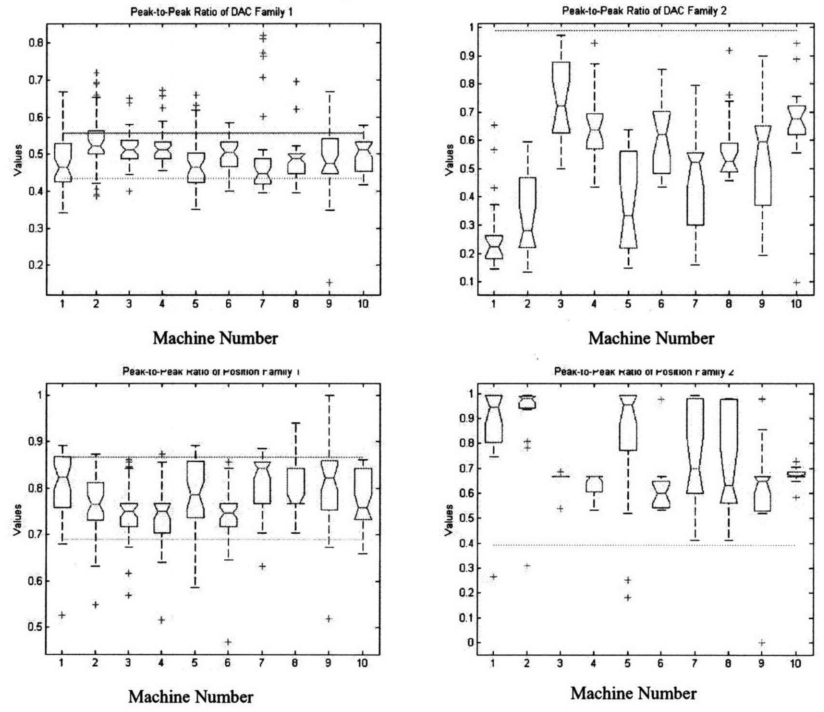


Fig. 16. Boxplot of Peak-to-Peak Ratio

The above results tell us that Peak-to-Peak Ratio can also serve as a metadata variable. Most of the boxes stay within the control limits. However, data vary from machine to machine in Family 2. This indicates that it may not be a good idea to treat every machine as the same thing in Family 2. Having SPC limits for each machine may be a good solution.

2.5 Overlay Cycles for Tilt

Conceptually, we could track multiple variables for any motor; for example, we have DAC, position, and position degree as three different raw data streams for the tilter. However, position and position degree have a perfect positive correlation, which is exactly 1; thus, we only take DAC and position as the raw data for the tilter, as shown in Fig. 17.

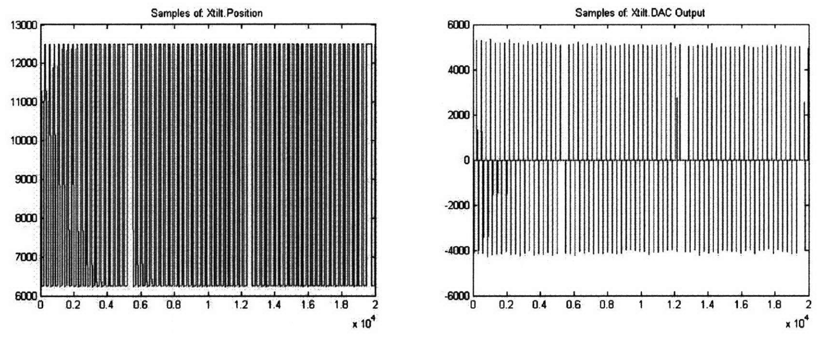


Fig. 17 Overview of position and DAC of Tilt

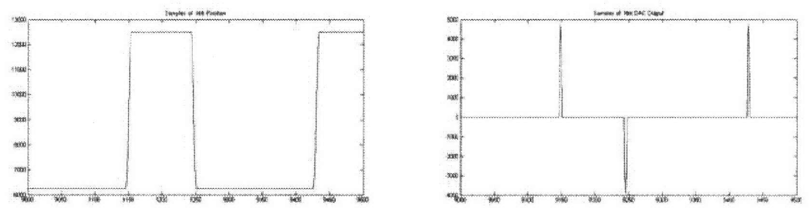


Fig. 18. Zoomed-in of Position and DAC of Tilt

We can see from the raw data plotted in Fig. 17 that there are no different families for tilt as there are for the orienter. In the zoomed-in plot (Fig. 18), we can find that there are two kinds of DAC, which is positive and negative, and happens on the rising and falling part of the position separately. In this thesis, we investigate the positive DAC specifically. Negative DAC can be investigated in the same way.

We overlay sixty cycles of DAC and position data in Fig. 19. The cycle for the position variable is extracted based on the time at which the position first exceeds some threshold, in this case 12,000. The cycle for the DAC variable is extracted based on the time at which the DAC first exceeds 2000 and ends when the DAC first goes below 100.

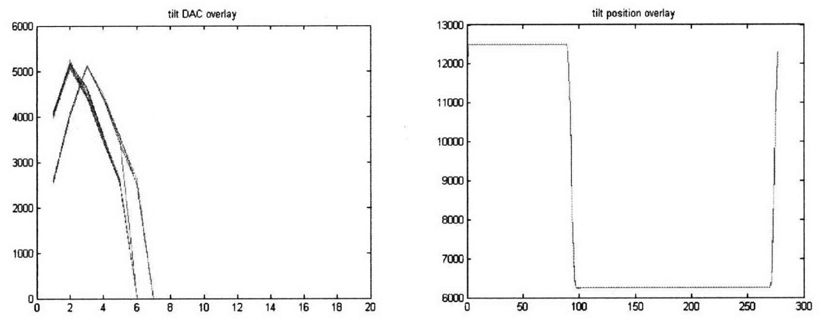
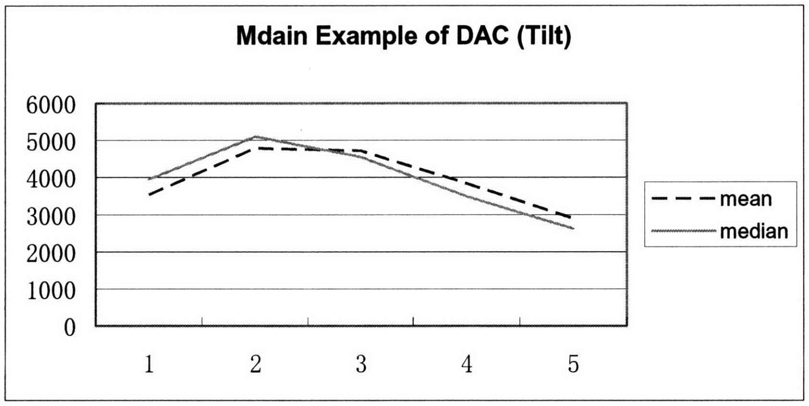


Fig. 19. Overlay of cycles

From the result in Fig. 19, we can see that cycles in tilt DAC and position signals are quite repeatable, with some difference in cycle time. That convinces us that it is worthwhile to investigate repeatable patterns for Tilt DAC and position signals in Section 2.6.

2.6 Find median example of a cycle for the tilter

We plotted the median example using the same method as for the orienter, and compare it with mean behavior in the following plots in Fig. 20. We learn from the result that tilt has highly repeatable patterns in both DAC and position data.



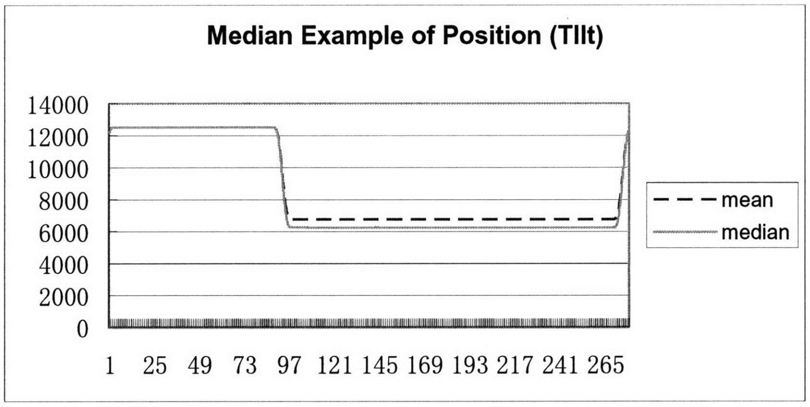


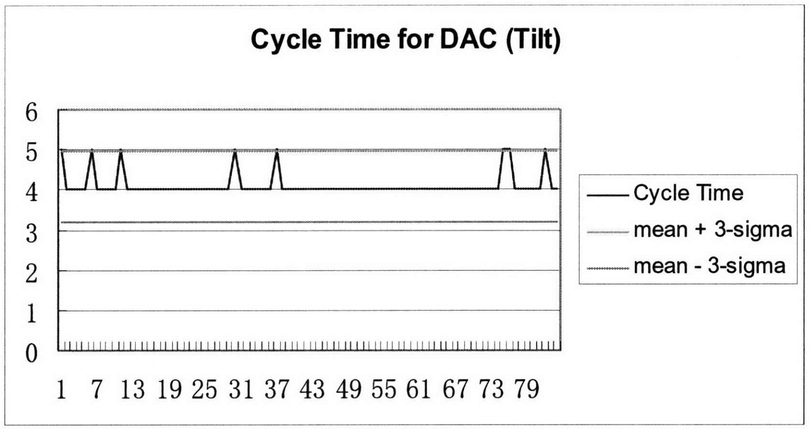
Fig. 20. Median example for DAC and position of the tilter

2.7 Find other generic and specific metadata in the cycles for the tilter

2.7.1 Cycle Time (Generic Metadata)

Cycle time for DAC in the following figure tells that cycle time is almost a constant value for DAC. It is not suitable to evaluate using SPC control chart approaches.

