Project Number: MXC-I002 - 41

Incoming Inspection Database

An Interactive Qualifying Project Report

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

[illegible]

Tyler R. Kennedy

Date: April 30, 2001

Approved:

[illegible]

Professor Michael J. Ciaraldi, Advisor

ABSTRACT

An electronic database was designed and built to automate the Incoming Inspection documentation process at Vicor Corporation. Measurements before and after deployment show that the workers expanded their capabilities while performing the current workload with greater accuracy and significantly greater efficiency.

TABLE OF CONTENTS

Chapter 1 : Introduction

The purpose of the design of the Incoming Inspection database was to replace a manual documentation of an inspection process with an electronic database. Vicor Corporation, a manufacturer of component power supplies located in Andover, Massachusetts sponsored the project with the goal to eliminate much of the unnecessary and repetitive work of the manual documentation process. This would allow the workers to expand their capabilities while performing the current workload more efficiently and accurately.

The design of the Incoming Inspection database began with Vicor's vision and design requirements. Vicor envisioned a seamless interaction of the Incoming Inspection database with their preexisting databases. One of the important requirements of the Incoming Inspection database was that it allowed data communication with Vicor's other databases. The database was researched, defined between the client and engineers, and then created. The database was then tested for problems and to establish that the Incoming Database design objectives had met Vicor's specifications. Upon successfully completing the design objectives, the Incoming Database went online on January 2, 2001.

Chapter 2 : Background Information

Vicor's Incoming Inspection Department is responsible for inspecting incoming parts and supplies to make sure they meet specified quality levels. These parts and supplies are used during the manufacture of Vicor's component power supplies. Identifying defective parts before they are manufactured into a power supply is an important part of Vicor's quality control. To completely understand the design requirements for the Incoming Inspection database it is necessary to describe the inspection and documentation process.

2.1 Receipt and Inspection Process

When an order is placed by Vicor to a supplier for a new lot of parts, a unique receipt is created containing information particular to the lot. The information on the receipt is maintained electronically on an Oracle database. Usually, this information includes the receipt number, part number, PO number and the supplier. Upon arrival at Vicor, the receipt has been updated to include the quantity of parts within the lot, as well as the date the lot was received.

After a lot has been received, the lot, along with a copy of the receipt, is sent to Incoming Inspection to undergo an inspection. The set of criteria for the inspection varies between parts. Electrical parts (inductors, capacitors etc) require electrical tolerance testing. Mechanical parts such as base plates, may require dimensional testing, or even just a visual check to make sure the part is free of scuffs and burrs. An inspector references the part number of the lot to find the correct Inspection Directions. These directions list which tests must be performed before the lot is accepted and sent to the stockroom.

During the inspection process, the inspector records all test results and then catalogues the information. If the lot has passed the inspection process, it is sent to the stock room. If, however, the lot is rejected, a review board meets to determine the root cause and corrective actions necessary to fix the problem.

In summary, the process is rather simple: order the lot, receive the lot, inspect the lot, and either accept the lot or fix the defect. This summary, perhaps, oversimplifies the process. It can take many weeks to a year until the causal loop is complete. As time passed, Vicor's business grew, and so did the number of inspections. To further complicate the matter, Vicor began to produce a variety of products and consequently the types of parts inspected rose from the hundreds to the thousands.

To combat the increasing workload, new policies were instituted. If a part number passed five consecutive inspections then the part was qualified to bypass inspection and sent directly to the stock room. This is referred to as Doc to Stock. From then on, every 20th lot received would undergo inspection to assure that quality was being maintained. The inspection of every 20th lot is referred to as a Doc to Stock Audit.

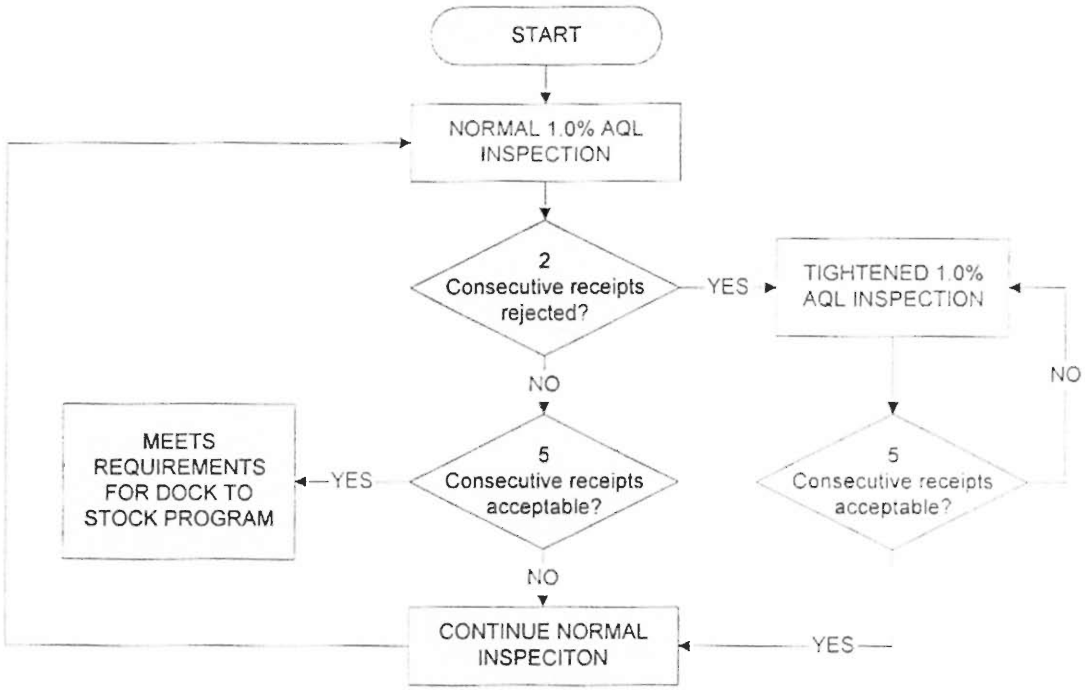

START
NORMAL 1.0% AQL INSPECTION
2 Consecutive receipts rejected?
YES
TIGHTENED 1.0% AQL INSPECTION
NO
NO
MEETS REQUIREMENTS FOR DOCK TO STOCK PROGRAM
YES
5 Consecutive receipts acceptable?
5 Consecutive receipts acceptable?
NO
CONTINUE NORMAL INSPECITON
YES

Figure 2.0.1: Vicor Inspection Plan

Figure 2.0.1 displays the Vicor inspection plan. Vicor inspects to a 1.0% acceptable quality level (AQL) which means they accept up to 1.0% rejects in a lot. Tightened Inspections are required after two consecutive receipts have been rejected. Tightened Inspections still inspect to a 1.0% AQL, but they require using a larger sample size for a more accurate inspection.

A new packaging and labeling policy was also created to help ease the workload on the Incoming Inspection Department. This policy forbade the use of messy packaging material, such as styrofoam 'peanuts', as well as required that the Vicor part number, receipt number and lot quantity be labeled on the inside and outside of the packaging. The new packaging policy was intended to eliminate time wasted determining the inspection criteria because the part number or receiver number were lost or not referenced. Unfortunately, this policy also required that each lot be inspected for the new packaging criteria. The new packaging requirements coupled with occasional/random quality defects, gave little chance for parts to pass five consecutive inspections. The ensuing increase in rejected lots quickly backlogged the review board.

2.2 The NCMR Database

In the summer of 1999, a commitment was made by the managers of the review board to create a rejected lot database, since the wealth of data given to them from Incoming Inspection was overwhelming. Four months later when the NCMR Database (Non-Conforming Material Report) went online, it was quickly discovered the over 80% of the rejected lots were repeat offenders and fully 40% of rejected lots were termed

Vicor Fault by the decisions of the review board. Vicor Fault rejections were usually caused by improper documentation or specifications. In actuality the parts perform as designed but the Vicor specifications were either not updated with later revisions of the part, or stated incorrect information initially.

To reduce the amount of rejected lots, Vicor quickly began to correct part specifications and drawings. This is not a quick process, by ISO design, and requires the approval of various engineers before the correction is authenticated. As specifications began to be corrected, a larger, more troubling trend developed. The instructions for lots were outdated or incorrect. This caused inspectors to perform incorrect or unnecessary tests that consequently resulted in unnecessary rejections.

To understand the scope of this problem, a more detailed description is needed. Similar parts are all grouped together to form Class Codes. For example, capacitors belong to one Class Code while resistors belong to another. Instructions are not written for each part, but rather for each Class Code. Therefore, the instructions for a Class Code are intended to apply to all the parts belonging to that Class Code. When a new part entered the system, an instruction card was created specifically for that part. However, there was no master list of Class Code instructions, so the criteria for the new instruction card was copied from a different part's instruction card that belonged to the same Class Code.

The result of this method for creating a new instruction card was that errors were copied from instruction card to instruction card. Worse, when the instructions needed to be updated there were hundreds of cards to update. It soon became apparent that instructions were not uniform per Class Code as designed.

To solve these problems, it was decided that a database be created to store the results of all inspections. The database would also be able to provide uniform instructions as well as quick links to technical drawings and specifications. The inspectors would be relieved of many tedious hours of copying and verifying data that the current system imposed.

2.3 Lessons Learned from the NCMR Database

The NCMR Database had taught Vicor many lessons in database design and implementation. One of the largest problems was not in fact related to database design at all. Many of the personnel that were intended to benefit from the NCMR database were not computer literate. In attempt to compensate for this, the design was made as simple and intuitive as possible. However, despite the design effort, simple computer literacy classes could have avoided many of the problems encountered during the first few months of use. Most of these problems were related to using improper logins when booting the computer. If the correct login was not used, or the user canceled the login process, the NCMR server would not allow access. Other problems related simply to the users inexperience with computers. These were relatively minor issues, such as the database 'disappearing' because the user accidentally minimized the program or because a second window had become the focus in a multitasking environment.

Data storage and recall methods were also improved upon with later revisions of the NCMR system. The NCMR system was developed and began operation in a very short period of time. The decision was made to go online despite missing several features. One of the most crucial features missing were Discrepancy Codes. They are a list of codes listing the cause of rejection. Discrepancy Codes were needed to provide keys for querying. These codes are necessary to guarantee a common reference for similar rejections. These codes were not developed in time, so the first month of data collected by the NCMR system is limited in its statistical use.

The NCMR system made liberal use of drop-down or combo boxes. The combo boxes allow the programmer to provide a list of choices for the user to choose. This is a method of controlling data input by using known variables. Since it was known that in some circumstances, the correct choice would not be listed, an unfortunate design decision was made to allow the user to select from the list or type in data. The result of this decision was an increase in information, which was too specific and could not be statistically quantified by the database. The goodhearted users typed in specific answers rather then selecting general categories. Unfortunately this ruins data integrity since data is being entered in a format not specified or known by the database. The situation was remedied by providing an "Other" selection in the list and removing the user privileges to type in answers. The users were then asked to select the best answer.[1]

The NCMR system also developed methods for interacting with Vicor's company database, which is an Oracle system. Write authorization to the Oracle database is very limited but read authorization is granted company wide. The NCMR system uses the receipt number, part number, PO number, supplier and other information that is all included on a lot receipt. Before the NCMR system, the inspectors copied this information from the lot receipt on to a written NCMR. When the NCMR database first went online, the inspectors typed this information into the system. However, it was

[1] The removal of the privileges led to heated arguments since the users felt they were losing authority. From a social standpoint it became obvious that it was much easier to add features then to remove them.

obvious that the Oracle database had all the required information since it prints the lot receipt. The only problem was to find the information in Oracle and communicate it with the NCMR system.

While this sounds rather simple, several months were needed for the systems to communicate properly. Even as this report is being written, more then a year since the NCMR system went online in November 1999, enhancements are still being made. The largest difficulty was returning the information faster than a person could type it in. Early attempts took over three minutes to return the data, and were plagued by system crashes if no data was returned. Worse, the information available from the Oracle database was not real time; instead only information up to the previous day was available. This was fine for 70% of the lots inspected; but roughly 30% were "hot" lots that are inspected the day they are received.

However, new methods of querying the Oracle database were found to improve performance. A local copy of the Oracle table in the NCMR server reduced performance delays to under ten seconds, but required the NCMR server to store much more temporary data. A script was written to update from the Oracle system every ten minutes while expunging records older then three months. Today, the interface with Oracle data is nearly seamless, and self-maintaining. The NCMR system can even operate when the Oracle system is down due to the local table copy.

2.4 The Incoming Database

Despite policy changes and the NCMR system, little had been done to improve the workload of the inspector. The majority of incoming lots are not defective, so no NCMR is ever required. However, the inspectors still test these lots and record the results. The workload required in writing an NCMR is simply reporting the results of a failed inspection. The NCMR system had provided management the tools to isolate the problem suppliers but reducing the amount of defective lots only reduces the inspector's workload by a little. There is also a large delay between when the problem is reported until it is fixed. Often the problem may last months, and in some cases over a year. All the meanwhile, the inspector is rejecting the same part again, for the same reason, again.

The managers realized that the inspectors' morale was low. They also knew that the reason for the low morale was the increasing workload, coupled with the seemingly pointless results from their efforts. The managers decided they could either hire more people to ease the workload, or increase the efficiency of the inspectors.

It was decided that another database, an Incoming Database, be created to capture the results from the lot inspections. This database would do a lot more, however. It would provide and manage Class Code instructions, provide a recent and detailed inspection history of any part, provide links for part technical drawings and specifications, calculate sample sizes and reject quantities as well as interface with the NCMR and Oracle databases.

2.5 Preliminary Design of the Incoming Database

The first several weeks of development in March 2000 were spent in meetings outlining the requirements for the NCMR database. The creation of the database allowed Vicor to reevaluate the entire inspection process. Since a new method of capturing the inspection results was under development, it only made sense to revise any processes or procedures related to incoming inspection. Simple revisions that perhaps would save only seconds per inspection would save minutes a day, and many hours a year.

The author is inexperienced in inspection procedure, which allowed him to have a unique perspective. This lack of experience was beneficial because the author was not susceptible to procedures that were 'grand fathered' into place. For example, for years the inspectors had written the sample size used for the inspection and then calculated the quantity passed *and* the quantity failed. Obviously, only one of these quantities is needed since the remaining quantity could be determined using the sample size. The initial instructions from Vicor were to replicate this procedure on the forms[2] of the NCMR database and to store this information in the tables. The author offered that the database calculate the quantity passed automatically based on the quantity failed and the sample size. It was during consideration of this offer that it was determined that the quantity passed was not needed to be stored at all since it could be quickly calculated if needed.

A large issue that required much discussion involved the recording of lot codes and date codes. When a lot is received it may have many different lot or date codes. If a lot is manufactured on two different manufacturing lines the parts are kept separate and assigned different lot codes. If the lots are manufactured on different dates, they are

[2] A form is the graphical interface the user interacts with in the database. The information entered into the form is stored in a database table.

given different date codes. The logic behind this is simple; if a problem with the product is limited to one date or manufacturing line, the bad product can be easily separated from the good product.

The issue involving lot codes or date codes is related to how Vicor inspects a lot with many sub-lots (differing lot codes or date codes). Traditionally, Vicor determines the sample size of the inspection based on the total lot size. The sample quantity is then chosen from the sub-lots based on the relative size of the sub-lot compared to the lot. For example, if a lot of 100 parts consists of two sub-lots, one of 60 parts, the other of 40 parts, then 60% of the sample will be taken from the larger lot and 40% from the smaller. Conversely an alternate inspection method for determining sample size would be to treat each sub-lot as a lot, and the sample size would be determined by the sub-lot's quantity.

Each method has serious problems. The traditional inspection method may allow bad sub-lots to reach manufacturing since the sample size is diluted with a mixture of all sub-lots. The proposed method would eliminate this problem but vastly increase the workload of the inspector. This is clear with a brief description of the sampling plan. The sampling plan informs the inspector of what size sample they should inspect for a given lot size. For a lot of 1000 parts, the sample size might be 100 parts. However, for a lot of 100 the sample size is a larger percent of the total, for example 50 parts. So for one lot of 1000, an inspector must inspect 100 parts but for a lot with 10 sub-lots of 100 parts each, the total inspection would be 500 parts. The debate between the two methods reduced to the simple question, is the increase in part quality worth the increase in work hours? It was finally decided that the database should have the flexibility for either method, so that Vicor could experiment to determine the best method.

