Interfacing a LabVIEW-driven DAQ instrument with LabWare LIMS

Swati Lakhanpal

Submitted to the faculty of the School of Informatics
in partial fulfillment of the requirements
for the degree
Master of Science
in Laboratory Informatics
Indiana University
May 2007

Master's Thesis Committee

Mahesh Merchant, Ph.D.

Chair

Douglas Perry, Ph.D.

Silpa Wairatpanij, MS.

Accepted by the Faculty of Indiana University
in partial fulfillment of the requirements for the degree of Master of Science in
Chemical Informatics (Laboratory Informatics Specialization)

ACKNOWLEDGEMENTS

I would like to thank everyone who has been a help at Indiana University-Purdue University Indianapolis (IUPUI) throughout my two years at the School of Informatics.

I would like to thank Dr. Douglas Perry for his efforts in creating the Laboratory Informatics program at IUPUI. I am grateful to Dr. Mahesh Merchant for the support and encouragement that I received from him throughout this project. I am indebted to Silpa Wairatpanij for his invaluable guidance and support right from the beginning of my work. Many thanks go to the staff at School of Informatics, especially Ms. Mary O' Neill and Ms. Elisabeth Hinshaw-Osgood.

Special thanks are due to LabWare for awarding me with the LabWare Fellowship. I would like to mention a special thanks to my parents, sister and Sumit for their whole-hearted support and encouragement throughout my program and I would like to dedicate my thesis to them.

ABSTRACT

The main purpose of this project is to automate information management and analysis, as well as to integrate different types of data. We intend to interface data acquisition (DAQ) instruments with LIMS by using LabVIEW^{TM}.

LabVIEW (**Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench) is graphical programming software from National Instruments (NI). LabVIEW is the tool of choice due to its unparalleled connectivity to instruments, powerful data acquisition capabilities, natural dataflow-based graphical programming interface, scalability, and overall function completeness and designed specially for data acquisition. NI has many kinds of DAQ instruments that can be used with LabVIEW DAQ VI's (LabVIEW programs). LabVIEW, plug-in data acquisition (DAQ) boards will help perform single point and continuous measurements.

LabWare LIMS is able to interface to a variety of instruments using LabWare LabStation module. Integrated LabStation is used to validate the results. The data used and generated by the instruments using LabVIEW will be maintained by a LIMS.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

LabVIEW

LabVIEW is an acronym for **Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench. It is a computer software development application created by National Instruments™ (Austin, Texas) that aims to aid scientists and researchers in gathering and understanding data using computer programs. LabVIEW is graphical (G) programming software that utilizes graphical objects to symbolize lines of code instead of text-based languages. In the source code of this graphical program, data execution depends on the flow of data. What may take days in C++ or Java written code is cut down to hours in G programming. Moreover, LabVIEW has built a general purpose library of functions and subroutines for most programming tasks. The time saved allows the user to fully focus on and understand the data. Even though some type of programming experience is useful, a novice in text-based programming language(s) can grasp the mechanics of LabVIEW because it is a graphical programming language utilizing iconic symbols to illustrate program action.

Operating LabVIEW

The building blocks of the LabVIEW program are called Virtual Instruments (VI). The name comes from the fact that the program emulates the appearance and tasks of physical instruments while still operating in the same capability as a text-based program. There are three key components to any VI: the front panel, the block diagram, and the icon and connector pane. The figures 1 & 2 below show the block diagram and the front panel of a typical VI.

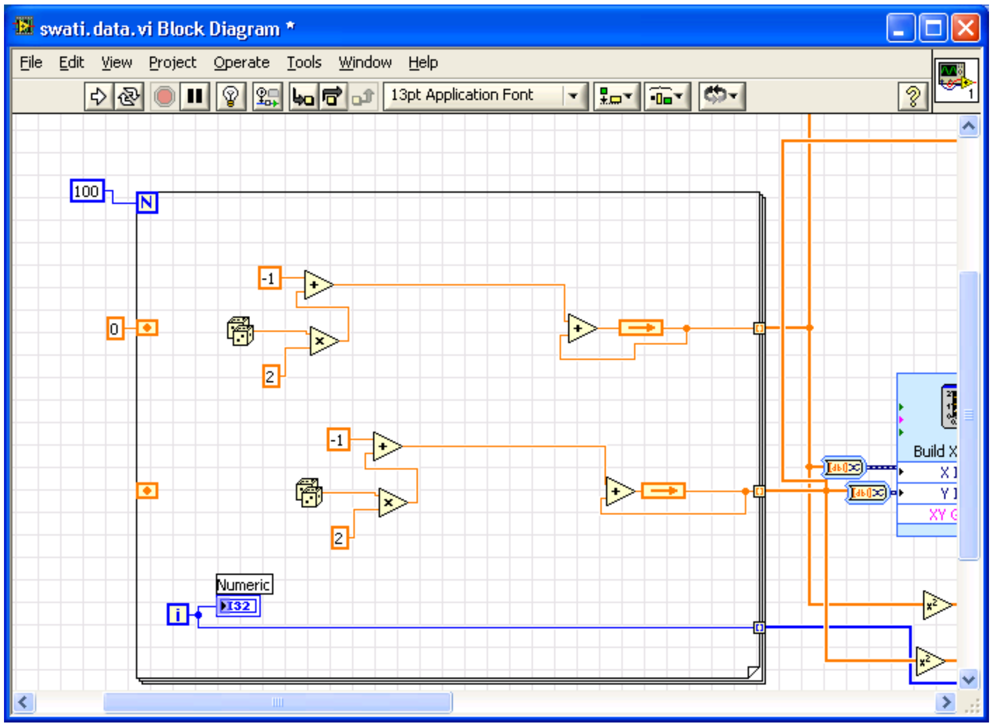

swati.data.vi Block Diagram *
File Edit View Project Operate Tools Window Help
13pt Application Font
100
N
0
-1
2
Numeric
I32
i
Build X
X
Y
XY

Figure 1: Screen shot of the block diagram of VI

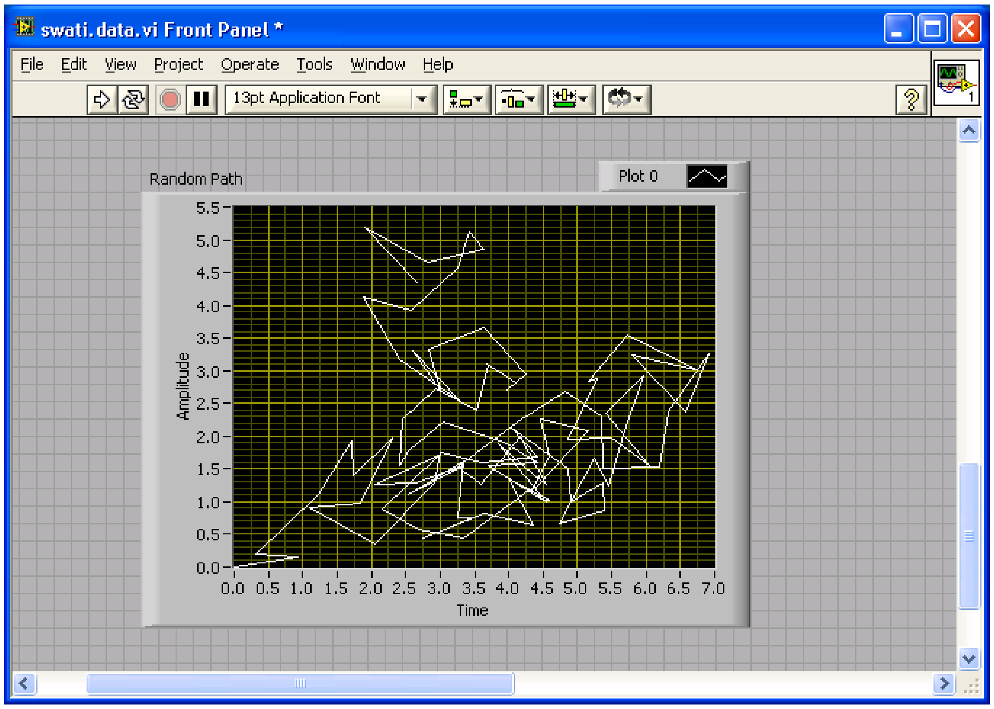


Figure 2: Screen shot of the front panel of VI

Front Panel

The front panel is the VI's interactive interface built with controls (inputs) and indicators (outputs) that replicate conventional instruments, like thermometer etc. It can also represent the output in the form of graphs, charts, boolean, digital waveforms, and other controls that may be found in the real world. The LabVIEW programmer can simply drag and drop the input data to the panels using the mouse, wire them and get the results on the front panel. Figure 3 illustrates the 'Function' palette in the front panel.

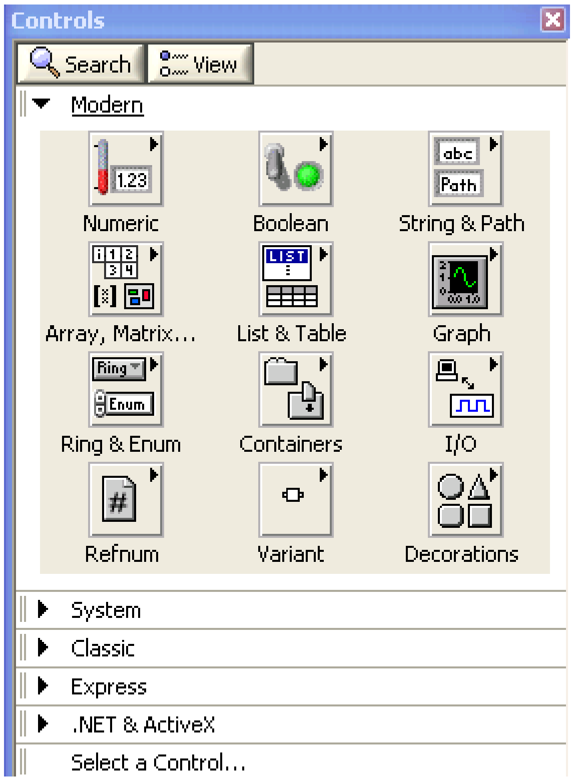


Figure 3: 'Function' palette in front panel

Block diagram

The block diagram is the source code location for executing programs in LabVIEW and is equivalent to text-based programming such as C++. The block diagram has three central parts to its source code: nodes, wires, and terminals. This is the site where the wiring of graphical objects or terminals is connected to the functional nodes. Terminals are the controls (inputs) and indicators (outputs) of the block diagram. When compared with written source code, nodes are equal to statements, functions, and subroutines. An example of a node may be an addition function or a while loop. The addition function is considered a lower-level VI. It is part of the general purpose LabVIEW library of functions and subroutines for programming tasks. Execution of the program, with all three—nodes, terminals, and wires—can occur when wiring between the terminals and nodes has been connected to designate the flow of data. The terminals in the front panel have parallel terminals on the block diagram; thereby data continuously flows from the interface in the front panel to the source code in the block diagram and back to the interface. Figure 4 illustrates the ‘Function’ palette in the block diagram. Some subpalettes like Instrument I/O, Structures, Numeric, Array, Boolean, etc. are subVIs which are used to perform subroutines like opening, closing, or inserting a database.

Icon/Connector

An icon symbolizes a VI in another block diagram. When an icon is placed in a foreign block diagram, it is there to act as a subroutine. A VI within another VI is called a subVI and is considered a lower-level call. A subVI connects with a top-level VI by means of a connector that passes data to outputs/indicators in the block diagram and receives data from inputs/controls in the front panel of the lower call terminal. The icon is the graphical representation of the subVI and may be likened to constraints of a subroutine in written code. To function as a subtask in a top-level call, there must be an icon and a connector.

The LabVIEW graphical program is both hierarchical and modular. Top-level and low-level calls, such as routines within a subroutine, are examples of this program's hierarchical nature. Each subroutine, or subVI, can execute independently of the top-level VI. A subVI can be built upon other subVIs.

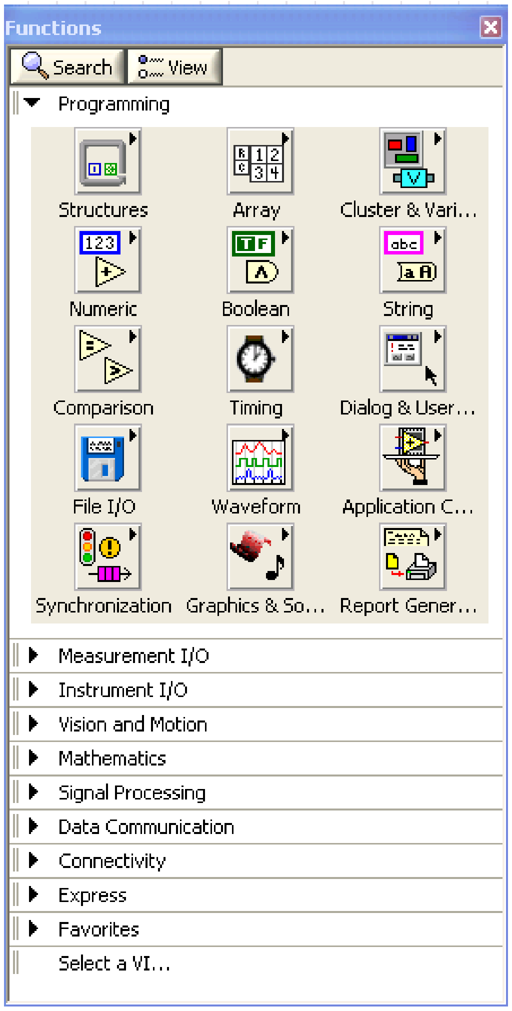


Figure 4: 'Function' palette in block diagram

Laboratory Information Management Systems

Laboratory Information Management Systems (LIMS) are computer software that is used in the laboratory for the management of samples, standards, laboratory users and other laboratory functions. LIMS are generally targeted towards environmental, research, or commercial use, viz. pharmaceutical or petrochemical. The main goal of using LIMS is to create an organization where:

- Instruments used in the laboratories can be integrated with LIMS and can receive instructions, return the results back to LIMS where they can be stored.
- Laboratory users can perform calculations, document and review the results received from integrated instruments and other resources.
- Management can supervise the laboratory process to ensure the regulatory demands.

LabWare LIMS

LabWare LIMS is a customer-configurable LIMS with a variety of optional modules. The use of modules allows the LIMS to integrate into any computing environment. LabWare LIMS offers major capabilities for pharmaceutical companies that help automate the laboratories. It is an object oriented application designed for Windows users. It has a built-in scripting language called LIMS Basic, and therefore allowing the user to configure the extensions. It also includes Database Management Utility that allows records to be imported/exported to LIMS via .CSV files. Instrument interfacing is accomplished in LIMS using the built-in LabStation module (discussed next). In this project, the LabStation module is used with the LabWare LIMS.