Cycle time for position is also almost a constant value, as it just varies between 276 and 277. So, it is not valuable to draw SPC control chart for it. We plot the cycle time versus duty ratio (which is the period for values above mean divided by the total cycle) for position data in Fig. 21, and find that duty ratio goes up when cycle time changes. We also have calculated the frequency of that change, and find the frequency is also a constant value. All these tell us that cycle time and duty ratio may not be a valuable indicator for tilt robot health.



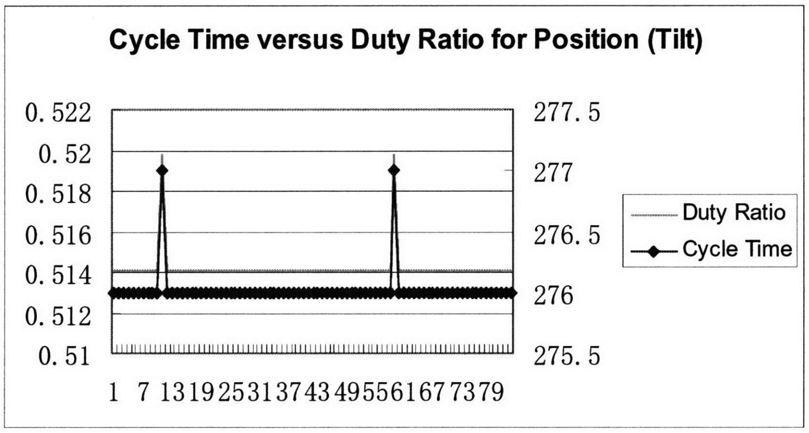
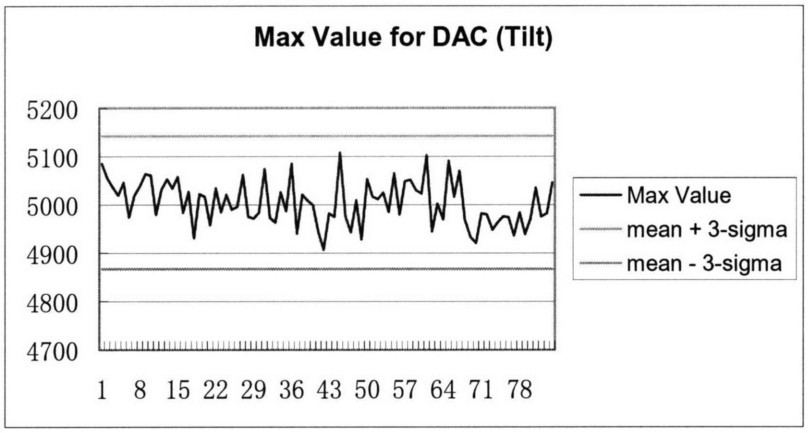


Fig. 21. Cycle Time

2.7.2 Max Value (Generic Metadata)

We can see from the first plot in Fig. 22 that max value for the tilter cycles stays within the SPC limits of DAC. From the second plot of Fig. 22, we can tell that max value is exactly constant for position data. Thus, we can conclude that max value is a good metadata for DAC, but is not a good one for position.



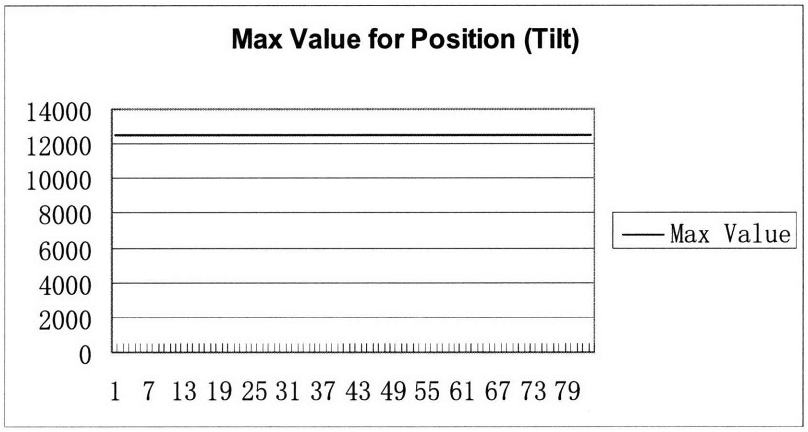
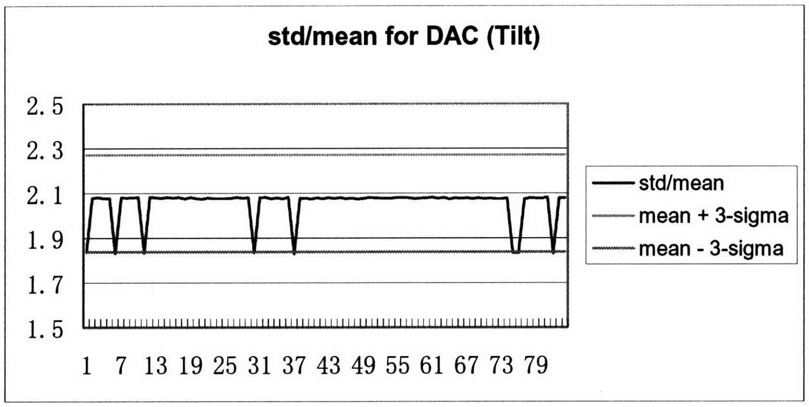
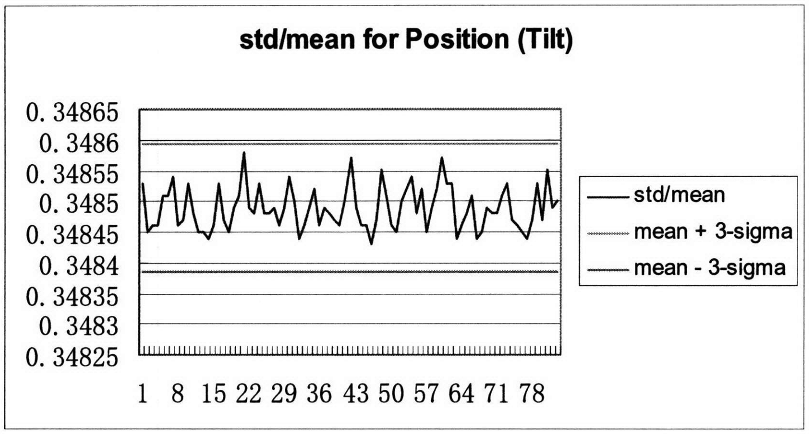


Fig. 22. Max Value

2.7.3 std/mean (Generic Metadata)

We can tell from Fig. 23 that std/mean is not a good metadata for DAC, but is a good one for position data. And, we can also find that position data does not vary very much. It stays closely to its mean, although the small relative deviations we observe in Fig. 23 may still be relevant and useful as indicates of robot health.