Chapter 3 : Development of the Incoming Inspection Database

During the first several weeks of development, meetings were scheduled to discuss the design and implementation of the Incoming Database. While the Incoming Database simulates the original manual process, several issues needed addressing. One of the largest issues involved database security.

3.1 Database Security

A database is potentially much more susceptible to malicious and accidental damage then a manual system. It is much easier for thousands of records to be deleted in a database then to destroy files from a filing cabinet. Since the database is hosted on a private server, access is limited to authorized users. Additionally, only network and database administrators have authorization to delete files; the remaining users only have read/write authorization. All of Vicor's servers are backed up daily reducing potential database damage to the 24 hours of data entry following a backup.

During development of the NCMR database, three different views were actually created, one for managers, one for inspectors, and another for read only users. In the rush to develop the NCMR, system this solution was easier then designing and incorporating a security system to limit or grant privileges. The three views were very easy to create; basically the original view was copied three times and all linked to the same back end or source tables. Then each view was modified for the three levels of privileges; this was accomplished mainly be removing links to features rather than through any additional development or code.

The drawback of this method is the increased maintenance time. A change in functionality or simply aesthetics has to be replicated for each database. Since the

Incoming Database development timeline was not as rushed as the NCMR Database, it was decided to design the Incoming Database with security implementation from the beginning.

A total of four different levels of privileges were designed into the Incoming Database. The database does not require you to login unless performing a task where permission is required. Currently, the database is not accessible via the internet, which restricts access to Vicor employees only. The anonymous access is the most basic. It allows for any user to view (read only) previously entered data as well as run reports. The user may also query to view data by company, part number, date range etc… An inspector privilege allows for the creation of a new inspection receipt. This simply provides access for an inspector to enter the results from an inspection into the database. The edit privilege allows the user to edit a previously entered and saved record. The edit privilege does not include the privileges of the inspector privilege. This means that while an editor may change the data saved by an inspector, the editor does not have permission to create a new record. In practice, an editor usually has inspector privileges as well. Finally, the edit instruction privilege allows the user to edit the class code instructions used as the criteria for inspections.

The name of the user and the date is recorded with any use of a privilege. For an inspection receipt, the inspector and most recent editor (in the case of multiple edits) are recorded. For instructions, the name and date of all edits is recorded. This provides a record of changes if any investigation in the history of an inspection receipt or class code instruction is required. In addition, the edit of a class code requires a brief description of what was modified. The database does not, however, save the previous versions of class code instructions although future revisions of the database may include this feature.

When the Incoming Database is opened the user is first greeted by the Incoming Database switchboard. The switchboard is a custom menu that allows users to navigate through the various features of the database. The first level of the menu selects the inspector or manager utilities of the database (Figure 3.1). Selecting the manager utilities brings up the management menu and management features (Figure 3.2).

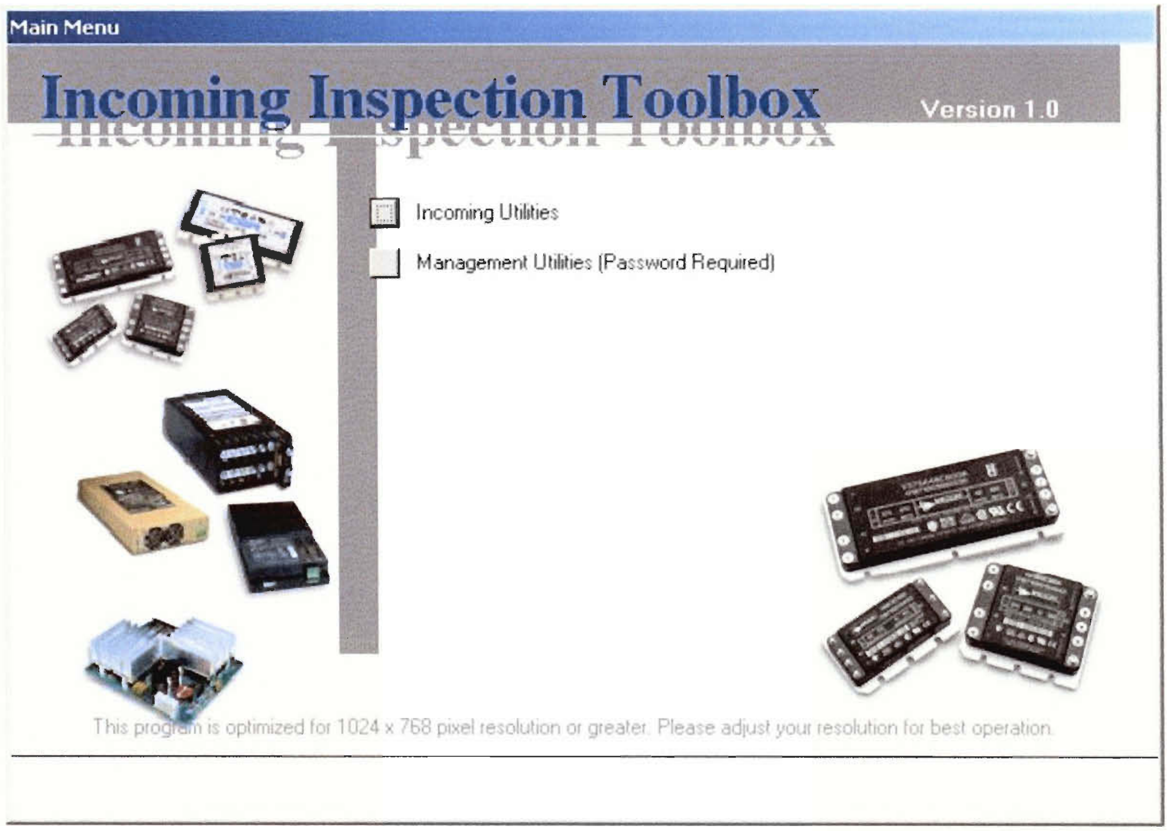

Main Menu
Incoming Inspection Toolbox
Version 1.0
Incoming Utilities
Management Utilities (Password Required)
This program is optimized for 1024 x 768 pixel resolution or greater. Please adjust your resolution for best operation.

Figure 3.1: Main Menu

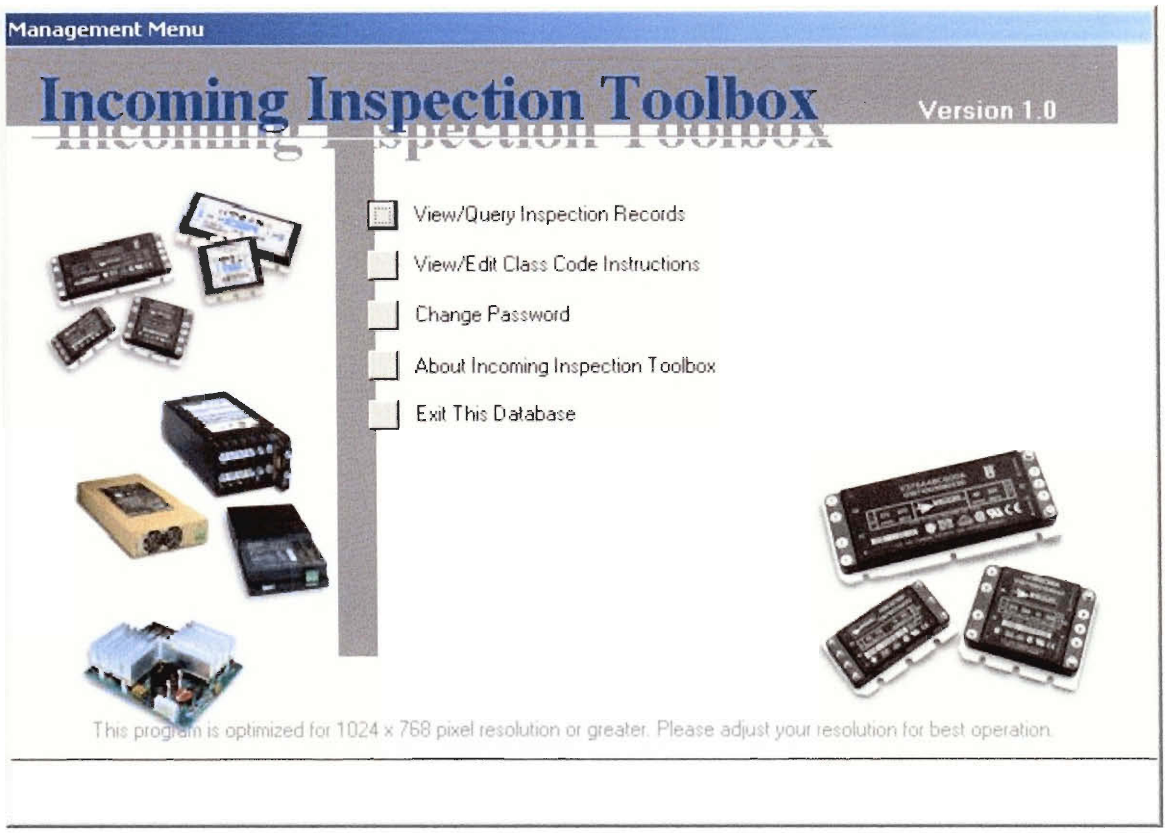
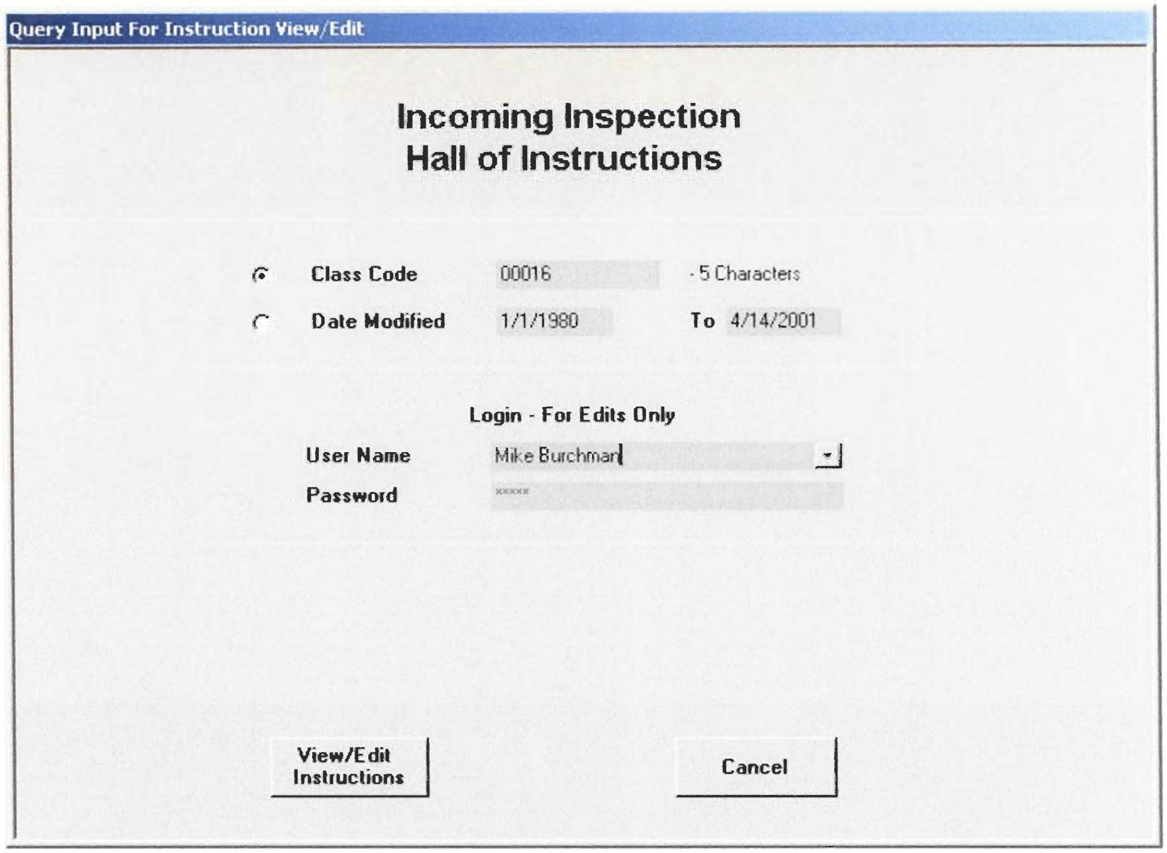

Management Menu
Incoming Inspection Toolbox
Version 1.0
View/Query Inspection Records
View/Edit Class Code Instructions
Change Password
About Incoming Inspection Toolbox
Exit This Database
This program is optimized for 1024 x 768 pixel resolution or greater. Please adjust your resolution for best operation.

Figure 3.2: Management Menu

3.2 Management Features

The management menu provides several features for database maintenance as well as some querying and reporting capabilities. The most powerful tool is the Class Code Instruction Editor. This editor allows authorized users to modify class code instructions that direct inspectors on how to perform inspections. The greatest improvement this feature provides over the previous manual methods is the centralizing and easy update of class code instructions. Before the Incoming Database, updating class code instructions was a daunting job, and there was no guarantee that all the instructions were updated. Now, the class codes provide real time instructions to users. Once a change is implemented in a class code, the new instructions are immediately available.

All users may view and print any class code instruction using a simple search feature. The editing of instructions, however, is password limited to a few authorized personnel.


Query Input For Instruction View/Edit

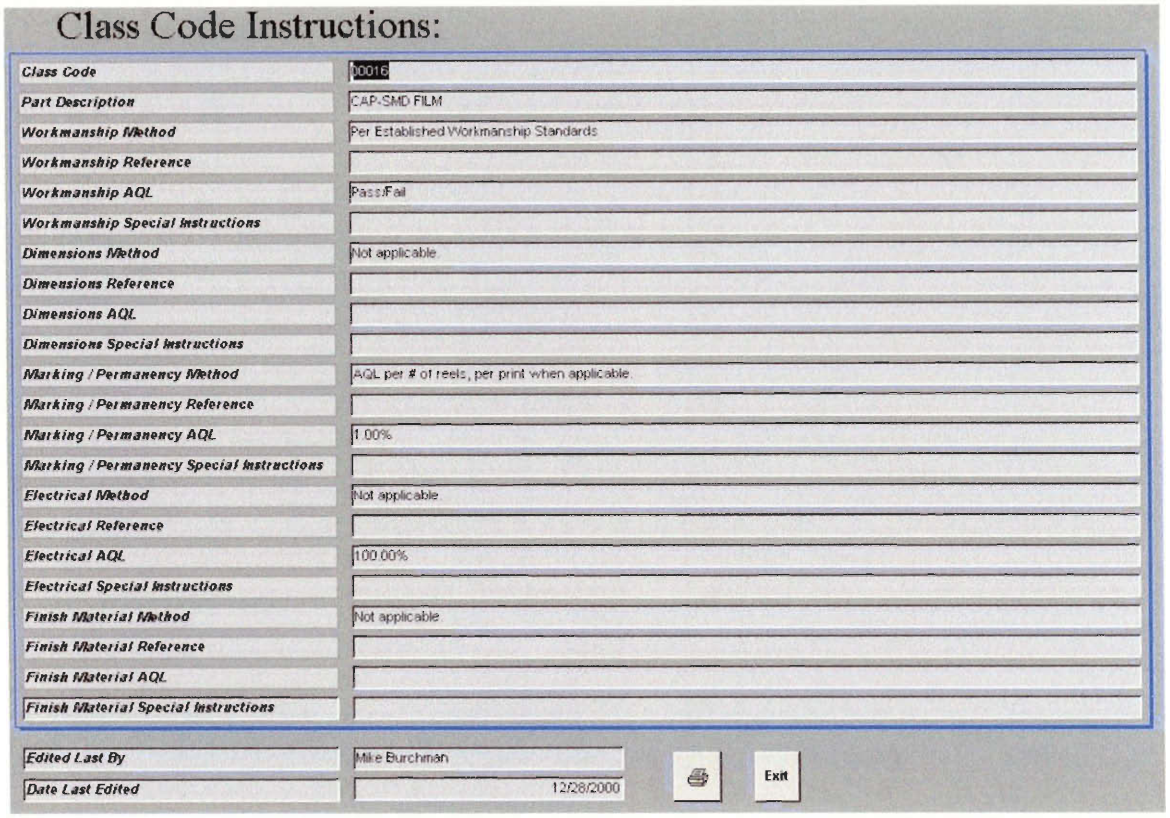
Incoming Inspection
Hall of Instructions

(●) Class Code: 00016 - 5 Characters
( ) Date Modified: 1/1/1980 To 4/14/2001

Login - For Edits Only
User Name: Mike Burchman
Password: *****

Figure 3.3: Class Code Instructions Query Screen

The screen shot in Figure 3.3 is the Class Code Instruction query screen. A user may search for Class Codes by the Class Code number, or by the date the Class Code was last modified. The login is required only if an edit is to be performed. If no login is entered, the Class Code Instruction screen is opened in read only mode (See Figure 3.4).