LabStation^{TM}

The integrated LabStation by LabWare LIMS is used for instrument integration. It is a general-purpose instrument interfacing tool that can accept results from almost any type of instrument and enter those results into LIMS automatically or upon user’s review and approval. The built-in instrument integration software assures regulatory compliance issues like 21 CFR Part 11. Title 21 Code of Federal Regulations (21 CFR Part11) applies to records in electronic form that are created, modified, maintained, archived, retrieved, or transmitted under any records requirements set forth in Agency regulations. Part 11 also applies to electronic records submitted to the Agency under the Federal Food, Drug, and Cosmetic Act (the Act) and the Public Health Service Act (the PHS Act), even if such records are not specifically identified in Agency regulations.

LabStation offers various mechanisms to integrate various instruments to LIMS, such as:

- Serial RS-232 Communications (COM ports)
- File based Import
- Reports stored in the NuGenesis Unify/ Vision Product (Version 5.2 or higher)

Screenshot

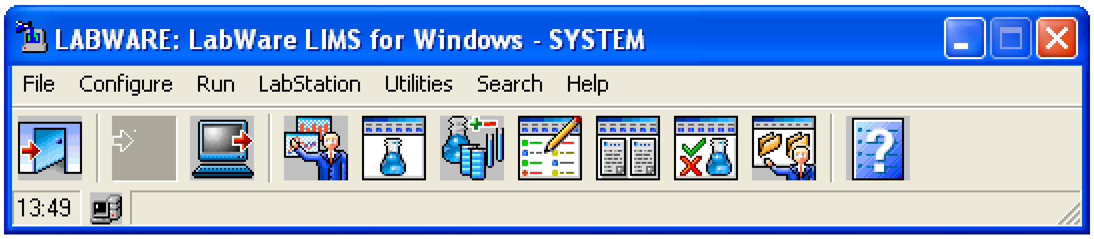


Figure 5: Screen shot of LabWare LIMS with Integrated LabStation

There are fundamentally two types of instruments that can be used with LabWare LIMS/LabStation: RS232 and file generating instruments. In this project, file based instrument integration is used.

LabStation comes with a complete library of parsing functions that allows users to use a parsing script for each type of instrument interface. It also provides a number of pre-built parsing scripts for many popular brands of instruments. Once the result has been acquired, LabStation can apply a user-defined set of parsing instructions to parse out the key result information that LIMS is expecting from the instrument.

CHAPTER 2

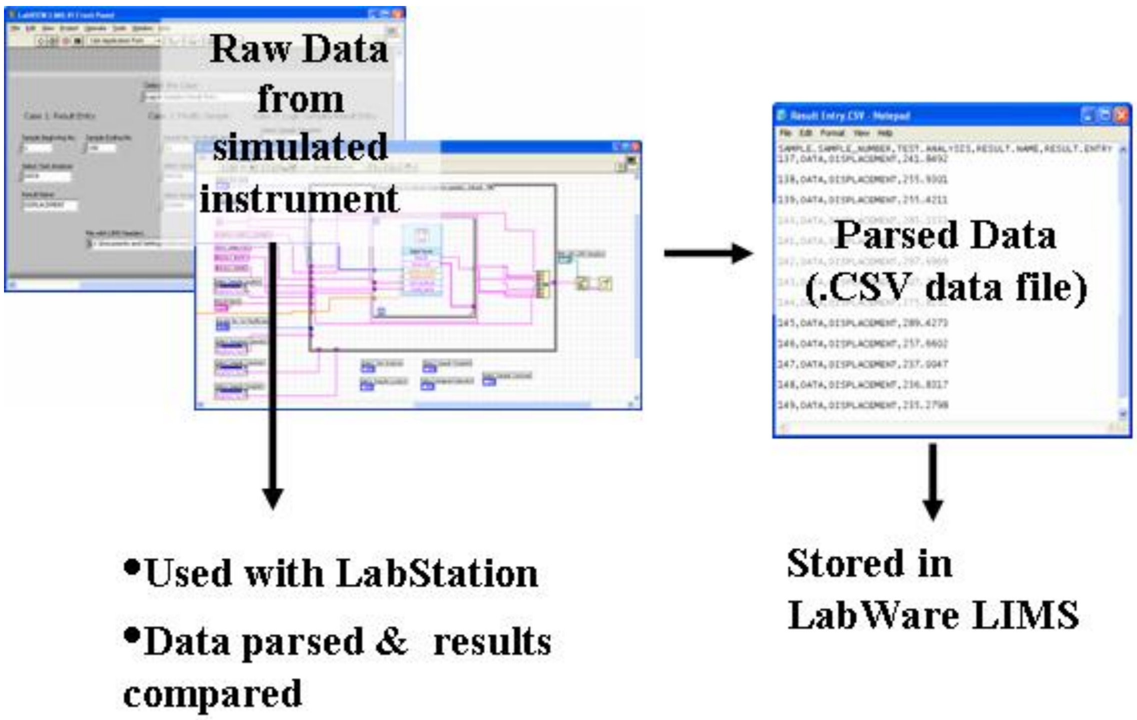
BACKGROUND

Laboratory Information Management Systems (LIMS) are information management systems designed specifically for the analytical laboratory. This includes research and development (R&D) labs, in-process testing labs, quality assurance (QA) labs, and more.

Originally, LIMS were developed in-house by organizations wishing to streamline their data acquisition and reporting processes. In-house developed LIMS take considerable time and resources to implement, therefore the need for a more immediate solution helped drive LIMS to the next stage in the 1970s. Custom-built systems became available during this time and were designed by independent systems development companies to run in specific laboratories. These commercial systems, while typically developed for a particular industry (such as the pharmaceutical industry), still required considerable customization to meet a specific laboratory's needs. In 1990s open systems emphasizing user configurability were launched. Today's commercial LIMS offer a high degree of flexibility and functionality.

The rise of Informatics, coupled with the increasing speed and complexity of the analytical instruments, is driving more sophisticated data manipulation and warehousing tools that work hand-in-glove with LIMS to manage and report laboratory data with ever greater accuracy and efficiency. Data collection from various instruments is a routine for most of the laboratories.

The purpose of this thesis project is to create an interface between LabVIEW and LabWare LIMS. LabVIEW program will be used to simulate the data from the instrument. The raw data will be parsed, and thus getting meaningful information that can be stored in LIMS. LabStation is also used to parse the same raw data file generated by LabVIEW, and therefore validating the results coming from LabVIEW (Figure 6).


Raw Data from simulated instrument
Parsed Data (.CSV data file)
Result Entry.CSV - Notepad
File Edit Format View Help
•Used with LabStation
•Data parsed & results compared
Stored in LabWare LIMS

```
SAMPLE.SAMPLE_NUMBER,TEST.ANALYSIS,RESULT.NAME,RESULT.ENTRY
137,DATA,DISPLACEMENT,241.8492
138,DATA,DISPLACEMENT,215.9301
139,DATA,DISPLACEMENT,211.4211
140,DATA,[illegible]
141,DATA,[illegible]
142,DATA,DISPLACEMENT,297.6969
143,[illegible]
144,[illegible]
145,DATA,DISPLACEMENT,289.4273
146,DATA,DISPLACEMENT,257.6602
147,DATA,DISPLACEMENT,237.0047
148,DATA,DISPLACEMENT,236.8317
149,DATA,DISPLACEMENT,215.2798
```

Figure 6: Project Outline

CHAPTER 3

METHODS

LabVIEW™ (Laboratory **V**irtual **I**nstrument **E**ngineering **W**orkbench**)**

LabVIEW™ version 8.0 was used to build the LabVIEW-LIMS.VI. A LabVIEW™ program – virtual instrument (VI) is written to produce random data that can be used with LabWare LIMS. The VI has two significant parts. The first part of the VI creates an example to acquire data from an instrument. In the first part, the VI calculates the random path and the displacement followed by a particle (Figure 7).

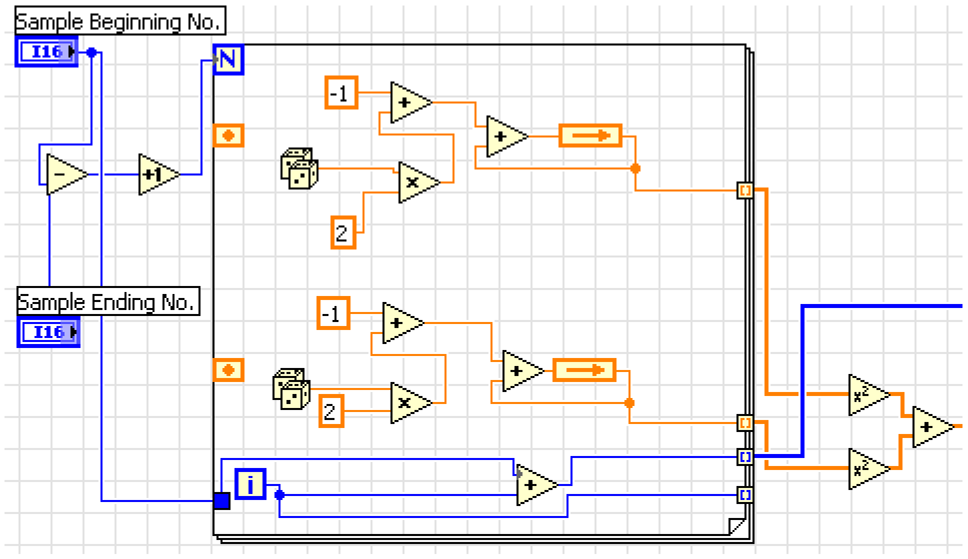


Figure 7: Block Diagram of the VI calculating random path and displacement of a particle

Since this data is to be used and stored in a LIMS system, it has to be formatted in a way such that it can be parsed and read by LabWare LIMS. Therefore, in the second part of the VI, the data is made compatible with LabWare LIMS system. The data calculated in the first part of the VI is parsed, and written on a comma separated value

(CSV) file, usually depicted as .CSV file. The details of parsing the data in LabVIEW are discussed later in the chapter.

Following three different cases are defined in the second part of the VI. The data is parsed depending on the case selected by the user.

Case 1: Result Entry

Case 2: Modify Sample

Case 3: Login Samples to Result Entry

The details of each case are discussed below:

Case 1: Result Entry

This case is selected if the user wants to assign tests and enter results for a range of *defined* sample numbers. In this case, the samples should be logged in manually by the user in LabWare LIMS. Before executing the VI, the user needs to define few field values which will be used by the program to parse the raw data, so that it is ready to store in LIMS. The user needs to define the sample beginning and the ending number for the result entry, select the 'test analysis' from the list on the front panel, and define the file path where the formatted data file will be saved. The test analysis 'LABV_DATA' and 'DATA' are predefined in LabWare LIMS. The test analysis – DATA is used by default. The sample beginning and ending numbers and the file path can be defined on the front panel. The result name is also needed to import the results in LIMS. In this case, the result name is set to default - 'DISPLACEMENT' (because the calculation is not changing in any case, and it's known that displacement results are imported). The final form of the data file is shown in Figure 14. The screen shot of the block diagram for this case is shown in Figure 8.

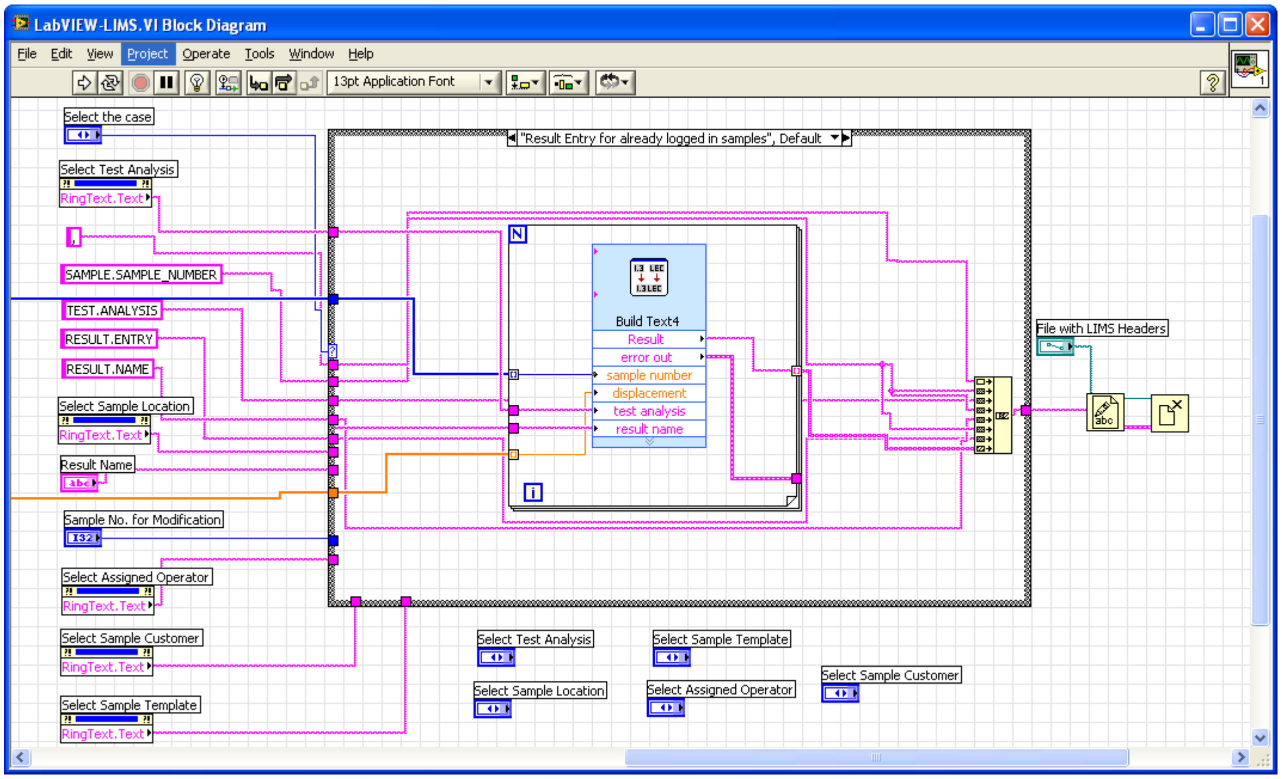


Figure 8: Screen shot of the block diagram for Case 1: Result Entry

Case 2: Modify Sample

This case is selected if the user wants to modify any sample that has already been logged in LabWare LIMS. In this case, the sample location and/or the assigned operator can be modified by defining the sample number. Before executing the VI, the user needs to define few field values which will be used by the program to parse the raw data, so that it is ready to store in LIMS. The user needs to define the sample number that is to be modified, select the 'sample location', the 'assigned operator' from the list on the front panel, and define the file path where the formatted data file will be saved. The Sample Locations: WK316 and Laboratory; Assigned Operator: System, Swati, Lab User; are predefined in LabWare LIMS. There can be a case where the user wants to modify only the sample location and not the assigned operator. In this case the user can select the new sample location from the front panel, and keep the assigned operator unchanged. No field value can be left blank. The screen shot of the block diagram for this case is shown in figure 9. The final form of the data file is shown in figure 16.