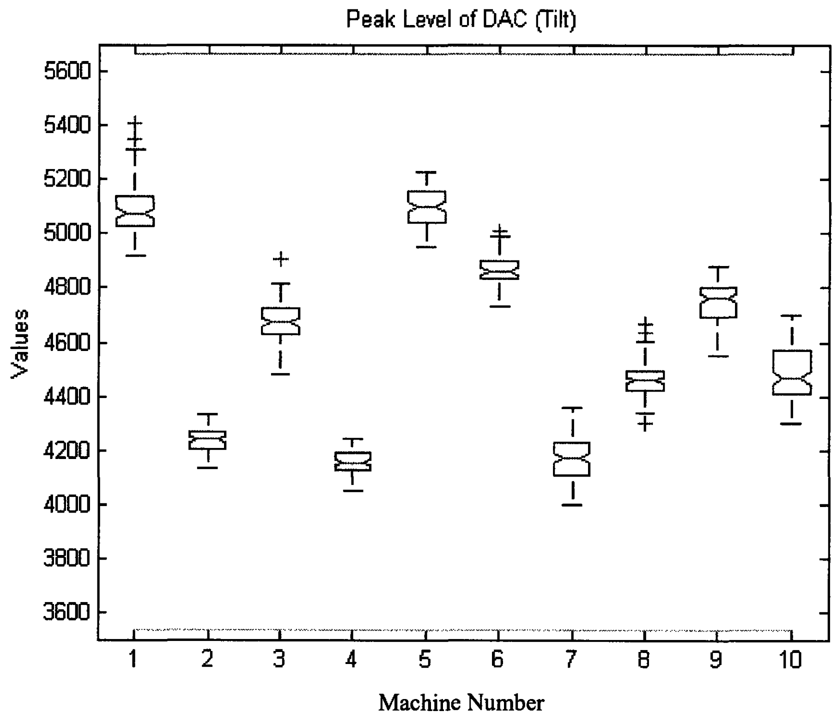
s

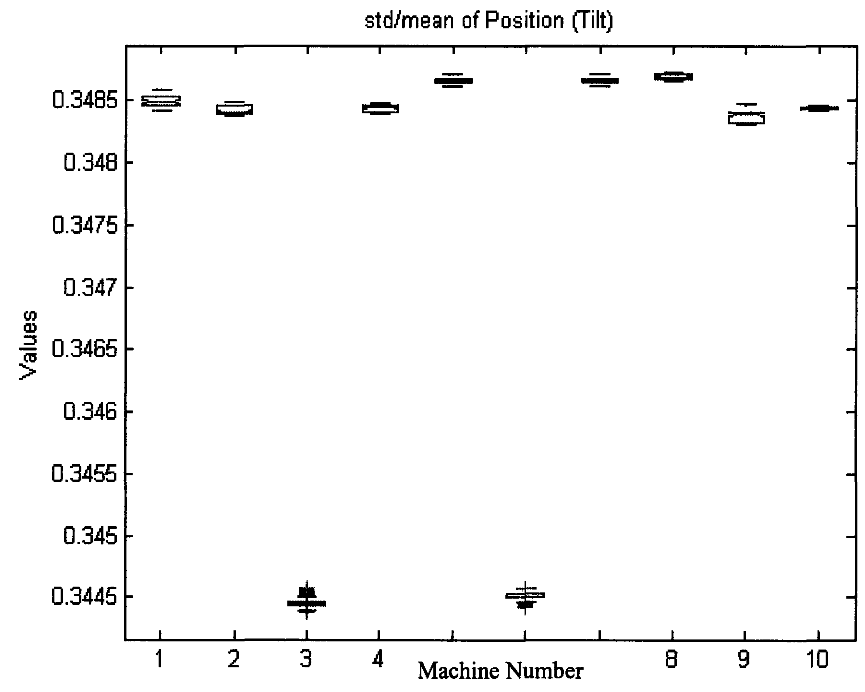
Fig. 23. std/mean for Tilt

2.8 Test Metadata on 10 Machines for Tilt

According to the analysis in Section 2.7, we can find that max value for the tilter has all the data within the SPC control limits for the DAC, and std/mean data stays within SPC control limits for position. Thus, we choose max value as the metadata to test on DAC, and std/mean as the metadata to test on position. As we did for the orienter, we test tilt data on that ten machines as well. The corresponding box plots are shown in Fig. 24.

The results tell us that max value works well on all the ten machines, with all of the control boxes staying within the three sigma SPC limits calculated using mean across all ten machines. Std/mean works well on all of the machines except Machine 3 and 6. In Fig. 24, we see that each box (the range of values for a given machine) are roughly comparable in size, but with substantial mean differences between machines. This indicates that a single set of SPC limits across all machines may not be appropriate, as these limits would have to be very wide to encompass all of the data. Instead, calibration on a per-machine basis might be appropriate, if the box (mean and range) for any given machine is stable over very long periods of time.


Peak Level of DAC (Tilt)
Values
5600
5400
5200
5000
4800
4600
4400
4200
4000
3800
3600
1
2
3
4
5
6
7
8
9
10
Machine Number



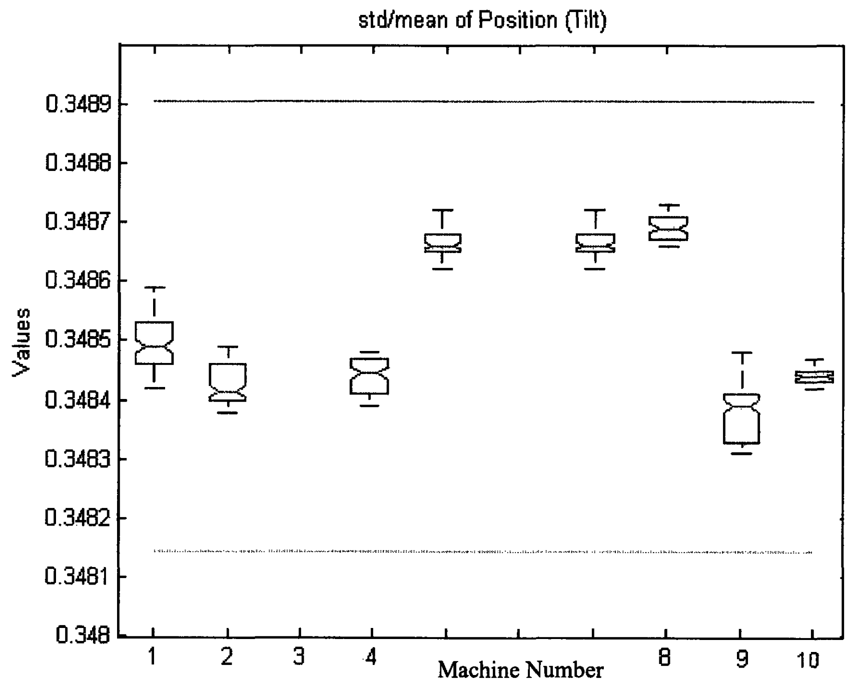


Fig. 24. BoxPlot for Tilt

Chapter 3. Exploration in Models for Predictive Maintenance

In this chapter, we explore models to give an alarm before the machine error. We take a look at the scanner, which is another robot. The error data we take come from a historical error log file that Varian has, which is named “Scan Following Error”. We extract three phases of data. Phase1 is the data before the error alarm goes on; phase2 is the data during the time when error alarm is on; and phase3 is the data after the machine is repaired. The cycle for the position variable is extracted based on the time at which the position first exceeds some threshold, in this case 0. The cycle for the position error variable is the same as of the position variable. Scan following error is known as the error when the scanner cannot follow the required position during each movement.

In Fig. 25, we plot the position data and its corresponding position error data during the last two cycles of phase1 and the first cycle of phase2. The density of the position error peaks in phase2 is greater than that of phase1. This is also the case for the rest of the data. Thus, we decide to calculate the total abstract magnitude in a cycle divided by its cycle time, and take that as a metadata to predict “Scan Following Error.” We name that metadata as “cycle-wise position error density”. In order to get rid of the noise, in further calculation, we cut off the position error values which are less than 20 units around the mean of position error in that cycle.

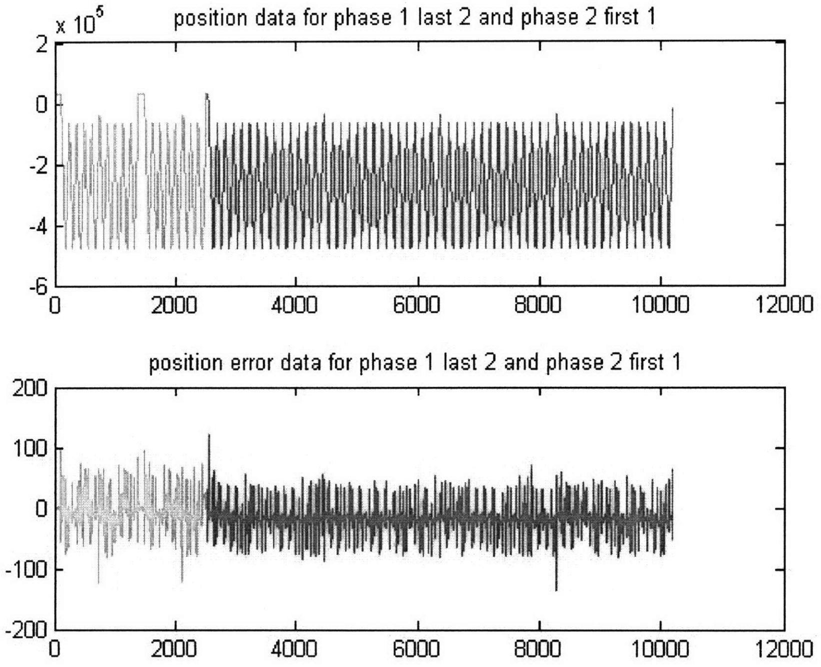


Fig. 25. Data before and after the alarm goes on, for three cycles.

We plot the cycle-wise position error density in Fig. 26, and find that it decreases shortly before the machine goes down, shoots up during the down time of the machine, and goes back to normal after the machine is repaired.

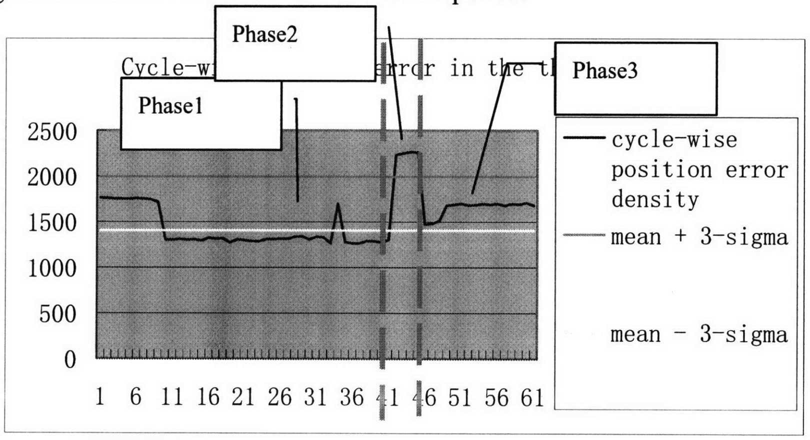


Fig. 26. Cycle-wise position error in the three phases

The 3-sigma is Fig. 26 is calculated based on healthy data, which are points one to ten and points fifty-one to sixty-one. We can see from the plot that phase2 data is out of the 3-sigma limit, which is to say that this metadata are able to identify failure. We can also tell from the plot that phase1 data is out of the 3-sigma limit, which is to say that this metadata may be able to predict the failure many cycles before it occurs.

In Table 2, we plot the detail information for each data point shown in Fig26. "X-sigma" in Table2 means how many sigma the metadata is on that cycle point. Particularly, the phase2 data (in yellow) lie beyond the 7 sigma limits.