Class Code Instructions:

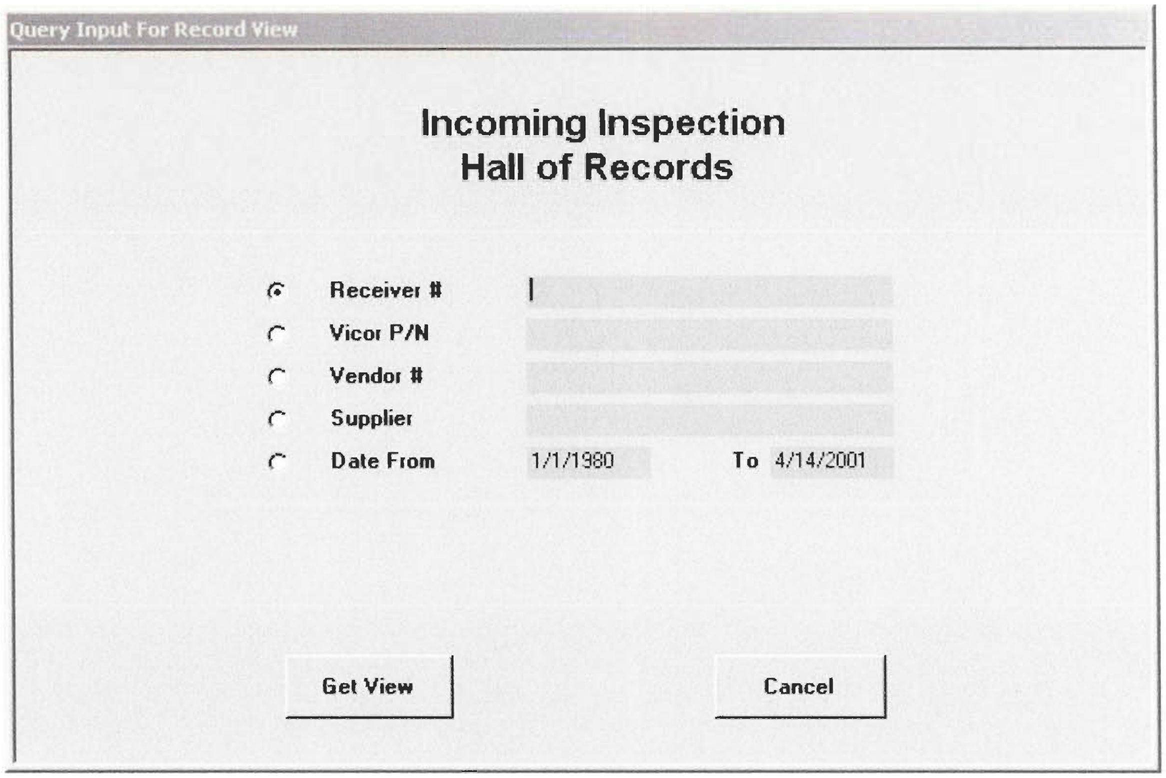
Class Code	00016
Part Description	CAP-SMD FILM
Workmanship Method	Per Established Workmanship Standards
Workmanship Reference	
Workmanship AQL	Pass/Fail
Workmanship Special Instructions	
Dimensions Method	Not applicable
Dimensions Reference	
Dimensions AQL	
Dimensions Special Instructions	
Marking / Permanency Method	AQL per # of reels, per print when applicable
Marking / Permanency Reference	
Marking / Permanency AQL	1.00%
Marking / Permanency Special Instructions	
Electrical Method	Not applicable
Electrical Reference	
Electrical AQL	100.00%
Electrical Special Instructions	
Finish Material Method	Not applicable
Finish Material Reference	
Finish Material AQL	
Finish Material Special Instructions	
Edited Last By	Mike Burchman
Date Last Edited	12/28/2000

Figure 3.4: Class Code Instruction Edit/View Screen

Figure 3.4 displays a sample Class Code instruction. Some of the criteria are not applicable to the inspection. In this example the dimensional, electrical, and finish criteria are not used during inspection. Also notice at the bottom of Figure 3.4, the name and date of the last editor is recorded.

The remaining portion of the management menu is dedicated to the querying and reporting of data. As of the time of this writing, only a few reports had been created mostly relating to the weekly performance of the Incoming Inspection Department. These reports return the total throughput of lots, the average time per inspection, and the total rejections. These reports recreate existing reports that used to be created through a manual process of gathering data. Examples of some of the reports created by the Incoming Database are shown and discussed in detail later.

The number of reports will undoubtedly grow if the NCMR system is used as a reference. After a six-month period of adjustment the users of the NCMR database became comfortable with the program and became aware of its potential. The amount of reports required by the managers and review board grew into the dozens. The reports are so numerous, that a new view is being developed for the NCMR Database just for reports.

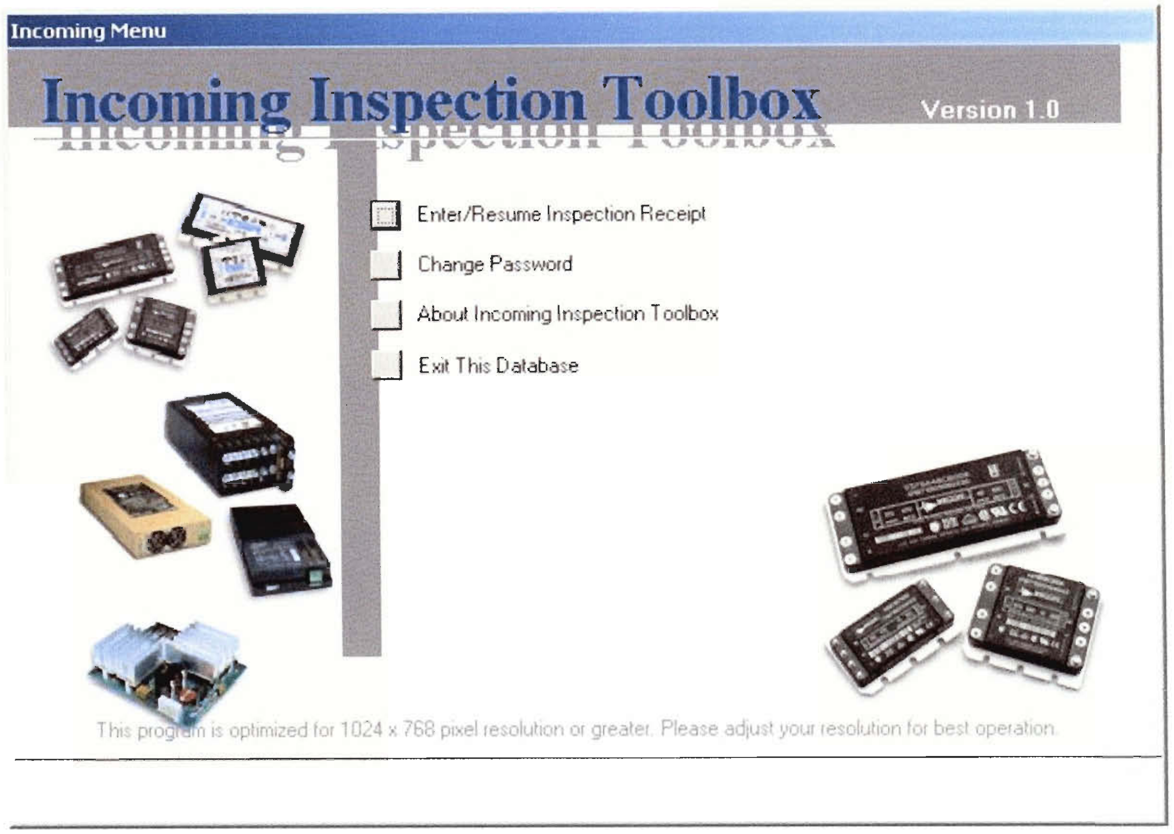

Query Input For Record View

Incoming Inspection
Hall of Records

[✓]	Receiver #			
	Vicor P/N			
	Vendor #			
	Supplier			
	Date From	1/1/1980	To	4/14/2001

Figure 3.5: Incoming Inspection Records Query Screen

Another useful application within the management directory is the records query screen as seen in Figure 3.5. This utility allows users to search for records using a variety of different criteria. The user may query by receiver number, Vicor part number, vendor number, supplier name, or by a date range. The date range works cooperatively with other criteria. This allows the user to search for records by Vicor part number within a certain time period for example.


Incoming Menu
Incoming Inspection Toolbox
Version 1.0
Enter/Resume Inspection Receipt
Change Password
About Incoming Inspection Toolbox
Exit This Database

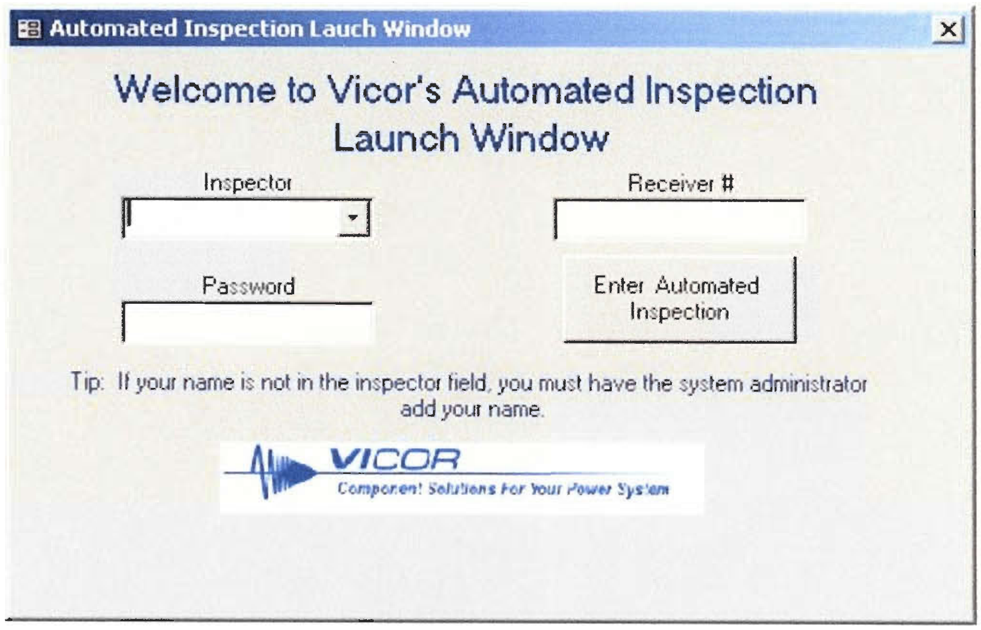
This program is optimized for 1024 x 768 pixel resolution or greater. Please adjust your resolution for best operation.

Figure 3.6: Inspector's Utilities Menu

3.3 Inspector Features

The inspector's utilities of the Incoming Database Switchboard are much more limited then the manager utilities. Inspectors have only two options, create or resume an inspection, or change their password. Selecting to change their password brings the inspector to a new screen where they can select the new password of their choice. The password is necessary to verify authorization during the creation of an inspection record. If the inspector chooses to create or resume an inspection, he is brought to the login screen (Figure 3.7).

To begin the process of creating a new inspection record, the user must enter his/her name, password and the receiver number from the lot receipt. The database first checks the name and password for proper authorization. If the user has inspector permission the database then searches past records to see if the receiver number has been entered before. If the receiver number has been entered before, the database returns who entered the receipt. If, the inspector happens to have edit privileges, the database will prompt them if they wish to edit the record else the database returns the user to the login screen. Since the receiver number is unique, this guarantees that duplicate records won't be entered into the database even if multiple users are logged in at once.


Automated Inspection Lauch Window
Inspector
Receiver #
Password
Enter Automated Inspection
VICOR
Component Solutions For Your Power System

Welcome to Vicor's Automated Inspection Launch Window

Tip: If your name is not in the inspector field, you must have the system administrator add your name.

Figure 3.7: Login Screen

If the receiver number is a new record, the database will begin to search the oracle tables for the corresponding lot information. If no receiver number is found, the user is notified that the receiver number they entered is incorrect and the program returns to the login screen. Finally, if the receiver number is found, the program opens to the inspection screen.

The database performs many operations during this time. First it gathers all the necessary information form the Oracle tables and inputs it into the newly created record. Using this information, the database returns the 15 most recent inspections of the same part number and company for the inspector to reference. The program references the part number with a look up table to determine the class code and then returns the appropriate inspection instructions. The database also creates the hyperlink to the part technical drawings. Finally, the database searches past archives for previous inspection comments.

In the developmental stages of the database, it was thought that the inspector would only create a record after the inspection had been performed. However, it was realized that the information that was necessary for the inspection was only available after the record had been created. For example, providing the inspectors with instructions, technical drawing, past inspection results and comments are all necessary and important tools for an inspection. While this seems relatively obvious, this design flaw was discovered only after operational tests had begun.

The solution to this problem was to allow the inspectors to create, save and resume their work. This solution was initially avoided because the NCMR system had been plagued early on with half complete and duplicate records. However, since the database only allows for a receiver number to be saved once, duplication was not a problem. Designing the database to allow the inspector to create a record before the inspection also created opportunities to increase the quality and efficiency of inspections. If the inspector enters in the lot code, date code, and lot quantity, the database will print a custom inspection sheet complete with instructions, sample sizes and the maximum rejection quantity as seen in Appendix B.

3.3.1 Line Items

The lot code, date code and lot quantity are entered as a line item. The line item also includes the sample type, results of the inspections, and sample quantity. There are two different sampling plans currently in use by Vicor, single and double sampling. Vicor inspects to a 1.0% AQL or acceptable quality level, meaning that Vicor accepts 1.0% rejects from their suppliers. Both sampling plans inspect to this level. The double sampling plan is more thorough as it may require inspecting an additional sample if the results from the first inspection were within a borderline pass/fail range. In addition each sampling plan also allows for tightened inspection. This is a more stringent inspection that increases the sample size. In total, an inspector may choose between four sampling types, Single Normal, Single Tightened, Double Normal, and Double Tightened. This may be better illustrated by examining the table displaying the various characteristics of each inspection.

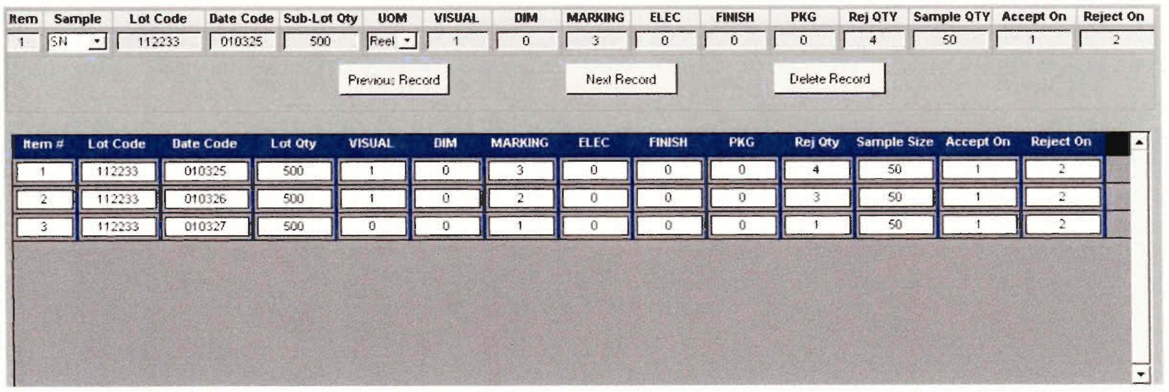
Table 3.1: Inspection Types

Inspection Type	Lot Lower Limit	Lot Upper Limit	Sample Size	Accept Level	Reject Level	Second Inspection Sample	Second Inspection Accept	Second Inspection Reject
Single Normal	151	500	50	1	2	N/A	N/A	N/A
Single Tightened	151	500	80	1	2	N/A	N/A	N/A
Double Normal	151	500	32	0	2	64	1	2
Double Tightened	151	500	50	0	2	100	1	2

Table 3.1 displays the inspection instructions for all four inspections types for lot size of between 151 to 500 pieces. The sample size is the amount of pieces the inspector inspects from the lot. Accept is the maximum allowed pieces rejected from the sample

size for the lot to still pass. Reject is the minimum amount of pieces rejected from the sample size for the lot to be rejected. Notice that for the two double sampling types, it is possible to have a rejection quantity that is above the accept level but below the reject level. If this occurs, the inspector performs a second inspection using the second inspection information from the table. It is important to realize the second inspection includes the results from the first inspection, so it is a cumulative inspection. For example, an inspector using the double normal sampling type finds one reject in his 32-piece sample. This rejection quantity is above the accept level of zero pieces, but below the rejection level of two pieces, so the inspector performs a second inspection. The second inspection consists of another sample size of 32 pieces for a cumulative sample size of 64 pieces. If the inspector finds no rejects, the cumulative rejection quantity will be one; below the second inspection accept level and the lot will pass. If the inspector finds any rejects in the second 32-piece sample, the cumulative rejection quantity will be greater or equal to two; which is above or equal to the second inspection reject level and consequently, the lot will fail.