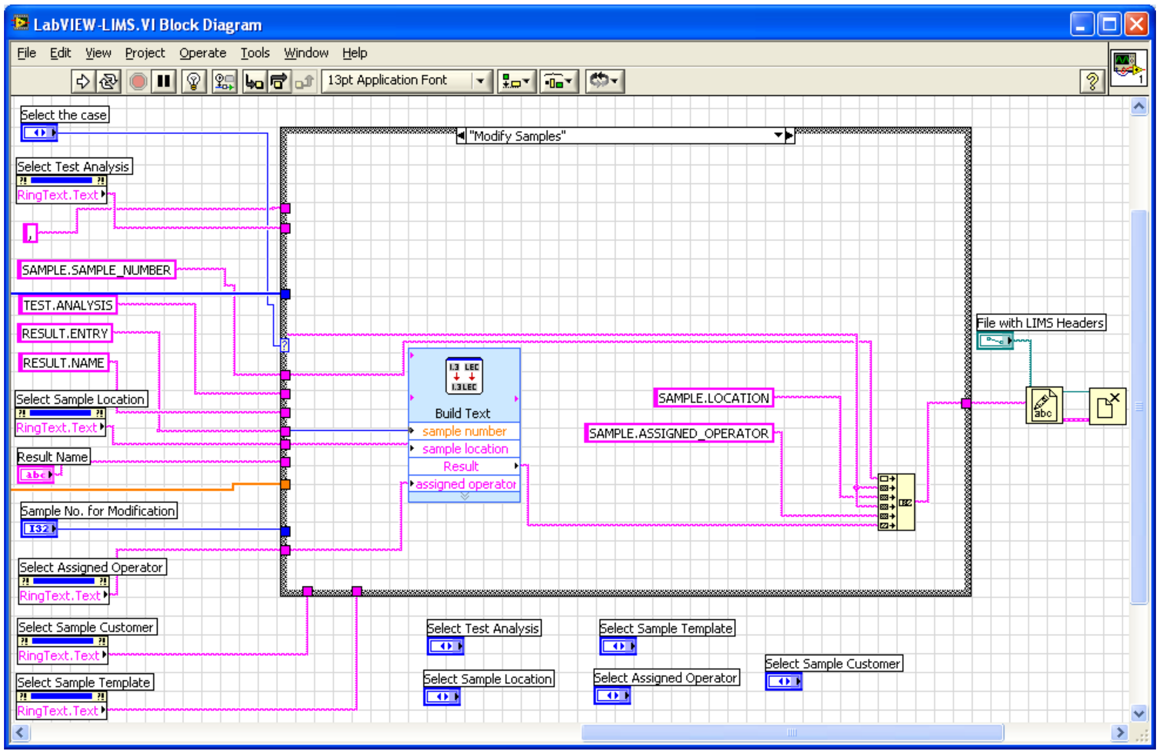


Figure 9: Screen shot of the block diagram for Case 2: Modify Sample

Case 3: Login Samples to Result Entry

By selecting this case, the user can perform major functions with one .CSV extension file. The user can log a sample, assign tests, and enter results using one data file. The user needs to define the sample text_Id, select the sample template, sample customer, test analysis from the list on the front panel, and define the file path where the formatted data file will be saved before executing the VI.

Since the sample does not yet exist, proxy identification of the text id for the sample is required in order to specify which records belong to same sample. The Sample

Customer: WK316 and LAB; Sample Template: LV_DATA, TRY; Test Analysis: DATA is predefined in LabWare LIMS. The result name (DISPLACEMENT) is set to default, and the Result Entry is the actual data. The information entered before executing the VI is used to parse the raw data, such that it is ready to be stored in LIMS. The screen shot of the block diagram for this case is shown in figure 10. The final form of the data file is shown in figure 18.

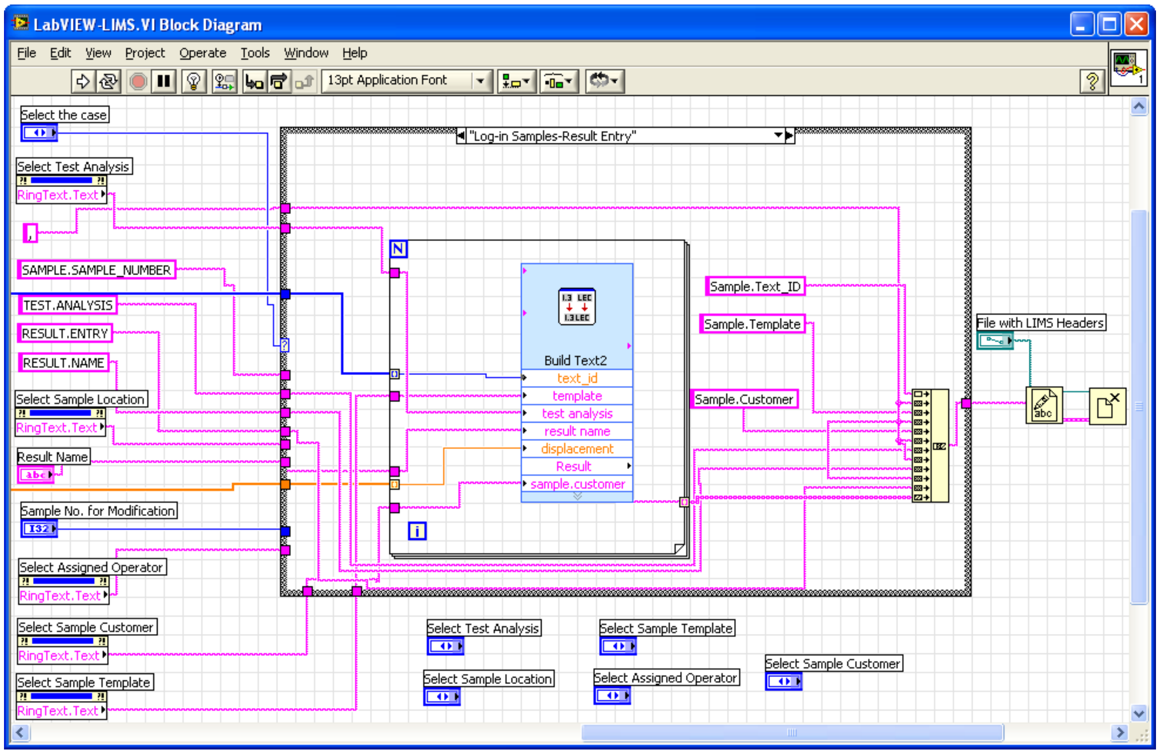
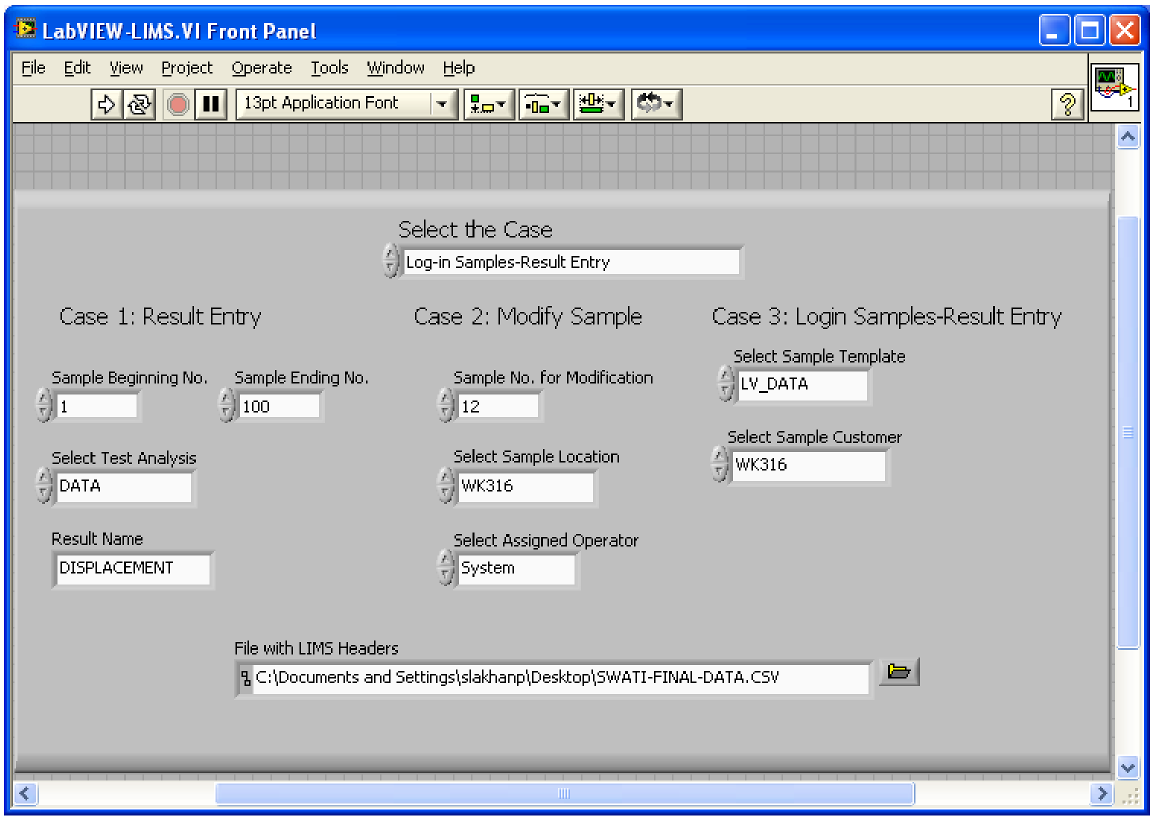


Figure 10: Screen shot of the block diagram for Case 3: Login Samples-Result Entry

Figure 11 shows the screenshot of the front panel of LabVIEW-LIMS.VI. It can be seen from the screen shot that the user can select the case from the front panel and fill the required field values. The field values like sample beginning and ending number, sample number for modification, test analysis, sample customer, sample location etc. can be entered in the front panel.


LabVIEW-LIMS.VI Front Panel
File Edit View Project Operate Tools Window Help
13pt Application Font

Select the Case
Log-in Samples-Result Entry

Case 1: Result Entry

Sample Beginning No. 1

Sample Ending No. 100

Select Test Analysis DATA

Result Name DISPLACEMENT

Case 2: Modify Sample

Sample No. for Modification 12

Select Sample Location WK316

Select Assigned Operator System

Case 3: Login Samples-Result Entry

Select Sample Template LV_DATA

Select Sample Customer WK316

File with LIMS Headers
C:\Documents and Settings\slakhanp\Desktop\SWATI-FINAL-DATA.CSV

Figure 11: Front panel of the LabVIEW-LIMS.VI

Details of the Data file: As mentioned before, the raw data is parsed such that it can be read and imported to LabWare LIMS. The procedure and approach used to import the results file is discussed in the next section. The data file is formatted; the headers are added to the .CSV file and saved by the VI in the directory from where it can be imported to LabWare LIMS. The first line of the file contains the table and field names separated by commas. For example, Sample.Sample_Number is used to indicate the sample number field in the sample table. The lines following the first line contain the data corresponding to the table and field names where data is separated by commas.

Three different headers are used for the three cases, and therefore, there is a different file format for each case (Refer Table 1).

Cases		File Headers
Case 1	Result Entry	Sample.Sample_Number,Test.Analysis,Result.Name,Result.Entry
Case 2	Modify Sample	Sample.Sample_Number,Sample.Location,Sample.Assigned_Operator
Case 3	Log-in Samples to Result Entry	Sample.Text_ID,Sample.Template,Sample.Customer, Test.Analysis,Result.Name,Result.Entry

Table 1: File headers for each of the three cases

LabWare LIMS

LabWare LIMS version 5.0 is used to store the results from LabVIEW^{TM}. The data is imported using the .CSV data file from LabVIEW^{TM}. The user can perform major functions using one file. Also, results can be entered automatically in one instance. As mentioned before, some field values are pre-defined in LabWare LIMS, and the same field values are used in LabVIEW so that the data can be integrated. The following table shows the field names that are pre-defined in LabWare LIMS (Table 2).

Importing Files to LabWare LIMS

The files can be imported in two ways into LIMS, either, by manually selecting the file path each time we want to import the data, or automating this process.

To import the files manually, select the desired file. The import files must have an extension of .CSV. The selected file will be parsed and the data will be displayed in the Import File Dialog. If an error occurs during parsing, it will be displayed in the Error Message field. If the operation finishes successfully, the .CSV file is renamed with an extension of .ARK and located in the same directory as the .CSV file. If an error occurs on any record in the file, the entire operation fails and no data is imported.

TABLE NAME	**FIELDS**
Analysis	DATA
Analysis Type	READ_DATA
Common Name	LABVIEW-DATA
Customers	LAB, WK316
Instruments	DAQ_INST, LABVIEW, LV-LIMS
LabStation	LAB-LAKH
Lists	INST_GRPS – LV driven DAQ inst.
Parsing Scripts	MY_SCRIPT
Sample Login Template	LV_DATA, TRY
Test List	LABV_DATA
Sample ID Configuration	Sample Number

Table 2: Pre-defined fields in LabWare LIMS

The .CSV file is renamed with an extension of .ERR and another file is created with an extension of .LOG, which contains the error message. The importing of files can be set as an automatic process on any LIMS client PC. To set a client PC for automatic importing, the following steps were followed:

1. Select “System” from the “Configure” menu, then Select “Client”.
2. Set the “File Import Directory” to the drive and path where the CSV files will be located (in this project: C:\LW-LIMS-V5\LW-LIMS_LAKHANPAL)
3. Set “Import files” to ‘Yes’ for automatic operation.
4. Click on OK to confirm the settings.

LIMS client PC will now poll for CSV files in the background and process them without user prompting or interaction. The activity is indicated by the status pane of the main menu. Again, the completion of any importing can be verified by the CSV file being renamed with an ARK extension, or ERR and LOG extensions.