Table. 2. Data to predict failure

cycle	Data	X-sigma	Mean + 3-sigma	Mean – 3-sigma
1	1764.1	1.36	1898.66	1405.25
2	1759.4	1.31	1898.66	1405.25
3	1755.3	1.26	1898.66	1405.25
4	1750.3	1.20	1898.66	1405.25
5	1752.1	1.22	1898.66	1405.25
6	1757.1	1.28	1898.66	1405.25
7	1754.3	1.24	1898.66	1405.25
8	1746.2	1.15	1898.66	1405.25
9	1715.4	0.77	1898.66	1405.25
10	1302.8	**-4.25**	1898.66	1405.25
11	1304.1	**-4.23**	1898.66	1405.25
12	1308.4	**-4.18**	1898.66	1405.25
13	1302.7	**-4.25**	1898.66	1405.25
14	1305	**-4.22**	1898.66	1405.25
15	1294.8	**-4.34**	1898.66	1405.25
16	1322.9	**-4.00**	1898.66	1405.25
17	1311.1	**-4.14**	1898.66	1405.25
18	1317.1	**-4.07**	1898.66	1405.25
19	1272.2	**-4.62**	1898.66	1405.25
20	1306.9	**-4.20**	1898.66	1405.25
21	1297.5	**-4.31**	1898.66	1405.25
22	1289.3	**-4.41**	1898.66	1405.25
23	1284.3	**-4.47**	1898.66	1405.25
24	1307.9	**-4.18**	1898.66	1405.25
25	1310	**-4.16**	1898.66	1405.25
26	1314.2	**-4.11**	1898.66	1405.25
27	1313.4	**-4.12**	1898.66	1405.25
28	1336.2	**-3.84**	1898.66	1405.25
29	1338.3	**-3.81**	1898.66	1405.25
31	1334.9	**-3.86**	1898.66	1405.25
32	1329	**-3.93**	1898.66	1405.25
33	1267.4	**-4.68**	1898.66	1405.25
34	1695.7	**0.53**	1898.66	1405.25
35	1279.6	**-4.53**	1898.66	1405.25
36	1264.3	**-4.71**	1898.66	1405.25
37	1267.7	**-4.67**	1898.66	1405.25
38	1291.4	**-4.38**	1898.66	1405.25
39	1288.6	**-4.42**	1898.66	1405.25
40	1275.4	**-4.58**	1898.66	1405.25
41	1302.3	**-4.25**	1898.66	1405.25
42	2234.5	7.08	1898.66	1405.25
43	2251.2	7.29	1898.66	1405.25
44	2266.4	7.47	1898.66	1405.25
45	2261.8	7.42	1898.66	1405.25
46	1471.5	-2.19	1898.66	1405.25
47	1482.9	-2.06	1898.66	1405.25
48	1509.1	-1.74	1898.66	1405.25
49	1678.3	0.32	1898.66	1405.25
50	1687.9	0.44	1898.66	1405.25
51	1700.2	0.59	1898.66	1405.25
52	1683	0.38	1898.66	1405.25
53	1683.6	0.38	1898.66	1405.25
54	1696.8	0.55	1898.66	1405.25
55	1687	0.43	1898.66	1405.25
56	1699.3	0.58	1898.66	1405.25
57	1677.6	0.31	1898.66	1405.25
58	1696.8	0.55	1898.66	1405.25
59	1689.5	0.46	1898.66	1405.25
60	1710.2	0.71	1898.66	1405.25
61	1677.6	0.31	1898.66	1405.25

We can see that our metadata, the cycle-wise position error density, successfully identifies and predicts the scan following error. Specifically, the first indication of unusual behavior occurs at cycle number 10, which is 32 cycles before the machine fails at cycle 42.

Chapter 4. Implementation

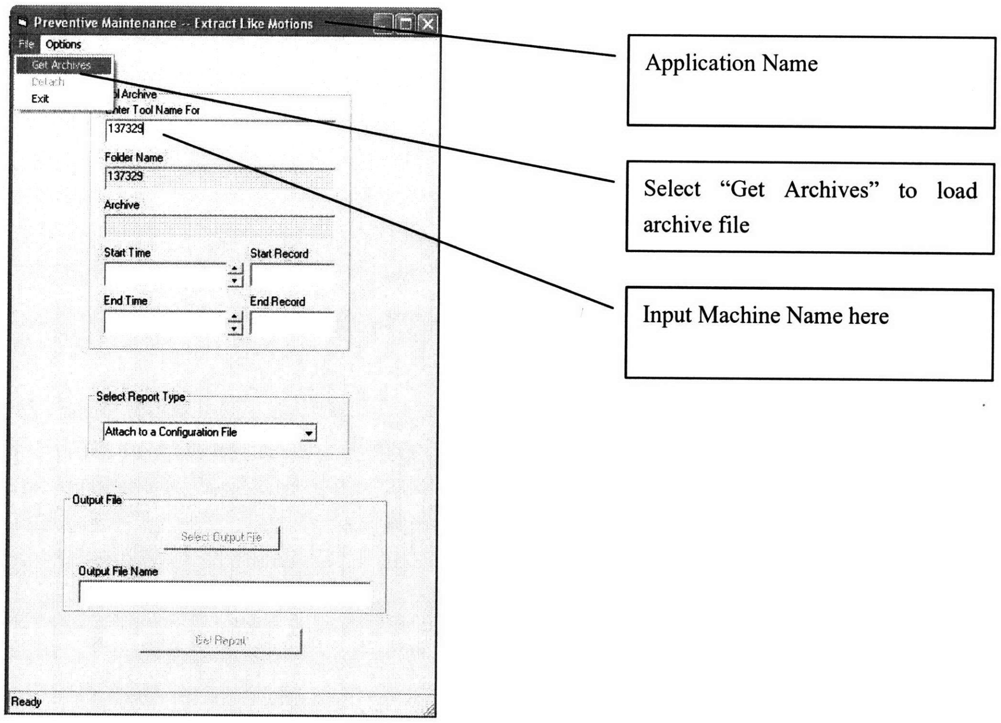
The approach and model discussed in Chapter 2 for metadata extraction are programmed into an application using Visual Basic. We compiled them into two extractable files ease of use. They are "Extract Like Motions.exe", and "Extract Metadata.exe". Major components of the Visual Basic codes are listed in the appendix.

4.1 Application Description and User Guide for "Extract Like Motions.exe"

"Extract Like Motions.exe" is designed to extract like motions and their matching current traces. PMAC IanArchive file should be ready for the application. By default, fifty continuous cycles will be captured and generated in 50 csv files by the program; this number can be changed in the VB code if desired.

Here is the User Guide for that application.

Double clicking the exe file, the user will see the application user interface as shown in Fig. 27. In Fig. 28 through 31, the step-by-step sequence of actions is illustrated, to the right of each screen shot are explanations of what each user input field means. Fig. 32 shows an example of a cycle output file. The record number of each data point, its DAC output, and position data are listed in the file for that cycle. The user is able to plot the figure if he or she opens the csv file using Excel.



Preventive Maintenance -- Extract Like Motions

File Options

Get Archives
Detach
Exit

ol Archive
nter Tool Name For: 137329
Folder Name: 137329
Archive:
Start Time:
Start Record:
End Time:
End Record:

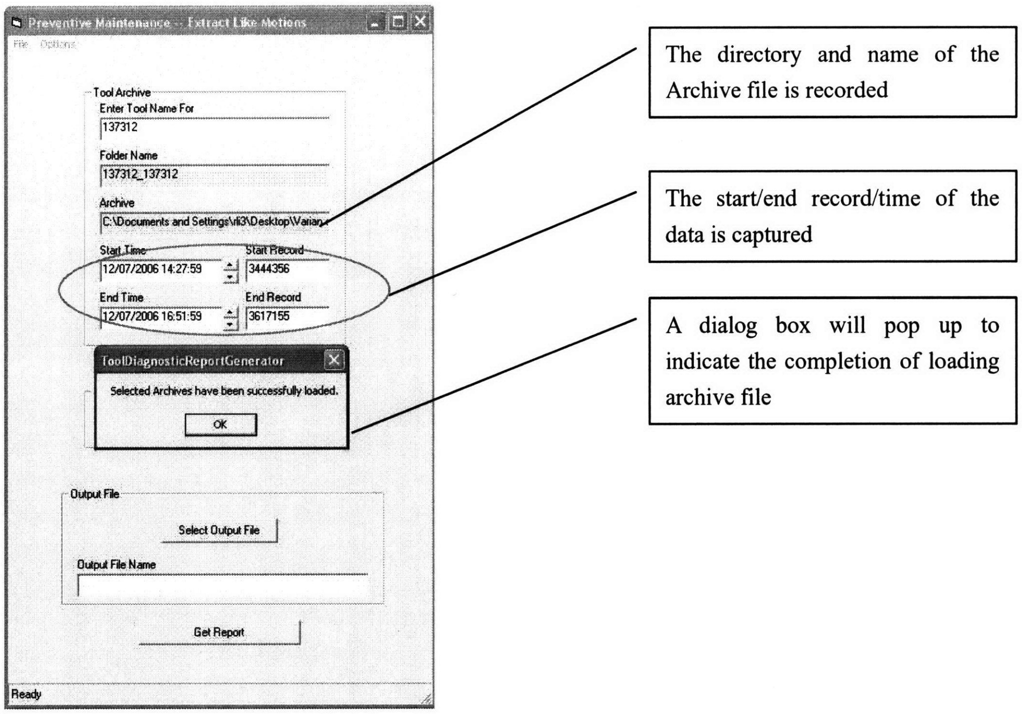
Select Report Type: Attach to a Configuration File

Output File
Select Output File
Output File Name:

Get Report

Ready

Fig. 27. step1—open the exe file, and load archives.