The single sampling types require one line item per unique lot or date code, while double sampling may require two. The database automatically numbers the line items for the inspector. Figure 3.8 displays an example of three line items using single normal inspection.



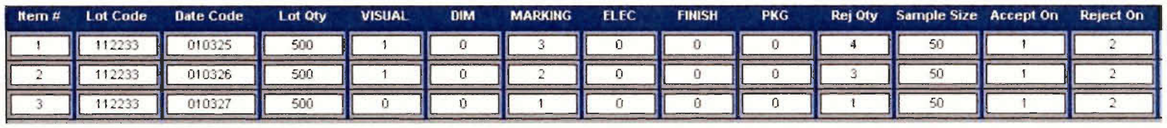
Item	Sample	Lot Code	Date Code	Sub-Lot Qty	UOM	VISUAL	DIM	MARKING	ELEC	FINISH	PKG	Rej QTY	Sample QTY	Accept On	Reject On
1	SN	112233	010325	500	Reel	1	0	3	0	0	0	4	50	1	2

Item #	Lot Code	Date Code	Lot Qty	VISUAL	DIM	MARKING	ELEC	FINISH	PKG	Rej Qty	Sample Size	Accept On	Reject On
1	112233	010325	500	1	0	3	0	0	0	4	50	1	2
2	112233	010326	500	1	0	2	0	0	0	3	50	1	2
3	112233	010327	500	0	0	1	0	0	0	1	50	1	2

Figure 3.8: Inspector Line Items

In the screenshot above, three different line items have been entered. Notice that each line item has the same lot code but differing date code; this is a requirement for a line item. The row at the top of the screen shot displays the line item currently selected for changes or additions, in this example it is line item one. The inspector is required to select the sample type, date code, lot code and enter the sub lot quantity. Notice that the sample quantity and accept/reject numbers have already been automatically calculated and are visible as the far right columns of the screen shot. The reject quantity is the sum of all rejects entered under the criteria settings (Visual, Dimensional, Electrical, Finish, and Packaging). Here, the reject quantity is four from the sum of three failures under Marking and one failure from Visual.

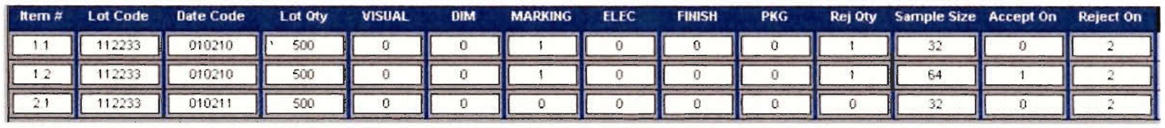
The auto numbering code for the line items was particularly difficult to develop. Two different standards were developed for use with the two different types of sampling plans, single and double sampling. The single sampling plan has a normal numbering scheme where one line item corresponds to a unique lot or date code. The single sampling plan uses a unique integer value to number the line items, 1, 2, 3, etc…(See Figure 3.9).



Item #	Lot Code	Date Code	Lot Qty	VISUAL	DIM	MARKING	ELEC	FINISH	PKG	Rej Qty	Sample Size	Accept On	Reject On
1	112233	010325	500	1	0	3	0	0	0	4	50	1	2
2	112233	010326	500	1	0	2	0	0	0	3	50	1	2
3	112233	010327	500	0	0	1	0	0	0	1	50	1	2

Figure 3.9: Line Item Example using the Single Sampling Plan

The double sampling plan makes allowance for a possible two line items per unique lot or date code, although only one line item may be needed. To signify that the two line items refer to the same lot or date code, decimals were used. The number to the left of the decimal refer to the date or lot code, the number to the right whether it was the first or second inspection in the double inspection. In Figure 3.10 a receipt is numbered like this: 1.1, 1.2, 2.1. The 1. refers to the first unique lot or date code entered and the .1 or .2 refers to the first or second inspection respectively. The 2.1 signifies the second unique lot or date code, only the first inspection was needed thus there was no need for line item 2.2 and hence it is absent in the example below.



Item #	Lot Code	Date Code	Lot Qty	VISUAL	DIM	MARKING	ELEC	FINISH	PKG	Rej Qty	Sample Size	Accept On	Reject On
1.1	112233	010210	500	0	0	1	0	0	0	1	32	0	2
1.2	112233	010210	500	0	0	1	0	0	0	1	64	1	2
2.1	112233	010211	500	0	0	0	0	0	0	0	32	0	2

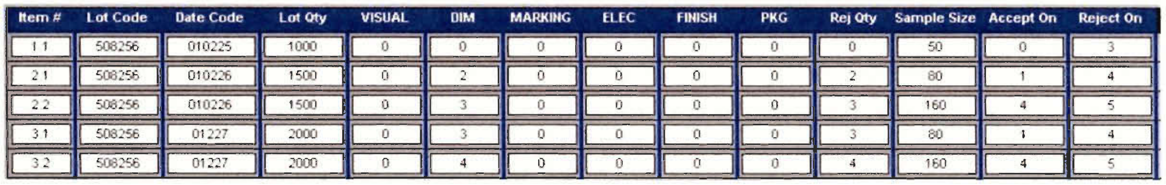
Figure 3.10: Line Item Example using the Double Sampling Plan

To implement the two different auto-numbering schemes, the program detects which sampling type the user is using. The sampling type is also necessary to calculate the sample quantity as well as the accept/reject quantities. The user selects from four different categories, SN, ST, DN, DT for Single Normal, Single Tightened, Double Normal, Double Tightened. The user must select the sample type before he saves the first line item since a user is not allowed to change this information later. If a user selects DN or DT, the line item is renumbered from 1 to 1.1. When he saves the first line item, 1.2 is automatically created and 2.1 becomes the next available item number. In beta versions, the database would prompt the user to create the second inspection (n.2) however it was easier and quicker to create the line item automatically and delete it later if it was unnecessary.

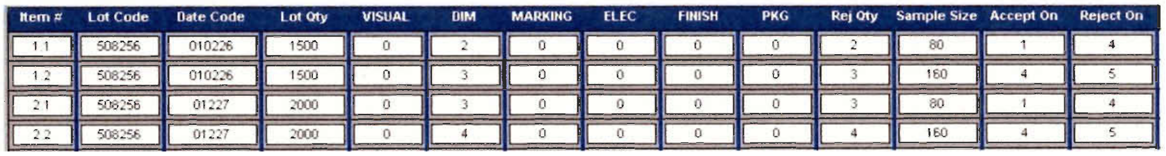
When a line item is deleted, the program must also determine which, if any, other item numbers should be renumbered. Suppose, for example, a normal sample type receipt was created with line items 1,2,3,4 and 2 was deleted. The program would first delete the second line item and then renumber the third line item 2, and the fourth line item 3.

For a double inspection, this is a much more complex operation. If the line item is the second inspection, n.2, then only that record should be deleted. If the line item is the first inspection, n.1 the program must check if there is second inspection, n.2. If n.2 exists the code deletes n.1 and renumbers n.2 as n.1. If n.2 does not exist then and item number greater then n.1 is reduced by 1. For example a double inspection receipt has 3 different lot or date codes but 5 line items, 1.1, 2.1, 2.2, 3.1, 3.2. If 1.1 is deleted, 2.1 is renumbered 1.1, 2.2 becomes 1.2, 3.1 becomes 2.1 and 3.2 becomes 2.2.

Figure 3.11: Double Inspection Record Before Delete



Item #	Lot Code	Date Code	Lot Qty	VISUAL	DIM	MARKING	ELEC	FINISH	PKG	Rej Qty	Sample Size	Accept On	Reject On
1.1	508256	010225	1000	0	0	0	0	0	0	0	50	0	3
2.1	508256	010226	1500	0	2	0	0	0	0	2	80	1	4
2.2	508256	010226	1500	0	3	0	0	0	0	3	160	4	5
3.1	508256	01227	2000	0	3	0	0	0	0	3	80	1	4
3.2	508256	01227	2000	0	4	0	0	0	0	4	160	4	5



Item #	Lot Code	Date Code	Lot Qty	VISUAL	DIM	MARKING	ELEC	FINISH	PKG	Rej Qty	Sample Size	Accept On	Reject On
1.1	508256	010226	1500	0	2	0	0	0	0	2	80	1	4
1.2	508256	010226	1500	0	3	0	0	0	0	3	160	4	5
2.1	508256	01227	2000	0	3	0	0	0	0	3	80	1	4
2.2	508256	01227	2000	0	4	0	0	0	0	4	160	4	5

Figure 3.12: Double Inspection Record Following Delete

When the results of the inspection have been entered into the various line items, the database adds the total rejected quantity and indicates to the user if the lot has passed. Before the record may be saved the program prompts the inspector for the time spent during inspection and, if necessary, the NCMR number. Future enhancement for the Incoming Database includes plans to automatically link with the NCMR system to create an NCMR, when necessary and return the NCMR number to the Incoming Database. As of the time of this report, however, the inspectors must create an NCMR and manually enter the number into the Incoming Database.

3.3.2 Saving and Deleting Records

To return to the database switchboard, the inspector has three options. First the inspector may choose to 'Save and Exit.' The inspector chooses this option when the record is complete and ready to be made official. The inspector will be unable to further edit or add to the record unless they have an edit privilege. The inspector is asked at this time to enter the total time taken inspecting and documenting the lot for performance tracking. If the inspector is not finished with the inspection and wishes to continue the record at a later time, he may choose 'Exit with Resume.' The exit with resume option allows the inspector to return to the record at a future time as well as print instructions and inspection worksheets if needed. Only the inspector who created the record or a user with an edit privilege is allowed to resume the record. The final option is 'Exit without Saving' which will delete the record and return the user to the switchboard.

3.4 Incoming Inspection Reports

One final aspect to the development of the NCMR database was the creation of automated reports and charts. The Incoming Inspection Department had previously created these reports on a weekly basis to track the performance of suppliers and the Incoming Inspection Department. At the end of each quarter, a more detailed report is created explaining trends in supplier and department performance as well as estimates for future performance. Obviously, some of the trend analysis required for the reports is beyond the capabilities of the Incoming Database, but the Incoming Database can be used to create many of the weekly and quarterly charts needed in the reports.

To create these charts, the structure of the manual charts were studied and duplicated by the Incoming Database. Some of the charts were impossible to create because they require information that the Incoming Database does not possess. Other reports had to be redefined to be replicated by the database. The Chart in Figure 3.13 for example, tracks inspector time per lot inspected. This provides an average time for the inspection and documentation of a lot. The chart was created manually by dividing the total work hours of all inspectors by the number of lots inspected. The Incoming Database does not know the total work hours of the inspectors, so the data cannot be calculated in the same manner. Fortunately, the database can calculate the chart based upon the time the inspector reported when he saved the inspection record. Each method has some flaws as to the accuracy of the chart. The manual method includes hours that inspectors may be performing some other tasks as time inspecting lots. The database is dependent on accurate times being reported by the inspectors. Fortunately, the database charts show little deviation from the manual charts with the average time around an hour per lot.

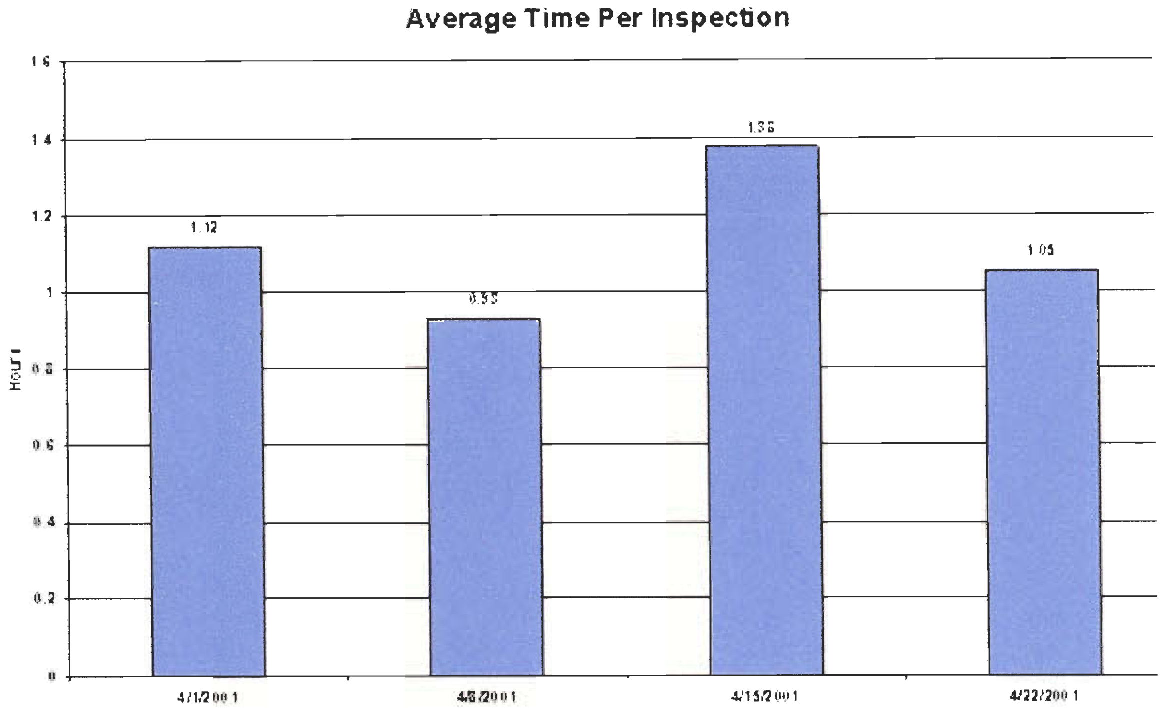


Figure 3.13: Average Time per Inspection

The remaining charts were much easier to duplicate since the Incoming Database contains all the pertinent information to calculate the charts. Some of the charts were combined since the information was more useful in conjunction with other data then by themselves. Figure 3.14 shows the conjunction of the total lots inspected per week with the total hours per week. While one can get an estimate for the inspector time per lots inspected chart discussed earlier, this chart provides information on the workload of the

Incoming Inspection Department. Increases or reductions in total lots inspected will be reflected in this chart.