File Import Logics

Results can be imported by using different logics and result values. In this project, the results are imported using three different file formats. Now we discuss the logic behind setting result values using three different the import file structures.

Case 1: Result Entry

The following flow chart represents the logic behind importing results using Sample.Sample_Number, Test.Analysis, Result.Name, Result.Entry (Figure 12).

The import parser must distinguish between records that are intended to update information on a sample. If the Sample Number is valid, then the fields will be updated on the specified sample. If the Sample Number is not valid, an error will be generated. Similarly, when importing the data in the above mentioned format, it should be made sure that the test details exist and that the right tests are added to the right sample. Each test is identified by Test.Analysis field. In this case, Test.Analysis is used for the results entry; therefore, a match is first attempted on the Analysis Names on the sample, then on the Alias Names of the Analyses on the Sample. Also for the result entry data, each result must be identified by the Result.Name field. Anytime the Result.Name is specified, a match is first attempted on the result names on the tests assigned to that sample, then on the Alias Names of the results on the tests assigned to that sample.

If the Result.Name is not found then the sample is searched for a result that has a result alias that matches the name in the import file. If the Result.Name does not match any of the existing results on the specified test, it is added to the test as an ad-hoc result if the matching test has a status of incomplete. If the Result.Name does not match any of the existing results on the specified test and all matching tests have status of complete then a new test matching the name in the import file is added, and an ad-hoc result is added to that test and selected for update. The new ad-hoc result added to the test will not be reportable. It can be made reportable by adding the Result.Reportable field to the import file and setting its value to ‘T’.

Case 2: Modify Sample

The import parser must distinguish between records that are intended to update information on a sample. If the Sample Number is valid, then the fields (Sample Location, Assigned Operator) will be updated on the specified sample. If the Sample Number is not valid, an error will be generated.

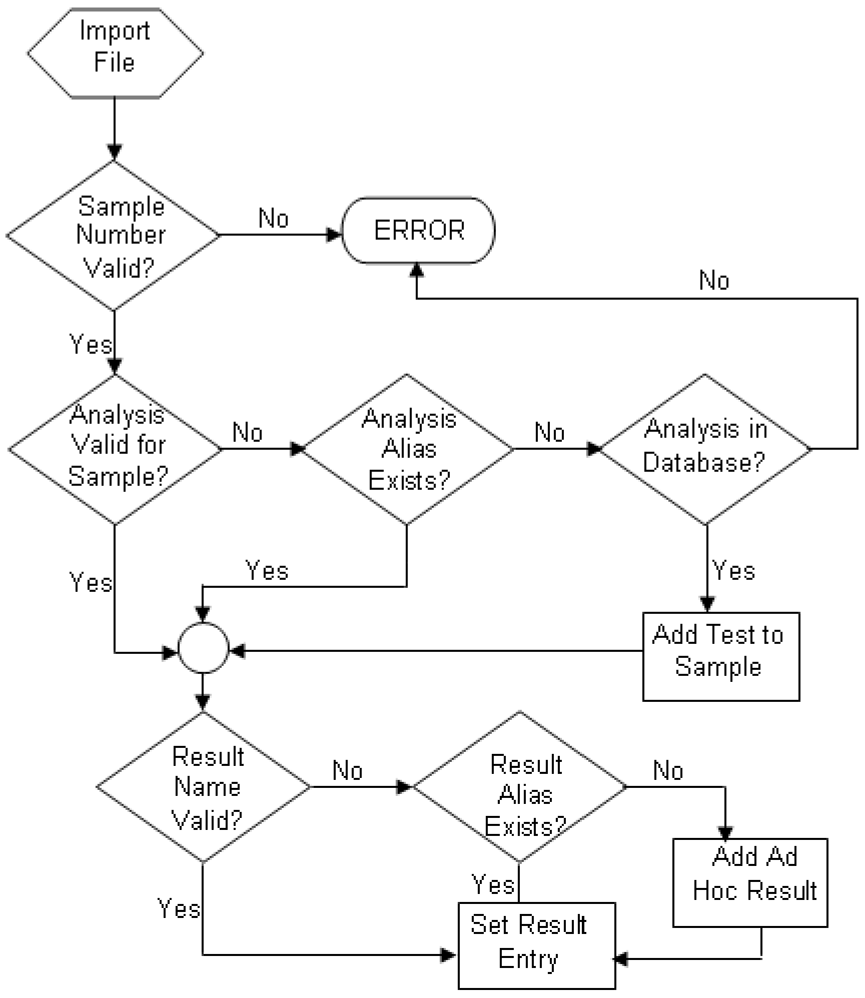


Figure 12: Flow chart representing logic behind importing results using Sample.Sample_Number, Test.Analysis, Result.Name, Result.Entry

Case 3: Login Samples to Result Entry

The flow chart in figure 13 represents the logic behind importing results using Sample.Text_Id, Sample.Template, Test.Analysis, Result.Name, Result.Entry. Since in this case, the sample does not yet exists, proxy identification of the text id for the sample is required in order to specify which records belong to the sample. Either its Sample Number or Text ID must identify each sample. If the Text ID is valid, then all remaining fields will update information on the specified sample. These fields can be sample, test, or result fields. If a sample template has been specified, then the record will log a new sample. If a sample is being logged and a Text ID was specified, this Text ID will be used for the new sample if the sample template contains the Text ID field as a user entry field. Otherwise, the Text ID is treated as proxy identification for the records in the import file and is only used for that purpose. In either case, the import parser grid will display the actual Text ID for the new sample. If neither Sample Number nor Text ID is specified, and a sample template is not specified, an error is generated. Each test is identified by Test.Analysis field. In this case, Test.Analysis is used for the results entry, therefore, a match is first attempted on the Analysis Names on the sample, then on the Alias Names of the Analyses on the Sample. Also for the result entry data, each result must be identified by the Result.Name field. Anytime the Result.Name is specified, a match is first attempted on the result names on the tests assigned to that sample, then on the Alias Names of the results on the tests assigned to that sample. If the Result.Name is not found then the sample is searched for a result that has a result alias that matches the name in the import file. If the Result.Name does not match any of the existing results on the specified test, it is added to the test as an ad-hoc result if the matching test has a status of incomplete. If the Result.Name does not match any of the existing results on the specified test and all matching tests have status of complete then a new test matching the name in the import file is added, and an ad-hoc result is added to that test and selected for update. The new ad-hoc result added to the test will not be reportable. It can be made reportable by adding the Result.Reportable field to the import file and setting its value to ‘T’. It should be made sure that the sample status must not be canceled, authorized or rejected for any of the case, because no change is permitted to these samples.

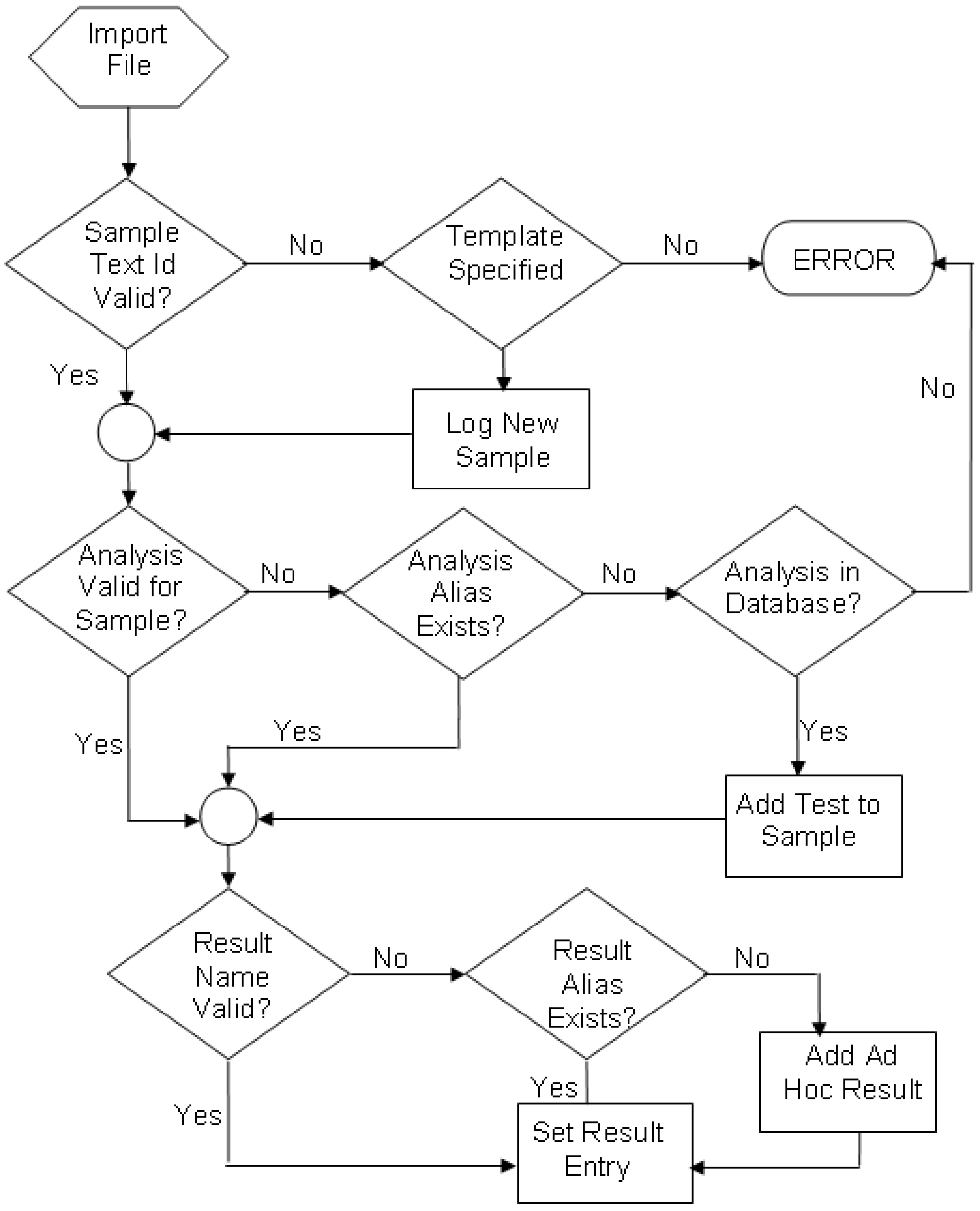


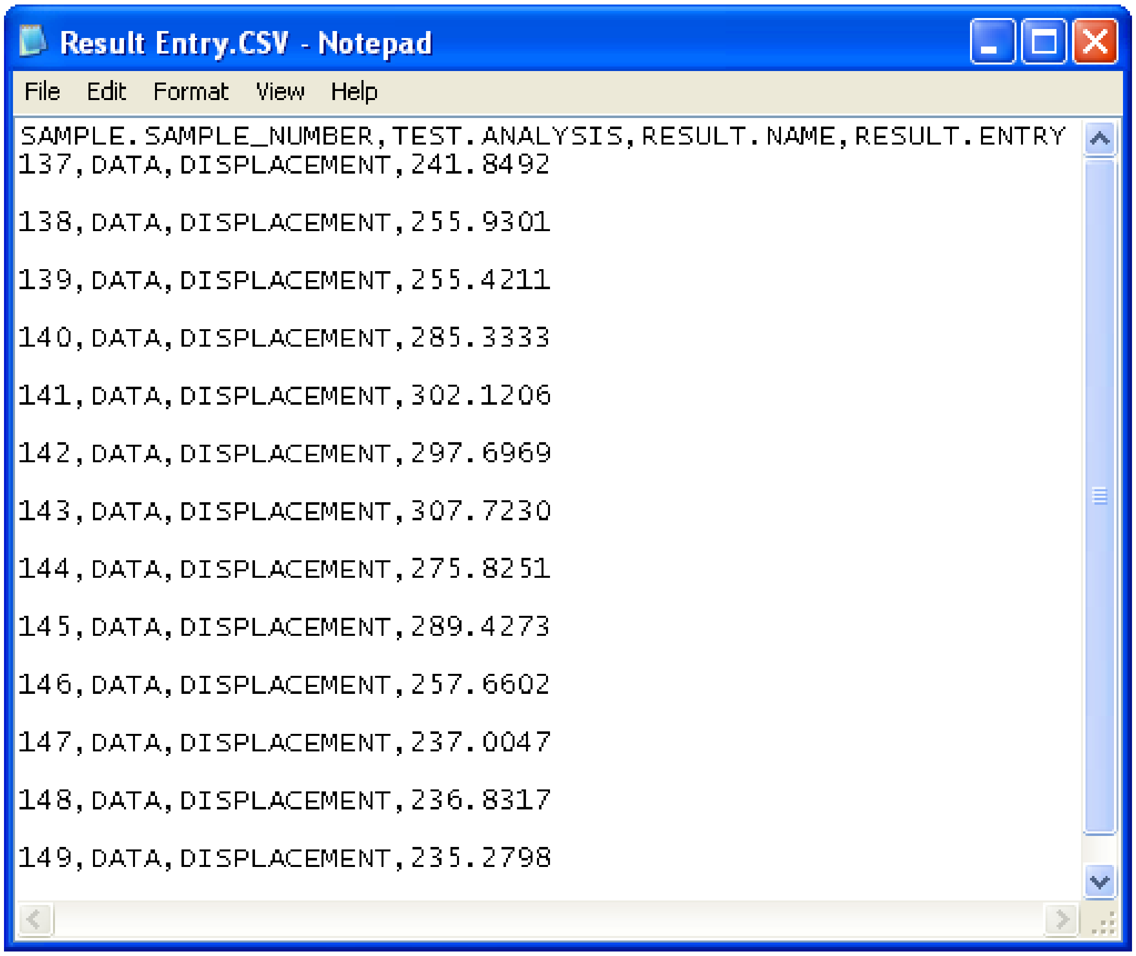
Figure 13: Flow chart representing the logic behind importing results using Sample.Text_Id, Sample.Template, Test.Analysis, Result.Name, Result.Entry

CHAPTER 4

RESULTS

LabVIEW^{TM} & LabWare LIMS

Figure 14 shows the results of the .CSV data file for Case 1: Result Entry. The information entered in the front panel, like Sample beginning number, Sample ending number, Test Analysis, Result Name is used to parse the raw data generated by LabVIEW.