Preventive Maintenance -- Extract Like Motions

File Options

Tool Archive

Enter Tool Name For: 137312

Folder Name: 137312_137312

Archive: C:\Documents and Settings\rli3\Desktop\Varian

Start Time	Start Record
12/07/2006 14:27:59	3444356

End Time	End Record
12/07/2006 16:51:59	3617155

ToolDiagnosticReportGenerator

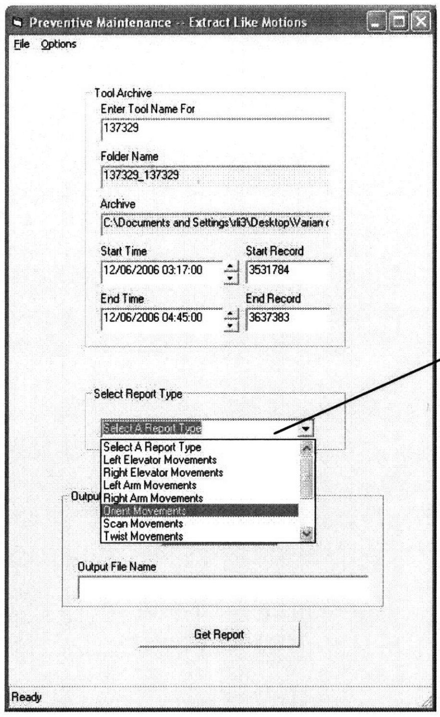
Selected Archives have been successfully loaded.

The directory and name of the Archive file is recorded

The start/end record/time of the data is captured

A dialog box will pop up to indicate the completion of loading archive file

Fig. 28. Step2--Wait for the system to get archive information.


Preventive Maintenance -- Extract Like Motions

File Options

Tool Archive

Enter Tool Name For: 137329

Folder Name: 137329_137329

Archive: C:\Documents and Settings\rli3\Desktop\Varian (

Start Time: 12/06/2006 03:17:00 Start Record: 3531784

End Time: 12/06/2006 04:45:00 End Record: 3637383

Select Report Type: Select A Report Type

- Select A Report Type
- Left Elevator Movements
- Right Elevator Movements
- Left Arm Movements
- Right Arm Movements
- Orient Movements
- Scan Movements
- Twist Movements

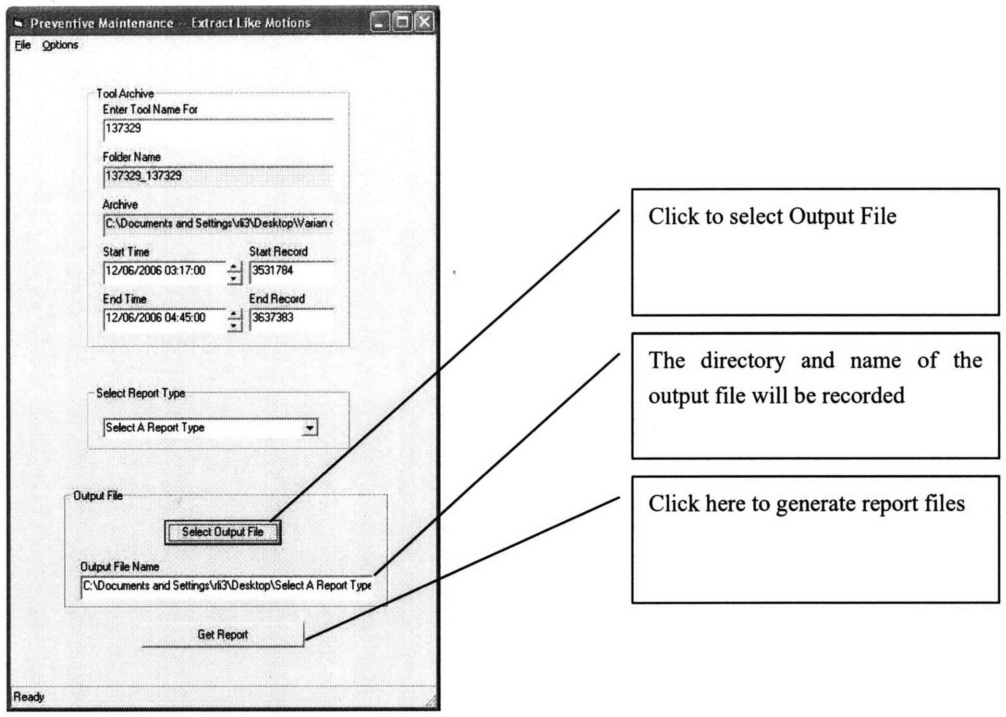
Output File Name

Get Report

Ready

Select Target Robot from the list. The list includes Orient and Tilt

Fig. 29. Step3--Select Target Robot.



Tool Archive

Enter Tool Name For: 137329

Folder Name: 137329_137329

Archive: C:\Documents and Settings\rli3\Desktop\Varian (

Start Time: 12/06/2006 03:17:00 Start Record: 3531784

End Time: 12/06/2006 04:45:00 End Record: 3637383

Select Report Type: Select A Report Type

Output File

Output File Name: C:\Documents and Settings\rli3\Desktop\Select A Report Type

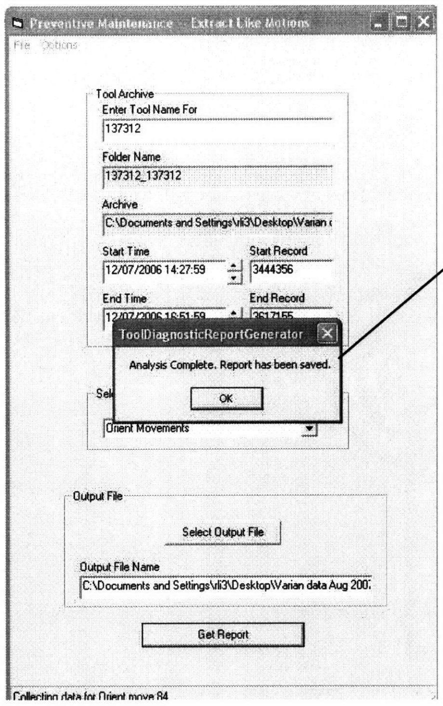
Ready

Click to select Output File

The directory and name of the output file will be recorded

Click here to generate report files

Fig. 30. Step4--Select Output File.



Tool Archive

Enter Tool Name For: 137312

Folder Name: 137312_137312

Archive: C:\Documents and Settings\rli3\Desktop\Varian c

Start Time: 12/07/2006 14:27:59

Start Record: 3444356

End Time: 12/07/2006 16:51:59

End Record: 3617155

ToolDiagnosticReportGenerator

Analysis Complete. Report has been saved.

Sel

Orient Movements

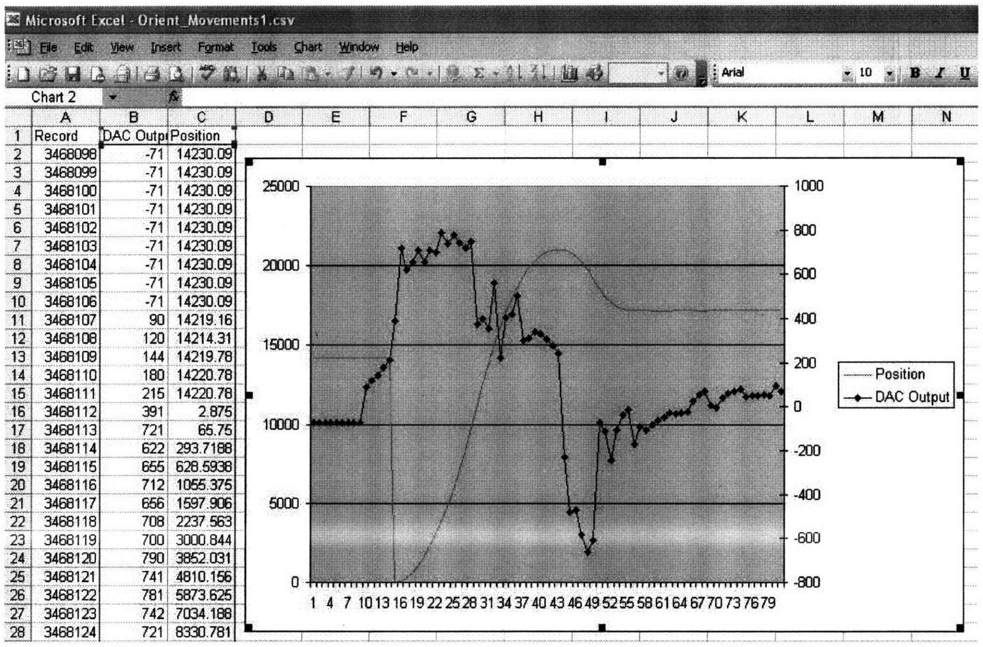
Output File

Output File Name: C:\Documents and Settings\rli3\Desktop\Varian data Aug 2007

Collecting data for Orient move 84

A dialog box will pup up to indicate the completion of generating report files.