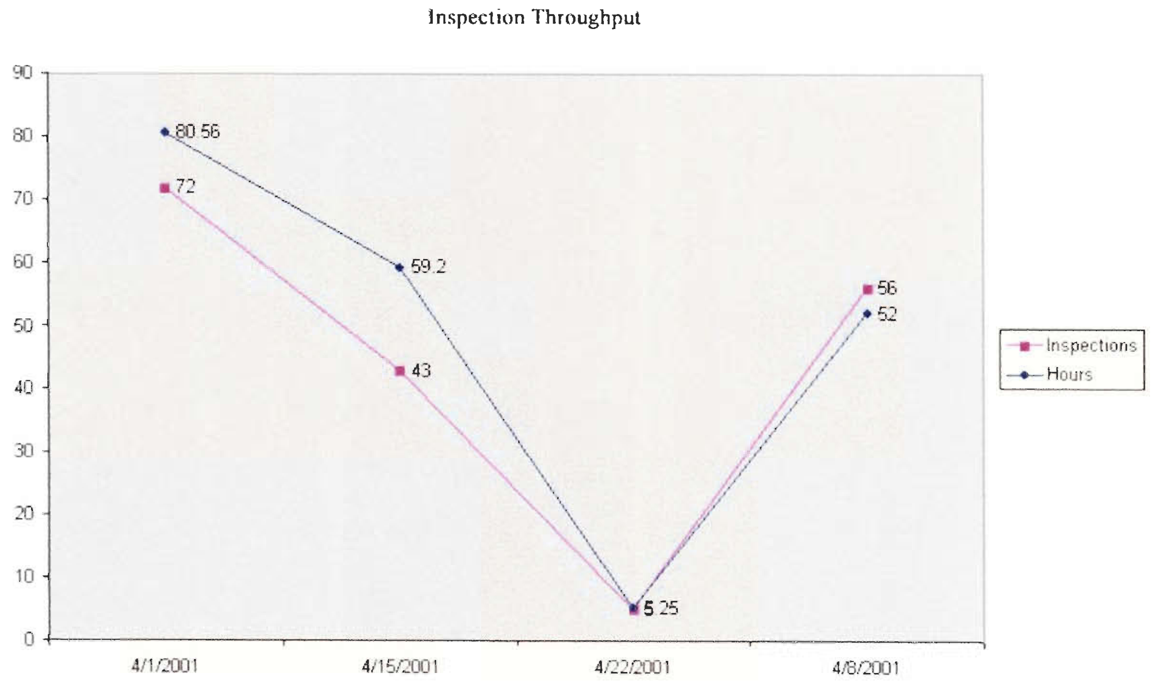


Figure 3.14:

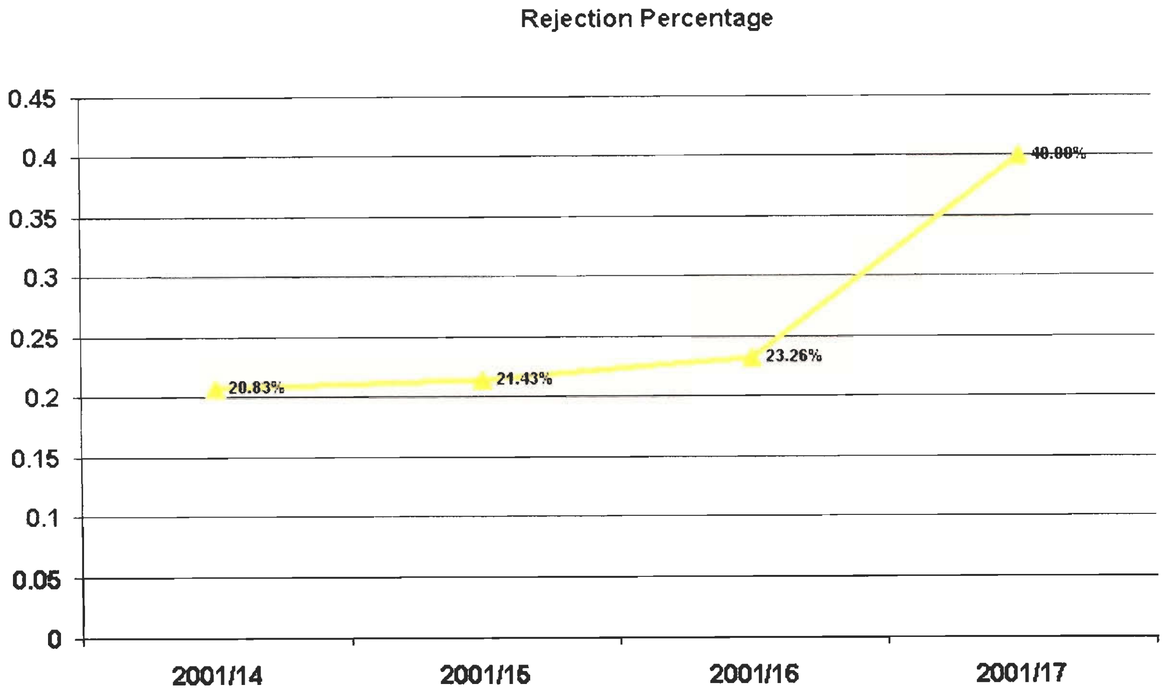


Figure 3.15: Rejection Percentage

The chart shown in Figure 3.14 does not portray the total Incoming Inspection Department since it does not mention the rejection of lots, which require additional work. The rejection of lots is the focus of the next two charts seen in Figure 3.15 and Figure 3.16. Figure 3.15 displays the rejection percentage, or the probability of a lot rejection per random lot inspected. Figure 3.16 displays the total rejections by week contrast to the total inspections. This was another chart that was modified from the manual original in providing the total inspections. It was felt that by adding the total inspections a better sense of scale would be provided in evaluating the rejection data.

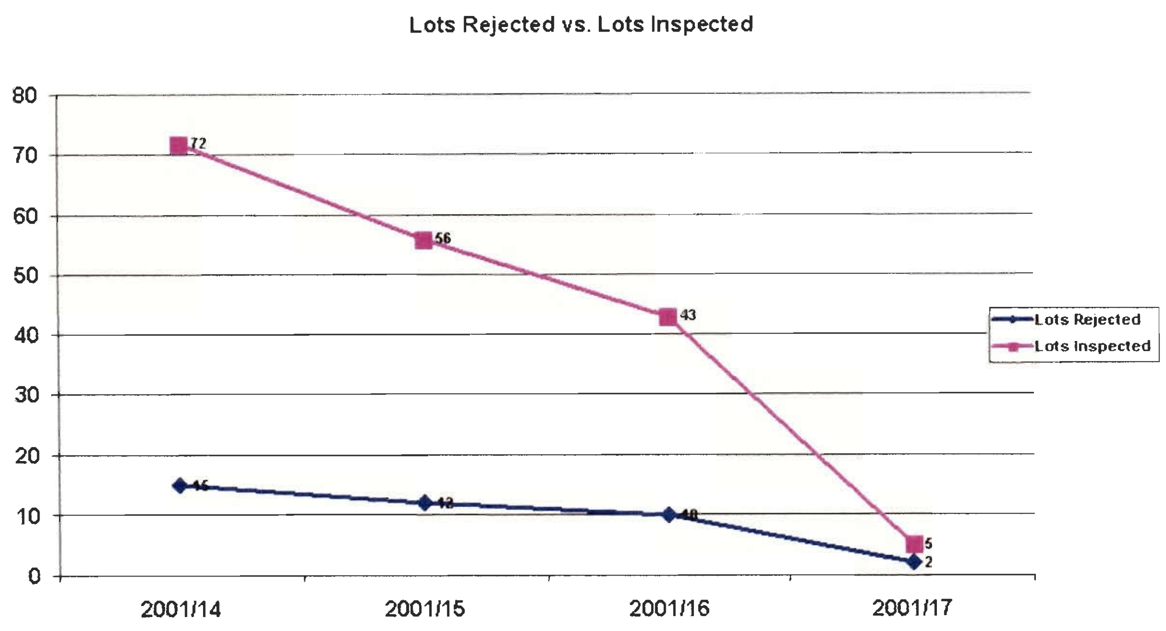
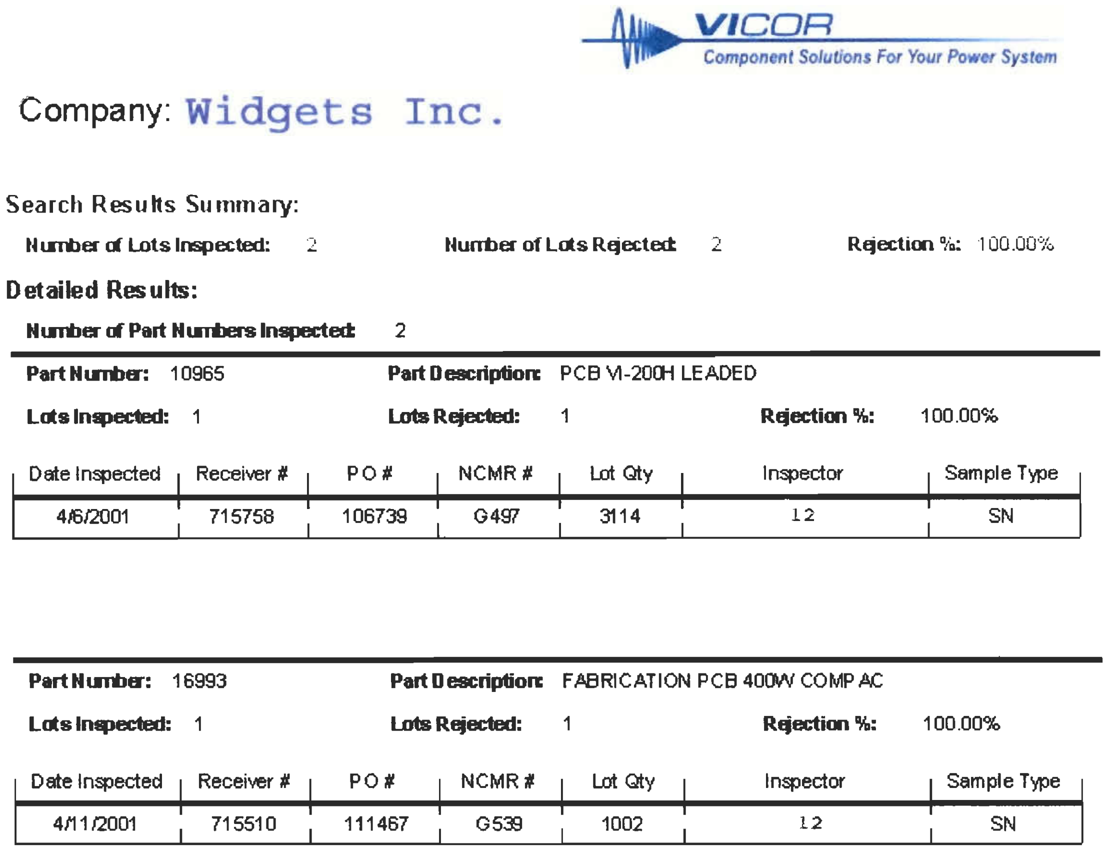


Figure 3.16

Some remaining charts could not be created with the Incoming Database. These charts break down rejected lots into percentage groups and require information from the NCMR system. The information could be easily linked from the NCMR system, however Vicor wanted to wait until they were confidant that undiscovered bugs in the Incoming Database would not corrupt the NCMR Database.

One final report created for the Incoming Inspection Database provides a supplier history performance. This report provides a supplier summary then breaks down the performance into reports on the various part numbers the supplier provides. Finally, a listing of the rejected lots with the inspection results is provided. This report is an excellent tool to evaluate supplier performance since you can easily determine overall supplier problems as well as issues only affecting a limited amount of part numbers. A fictional example is provided in Figure 3.17.



Company: Widgets Inc.

Search Results Summary:

Number of Lots Inspected: 2 **Number of Lots Rejected:** 2 **Rejection %:** 100.00%

Detailed Results:

Number of Part Numbers Inspected: 2

Part Number: 10965 **Part Description:** PCB VI-200H LEADED

Lots Inspected: 1 **Lots Rejected:** 1 **Rejection %:** 100.00%

Date Inspected	Receiver #	PO #	NCMR #	Lot Qty	Inspector	Sample Type
4/6/2001	715758	106739	G497	3114	12	SN

Part Number: 16993 **Part Description:** FABRICATION PCB 400W COMP AC

Lots Inspected: 1 **Lots Rejected:** 1 **Rejection %:** 100.00%

Date Inspected	Receiver #	PO #	NCMR #	Lot Qty	Inspector	Sample Type
4/11/2001	715510	111467	G539	1002	12	SN

Figure 3.17: Suppler History Report

As can be seen in Figure 3.17, the supplier history report provides a great deal of information. In this example, Widgets Inc. has had a total of two lots inspected, both of which failed. Furthermore, the report indicates the Widgets Inc. supplies two different part numbers. In this example, one of each part number supplied has been rejected. The header information provides information about Widgets Inc. overall performance, while the detailed results display information on the part number and inspections.

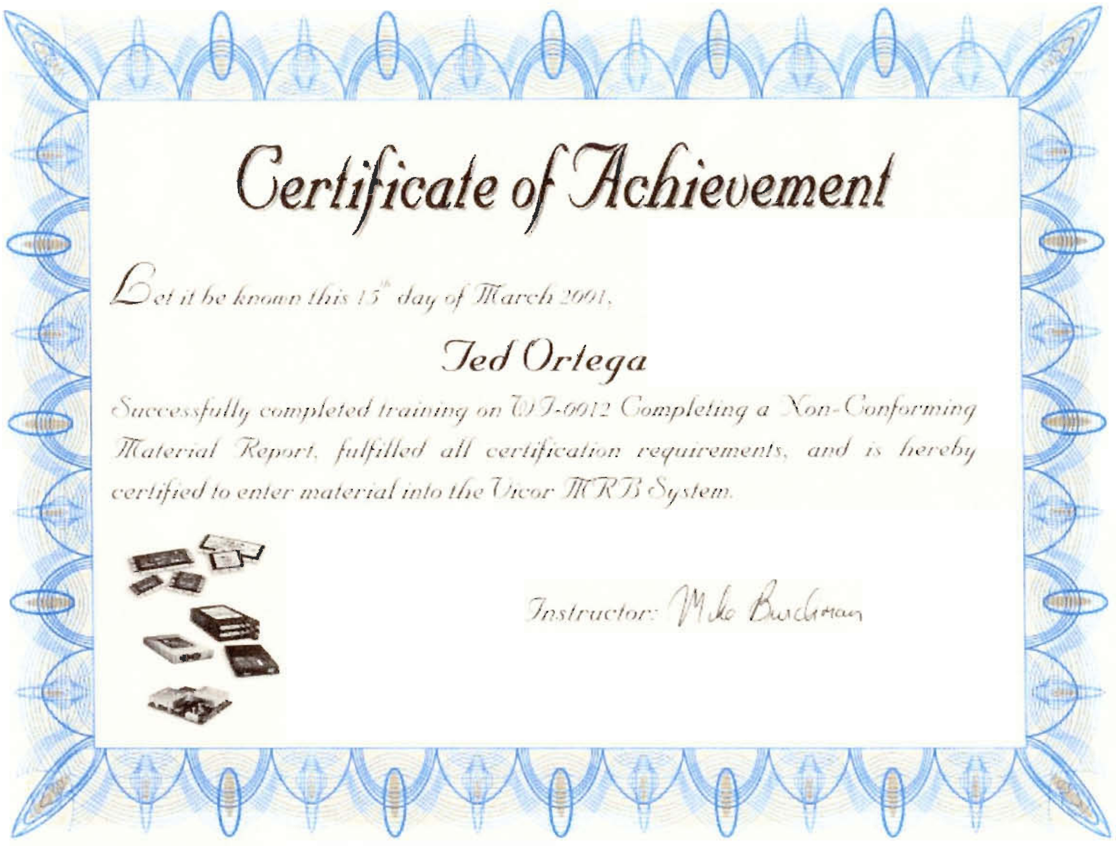
3.5 Beta Testing

The Incoming Database began beta testing in the fourth quarter of 2000. Certain inspectors were chosen specifically because of their help resolving difficulties with the NCMR database to test the database for any bugs or data corruption. Fortunately, there were no major problems discovered, and most of the changes were of an aesthetic nature. The inspectors also made suggestions in several areas for the use of more appropriate vocabulary.

The Incoming Inspection began use by the full Incoming Inspection Department staff on January 2, 2001. At the time of this report, the inspectors were recording data using the Incoming Database as well as recording the data in the traditional manner by hand. This was a decision by Vicor to test the database for at least six months before abandoning the manual documentations. This is an appropriate, cautious decision. If some flaw in the database corrupts data, Vicor would not lose the manual records. Records from the NCMR Database were printed and stored for over a year until Vicor was comfortable with an electronic, “paperless” system.