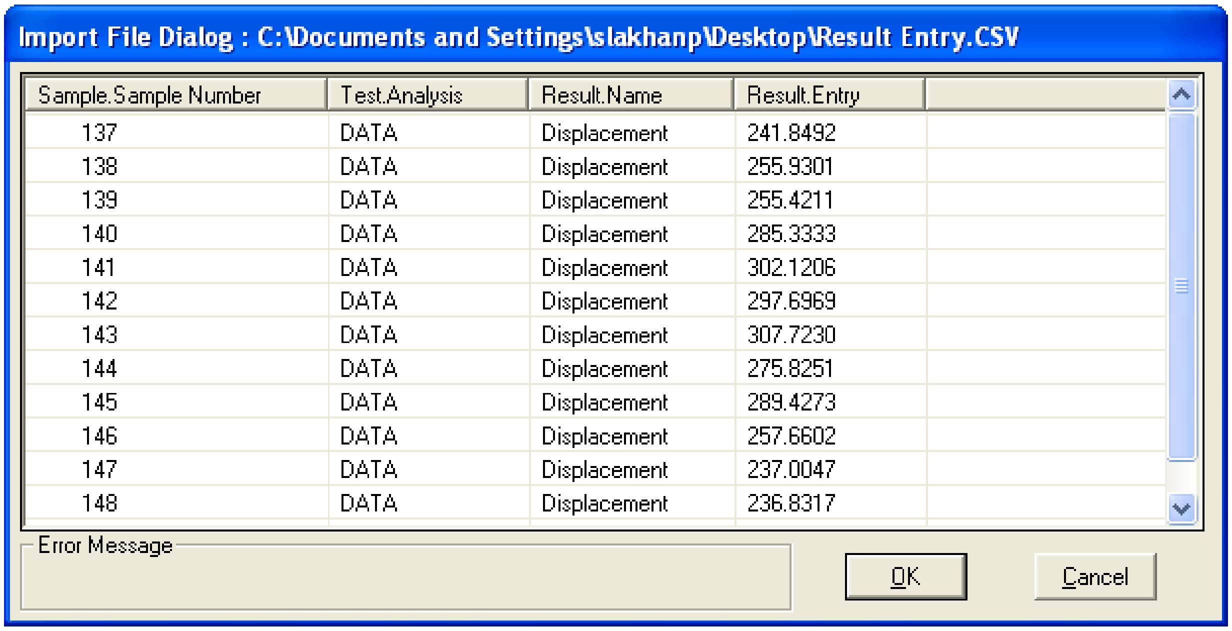
```
SAMPLE.SAMPLE_NUMBER,TEST.ANALYSIS,RESULT.NAME,RESULT.ENTRY
137,DATA,DISPLACEMENT,241.8492

138,DATA,DISPLACEMENT,255.9301

139,DATA,DISPLACEMENT,255.4211

140,DATA,DISPLACEMENT,285.3333

141,DATA,DISPLACEMENT,302.1206

142,DATA,DISPLACEMENT,297.6969

143,DATA,DISPLACEMENT,307.7230

144,DATA,DISPLACEMENT,275.8251

145,DATA,DISPLACEMENT,289.4273

146,DATA,DISPLACEMENT,257.6602

147,DATA,DISPLACEMENT,237.0047

148,DATA,DISPLACEMENT,236.8317

149,DATA,DISPLACEMENT,235.2798
```

Figure 14: CSV data file for Case 1: Result Entry

The following field values were entered in the front panel:

Sample Beginning Number = 137, Sample Ending Number = 149,

Test Analysis = DATA, Result Name = Displacement (it is set to default).



Import File Dialog : C:\Documents and Settings\slakhanp\Desktop\Result Entry.CSV

Sample.Sample Number	Test.Analysis	Result.Name	Result.Entry
137	DATA	Displacement	241.8492
138	DATA	Displacement	255.9301
139	DATA	Displacement	255.4211
140	DATA	Displacement	285.3333
141	DATA	Displacement	302.1206
142	DATA	Displacement	297.6969
143	DATA	Displacement	307.7230
144	DATA	Displacement	275.8251
145	DATA	Displacement	289.4273
146	DATA	Displacement	257.6602
147	DATA	Displacement	237.0047
148	DATA	Displacement	236.8317

Figure 15: File Import Grid for Case 1: Result Entry

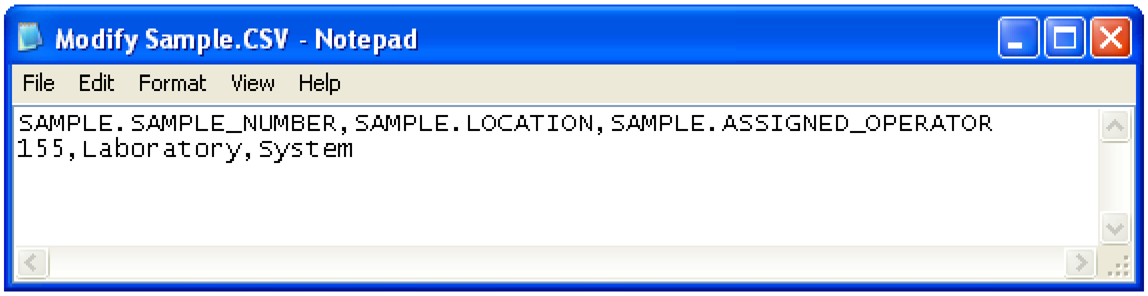
The data file shown in figure 14 is imported to LabWare LIMS. Figure 15 shows the Import grid of LabWare LIMS for Case 1. There is no ‘Error Message’; therefore, the results are ready to be stored in the LIMS.

Figure 16 shows the results of the .CSV data file for Case 2: Modify Sample. The information entered in the front panel, like Sample number for modification, Sample Location, Assigned Operator is used to parse the raw data.

The following field values were entered in the front panel:

Sample Number for Modification = 155, Sample Location = Laboratory,

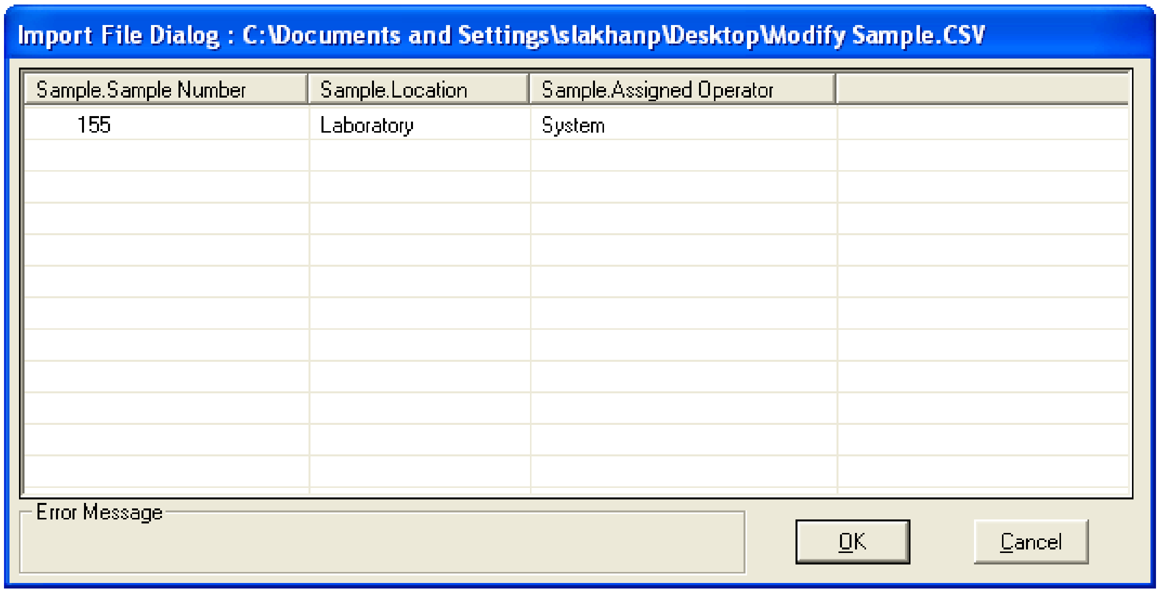
Assigned Operator = System.



```
SAMPLE.SAMPLE_NUMBER,SAMPLE.LOCATION,SAMPLE.ASSIGNED_OPERATOR
155,Laboratory,System
```

Figure 16: CSV data file for Case 2: Modify Sample

The data file shown in figure 16 is imported to LabWare LIMS and the Import Grid is shown in figure 17. There is no ‘Error Message’; therefore, the results are ready to be stored in the LIMS.

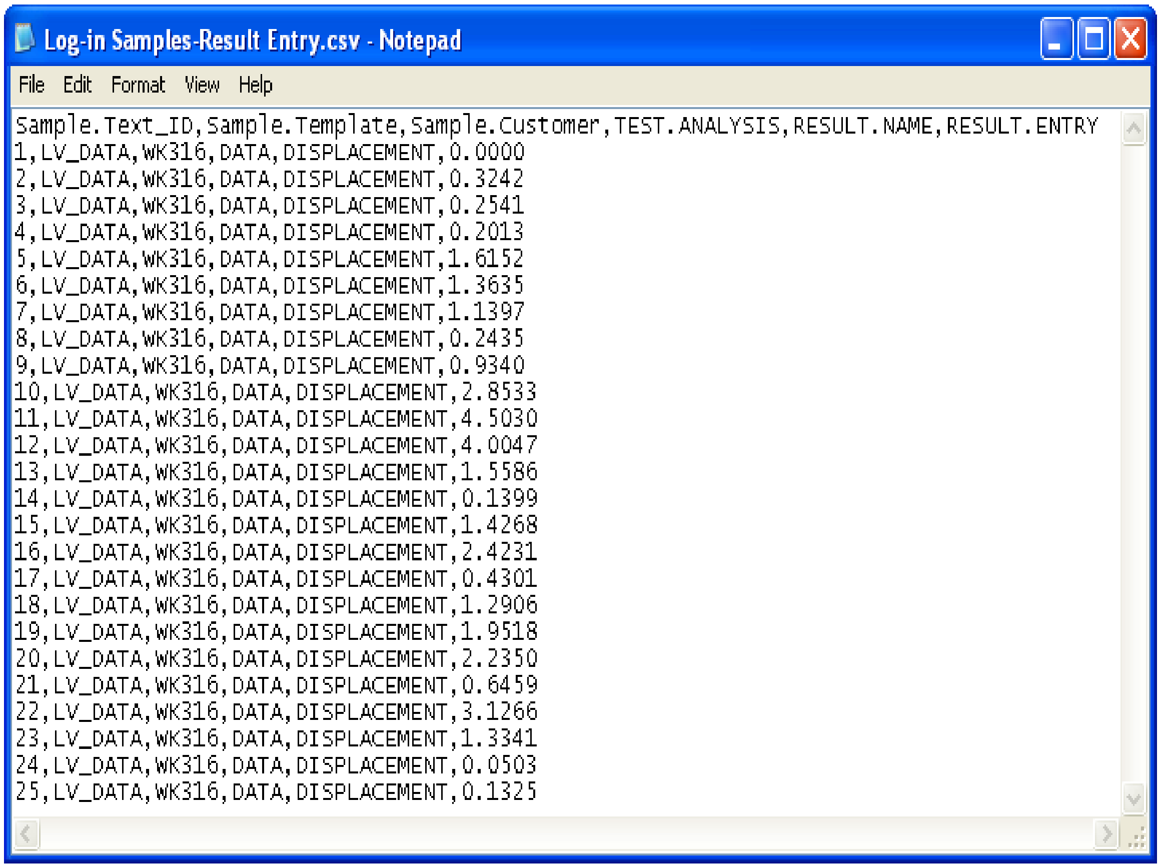

Import File Dialog : C:\Documents and Settings\slakhanp\Desktop\Modify Sample.CSV
Error Message
OK
Cancel

Sample.Sample Number	Sample.Location	Sample.Assigned Operator
155	Laboratory	System

Figure 17: File Import Grid for Case 2: Modify Sample

Figure 18 shows the .CSV data file for Case 3: Login Samples-Result Entry. The following field values were entered in the front panel and where used to parse the raw data:

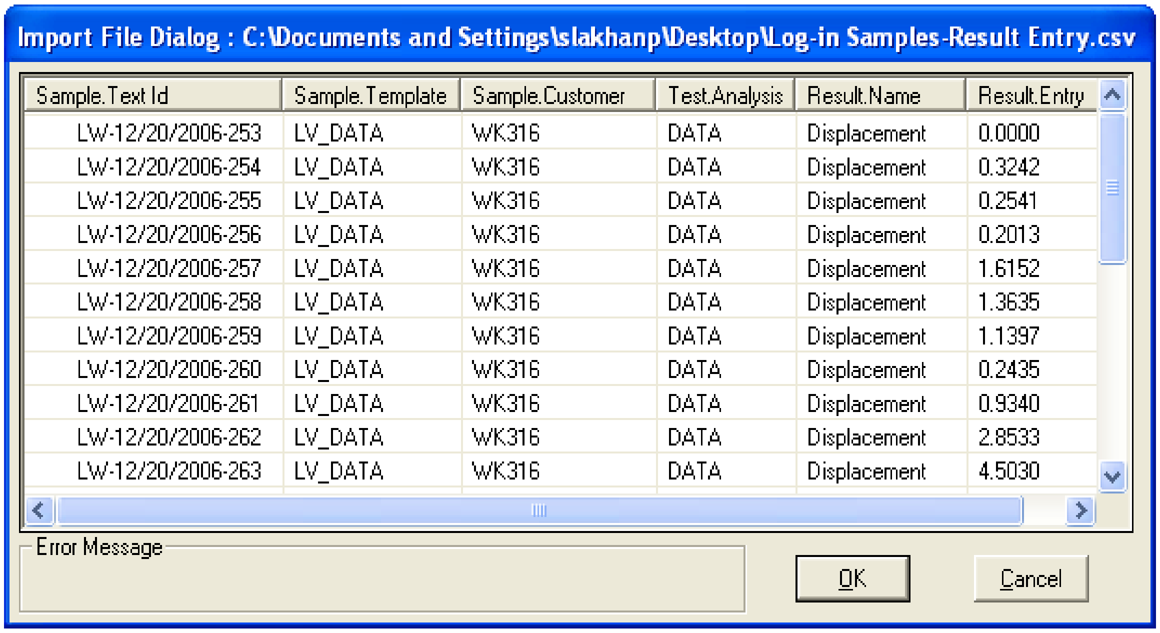
Sample Beginning Number = 1, Sample Ending Number = 25, Sample Template = LV_DATA, Sample Customer = WK316, Test Analysis = DATA, Result Name = Displacement.