Fig. 31. Step5--Wait for the system to generate report files



Microsoft Excel - Orient_Movements1.csv
File Edit View Insert Format Tools Chart Window Help
Arial 10 B I U
Chart 2

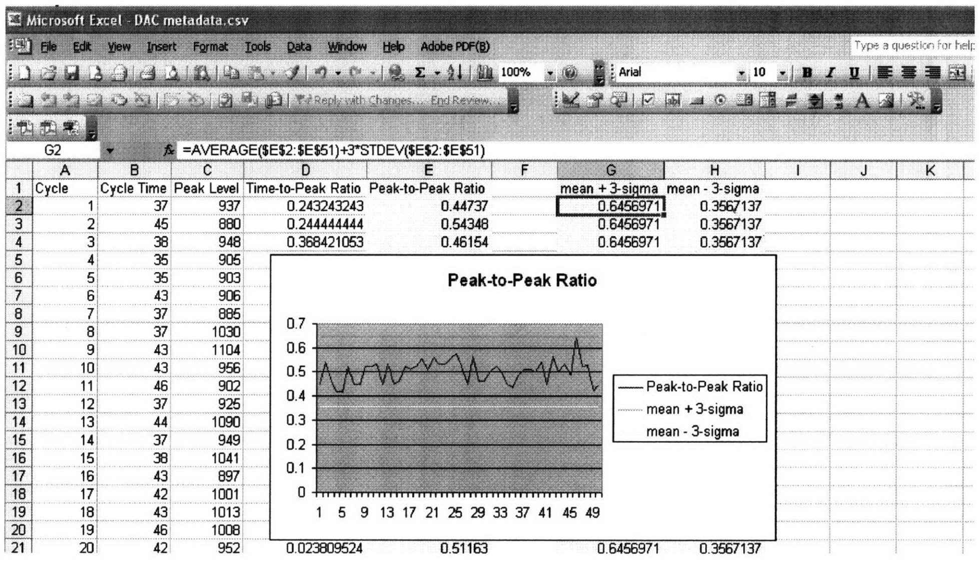
	A	B	C
1	Record	DAC Outp	Position
2	3468098	-71	14230.09
3	3468099	-71	14230.09
4	3468100	-71	14230.09
5	3468101	-71	14230.09
6	3468102	-71	14230.09
7	3468103	-71	14230.09
8	3468104	-71	14230.09
9	3468105	-71	14230.09
10	3468106	-71	14230.09
11	3468107	90	14219.16
12	3468108	120	14214.31
13	3468109	144	14219.78
14	3468110	180	14220.78
15	3468111	215	14220.78
16	3468112	391	2.875
17	3468113	721	65.75
18	3468114	622	293.7188
19	3468115	655	628.5938
20	3468116	712	1055.375
21	3468117	656	1597.906
22	3468118	708	2237.563
23	3468119	700	3000.844
24	3468120	790	3852.031
25	3468121	741	4810.156
26	3468122	781	5873.625
27	3468123	742	7034.188
28	3468124	721	8330.781

Fig. 32. the output file for one cycle of the orienter

4.2 Application Description and User Guide for "Extract Metadata.exe"

"Extract Metadata.exe" is designed to extract metadata as detailed in Chapter 2. PMAC IanArchive file should be ready for the application. "DAC metadata.csv" and "Position metadata.csv" will be generated by the application. To use the application, the user will perform the same steps as are described in Section 4.1 to locate and load the data, and generate the desired data. After that, the user will take one more step if he or she wants to analyze the data using SPC control charts.

To generate the SPC control charts described in Chapter 2, the user will need to plot the 3-sigma control line on the data he or she desires. For example, if the user wants to plot an SPC control chart on the orienter DAC peak-to-peak ratio metadata, what he or she needs to do is to open that Excel CSV file, and generate two columns by typing in Excel statistical functions as is shown in Fig. 33. Then, he or she can plot those three columns to get the SPC control chart for the specific metadata.



G2 =AVERAGE($E$2:$E$51)+3*STDEV($E$2:$E$51)

	A	B	C	D	E	F	G	H	I	J	K
1	Cycle	Cycle Time	Peak Level	Time-to-Peak Ratio	Peak-to-Peak Ratio		mean + 3-sigma	mean - 3-sigma			
2	1	37	937	0.243243243	0.44737		0.6456971	0.3567137			
3	2	45	880	0.244444444	0.54348		0.6456971	0.3567137			
4	3	38	948	0.368421053	0.46154		0.6456971	0.3567137			
5	4	35	905								
6	5	35	903								
7	6	43	906								
8	7	37	885								
9	8	37	1030								
10	9	43	1104								
11	10	43	956								
12	11	46	902								
13	12	37	925								
14	13	44	1090								
15	14	37	949								
16	15	38	1041								
17	16	43	897								
18	17	42	1001								
19	18	43	1013								
20	19	46	1008								
21	20	42	952	0.023809524	0.51163		0.6456971	0.3567137			

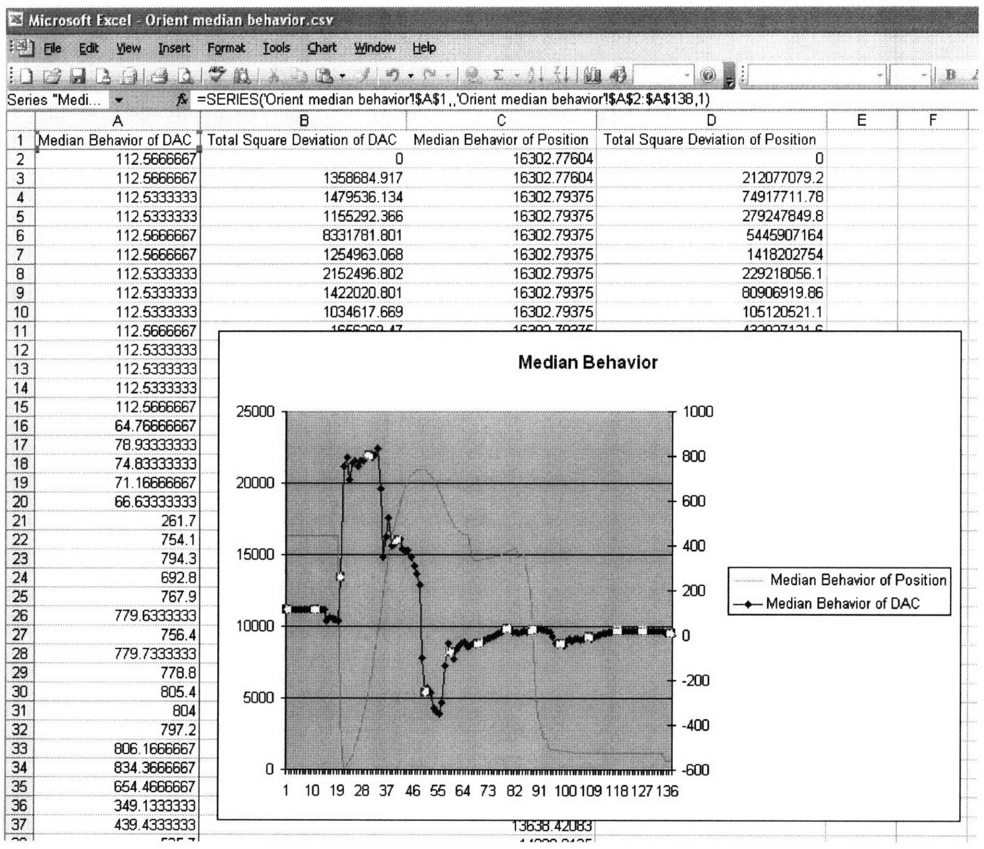
Fig. 33. Analyzing the output file of "extract metadata.exe", to generate an SPC control chart with the 3-sigma control lines added.

4.3 Application Description and User Guide for "Extract Mean Behavior.exe"

"Extract Mean Behavior.exe" is designed to extract mean or median behavior as well as the total squared deviation as detailed in Chapter 2. In order to make the application run fast, in this implementation we take the mean behavior as the "usual" behavior or template for comparison with later cycle data. According to the discussion in Section 2.2.1, there is not much difference between mean and median behavior. Thus, it is reasonable to take the mean as example or template behavior; if the median template is desired, extensions to the VB code should be implemented, based on finding the minimum total squared deviation example as discussed in Section 2.2.1.

PMAC IanArchive file should be ready for the application. "Orient mean behavior.csv" and "Tilt mean behavior.csv" will be generated by the application based on the robots chosen by the user. 30 cycles will be taken to extract the mean behavior and total squared deviation. To use the application, the user will perform the same steps as are described in Section 4.1 to locate and load the data, and generate the desired data. After that, the user will find the result in the csv files.

There are four columns in the csv files for each of the results separately, which are: "Mean Behavior of DAC", "Total Square Deviation of DAC", "Mean Behavior of Position", and "Total Square Deviation of Position" as shown in Fig. 34.


Microsoft Excel - Orient median behavior.csv
File Edit View Insert Format Tools Chart Window Help
Series "Medi...

```
=SERIES('Orient median behavior'!$A$1,,'Orient median behavior'!$A$2:$A$138,1)
```

	A	B	C	D	E	F
1	Median Behavior of DAC	Total Square Deviation of DAC	Median Behavior of Position	Total Square Deviation of Position		
2	112.5666667	0	16302.77604	0		
3	112.5666667	1358684.917	16302.77604	212077079.2		
4	112.5333333	1479536.134	16302.79375	74917711.78		
5	112.5333333	1155292.366	16302.79375	279247849.8		
6	112.5666667	8331781.801	16302.79375	5445907164		
7	112.5666667	1254963.068	16302.79375	1418202754		
8	112.5333333	2152496.802	16302.79375	229218056.1		
9	112.5333333	1422020.801	16302.79375	80906919.86		
10	112.5333333	1034617.669	16302.79375	105120521.1		
11	112.5666667					
12	112.5333333					
13	112.5333333					
14	112.5333333					
15	112.5666667					
16	64.76666667					
17	78.93333333					
18	74.83333333					
19	71.16666667					
20	66.63333333					
21	261.7					
22	754.1					
23	794.3					
24	692.8					
25	767.9					
26	779.6333333					
27	756.4					
28	779.7333333					
29	778.8					
30	805.4					
31	804					
32	797.2					
33	806.1666667					
34	834.3666667					
35	654.4666667					
36	349.1333333					
37	439.4333333		13638.42083			

Fig. 34. Get results from the output file of "Extract Mean Behavior.exe"

Chapter 5. Conclusion and Future Work

The ability to automatically perform early detection of equipment failures in a production line can lead to significant improvements in the overall capability and profitability of the process [5]. Predictive maintenance is helpful because it can detect the equipment failures and provide alarms, which can help a factory in both quality control and productivity. First, it can help to improve the production quality, because it can suggest proper maintenance method to avoid the failure. Second, it can help to increase the productivity, because it can leave sufficient time for the factory to buy spare parts so as that to reduce the time to repair or maintain the machine.