Overall, the implementation of the Incoming Database has been a great success. To date, there have been no major problems or any data loss or corruption issues. Any problems encountered have mostly been due to an operator’s unfamiliarity with the Incoming Database. To correct these issues, Vicor has launched an official training program where users are certified to use the NCMR and Incoming Database. At the time of this report, only the NCMR Database classes had begun. The Incoming Database classes are scheduled to begin in May 2001, following the completion of the NCMR

Database classes. Seen in Figure 3.18 is an example of the NCMR certification the users receive upon successfully completing the training classes.



Certificate of Achievement

Let it be known this 15th day of March 2001,

Ted Ortega

Successfully completed training on WI-0012 Completing a Non-Conforming Material Report, fulfilled all certification requirements, and is hereby certified to enter material into the Vicor MRB System.

Instructor: [illegible]

Figure 3.18: NCMR Certification

Chapter 4 : Results

The introduction of the Incoming Inspection Database was a great success. The lack of major problems shows the importance of design methodology and proper testing. The users previous experience with the NCMR Database has reduced the amount of user errors when compared to the introduction of the NCMR Database. To further reduce the amount of user errors Vicor has embarked on training programs to certify each user as qualified to use the NCMR Database.

4.1 Incoming Inspection Database Performance

Performance was a key issue in the development of the Incoming Inspection Database. Vicor wanted to provide the database as a tool to make the work of an inspector easier and quicker. Measures of performance were tracked and reported with the old manual documentation methods. Vicor was hopeful and eager to see the performance comparisons to the Incoming Inspection Database.

Figure 4.1 shows the progression of time spent per inspection record from January to part of April. These numbers are not how long it took to complete an inspection, but the time needed to record the results. The manual results were taken from the fourth quarter 2000 with an average of 10.2 minutes per record. This includes retrieving and filing the record as well as the data input. The predicted value is based upon the proposal to Vicor management on the expected performance of the inspectors with the Incoming Database. This proposal was in the spring of 2000 and performance estimates were based upon inspector performance in the NCMR Database. The database value measures how long the record was open before saving. This may be a misleading number since undoubtedly some inspectors left the window open for significant time while performing other tasks. However, during the data collection, any time over 15 minutes was filtered out and not included. The database time also does not include the time required to load the program and open the record, although this would be a maximum of one minute.

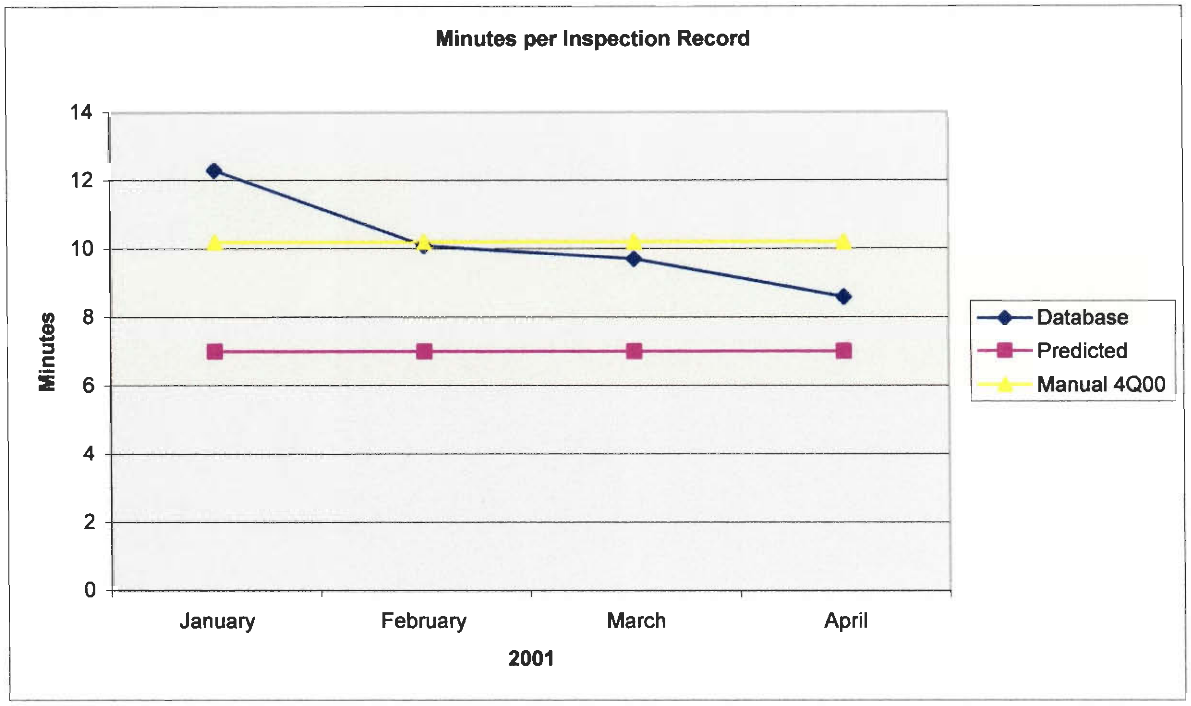


Figure 4.1: Minutes per Inspection Record

The recorded times for the database are steadily improving as the inspectors become more familiar with the Incoming Database. The numbers are slightly disappointing, as the predicted numbers were actually considered conservative at the time of the proposal. Still, the Incoming Database is over a minute quicker per record then the manual data collection method. The results could also be skewed because we present the inspector with more information than with the manual method. For example, hyperlinks to part drawings and specifications are provided. If an inspector is using these features, the time per record is still being accumulated. The sample size and accept/reject criteria are also provided. This information is all provided for the inspector's benefit and undoubtedly increases the efficiency and resources of the inspector. Since retrieving this information was *not* included during the manual times, the graph in Figure 4.1 does not portray an accurate representation of the improvement from the Incoming Database.

The biggest improvement in performance comes from the report generation. Weekly, monthly and quarterly charts are generated for the inspectors and managers to track performance. Many of these charts not only track inspector performance such as throughput etc… but also provide overall supplier performance. For instance, the reject percentage of lots inspected is reported as well as the total lots inspected and rejected. These old charts were created by manually recording data during the week and entering the data into an Excel spreadsheet. Vicor reported that four hours per week were spent recording and creating the weekly charts. Quarterly charts simply averaged the weekly results for the entire quarter, but since this process had to be repeated on eight charts, the entire process adds an additional 4 hours.

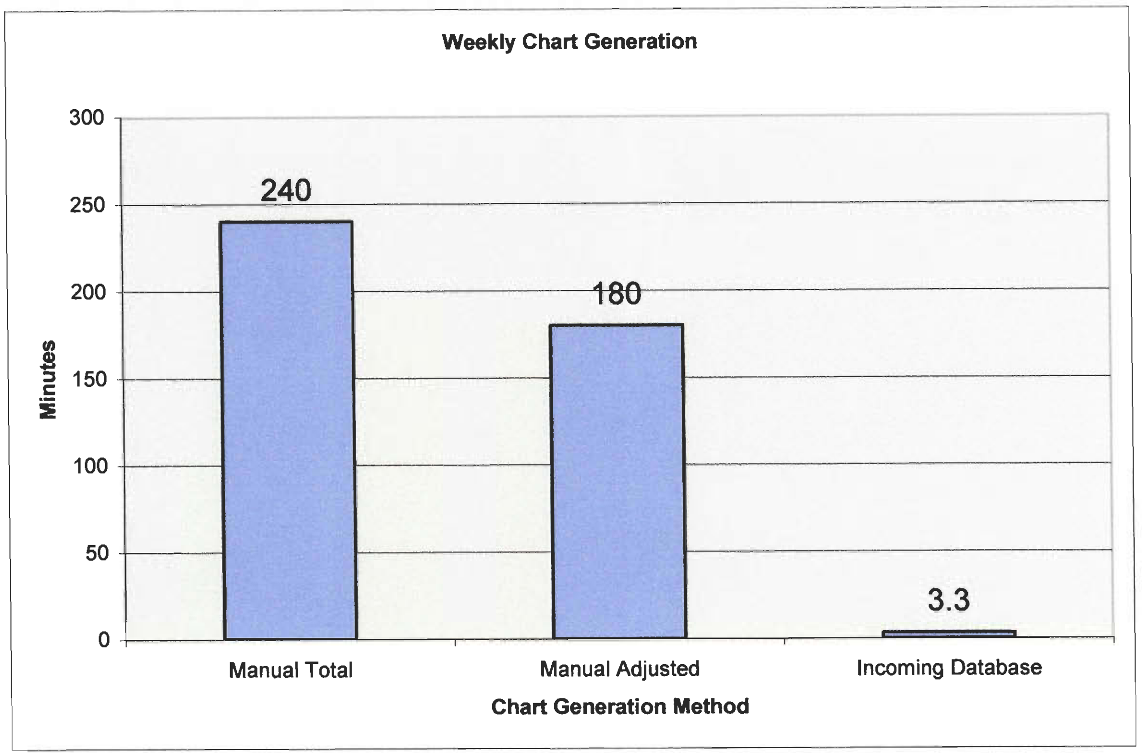


Figure 4.2: Time Expense for Weekly Chart Generation

Figure 4.2 shows the total time in minutes for the creation of weekly charts by the Incoming Database and two different methods of calculating manual method times. Two different times for the manual collection method are shown because the Incoming Database only contains information to replicate six of the eight weekly charts. Since the Vicor estimate of fours hours is based upon the creation of all eight reports, the adjusted manual category displays the estimated time to create six charts. The two charts that the Incoming Database cannot create require information from the NCMR system. These two charts track follow up information to rejected lots. The two charts could be created from the NCMR or the Incoming Database could link to the NCMR data to create the chars. However, at the time of this report, Vicor did not want the two databases sharing data until they were assured the Incoming Database was free of bugs. The time for the

Incoming Database was determined by timing the database while creating the charts. The Incoming Database is programmed to include the trend from the 15 most recent weeks, so the time needed, roughly 200 seconds includes time to calculate the previous weeks as well.

As shown in Figure 4.3, the timesavings by creating the weekly charts by a database over a quarter are enormous. Creating the charts through the Incoming Database saves over 38 hours a quarter in man-hours. This data includes an estimated 10% reduction in database performance resulting from the calculations of additional records. Clearly, the Incoming Database is a vastly superior method of report generation the previous manual method.

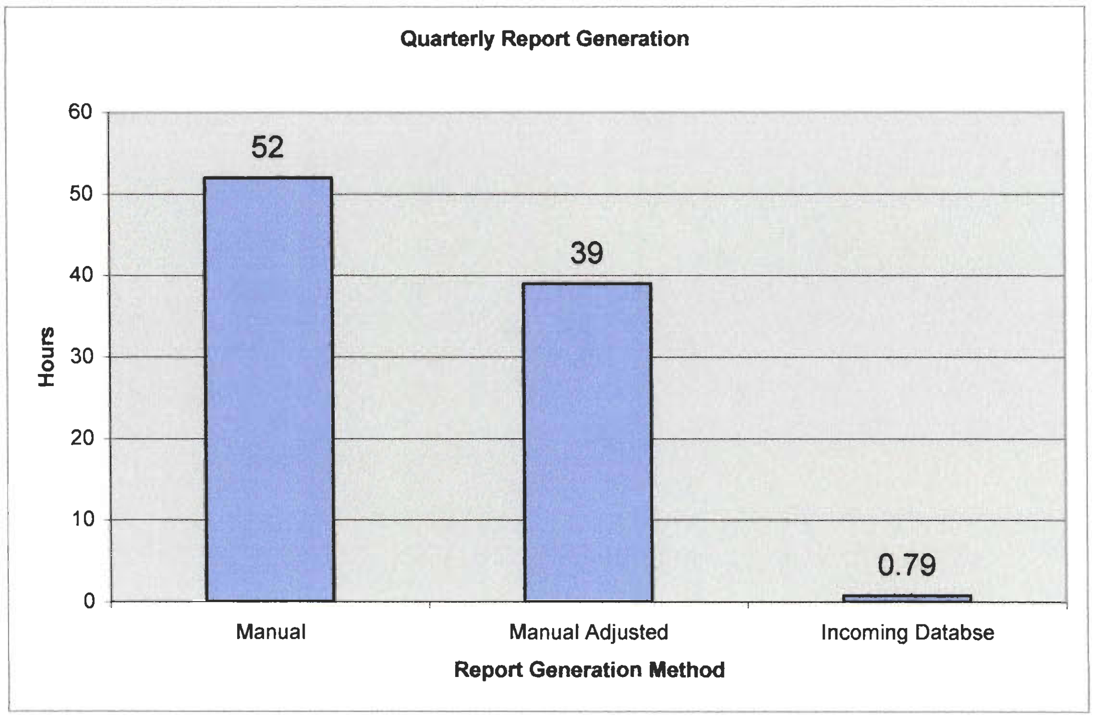


Figure 4.3: Time Expense for Quarterly Chart Generation per Quarter

Chapter 5 : Conclusions & Recommendations

The final step of any design project is to review the final design and report any information garnered that would help in a similar project. Recreating a manual process with a database requires many iterations of planning, programming, reassessment and then programming again. It was often impossible to recreate the manual documentation process exactly, so in many places innovation was needed for a successful final database.

The lessons learned from development of the NCMR Database proved invaluable in the design and implementation of the Incoming Database. Because of the NCMR Database, problems such as interactions with the Oracle Database, proper security measures, user-friendly interface and table structure had already been resolved before the creation of the Incoming Database. This allowed much of development to concentrate on additional features and functionality. Providing the inspectors with hyperlinks to view technical drawings and the addition of the double sampling plan were two such additions that were not included in the original proposal.

The Incoming Database appears to be a great success. It allows the staff of the Incoming Inspection department to job more easily, accurately and efficiently. The Incoming Database allows Vicor to quickly and easily analyze inspection data that will allow Vicor to reassess their allocation of resources. Vicor can now easily identify and target problem areas. Finally, Vicor now has complete manageability of the Class Codes. This provides consistent and ECO controlled inspection criteria, a major difficulty with the manual process.

One of the greatest lessons learned was to keep to a schedule and provide the structure of the database before the addition of superfluous helpful features. One of the problems encountered with Vicor were changing project specifications as the development process proceeded. Many of the changes were incorporated during the final revisions of the database but time and program limits restricted a few requests. However, many of these changing specifications could be accommodated for because the basic functionality of the database had been finished. Since these additional features were not necessary for database functionality, the features could be added even after the Incoming Database had been launched and was in use. Furthermore, it was only after the Incoming Database was in regular use before it became clear what additional features were helpful and which were useless. Fortunately Vicor recognized the logic of waiting till the users had evaluated the database before features were added. Much design work was saved in this manner since only useful features were developed.

There are still many improvements that can be made to the database. The biggest change would be to save the data in Oracle tables. This would provide instant company wide access and allow experienced Oracle users the means to query and report the data. Vicor would also like to implement variable AQL for the Class Code instructions. Vicor inspects to a 1.0% AQL but would like the option to relax the AQL for parts where less accuracy is needed. Ideally the variable AQL would even be between inspection criteria. So, for example, the electrical criteria of a capacitor would remain at 1.0% AQL, but the mechanical criteria could be relaxed to 5.0%. This would require a major revision since this would require the database to calculate and record sample sizes for each criteria. The physical presentation of the database would also have to be modified to display the data. Still, this idea is feasible and would allow Vicor to specialize their inspections for more rigorous inspections on key criteria while relaxing criteria of secondary importance.

Ultimately, Vicor feels this would allow the Incoming Inspection Department to reduce the level of lots rejected that are still useable without compromising quality.