Log-in Samples-Result Entry.csv - Notepad
File Edit Format View Help

```
Sample.Text_ID,Sample.Template,Sample.Customer,TEST.ANALYSIS,RESULT.NAME,RESULT.ENTRY
1,LV_DATA,WK316,DATA,DISPLACEMENT,0.0000
2,LV_DATA,WK316,DATA,DISPLACEMENT,0.3242
3,LV_DATA,WK316,DATA,DISPLACEMENT,0.2541
4,LV_DATA,WK316,DATA,DISPLACEMENT,0.2013
5,LV_DATA,WK316,DATA,DISPLACEMENT,1.6152
6,LV_DATA,WK316,DATA,DISPLACEMENT,1.3635
7,LV_DATA,WK316,DATA,DISPLACEMENT,1.1397
8,LV_DATA,WK316,DATA,DISPLACEMENT,0.2435
9,LV_DATA,WK316,DATA,DISPLACEMENT,0.9340
10,LV_DATA,WK316,DATA,DISPLACEMENT,2.8533
11,LV_DATA,WK316,DATA,DISPLACEMENT,4.5030
12,LV_DATA,WK316,DATA,DISPLACEMENT,4.0047
13,LV_DATA,WK316,DATA,DISPLACEMENT,1.5586
14,LV_DATA,WK316,DATA,DISPLACEMENT,0.1399
15,LV_DATA,WK316,DATA,DISPLACEMENT,1.4268
16,LV_DATA,WK316,DATA,DISPLACEMENT,2.4231
17,LV_DATA,WK316,DATA,DISPLACEMENT,0.4301
18,LV_DATA,WK316,DATA,DISPLACEMENT,1.2906
19,LV_DATA,WK316,DATA,DISPLACEMENT,1.9518
20,LV_DATA,WK316,DATA,DISPLACEMENT,2.2350
21,LV_DATA,WK316,DATA,DISPLACEMENT,0.6459
22,LV_DATA,WK316,DATA,DISPLACEMENT,3.1266
23,LV_DATA,WK316,DATA,DISPLACEMENT,1.3341
24,LV_DATA,WK316,DATA,DISPLACEMENT,0.0503
25,LV_DATA,WK316,DATA,DISPLACEMENT,0.1325
```

Figure 18: CSV data file for Case 3: Login Samples-Result Entry

The data file shown in figure 18 is imported to LabWare LIMS and the Import Grid is shown in figure 19. There is no ‘Error Message’; therefore, the results are ready to be stored in the LIMS.



Import File Dialog : C:\Documents and Settings\slakhanp\Desktop\Log-in Samples-Result Entry.csv

Sample.Text Id	Sample.Template	Sample.Customer	Test.Analysis	Result.Name	Result.Entry
LW-12/20/2006-253	LV_DATA	WK316	DATA	Displacement	0.0000
LW-12/20/2006-254	LV_DATA	WK316	DATA	Displacement	0.3242
LW-12/20/2006-255	LV_DATA	WK316	DATA	Displacement	0.2541
LW-12/20/2006-256	LV_DATA	WK316	DATA	Displacement	0.2013
LW-12/20/2006-257	LV_DATA	WK316	DATA	Displacement	1.6152
LW-12/20/2006-258	LV_DATA	WK316	DATA	Displacement	1.3635
LW-12/20/2006-259	LV_DATA	WK316	DATA	Displacement	1.1397
LW-12/20/2006-260	LV_DATA	WK316	DATA	Displacement	0.2435
LW-12/20/2006-261	LV_DATA	WK316	DATA	Displacement	0.9340
LW-12/20/2006-262	LV_DATA	WK316	DATA	Displacement	2.8533
LW-12/20/2006-263	LV_DATA	WK316	DATA	Displacement	4.5030

Error Message

Figure 19: File Import Grid for Case 3: Login Samples-Result Entry

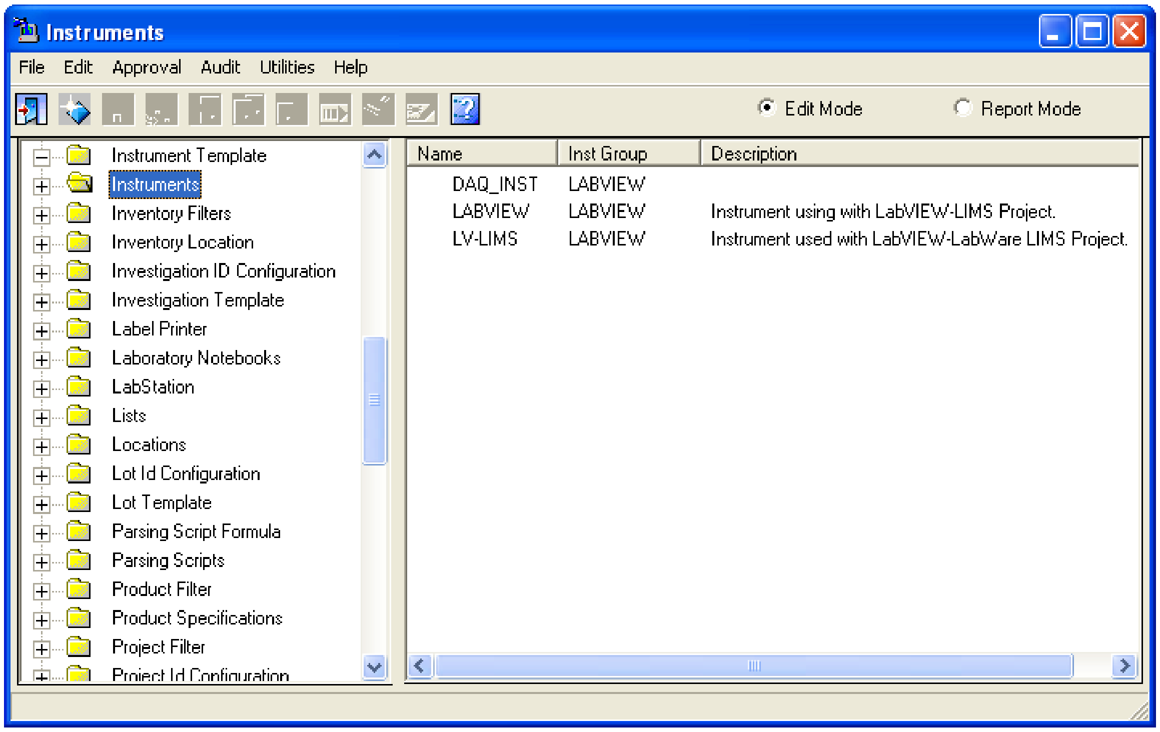
If an error occurs during parsing, it will be displayed in the Error Message field. If the operation finishes successfully, the .CSV file is renamed with an extension of .ARK and located in the same directory as the .CSV file. If an error occurs on any record in the file, the entire operation fails and no data is imported. The .CSV file is renamed with an extension of .ERR and another file is created with an extension of .LOG, which contains the error message.

LabStation^{TM}

The main purpose of using LabStation is to validate the results from LabVIEW. The raw data generated by LabVIEW-LIMS.VI is used with LabStation, and the standard procedure as explained below is followed to parse the raw data in LabStation. LabStation has a huge library of parsing commands that can be used to parse the data collected from various instruments connected to LabWare LIMS. Therefore, the parsed output file from

LabStation should be similar to the parsed output file generated by LabVIEW for proper validation.

File based instrument integration is used with LabWare LIMS. The instrument is added to the instrument table in LIMS. Figure 20 shows the screenshot for the instruments defined in LIMS. Instruments: DAQ_INST, LABVIEW, LV-LIMS are already added to the LIMS instrument table. 'Instrument Group: LABVIEW' was defined for each instrument defined in LIMS. The 'Instrument Group' is used to tag the information with LabStation to configure the instrument.



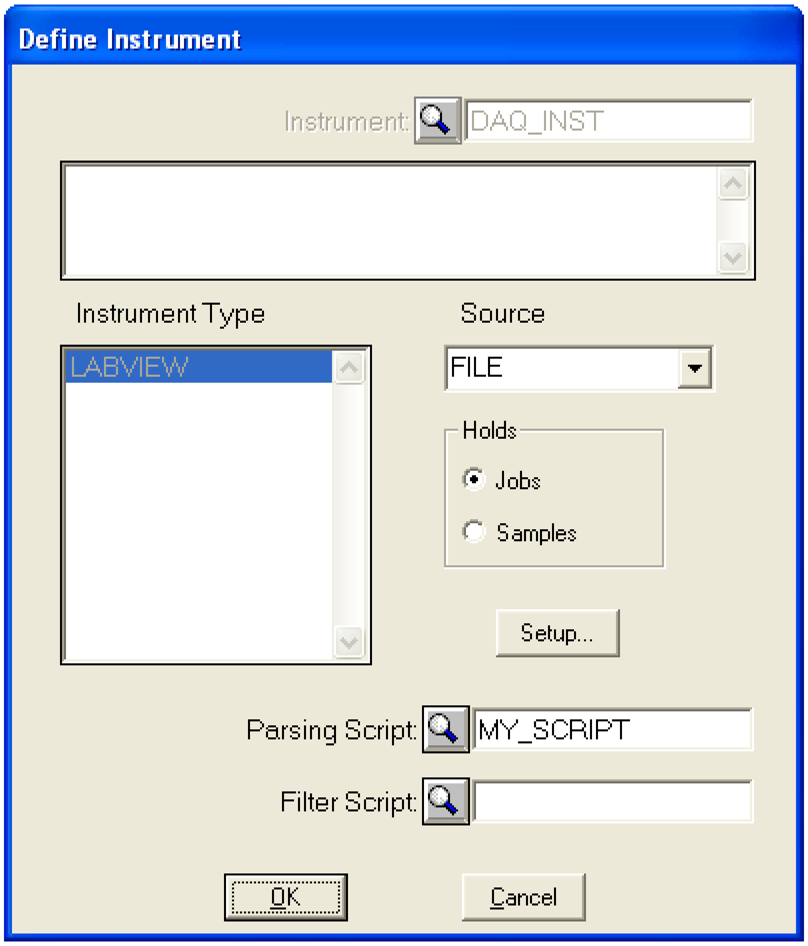
Name	Inst Group	Description
DAQ_INST	LABVIEW	
LABVIEW	LABVIEW	Instrument using with LabVIEW-LIMS Project.
LV-LIMS	LABVIEW	Instrument used with LabVIEW-LabWare LIMS Project.

Figure 20: Screen-Shot for the Instruments added to LabWare LIMS

'Instrument Group' was defined for each instrument defined in LIMS. The 'Instrument Group' is used to tag the information with LabStation to configure the instrument. This Instrument is now configured in LabStation. Select 'Instrument

Configuration' and add the instrument (Figure 21). Instrument Type is defined as 'LABVIEW', the source is selected as 'FILE', the instrument holds 'Jobs', parsing script-'MY_SCRIPT' is selected.

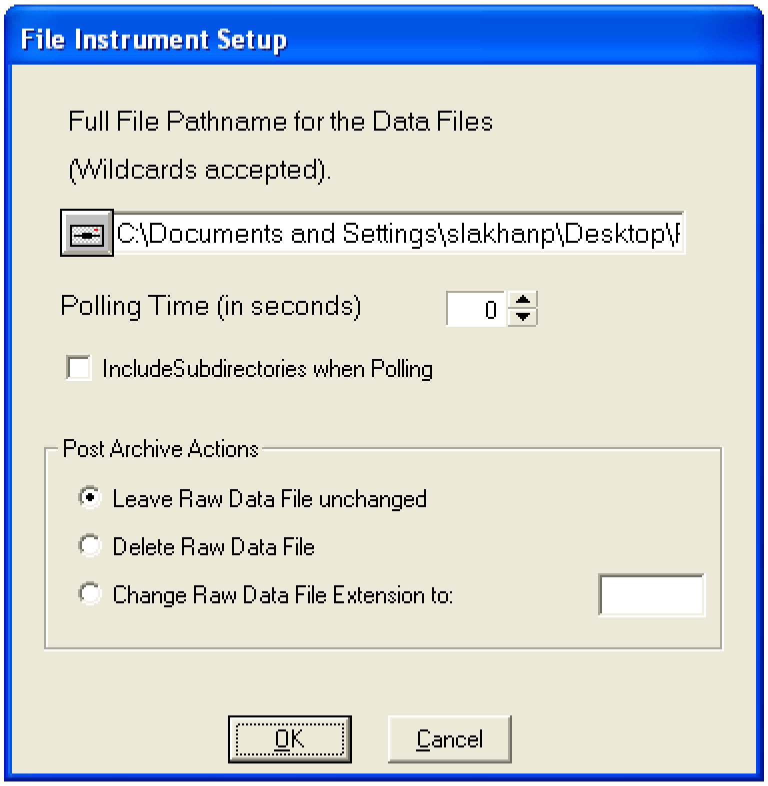
Since the source is defined as 'File', the user also needs to further define the 'File Instrument Setup'. In the file instrument setup, the full path name for the data file is defined and the polling time is set to 0 (Figure 22).



Instrument:	DAQ_INST
Instrument Type	LABVIEW
Source	FILE
Holds	[✓ Jobs] Samples
Parsing Script:	MY_SCRIPT
Filter Script:	

Figure 21: Instrument Configuration in LabStation

A job is created manually that requests for data. These requests are passed along to the instrument queue to await data from the instrument. Samples are added to the Job. First job/samples are assigned to an instrument queue in order to acquire data, then the job/sample is selected and ‘Assign Raw Data File’ is selected from the Edit menu (Figure 24). This will open a file selection dialog. Select the appropriate file, and the data is assigned to the selected Job/Sample(s). Now the data is parsed, and the status of the Job/Sample changes from incomplete to complete. The screenshot of the parsing script used is shown in Figure 23. The parsed data is saved in a new file named Limsdata.raw.



Full File Pathname for the Data Files (Wildcards accepted).