Varian Semiconductor Equipment Associates, Inc. (VSEA) desires a method to help predict failure of ion implanters. Predictive maintenance would help to reduce the unscheduled downtime of ion implanters, whose throughput and uptime is highly important to customers.

An efficient way to predict time-to-failure is desired both by the industry and by VSEA, to perform efficient predictive maintenance. However, much of predictive maintenance in industry still focuses on practical experience [14] [15] and mechanical analysis [12] [13], which is difficult to automate. An effective method in process and equipment diagnosis is statistical process control (SPC) [10]. SPC has traditionally been applied using a single sample to measure characteristics of the product produced in a piece of equipment, or using a single parameter representative of the operation of the equipment, for each process execution. But in this project, we have many real time measurements during each "run" or cycle of the equipment. Thus, proper metadata are required to be selected from that pool to serve as the indicators for machine health. Ideal metadata are those that can tolerant the fluctuation in healthy data, and at the same time, capture the unhealthy data sharply whenever they appear.

5.1 Contributions

We investigate the approaches to find failure prevention methods in Chapter 2. The orienter and tilter are taken as example robots to find metadata. The detailed steps we take to investigate metadata are as follows. First, we overlay cycles for the robot, to see whether it is possible to find repeatable patterns of the cycles. In order to do that, we perform signal filtering on raw data. Second, we try to find a median example of a cycle for that robot, to confirm that there is a repeatable pattern. We further investigate the total squared deviation to see if it can serve as a metadata. Third, we explore other metadata. We investigate two types of metadata. They are: general metadata, which are normal ones for all signal process data; and specific metadata, which are suggested by special features of the robot data. Finally, we test the metadata on 10 machines. Box plots have been used for the test, and SPC limits of the mean value of the box plots have been drawn to investigate the metadata.

The result in Chapter 2 tells that, for the orienter, std/mean and peak-to-peak ratio can serve as good metadata for both DAC and position data; for the tilt, max value is a good metadata for DAC, and std/mean is a good metadata variable for position data. Those data are all within SPC limits of both the individual machine data and the ten machine data.

In Chapter 3, we explore a model to predict failure before the machine fails. Data are taken from historical machine records when it is before the failure, during the failure, and after the failure. SPC is implemented on the data. The result suggests that it is possible to differentiate those statuses, and to provide an alarm before the machine fails.

In Chapter 4, we further develop the application using Visual Basic based on our study. The application enables the user to extract real-time data from the machine, and get analysis reports in CSV format. A user guide is also provided on how to generate the analysis reports on new data using our application.

5.2 Future Work

Future work may include improvements in four areas.

First, the use of multivariable SPC approaches should be investigated. That will include PCA (Principal Component Analysis) [1]. We may be able to get many effective metadata if we look at more signals of the robots. PCA will help us to tell what is the biggest impact factor from the pool of metadata, and to explore correlations between multiple data variables.

Second, some interesting metadata performance can be further investigated. For example, the max value of DAC data shows a trend in addition to "random" fluctuations. If there is a trend in time, this indicates that something may be changing over long periods of time in the equipment. Future work may be suggested to project or extrapolate forward in time, for example, potentially estimating a time in the future when this indicator might exceed same specification limit, signaling a need for future preventative maintenance.

Third, a more mature model to predict failure once we get sufficient unhealthy data can be developed. In Chapter 3, we do not have sufficient data to show the trend of data before the failure happens, and thus are not able to predict when the machine will fail. If we can have sufficient data, we can perform proof testing on the data before the failure happen, and then may be able to predict when the machine will fail, and its possibility to fail.

Finally, future work can strengthen the VB application. We can investigate more customer requirements, such as how many robots will use this application, and where do they expect to proliferate the application. We can then improve the program by adding more features which are able to be changed by users.

In a summary, with the help of Professor Boning and VSEA, this project has been successfully done to fulfill the customer requirement. An application has been delivered to perform the expected function. And interesting future work is proposed for future students.

References

[1] Hai-Fang Guo, Costas J. Spanos, and Alan J. Miller, "Real Time Statistical Process Control for Plasma Etching," *IEEE/SEMI Int'l Semiconductor Manufacturing Science Symposium,* pp. 113-118, 1991.

[2] Costas J. Spanos, "Statistical Process Control in Semiconductor Manufacturing," *Proceedings of the IEEE*, vol. 80, no. 6, pp. 819-830, June 1992.

[3] Costas J. Spanos, Hai-Fang Guo, Alan Miller, Joanne Levine-Parrill, "Real-Time Statistical Process Control Using Tool Data," *IEEE Transactions on Semiconductor Manufacturing*, vol. 5, no. 4, pp. 308-318, Nov. 1992

[4] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research,* vol. 4, pp. 237-285, 1996.

[5] S. Theodoridis and K. Koutroumbas, Pattern Recognition, Academic Press, 1999.

[6] T. Hastie, R. Tibshirani, and J. Fridman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer-Verlag, 2001.

[7] T. T. Lee, T. F. Liu, P. L. Mao, P. Wu, B. W. Lee, B. J. Ng and L. K. See, "An Object-oriented Automated SPC Analysis and Reporting System for Powder Manufacturing in the Specialty Chemical Plants," *Seventh International Conference on Control, Automation, Robotics and Vision (ICARCV'02),* pp. 1398-1401, Dec. 2002.

[8] S. M. Hanna, "Application of Statistical Process Control (SPC) in the Manufacturing of Medical Accelerators," *Proceedings of the 2003 Particle Accelerator Conference,* pp. 1077-1079, 2003.

[9] Chris Fredric, Geoffrey Crabtree, Konstantin Holderman, Lisa Mandrell, Jeff Nickerson, and Theresa Jester, "Statistical Process Control Driven Variation Reduction Critical to Manufacturing Success," *Photovoltaic Specialists Conference,* pp. 939-942, 2005.

[10] Gary S. May, and Costas J. Spanos, Fundamentals of Semiconductor Manufacturing and Process Control, Wiley-Interscience, 2006.

[11] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan, "Frequent Pattern Mining: Current Status and Future Directions," *Data Mining and Knowledge Discovery,* vol. 14, no. 1, pp. 55-86, 2007.

[12] Lou Morando, "Front line Condition Monitoring using Shock Pulse for Bearing Damage Detection and Lubrication Condition," *The Predictive Maintenance Technology Conference and Expo,* 2007.

[13] Greg Livingstone, "Revolutionary Contamination Control Technologies for Lubricants," *The Predictive Maintenance Technology Conference and Expo,* 2007.

[14] Brett Anders, “Driving Change using Condition Based Maintenance,” *The Predictive Maintenance Technology Conference and Expo*, 2007.

[15] Paul Dufrense, “Strategies for Effective Contamination Monitoring and Control,” *The Predictive Maintenance Technology Conference and Expo*, 2007.