Appendix A

Manual Documentation Inspection & Instruction Card

Part Number: 20637-0304

VENDOR HISTORY CARD

	Rec. # / P.O. #	REV	Date of Insp	Lot Qty	A Doc Chec/Lot	B VISUAL	C DIM	D MARKING	E ELEC	F FINISH	Qty Rej/Acc	(Norm)	Inspection Stamp(s) Mech	Inspection Stamp(s) Elec	Comments/ Remarks
14	725287 / 997851	05	5/10/01	575	0 / 1	N 0 / 50	N 0 / 20		N 0 / 50		0 / 575		[illegible]	[illegible]	2000037[illegible]
15	725804 / 997851	05	5/11/01	444	0 / 1	N 0 / 50	N 0 / 50		N 0 / 50		0 / 444		[illegible]	[illegible]	2000037[illegible]2
16	725206 / 997855	05	4.19.01	394	0 / 1	N 0 / 50	N 0 / 50		N 0 / 50		0 / 394		[illegible]	alta	200003772[illegible]
17	725305 / 997855	05	4.18.01	411	0 / 1	N 0 / 50	N 0 / 32		N 0 / 50		0 / 411		[illegible]	alta	20000377[illegible]3
18	725306 / 997855	05	4.19.01	423	0 / 1	N 0 / 50	N 0 / 50		N 0 / 50		0 / 423		[illegible]	alta	2000037[illegible]
19	725307 / 997855	05	4/18/01	1463	0 / 1	N 0 / 125	N 0 / 125		N 0 / 125		0 / 1463		[illegible]	alta	2000037[illegible]
20	725768 / 997857	05	4/13/01	446	0 / 1	N 0 / 50	N 0 / 50		N 0 / 50		0 / 446		[illegible]	[illegible]	2000037[illegible]
21	726080 / 997855	05	4.30.01	533	0 / 1	N 0 / 50	N 0 / 50		N 0 / 50		0 / 533		[illegible]	[illegible]	2000038037
22	726635 / 997855	05	4.30.01	603	0 / 1	N 0 / 50	N 0 / 50		N 0 / 50		0 / 603		[illegible]	[illegible]	2000038[illegible]2
23	726090 / 997855	05	4/21/01	348	0 / 1	U 0 / 50	U 0 / 50		U 0 / 50		0 / 348		GR	GR	2000038047
24	726071 / 997851	05	4/28/01	580	0 / 1	N 0 / 80	N 0 / 80		N 0 / 80		0 / 580		[illegible]	[illegible]	2000038[illegible]
25	721077 / 997851	05	4/30/01	860	0 / 1	N 0 / 80	N 0 / 80		N 0 / 80		0 / 860		[illegible]	[illegible]	200003803[illegible]
26															
27															
28															

Continuation Issued: Yes No

VICOR INSPECTION INSTRUCTION CARD **VICOR**

Sheet No. 1 of 1

Part Number: 20637-XXXX	Part Description: Orphan Core assy	Issue	1/ (3) 12/15/99 AT [illegible]

Characteristics	Tool/Method	Reference	AQL
Purchase Order Requirements	Visual	Receiving/ Inspection Record	DOC (A)
WORKMANSHIP		Per Print 20638	1.0 (B) %
DIMENSIONS		Per Print 20638	1.0 % (C)
MARKING/ PERMANENCY			(D) %
ELECTRICAL		Per Print 20638	1.0 % (E)
FINISH/ MATERIAL			% (F)

Special Instructions
(i.e.: Diagram, Schematic, Sketches and etc)

Use individual card for each dash number.

Inductance.

Vicor Part # 11037 Rev. 02

Appendix B

Inspector Worksheet & Instruction Card

Report #	Part #	Rev	Part Description	Supplier	Vendor #	Receiver	PO	Date	Lot Qty	DTS Audit
71	20591	#3	INSULATOR, INPUT ROD		17088	0000722554	112880	4/4/2001	50000	

Item #	Date Code	Lot Code	Visual	Dimensional	Electrical	Finish	Packaging	Total Rej.	Sub-Lot Qty	Rej #/Sample
1	N/A	010312-10							25000	8 / 315
2	N/A	010312-11							25000	8 / 315

Instructions Worksheet

Last Update	
Class Code	00050
Part Description	INDUCTORS, MISC.
Workmanship Method	Per Established Workmanship Standards
Workmanship Reference	
Workmanship AQL	0.01
Workmanship Special Instructions	
Dimensions Method	Per assembly drawing identified in BOM. Inspect critical dimensions when specified. Inspect all dimensions when no critical dimensions specified.
Dimensions Reference	
Dimensions AQL	1.0%
Dimensions Special Instructions	
Marking / Permanency Method	Per print when applicable.
Marking / Permanency Reference	
Marking / Permanency AQL	1.0%
Marking / Permanency Special Instructions	
Electrical Method	Per print or winding chart when applicable. Refer to BOM for winding chart. Measure inductance and DCR when applicable. Use specific test equipment when specified by print.
Electrical Reference	
Electrical AQL	1.0%
Electrical Special Instructions	
Finish Material Method	Per print when applicable.
Finish Material Reference	
Finish Material AQL	1.0%
Finish Material Special Instructions	
ECO #	
ECO Date	

Tuesday, April 24, 2001 Page 1 of 1

Appendix C

Visual Basic Code

C-1 Switchboard Code

```
Option Compare Database
Option Explicit

Private Sub Form_Open(Cancel As Integer)
' Minimize the database window and initialize the form.

    Dim dbs As Database
    Dim rst As Recordset


On Error GoTo Form_Open_Err
    Application.SetOption "Confirm Action Queries", 0
       Application.SetOption "Confirm Record Changes", 0
        Application.SetOption "Confirm Document Deletions", 0
    ' Minimize the database window.
    DoCmd.SelectObject acForm, "Switchboard", True
    DoCmd.Minimize
    ' Login_Query auto login with Oracle Database
    DoCmd.OpenQuery "LOGIN_QUERY"

    DoCmd.Hourglass False
    Set dbs = CurrentDb()
    Set rst = dbs.OpenRecordset("My Company Information")
    If rst.RecordCount = 0 Then
      rst.AddNew
      rst![Address] = Null
      rst.Update
      msgbox "Before using this application, you need to enter your
company name, address and related information."
      DoCmd.OpenForm "My Company Information", , , , , acDialog
    End If
    rst.close
    dbs.close

    ' Move to the switchboard page that is marked as the default.
    Me.Filter = "[ItemNumber] = 0 AND [Argument] = 'Default' "
    Me.FilterOn = True

Form_Open_Exit:
    Exit Sub

Form_Open_Err:
    msgbox Err.Description
    Resume Form_Open_Exit

End Sub
```

```
Private Sub Form_Current()
' Update the caption and fill in the list of options.

    Me.Caption = Nz(Me![ItemText], "")
    FillOptions

End Sub
```

```
Private Sub FillOptions()
' Fill in the options for this switchboard page.

    ' The number of buttons on the form.
    Const conNumButtons = 8

    Dim dbs As Database
    Dim rst As Recordset
    Dim strsql As String
    Dim intOption As Integer

    ' Set the focus to the first button on the form,
    ' and then hide all of the buttons on the form
    ' but the first.  You can't hide the field with the focus.
    Me![Option1].SetFocus
    For intOption = 2 To conNumButtons
        Me("Option" & intOption).Visible = False
        Me("OptionLabel" & intOption).Visible = False
    Next intOption

    ' Open the table of Switchboard Items, and find
    ' the first item for this Switchboard Page.
    Set dbs = CurrentDb()
    strsql = "SELECT * FROM [Switchboard Items]"
    strsql = strsql & " WHERE [ItemNumber] > 0 AND [SwitchboardID]=" &
Me![SwitchboardID]
    strsql = strsql & " ORDER BY [ItemNumber];"
    Set rst = dbs.OpenRecordset(strsql)

    ' If there are no options for this Switchboard Page,
    ' display a message.  Otherwise, fill the page with the items.
    If (rst.EOF) Then
        Me![OptionLabel1].Caption = "There are no items for this
switchboard page"
    Else
        While (Not (rst.EOF))
            Me("Option" & rst![ItemNumber]).Visible = True
            Me("OptionLabel" & rst![ItemNumber]).Visible = True
            Me("OptionLabel" & rst![ItemNumber]).Caption =
rst![ItemText]
            rst.MoveNext
        Wend
    End If

    ' Close the recordset and the database.
    rst.close
    dbs.close

End Sub
```

```
Private Function HandleButtonClick(intBtn As Integer)
' This function is called when a button is clicked.
' intBtn indicates which button was clicked.

    ' Constants for the commands that can be executed.
    Const conCmdGotoSwitchboard = 1
    Const conCmdOpenFormAdd = 2
    Const conCmdOpenFormBrowse = 3
    Const conCmdOpenReport = 4
    Const conCmdCustomizeSwitchboard = 5
    Const conCmdExitApplication = 6
    Const conCmdRunMacro = 7
    Const conCmdRunCode = 8

    ' An error that is special cased.
    Const conErrDoCmdCancelled = 2501

    Dim dbs As Database
    Dim rst As Recordset
    Dim Password As String

On Error GoTo HandleButtonClick_Err

    ' Find the item in the Switchboard Items table
    ' that corresponds to the button that was clicked.
    Set dbs = CurrentDb()
    Set rst = dbs.OpenRecordset("Switchboard Items", dbOpenDynaset)
    rst.FindFirst "[SwitchboardID]=" & Me![SwitchboardID] & " AND
[ItemNumber]=" & intBtn

    ' If no item matches, report the error and exit the function.
    If (rst.NoMatch) Then
        msgbox "There was an error reading the Switchboard Items
table."
        rst.close
        dbs.close
        Exit Function
    End If

    Select Case rst![Command]

        ' Go to another switchboard.
        Case conCmdGotoSwitchboard
            'Password Code for Managment screen
            'If rst![SwitchboardID] = 1 And rst![ItemNumber] = 2 Then
             '    Password = InputBox
              '   If Password = "Vicor" Then msgbox "success!"



            Me.Filter = "[ItemNumber] = 0 AND [SwitchboardID]=" &
rst![Argument]

        ' Open a form in Add mode.
        Case conCmdOpenFormAdd
            DoCmd.OpenForm rst![Argument], , , , acAdd

        ' Open a form.
        Case conCmdOpenFormBrowse
            DoCmd.OpenForm rst![Argument]

        ' Open a report.
        Case conCmdOpenReport
            DoCmd.OpenReport rst![Argument], acPreview
        ' Customize the Switchboard.
        Case conCmdCustomizeSwitchboard
            ' Handle the case where the Switchboard Manager
            ' is not installed (e.g. Minimal Install).
            On Error Resume Next
            Application.Run "WZMAIN80.sbm_Entry"
            If (Err <> 0) Then msgbox "Command not available."
            On Error GoTo 0
            ' Update the form.
            Me.Filter = "[ItemNumber] = 0 AND [Argument] = 'Default' "
            Me.Caption = Nz(Me![ItemText], "")
            FillOptions

        ' Exit the application.
        Case conCmdExitApplication
            CloseCurrentDatabase

        ' Run a macro.
        Case conCmdRunMacro
            DoCmd.RunMacro rst![Argument]

        ' Run code.
        Case conCmdRunCode
            Application.Run rst![Argument]

        ' Any other command is unrecognized.
        Case Else
            msgbox "Unknown option."

    End Select

    ' Close the recordset and the database.
    rst.close
    dbs.close

HandleButtonClick_Exit:
    Exit Function

HandleButtonClick_Err:
    ' If the action was cancelled by the user for
    ' some reason, don't display an error message.
    ' Instead, resume on the next line.
    If (Err = conErrDoCmdCancelled) Then
        Resume Next
    Else
        msgbox "There was an error executing the command.", vbCritical
        Resume HandleButtonClick_Exit
    End If

End Function
```