C:\Documents and Settings\slakhanp\Desktop\

Polling Time (in seconds) 0

☐ IncludeSubdirectories when Polling

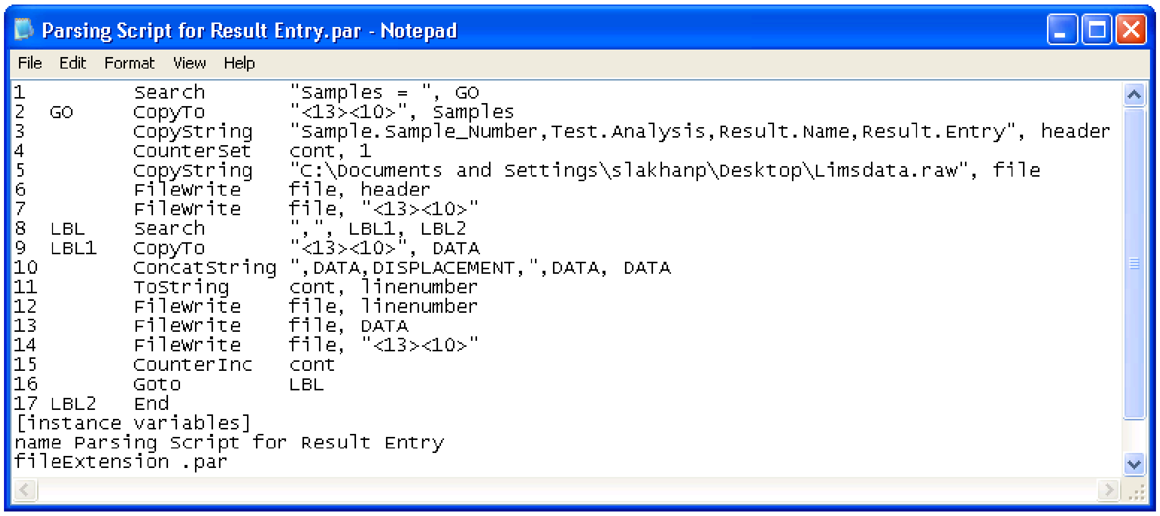
Post Archive Actions

● Leave Raw Data File unchanged

○ Delete Raw Data File

○ Change Raw Data File Extension to:

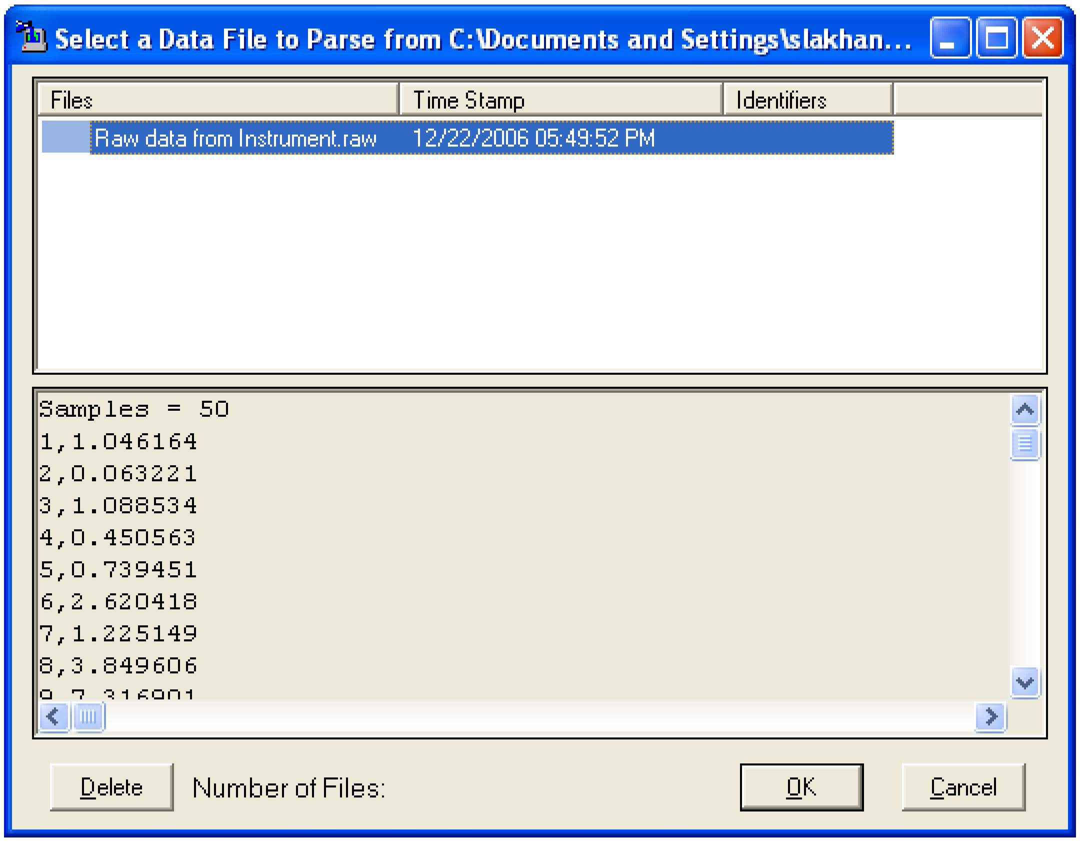
Figure 22: File Instrument Setup in LabStation



Parsing Script for Result Entry.par - Notepad

```
1               Search        "Samples = ", GO
2   GO          CopyTo        "<13><10>", Samples
3               CopyString    "Sample.Sample_Number,Test.Analysis,Result.Name,Result.Entry", header
4               CounterSet    cont, 1
5               CopyString    "C:\Documents and Settings\slakhanp\Desktop\Limsdata.raw", file
6               FileWrite     file, header
7               FileWrite     file, "<13><10>"
8   LBL         Search        ",", LBL1, LBL2
9   LBL1        CopyTo        "<13><10>", DATA
10              ConcatString  ",DATA,DISPLACEMENT,",DATA, DATA
11              ToString      cont, linenumber
12              FileWrite     file, linenumber
13              FileWrite     file, DATA
14              FileWrite     file, "<13><10>"
15              CounterInc    cont
16              Goto          LBL
17  LBL2        End
[instance variables]
name Parsing Script for Result Entry
fileExtension .par
```

Figure 23: Parsing Script used to parse the raw data for Case 1: Result Entry



Select a Data File to Parse from C:\Documents and Settings\slakhan...

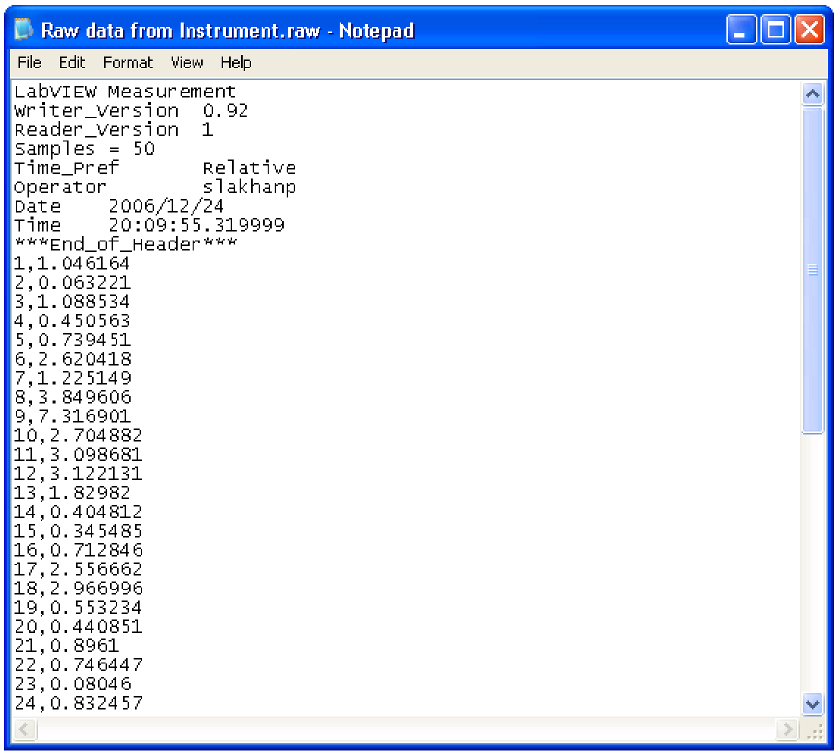
Files	Time Stamp	Identifiers
Raw data from Instrument.raw	12/22/2006 05:49:52 PM	

```
Samples = 50
1,1.046164
2,0.063221
3,1.088534
4,0.450563
5,0.739451
6,2.620418
7,1.225149
8,3.849606
```

Figure 24: Assigning a raw data file to parse in LabStation

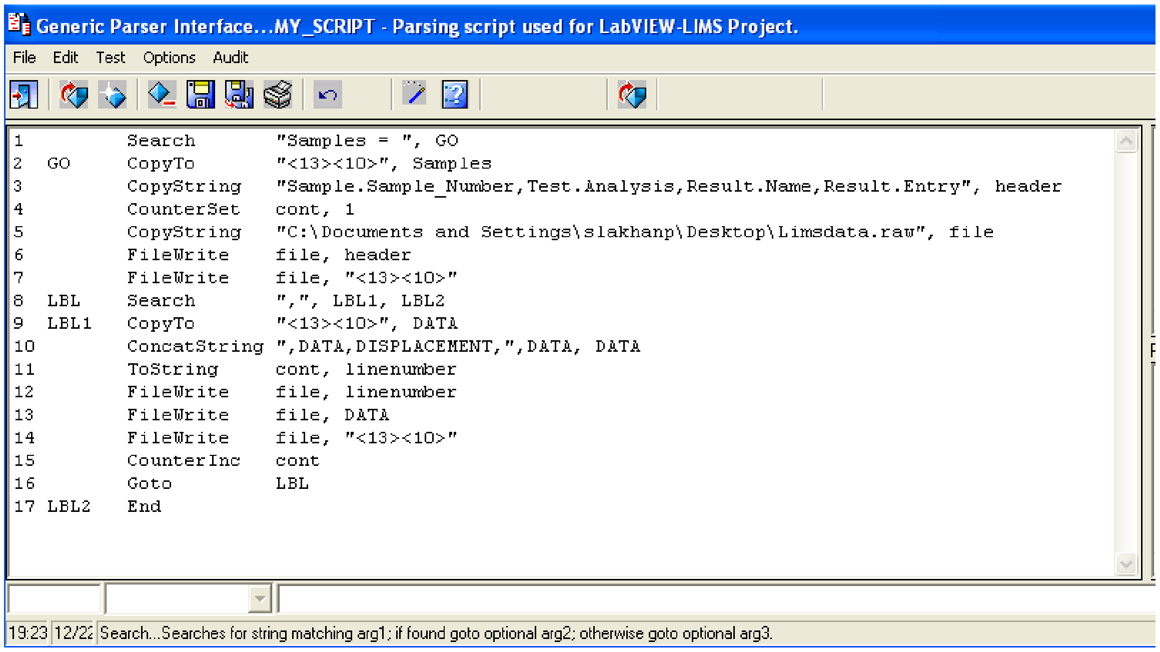
Since the parser interface is not very user friendly in LabStation, only one parsing script is defined and that is for Case 1 (Result Entry). Therefore, LabStation can be used to enter the results for the samples that are already logged in LIMS, whereas, LabVIEW is used to parse the data for each case.

Figure 25 shows the screen shot of the raw data generated by LabVIEW that will be parsed in LabStation. This raw data file will be parsed and put in a .CSV extension file so that the results can be imported and stored in LabWare LIMS. Figure 26 shows the screen shot of a portion of the parser interface in LabStation. Figure 27 shows the screen shot of a portion of the parser interface that includes the test file, and the parsing information interface.


Raw data from Instrument.raw - Notepad
File Edit Format View Help

```
LabVIEW Measurement
Writer_Version  0.92
Reader_Version  1
Samples = 50
Time_Pref       Relative
Operator        slakhanp
Date     2006/12/24
Time     20:09:55.319999
***End_of_Header***
1,1.046164
2,0.063221
3,1.088534
4,0.450563
5,0.739451
6,2.620418
7,1.225149
8,3.849606
9,7.316901
10,2.704882
11,3.098681
12,3.122131
13,1.82982
14,0.404812
15,0.345485
16,0.712846
17,2.556662
18,2.966996
19,0.553234
20,0.440851
21,0.8961
22,0.746447
23,0.08046
24,0.832457
```

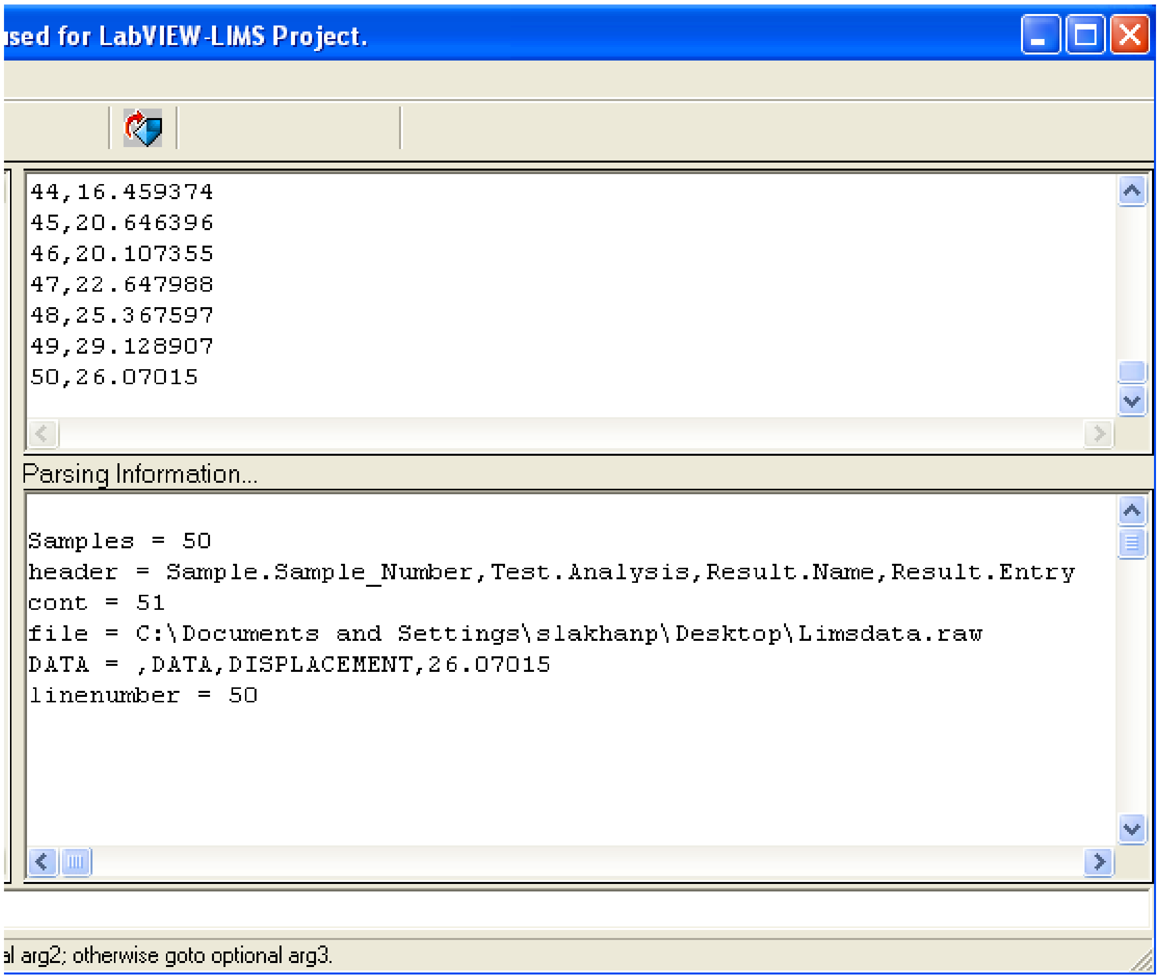
Figure 25: Screen shot of the raw data file used with LabStation


Generic Parser Interface...MY_SCRIPT - Parsing script used for LabVIEW-LIMS Project.
File
Edit
Test
Options
Audit

```
        Search        "Samples = ", GO
GO      CopyTo        "<13><10>", Samples
        CopyString    "Sample.Sample_Number,Test.Analysis,Result.Name,Result.Entry", header
        CounterSet    cont, 1
        CopyString    "C:\Documents and Settings\slakhanp\Desktop\Limsdata.raw", file
        FileWrite     file, header
        FileWrite     file, "<13><10>"
LBL     Search        ",", LBL1, LBL2
LBL1    CopyTo        "<13><10>", DATA
        ConcatString  ",DATA,DISPLACEMENT,",DATA, DATA
        ToString      cont, linenumber
        FileWrite     file, linenumber
        FileWrite     file, DATA
        FileWrite     file, "<13><10>"
        CounterInc    cont
        Goto          LBL
LBL2    End
```

19:23 12/22 Search...Searches for string matching arg1; if found goto optional arg2; otherwise goto optional arg3.