Appendix

VB Program to Generate Orient Metadata

```
'***********************************************
'This program is to generate the orienter metadata.
'Cursor is used to collect data and control the steps
'The structure is like: variable declarations, followed by the time-series steps to be executed
'***********************************************
Public Sub StoreOrientMovements()
Dim OrientPositionCommandHandle, DACOutHandle, PositionHandle, PositionErrHandle
Dim OrientPositionCommandCursor, DACOutCursor, PositionCursor, PositionErrCursor
Dim ReportDirectory As String, i As Long, cycle_within As Integer, StartRecord As Long, EndRecord As Long, LastRecordNumber As Long
Dim ittr As Integer, flag As Integer, ittr1 As Integer
Dim PositionErrMax As Single
Dim Family As Integer 'indicate whether it's Family 1 or 2
Dim CycleTime(50) As Double, PeakLevel_DAC(50) As Double, PeakLevel_Position(50) As Double, TimeToPeakRatio_DAC(50) As Double, TimeToPeakRatio_Position(50) As Double, PeakToPeakRatio_DAC(50) As Double, PeakToPeakRatio_Position(50) As Double 'metadata
Dim MainPeakRecord As Long, MinorPeakRecord As Long 'to help calculate PeakToPeakRatio_DAC

' Set up signal handles and cursors
With p_oVCSDataProvider

    OrientPositionCommandHandle = .GetGlobalSignalHandle("Orient.Position Command")
    DACOutHandle = .GetGlobalSignalHandle("Orient.DAC Output")
    PositionHandle = .GetGlobalSignalHandle("Orient.Position")
    PositionErrHandle = .GetGlobalSignalHandle("Orient.Position Error")

    Set OrientPositionCommandCursor = .GetCursor(OrientPositionCommandHandle)
    Set DACOutCursor = .GetCursor(DACOutHandle)
    Set PositionCursor = .GetCursor(PositionHandle)
    Set PositionErrCursor = .GetCursor(PositionErrHandle)

End With

ReportDirectory = Mid(frmAnalysis.txtOutputFile.Text, 1, InStrRev(frmAnalysis.txtOutputFile.Text, "\"))
```

```
With DACOutCursor

    .record = CLng(frmAnalysis.txtStartRecord.Text) + 10000
    cycle_within = 0
    While (.State < 3) And (Not CancelReport) And (.record <
CLng(frmAnalysis.txtEndRecord.Text))
        If cycle_within > 49 Then 'generate metadata file
            frmAnalysis.txtOutputFile.Text = ReportDirectory & "Orient DAC Metadata.csv"
            WriteReportHeader
            DoEvents
            For i = 1 To 50
                OutputFile.WriteLine i & "," & CycleTime(i) & "," & PeakLevel_DAC(i) & "," &
TimeToPeakRatio_DAC(i) & "," & PeakToPeakRatio_DAC(i)
            Next i
            OutputFile.Close
            frmAnalysis.txtOutputFile.Text = ReportDirectory & "Orient Position Metadata.csv"
            WriteReportHeader
            DoEvents
            For i = 1 To 50
                OutputFile.WriteLine i & "," & CycleTime(i) & "," & PeakLevel_Position(i) & ","
& TimeToPeakRatio_Position(i) & "," & PeakToPeakRatio_Position(i)
            Next i
            OutputFile.Close

            Exit Sub
        Else
            cycle_within = cycle_within + 1

        End If
        flag = 0
        While flag = 0
            ittr = 0
            While .value < 400 And .value > -400
                ittr = ittr + 1
                .MoveNext
            Wend
            If ittr > 100 Then
                flag = 1
                StartRecord = .record
            End If
        .MoveNextChange
        If .State > 2 Then
            DebugPrint "End of Archive reached when looking for StartRecord."
            OutputFile.Close
            Exit Sub
        End If
    Wend
    flag = 0
    While flag = 0
        ittr = 0
        While .value < 400 And .value > -400 And ittr < 101
            ittr = ittr + 1
            .MoveNext
        Wend
        If ittr > 100 Then
            flag = 1
            EndRecord = .record - 100
        End If
        .MoveNextChange
        If .State > 2 Then
            DebugPrint "End of Archive reached when looking for EndRecord."
            OutputFile.Close
            Exit Sub
        End If
    Wend
    flag = 0

    StartRecord = StartRecord - 50
    EndRecord = EndRecord + 30
    CycleTime(cycle_within) = EndRecord - StartRecord

    For i = StartRecord To EndRecord
        DACOutCursor.record = i
        PositionCursor.record = i
        PositionErrCursor.record = i
        .record = i
        If PeakLevel_DAC(cycle_within) < .value Then
            PeakLevel_DAC(cycle_within) = .value
            TimeToPeakRatio_DAC(cycle_within)    =    (.record    -    StartRecord)    /
CycleTime(cycle_within)
        End If
```

```
            If PeakLevel_Position(cycle_within) < PositionCursor.value Then
                PeakLevel_Position(cycle_within) = .value
                TimeToPeakRatio_Position(cycle_within) = (.record - StartRecord) /
CycleTime(cycle_within)
            End If
            PeakToPeakRatio_DAC(cycle_within) = PeakToPeakRatio_DAC(cycle_within)
+ .value   'total
            PeakToPeakRatio_Position(cycle_within) = PeakToPeakRatio_Position(cycle_within)
+ PositionCursor.value 'total
        Next i
        DACOutCursor.record = StartRecord
        While .record < EndRecord And .value < PeakToPeakRatio_DAC(cycle_within) /
CycleTime(cycle_within)
            .MoveNextChange
        Wend
        While .record < EndRecord And .value > PeakToPeakRatio_DAC(cycle_within) /
CycleTime(cycle_within)
            .MoveNextChange
        Wend
        MainPeakRecord = .record
        While .record < EndRecord And .value < PeakToPeakRatio_DAC(cycle_within) /
CycleTime(cycle_within)
            .MoveNextChange
        Wend
        While .record < EndRecord And .value > PeakToPeakRatio_DAC(cycle_within) /
CycleTime(cycle_within)
            .MoveNextChange
        Wend
        MinorPeakRecord = .record
        PeakToPeakRatio_DAC(cycle_within) = (MinorPeakRecord - MainPeakRecord) /
CycleTime(cycle_within)
        PositionCursor.record = StartRecord
        While PositionCursor.record < EndRecord And PositionCursor.value <
PeakToPeakRatio_Position(cycle_within) / CycleTime(cycle_within)
            PositionCursor.MoveNextChange
        Wend
        While PositionCursor.record < EndRecord And PositionCursor.value >
PeakToPeakRatio_Position(cycle_within) / CycleTime(cycle_within)
            PositionCursor.MoveNextChange
        Wend
        MainPeakRecord = PositionCursor.record
        While PositionCursor.record < EndRecord And PositionCursor.value <
PeakToPeakRatio_Position(cycle_within) / CycleTime(cycle_within)
            PositionCursor.MoveNext
        Wend
        While PositionCursor.record < EndRecord And PositionCursor.value >
PeakToPeakRatio_Position(cycle_within) / CycleTime(cycle_within)
            PositionCursor.MoveNext
        Wend
        MinorPeakRecord = PositionCursor.record
        PeakToPeakRatio_Position(cycle_within) = (MinorPeakRecord - MainPeakRecord) /
CycleTime(cycle_within)

        If CycleTime(cycle_within) > 250 Then 'error data
            cycle_within = cycle_within - 1
        End If
        .record = EndRecord - 30
        .MoveNextChange

    Wend

        'to change file name


End With

End Sub
Public Sub
```

VB Program to Generate Tilt Metadata

```
'***********************************************
'This program is to generate the tilter metadata.
'Cursor is used to collect data and control the steps
'The structure is like: variable declarations, followed by the time-series steps to be executed
'***********************************************
StoreXTiltMovements()
Dim PositionStatusHandle, DACOutHandle, PositionDegreesHandle, ImplantPositionErrHandle,
PositionHandle, PositionErrHandle
Dim PositionStatusCursor, DACOutCursor, PositionDegreesCursor, ImplantPositionErrCursor,
PositionCursor, PositionErrCursor
```

```
Dim ReportDirectory As String, i As Long, cycle_within As Integer, StartRecord As Long, EndRecord
As Long, LastRecordNumber As Long
Dim PositionErrMax As Single
Dim CycleTime(50) As Double, PeakLevel_DAC(50) As Double, PeakLevel_Position(50) As Double

' Set up signal handles and cursors
With p_oVCSDataProvider

    PositionStatusHandle = .GetGlobalSignalHandle("Xtilt Load Position.Status")
    DACOutHandle = .GetGlobalSignalHandle("Xtilt.DAC Output")
    PositionDegreesHandle = .GetGlobalSignalHandle("Xtilt.Position Degrees")
    ImplantPositionErrHandle = .GetGlobalSignalHandle("Xtilt.Implant Position Error")
    PositionHandle = .GetGlobalSignalHandle("Xtilt.Position")
    PositionErrHandle = .GetGlobalSignalHandle("Xtilt.Position Error")

    Set PositionStatusCursor = .GetCursor(PositionStatusHandle)
    Set DACOutCursor = .GetCursor(DACOutHandle)
    Set PositionDegreesCursor = .GetCursor(PositionDegreesHandle)
    Set ImplantPositionErrCursor = .GetCursor(ImplantPositionErrHandle)
    Set PositionCursor = .GetCursor(PositionHandle)
    Set PositionErrCursor = .GetCursor(PositionErrHandle)

End With

ReportDirectory = Mid(frmAnalysis.txtOutputFile.Text, 1, InStrRev(frmAnalysis.txtOutputFile.Text,
"\"))

With PositionCursor

    .record = CLng(frmAnalysis.txtStartRecord.Text) + 20000
    cycle_within = 0
    While (.State < 3) And (Not CancelReport) And (.record <
CLng(frmAnalysis.txtEndRecord.Text))
        If cycle_within > 49 Then
            frmAnalysis.txtOutputFile.Text = ReportDirectory & "Tilt DAC Metadata.csv"
            WriteReportHeader
            DoEvents
            For i = 1 To 50
                OutputFile.WriteLine i & "," & CycleTime(i) & "," & PeakLevel_DAC(i)
            Next i
            OutputFile.Close
        frmAnalysis.txtOutputFile.Text = ReportDirectory & "Tilt Position Metadata.csv"
        WriteReportHeader
        DoEvents
        For i = 1 To 50
            OutputFile.WriteLine i & "," & CycleTime(i) & "," & PeakLevel_Position(i)
        Next i
        OutputFile.Close

        Exit Sub
    Else
        cycle_within = cycle_within + 1
    End If
    While .value < 11000
        .MoveNextChange
        If .State > 2 Then
            DebugPrint "End of Archive reached looking for Position > 11000."
            OutputFile.Close
            Exit Sub
        End If
    Wend
    StartRecord = .record - 1
    While .value > 11000
        .MoveNextChange
        If .State > 2 Then
            DebugPrint "End of Archive reached looking for Position < 11000."
            OutputFile.Close
            Exit Sub
        End If
    Wend
    While .value < 11000
        .MoveNextChange
        If .State > 2 Then
            DebugPrint "End of Archive reached looking for Position > 11000."
            OutputFile.Close
            Exit Sub
        End If
    Wend
    EndRecord = .record - 1

    CycleTime(cycle_within) = EndRecord - StartRecord
    For i = StartRecord To EndRecord
```

```
            DACOutCursor.record = i
            ImplantPositionErrCursor.record = i
            PositionErrCursor.record = i
            PositionDegreesCursor.record = i
            PositionCursor.record = i
            If DACOutCursor.value > PeakLevel_DAC(cycle_within) Then
                PeakLevel_DAC(cycle_within) = DACOutCursor.value
            End If
            If PositionCursor.value > PeakLevel_Position(cycle_within) Then
                PeakLevel_Position(cycle_within) = PositionCursor.value
            End If
        Next i

        .MoveNextChange

    Wend

End With
End Sub
```