C-2 Line Item Code

```
Option Compare Database
Option Explicit
Dim db As Database
Dim rs As Recordset

Private Sub Delete_Record_Click()
'Sub to delete record line item
Set db = CurrentDb
Dim Child As Integer
Dim Item As Double
Dim Item2 As Double
Dim Item3 As Double
Dim strsql As String

If Me.Item_Num = 0 Then
    Me.Item_Num = Me.ItemView
End If
    'Check if line item is single or double inspection
    If Right(Me.Item_Num, 2) = 0.2 Then
        Item = Left(Me.Item_Num, 2) + 0.1
        Item2 = Me.Child_Num
        strsql = "SELECT * FROM [History Details] WHERE Child_ID=" &
Me.Child_Num & " AND Item_Num >=" & Me.Item_Num & " order by Item_Num"
        DoCmd.GoToRecord , , acLast
            If Me.Item_Num = 1.1 Then
                DoCmd.GoToRecord , , acNewRec
            Else
                DoCmd.GoToRecord , , acFirst
            End If
        Set rs = db.OpenRecordset(strsql, dbOpenDynaset)
        If Not rs.BOF Or Not rs.EOF Then rs.Delete
        rs.close
        strsql = "SELECT * FROM [History Details] WHERE Child_ID=" &
Item2 & " AND Item_Num =" & Item & " order by Item_Num"
        Set rs = db.OpenRecordset(strsql, dbOpenDynaset)
        If Not rs.BOF Or Not rs.EOF Then
        rs.Edit
        rs.Fields("Line_Item_Query_Key") = -1
        rs.Update
        End If
        rs.close
    Else

        If Right(Me.Item_Num, 2) = 0.1 Then
            Child = Me.Child_Num
            Item = Me.Item_Num
            Item3 = Left(Me.Item_Num, 2) + 1
            strsql = "SELECT * FROM [History Details] WHERE Child_ID="
                  & Me.Child_Num & " AND Item_Num=" & Me.Item_Num

            Set rs = db.OpenRecordset(strsql, dbOpenDynaset)
            If Not rs.BOF Or Not rs.EOF Then rs.Delete
            rs.close
            Item2 = Item + 0.1
            strsql = "Select * From [History Details] WHERE Child_ID ="
& Child & " AND Item_Num >=" & Item2 & " and Item_Num<" & Item3
            Set rs = db.OpenRecordset(strsql, dbOpenDynaset)

            If Not rs.BOF Or Not rs.EOF Then
                rs.Edit
                rs.Fields("Item_Num") = Item
                rs.Update
                rs.close
            Else
                rs.close
                strsql = "SELECT * FROM [History Details] WHERE
Child_ID=" & Child & "AND Item_Num > " & Item
                Set rs = db.OpenRecordset(strsql, dbOpenDynaset)
                If Not rs.BOF Or Not rs.EOF Then
                    rs.MoveLast
                    rs.MoveFirst
                    Do While Not rs.EOF
                        rs.Edit
                        rs.Fields("Item_Num") = rs.Fields("Item_Num") - 1
                        rs.Update
                        rs.MoveNext
                    Loop
                    rs.close
                End If
            End If
        Else
            Child = Me.Child_Num
            Item = Me.Item_Num
            strsql = "SELECT * FROM [History Details] WHERE Child_ID="
& Child & " AND Item_Num=" & Item
            DoCmd.GoToRecord , , acLast
            If Me.Item_Num = 1 Then
                DoCmd.GoToRecord , , acNewRec
            Else
                DoCmd.GoToRecord , , acFirst
            End If
            Set rs = db.OpenRecordset(strsql, dbOpenDynaset)
            If Not rs.BOF Or Not rs.EOF Then rs.Delete
            rs.close
            strsql = "SELECT * FROM [History Details] WHERE Child_ID="
& Child & "AND Item_Num > " & Item
            Set rs = db.OpenRecordset(strsql, dbOpenDynaset)
            If Not rs.BOF Or Not rs.EOF Then
                rs.MoveLast
                rs.MoveFirst
                Me.Count1 = Item
                Do While Not rs.EOF
                    If rs.Fields("Item_Num") > Me.Count1 Then Me.Count1
= rs.Fields("Item_Num")
                    rs.Edit
                    rs.Fields("Item_Num") = rs.Fields("Item_Num") - 1
                    rs.Update
                    rs.MoveNext
                Loop
                rs.close
            End If
        End If

    End If
    'Refresh Data following delete
    Requery
    Forms![History].[History Details Datasheet].Requery

    If Me.Item_Num = 0 Then
        Me.Count1 = 1
        If Me.Sample_Type = "DN" Or Me.Sample_Type = "DT" Then
            Me.ItemView = 1.1
        Else
            Me.ItemView = 1
        End If
        Me.Line_Item_Saved = 0
    Else
        Me.ItemView = Me.Item_Num
        Me.Count1 = 1
    End If
End Sub
```

```
Private Sub Dimensional_Exit(Cancel As Integer)
'After exit from inspection criteria
'check if lot fails
Check_if_Fail
End Sub
```

```
Private Sub Electrical_Exit(Cancel As Integer)
'After exit from inspection criteria
'check if lot fails
Check_if_Fail
End Sub
```

```
Private Sub Finish_Exit(Cancel As Integer)
'After exit from inspection criteria
'check if lot fails
Check_if_Fail
End Sub
```

```
Private Sub Form_Open(Cancel As Integer)
'After exit from inspection criteria
'check if lot fails
'Check_if_Fail
End Sub
```

```
Private Sub Lot_code_AfterUpdate()
'Initialize auto number upon record resume
If Me.ItemView < 2 And Not Me.Line_Item_Saved = -1 Then
    If Me.Sample_Type = "SN" Then Me.ItemView = Me.Count1
    If Me.Sample_Type = "ST" Then Me.ItemView = Me.Count1
    If Me.Sample_Type = "DN" Then Me.ItemView = Me.Count1 + 0.1
    If Me.Sample_Type = "DT" Then Me.ItemView = Me.Count1 + 0.1
    Forms![History]![Sample] = Me.Sample_Type
```

```
End If
End Sub
```

```
Private Sub Marking_Exit(Cancel As Integer)
'After exit from inspection criteria
'check if lot fails
Check_if_Fail
End Sub
```

```
Private Sub Packaging_Exit(Cancel As Integer)
'After exit from inspection criteria
'check if lot fails
Check_if_Fail
End Sub
```

```
Private Sub Previous_Record_Click()
'Go to previous line item
On Error GoTo Err_Previous_Record_Click
    If Me.Item_Num = 0 And IsNull(Me.Date_Code) And IsNull(Me.Lot_Code)
Or Me.Line_Item_Saved = -1 Then
    DoCmd.GoToRecord , , acPrevious
    Me.ItemView = Me.Item_Num
    Me.Count1 = Left(Me.Item_Num, 2)
    Check_if_Fail
    Else
    msgbox "Please click 'Next Record' to enter current record before
returning to previous records", vbOKOnly, "Error: Current Record Not
Yet Saved."
    End If
Exit_Previous_Record_Click:
    Exit Sub

Err_Previous_Record_Click:
    msgbox Err.Description
    Resume Exit_Previous_Record_Click

End Sub
```

```
Private Sub Next_Record_Click()
'Go to next record
'if next record exists go to it
'if next record does not exist, create it
Dim strLotCode As String
Dim strDateCode As String
Dim strLotQty As String
Dim SampleSize As Double
Dim RejNum As Integer
Dim AccNum As Integer
Dim tempcount As Double



    If IsNull(Me.Date_Code) And IsNull(Me.Lot_Code) Or Me.Sub_Lot_Qty =
0 Then
```

```
        msgbox "The Date Code, Lot Code and Sub-Lot Quantity must be
entered before you may save this line item."
    Else
        If Me.Item_Num = 0 Then
            If Right(Me.ItemView, 2) = 0.1 Then
            'for double inspection create the second line item
                Me.Item_Num = Me.ItemView
                tempcount = Me.Item_Num
                strLotCode = Me.Lot_Code
                strDateCode = Me.Date_Code
                SampleSize = Me.Sub_Lot_Sample * 2
                strLotQty = Me.Sub_Lot_Qty
                RejNum = Me.Cum_Reject
                AccNum = Me.Cum_Accept
                Me.Line_Item_Saved = -1
                DoCmd.GoToRecord , , acNext
                Me.Item_Num = tempcount + 0.1
                Me.Lot_Code = strLotCode
                Me.Date_Code = strDateCode
                Me.Sub_Lot_Sample = SampleSize
                Me.Sub_Lot_Qty = strLotQty
                Me.Reject_Num = RejNum
                Me.Accept_Num = AccNum
                Me.Line_Item_Saved = -1         'save line item
                Me.Line_Item_Query_Key = -1
                DoCmd.GoToRecord , , acNext
                Forms![History].[History Details Datasheet].Requery
                Me.Count1 = Me.Count1 + 1       'increment auto number
                Me.ItemView = Me.Count1 + 0.1
            Else
                Me.Item_Num = Me.ItemView
                Me.Line_Item_Saved = -1         'save line item
                Me.Line_Item_Query_Key = -1
                DoCmd.GoToRecord , , acNext
                Forms![History].[History Details Datasheet].Requery
                Me.Count1 = Me.Count1 + 1       'increment auto number
            End If
            'set auto number for next line item
            If Me.Sample_Type = "SN" Then Me.ItemView = Me.Count1
            If Me.Sample_Type = "ST" Then Me.ItemView = Me.Count1
            If Me.Sample_Type = "DN" Then Me.ItemView = Me.Count1 + 0.1
            If Me.Sample_Type = "DT" Then Me.ItemView = Me.Count1 + 0.1
        Else
            Me.Line_Item_Saved = -1
            DoCmd.GoToRecord , , acNext
            Check_if_Fail
            If Me.Item_Num = 0 Then
                Me.Count1 = Me.Count1 + 1
                Me.ItemView = Me.Count1
                If Me.Sample_Type = "SN" Then Me.ItemView = Me.Count1
                If Me.Sample_Type = "ST" Then Me.ItemView = Me.Count1
                If Me.Sample_Type = "DN" Then Me.ItemView = Me.Count1 + 0.1
                If Me.Sample_Type = "DT" Then Me.ItemView = Me.Count1 + 0.1
            Else
                Me.ItemView = Me.Item_Num
                Me.Count1 = Left(Me.Item_Num, 2)
            End If
        End If
    End If
End Sub
```

```
Private Sub Sample_Type_Click()
'initialize auto number once sample size is known
'sample size must be selected on first line item
If Me.ItemView < 2 And Not Me.Line_Item_Saved = -1 Then
    If Me.Sample_Type = "SN" Then Me.ItemView = Me.Count1
    If Me.Sample_Type = "ST" Then Me.ItemView = Me.Count1
    If Me.Sample_Type = "DN" Then Me.ItemView = Me.Count1 + 0.1
    If Me.Sample_Type = "DT" Then Me.ItemView = Me.Count1 + 0.1
    Forms![History]![Sample] = Me.Sample_Type
Else
    Me.Sample_Type = Forms![History]![Sample]
    msgbox "The Sample Type may only be selected on the first line item
before the line item is saved.  The Sample Type must remain the same
for all line items." & vbCrLf & vbCrLf & "If the wrong Sample Type was
initialy selected, all line items must be deleted and the data entered
again with the correct Sample Type", vbOKOnly, "Error: Sample Type
Selected After Line Item #1 Has Been Entered"
End If
End Sub
```

```
Private Sub Sub_Lot_Qty_Exit(Cancel As Integer)

'Determin sampling qty by querying "Sampling Table" using user defined
sample type and lot qty as criteria.
'Return sampling qty to form.

Set db = CurrentDb
Dim strsql As String
If Me.Sub_Lot_Qty >= 0 Then
strsql = "SELECT * FROM [Sampling Table] WHERE Lower_Limit<=" &
Me.Sub_Lot_Qty & " AND Upper_Limit>=" & Me.Sub_Lot_Qty
Set rs = db.OpenRecordset(strsql, dbOpenDynaset)
If Not rs.BOF Or Not rs.EOF Then
    If Me.Sample_Type = "SN" Then
        Me.Sub_Lot_Sample = rs.Fields("Normal_Sample_Size")
        Me.Reject_Num = rs.Fields("Normal_Reject")
        Me.Accept_Num = rs.Fields("Normal_Accept")
    End If
    If Me.Sample_Type = "ST" Then
        Me.Sub_Lot_Sample = rs.Fields("Tightened_Sample_Size")
        Me.Reject_Num = rs.Fields("Tightened_Reject")
        Me.Accept_Num = rs.Fields("Tightened_Accept")
    End If
    If Me.Sample_Type = "DN" Then
        Me.Sub_Lot_Sample = rs.Fields("Double_Norm_Sample_Size")
        Me.Reject_Num = rs.Fields("Double_Norm_Reject")
        Me.Accept_Num = rs.Fields("Double_Norm_Accept")
        Me.Cum_Accept = rs.Fields("Double_Norm_Cumulative_Accept")
        Me.Cum_Reject = rs.Fields("Double_Norm_Cumulative_Reject")
    End If
    If Me.Sample_Type = "DT" Then
        Me.Sub_Lot_Sample = rs.Fields("Double_Tight_Sample_Size")
        Me.Reject_Num = rs.Fields("Double_Tight_Reject")
        Me.Accept_Num = rs.Fields("Double_Tight_Accept")
        Me.Cum_Accept = rs.Fields("Double_Tight_Cumulative_Accept")
        Me.Cum_Reject = rs.Fields("Double_Tight_Cumulative_Reject")
    End If
End If
rs.close
End If
If Me.Sub_Lot_Sample > Me.Sub_Lot_Qty Then
    Me.Sub_Lot_Sample = Me.Sub_Lot_Qty
End If
End Sub
```

```
Private Sub Check_if_Fail()
'Check if sub-lot has failed.  If it has indicate failure by changing
text color to red.

Dim lngred As Long
Dim lngblack As Long
lngred = RGB(255, 0, 0)
lngblack = RGB(0, 0, 0)

Me.Reject_Qty = Me.Visual + Me.Dimensional + Me.Marking + Me.Electrical
+ Me.Packaging + Me.Finish

If Me.Reject_Qty >= Me.Rej_Num Then
Me.Reject_Qty.ForeColor = lngred
Else: Me.Reject_Qty.ForeColor = lngblack
End If
End Sub
```

```
Private Sub Change_View_Color()
'Check if any sub-lot has failed.  If it has indicate lot failure by
turning view red

Dim strsql As String
Dim fail As Integer
Dim rschange As Recordset
Set db = CurrentDb

fail = 0
strsql = "SELECT * FROM [History Details] WHERE Child_ID=" &
Me.Child_Num
    Set rschange = db.OpenRecordset(strsql, dbOpenDynaset)
    If Not rschange.BOF Or Not rschange.EOF Then
        rschange.MoveLast
        rschange.MoveFirst
        Do While Not rschange.EOF
            Me.Reject_Qty = Me.Visual + Me.Dimensional + Me.Marking +
Me.Electrical + Me.Packaging + Me.Finish
            If Me.Reject_Qty >= Me.Rej_Num Then fail = 1
            rschange.MoveNext
        Loop
        rschange.close
    End If

If fail = 1 Then
Forms![History]![History Details Datasheet].Form![Qty_Reject].ForeColor
= RGB(255, 0, 0)
Else
Forms![History]![History Details Datasheet].Form![Qty_Reject].ForeColor
= RGB(0, 0, 0)
End If
End Sub
```

```
Private Sub Visual_Exit(Cancel As Integer)
'After exit from inspection criteria
'check if lot fails
Check_if_Fail
End Sub
```

C-3 Login Code

```
Option Compare Database
Option Explicit
Dim db As Database
Dim rs As Recordset
Dim rs2 As Recordset
'This code verifies user and password then creates new record
'or resumes as necessary


Private Sub Command3_Click()
Dim strInput As String
Dim strsql As String
Dim strSQL2 As String
Dim strPass As String
Dim intanswer As Integer
Set db = CurrentDb

'Verify Login
If IsNull(Me.Inspector) Then
    msgbox ("Please Login Before Entering the VICOR Automated History
Card Database."), , "Vicor Automated Inspection Database"
Else
    If IsNull(Me.Receiver) Then
        msgbox "Please Enter the Receiver Number.", , "Vicor Automated
Inspection Database"
    Else
        strInput = Me.Inspector
        strsql = "Select * from [Inspector Personel] where inspector =
'" & strInput & "'"
        Set rs = db.OpenRecordset(strsql, dbOpenDynaset)
        If Not (rs.BOF And rs.EOF) Then strPass = rs.Fields("Password")
        rs.close
        If Not strPass = Me.User_Password Then
            msgbox "The password entered is incorrect.", , "Vicor
            Automated Inspection Database"
        Else
            'return record data from login
            strSQL2 = "Select * From [tblHistory] where Receiver=0000"
                  & Me.Receiver & " and Save=-1"
            Set rs2 = db.OpenRecordset(strSQL2, dbOpenDynaset)
            If Not (rs2.BOF And rs2.EOF) Then
            strsql = "Select * from [Inspector Personel] where User='"
                  & strInput & "' and Edit_Records = -1"
            Set rs = db.OpenRecordset(strsql, dbOpenDynaset)
            If Not (rs.BOF And rs.EOF) Then
            'result if record has been already entered
            'if user has edit authorization then prompt to open
            'otherwise return user to lgin screen
                  intanswer = msgbox("The Receiver Number you selected
                  has been previously entered and saved by " &
                  rs2.Fields("Inspector") & ".  " & vbCrLf & vbCrLf &
                  "You have authorization to edit this record,
                  proceeed?", vbYesNo, "Receipt Edit Authorized")
                        If intanswer = vbYes Then Open_Existing_Record
                        rs.close
                        rs2.close
                        Else
                  msgbox "The Receiver Number you selected has been
                  previously entered and saved by " &
                  rs2.Fields("Inspector") & ".  " & vbCrLf & vbCrLf &
                  "This reciept may not be entered again.  Please
                  contact System Administrator for problems.",
                  vbOKOnly, "Receipt Access Denied"
                        rs.close
                        rs2.close
                    End If
            Else
                'result if inspector may resume previous record
                strsql = "Select * From [tblHistory] where
                  Receiver=0000" & Me.Receiver & " and Save=0 and
                  Resume =-1"
                Set rs = db.OpenRecordset(strsql, dbOpenDynaset)
                If Not (rs.BOF And rs.EOF) Then
                    If Me.Inspector = rs.Fields("Inspector") Then
                        Open_Existing_Record
                        msgbox "The Receiver Number you selected has
                        been previously entered but not completed.  You
                        may continue with this record, or delete it and
                        begin again."
                        rs.close
                    Else
                        'record authored by another user
                        'return user to login screen
                        msgbox "The Receiver Number you selected has
been previously entered but not completed by " & rs.Fields("Inspector")
& ".  " & vbCrLf & vbCrLf & "This reciept may only be continued by " &
rs.Fields("Inspector") & ".", vbOKOnly, "Receipt Access Denied"
                        rs.close
                    End If
                Else
                  'if record can't be found create new record
                    Open_New_Record
                End If
            End If
        End If
    End If
End If
Forms![Launch Form]![User_Password] = ""
End Sub
```

```
Private Sub Inspector_Click()
'require inspector to reenter password after each use
Me.User_Password = ""
End Sub
```

```
Private Sub Open_Existing_Record()
'function to open an existing record in the Incoming Database
Dim stDocname As String
Dim stLinkCriteria As String
```

```
    stDocname = "History"
    stLinkCriteria = "[Receiver]=" & Me.Receiver
    DoCmd.OpenForm stDocname, acDesign, , stLinkCriteria
    DoCmd.OpenForm stDocname, , , stLinkCriteria
End Sub
```

```
Private Sub Open_New_Record()
'Collect record data from Oracle table, and create new
'record in Incoming Database
Dim strInpt As String
Dim strSQL3 As String

DoCmd.OpenForm "History", acNormal, , , acFormAdd, acWindowNormal
DoCmd.GoToRecord , , acNewRec
Set db = CurrentDb
strInpt = Forms![Launch Form]![Receiver]
strSQL3 = "Select * From [Receiver Data] where receiver_num = '0000" &
strInpt & "'"
Set rs = db.OpenRecordset(strSQL3, dbOpenDynaset)
'Input Oracle data into Incoming Database
If Not (rs.BOF And rs.EOF) Then
    Forms![History]![Part_Num] = rs.Fields("part_num")
    Forms![History]![Revision] = rs.Fields("revision")
    Forms![History]![Part_Description] = rs.Fields("part_description")
    Forms![History]![supplier] = rs.Fields("supplier")
    Forms![History]![Vendor_Num] = rs.Fields("vendor_num")
    Forms![History]![PO_Num] = rs.Fields("po_num")
    Forms![History]![Receiver_Num] = rs.Fields("receiver_num")
    Forms![History]![Qty_Received] = rs.Fields("qty_received")
    'Forms![History]![DTS_Audit] = rs.Fields("dts_audit")
End If

rs.close
Forms![History]![Inspector] = Forms![Launch Form]![Inspector]

'If receipt not found, return user to lauch screen.

If Forms![History]![Receiver_Num] = 0 Then
    msgbox "Vicor's Automated History Database was unable to find the
Receiver # 0000" & Forms![Launch Form]![Receiver] & ".  Please check to
make sure that this is the correct Reciever Number.", vbOKOnly, "Error
Finding Specified Receiver Number"
    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70
    DoCmd.close acForm, "History"
Else
    DoCmd.DoMenuItem acFormBar, acRecordsMenu, 5, , acMenuVer70
End If
End Sub
```