Figure 26: Screen-Shot of the Parser Interface with the Parsing Script used with raw data

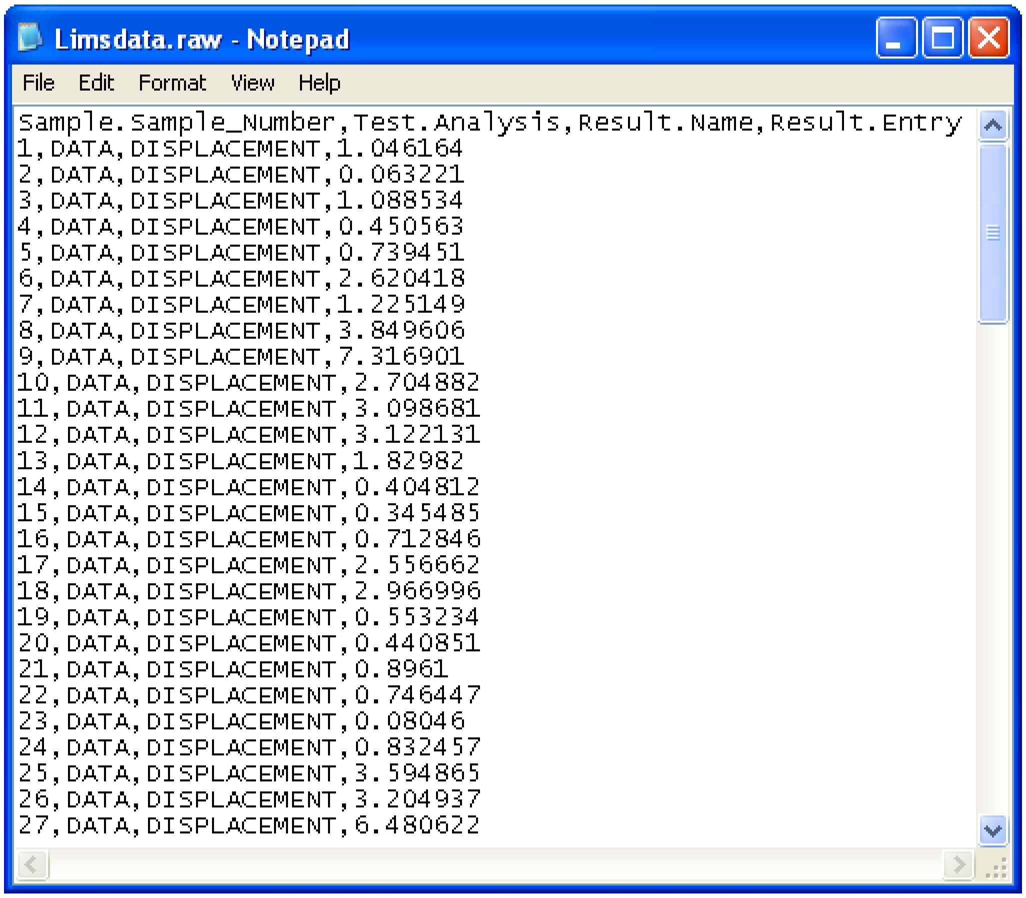

sed for LabVIEW-LIMS Project.
Parsing Information...
l arg2; otherwise goto optional arg3.

```
44,16.459374
45,20.646396
46,20.107355
47,22.647988
48,25.367597
49,29.128907
50,26.07015
```

```
Samples = 50
header = Sample.Sample_Number,Test.Analysis,Result.Name,Result.Entry
cont = 51
file = C:\Documents and Settings\slakhanp\Desktop\Limsdata.raw
DATA = ,DATA,DISPLACEMENT,26.07015
linenumber = 50
```

Figure 27: Screen-Shot of the Parsing Information from the Parser Interface

The final form of the parsed raw data file is shown in figure 28, which is similar to the output file from LabVIEW for Case 1, and therefore completes the validation.



```
Sample.Sample_Number,Test.Analysis,Result.Name,Result.Entry
1,DATA,DISPLACEMENT,1.046164
2,DATA,DISPLACEMENT,0.063221
3,DATA,DISPLACEMENT,1.088534
4,DATA,DISPLACEMENT,0.450563
5,DATA,DISPLACEMENT,0.739451
6,DATA,DISPLACEMENT,2.620418
7,DATA,DISPLACEMENT,1.225149
8,DATA,DISPLACEMENT,3.849606
9,DATA,DISPLACEMENT,7.316901
10,DATA,DISPLACEMENT,2.704882
11,DATA,DISPLACEMENT,3.098681
12,DATA,DISPLACEMENT,3.122131
13,DATA,DISPLACEMENT,1.82982
14,DATA,DISPLACEMENT,0.404812
15,DATA,DISPLACEMENT,0.345485
16,DATA,DISPLACEMENT,0.712846
17,DATA,DISPLACEMENT,2.556662
18,DATA,DISPLACEMENT,2.966996
19,DATA,DISPLACEMENT,0.553234
20,DATA,DISPLACEMENT,0.440851
21,DATA,DISPLACEMENT,0.8961
22,DATA,DISPLACEMENT,0.746447
23,DATA,DISPLACEMENT,0.08046
24,DATA,DISPLACEMENT,0.83245 7
25,DATA,DISPLACEMENT,3.594865
26,DATA,DISPLACEMENT,3.204937
27,DATA,DISPLACEMENT,6.480622
```

Figure 28: Parsed raw data file from LabStation for Case 1: Result Entry

CHAPTER 5

CONCLUSION

This project interfaces LabWare LIMS with LabVIEW. The different type of data is integrated using LabVIEW and the results are validated using LabStation.

The LabVIEW-LIMS.VI is an easy to use program that can login samples, modify samples and enter results in LabWare LIMS using one comma separated value data file. The user needs to select few field values from the front panel depending on the case selected, and then LabVIEW parses the raw data and writes it to a file that is automatically imported and stored in LabWare LIMS. Using LabVIEW created .CSV data file, the user can now continuously enter, store results and/or update any sample information in the LIMS system automatically. By pre-defining fields in LIMS, viz., sample template, users, customers and other relevant features that can be anchored with the sample information, the user can skip all other steps, and can directly review, approve, authorize or reject samples right after importing the results in LIMS.

The user does not have to follow the standard steps in LIMS and can login samples, assign tests, enter results, modify samples automatically that leads to automated information management. The first part of the VI can be replaced by any instrument and can be used in conjunction with the second part of the VI.

Future Work:

This project can be continued further by using different instrument integration types (RS232, COM Ports) in LabStation. As mentioned in the discussion this project can be configured as per the requirements mandated by the regulatory agencies.

CHAPTER 6

DISCUSSION

The results demonstrate the utility of LabVIEW^{TM} to make data files that are compatible with LabWare LIMS and hence, the results and other sample information can be stored automatically in LIMS system.

The integrated LabStation by LabWare LIMS is used for the validation of the results from LabVIEW. LabStation is used for file based instrument integration to store results in LIMS. The raw data generated by LabVIEW is parsed using a parsing script in LabStation and the results are stored in LabWare LIMS.

The main purpose of using LabStation is to validate the results generated by LabVIEW. Also, we discuss the user interface and functionalities of these two softwares. It is observed that parsing the data in LabVIEW is easier compared to parsing the same data in LabStation. The LabStation parser interface offers a huge number of parsing commands. However, they are not very direct to use and the parsing interface is not very user friendly. LabStation is a product of LabWare, still, there is no command or pre-defined parsing script that can be used for any standard instrument integration to make a raw data file into .CSV file or ASCII text files and thus compatible with LabWare LIMS. The advantage of using LabVIEW is that it is not limited to any number of RS232 connections and can acquire data from a number of instruments at the same time, whereas LabStation can acquire data from one instrument at a time. Also, LabVIEW can interface with any instrument on a network.

In this project, the field values are pre-defined in LabWare LIMS. To integrate LIMS with LabVIEW same fields are defined in the VI. For better data flow, the VI can be configured such that it can pull the data from LIMS database, and can update the required fields in LabVIEW VI.

The other important issue in interfacing LabVIEW with LabWare LIMS is the use of data file (results) in a regulatory environment. In this project, the data file is stored in a drive on the system therefore; it can be easily changed or edited by anyone using the system. In a regulatory environment, the Universal Naming Convention (UNC) should be used. In a network, the UNC is a way to identify a shared file in a computer without having to specify (or know) the storage device it is on. In Windows operating systems, and possibly other operating systems, the UNC can be used instead of the local naming system (such as the DOS naming system in Windows). UNC specifies a common syntax to describe the actual location on a network resource; such as a shared file, folder or printer. In the second part of the VI where the raw data is parsed, the date and time when the data is generated is not recorded. In this project, only the date and time when the data is stored in LIMS is recorded. When working in a regulatory environment, it is essential to record when the data was generated and when it was stored. Therefore, the VI needs some configuration so that it can be used in a regulatory environment.

REFERENCES

1.) Bishop, Robert. National Instruments Learning with LabVIEW 7 Express. Upper Saddle River, NJ: Pearson Education, Inc., 2004.

2.) “An Introduction to LIMS”:
Available: http://www.limsource.com/intro.html

3.) LabWare LIMS Customer Support
Available: http://www.labware.com/LWCustomerHomePage.NSF

4.) LabWare LIMS Version 5 User Guide
Available: http://www.labware.com/LWCustomerHomePage.NSF

5.) LabStation Version 1 User Manual
Available: http://www.labware.com/LWCustomerHomePage.NSF

6.) Universal Naming Convention
Available:
http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci214159,00.html

APPENDIX

SWATI LAKHANPAL

835 Camp Street, Phone: 317-603-2860
Indianapolis, IN 46202. Email: slakh11@yahoo.com

OBJECTIVE
To obtain a full-time position in Laboratory Informatics.

EDUCATION
Master of Science, Chemical Informatics (Specialization Laboratory Informatics),
Indiana University, Indianapolis, IN
GPA: 3.90 /4.0, **Expected graduation: December 2006**
Thesis Topic: *Task level programming of Zymate^{TM} laboratory robot using LabVIEW*
Bachelor of Engineering, Department of Chemical Engineering & Technology, Panjab University, Chandigarh, India
GPA: 3.40/4.0 (First class) **Graduation: May 2003**
Undergraduate Project: *Removal of Toxics from Industrial Waste Water*

WORK EXPERIENCE
Intern, Lab Rat Software Systems, Indianapolis, September 2006 till present
Potential duties include system testing, user documentation development, usability studies, method development and integration studies.
Graduate Research Assistant, Indiana University, August 2005 till present
Interfacing LabVIEW driven DAQ instrument with LabWare LIMS
Graduate Teaching Assistant, Indiana University, January 2005 – May 2005
Held weekly study sessions, grading quizzes in Chemistry.
Tutor, Math Assistance Centre, Indiana University, January 2005 – July 2005
Graduate Engineer Trainee, Gurdarshan Leathers Limited, India, August 2003- June 2004
Design of an effluent treatment plant and the removal of toxics from industrial wastewater.
Summer Intern, Rallis India Ltd., India, May - June 2003
Industrial waste water management system, setup of the new waste water recycle system, and a brief study of the effects of pesticides pollution in the environment.
Practical Surveying Training, Jagatjit Distilleries, Hamira, India, Jun 2002
Educational Training, India International Trade Fair (IITF), New Delhi, India, Dec 2002

LABORATORY INFORMATICS PROJECTS
Scientific Data Management & Analysis- National Cancer Institute Dataset: Database was populated by parsing the records of interest from the files in the dataset, the data was then analyzed and visualized using Spotfire. LabTrack, CyberLab and Waters NuGenesis eLab Notebook was also used.
LIMS for Gel-Electrophoresis of DNA: Application of LabWare and LabVantage Sapphire for developing the Gel-Electrophoresis LIMS system.
Programmatically Controlling the Zymate^{TM} Laboratory Robot: LabVIEW was used to develop joystick code for complete control of 4-axis of the Zymate^{TM} robot.

AWARDS
Inaugural Recipient of the LabWare Fellowship for Innovation in Laboratory Informatics, 2005-2006

PRESENTATIONS
"Scientific Data Management & Analysis: National Cancer Institute", School of Informatics, Indiana University Purdue University, Indianapolis, December 2005.
"Programmatically Controlling the Zymate^{TM} Laboratory Robot", School of Informatics, Indiana University Purdue University, Indianapolis, December 2005.

"Information Architecture of the Web As It Relates To Chemical Informatics", School of Informatics, Indiana University Purdue University Indianapolis, May 2005.
"Pre-Hospital Patient Data from the Marion County Patient Database: Database design, HIPAA Security, Products in Market, Problems and Future Work", School of Informatics, Indiana University Purdue University Indianapolis, May 2005.
"FTC and the Internet: Advertising and Marketing on the Internet", School of Informatics, Indiana University Purdue University Indianapolis, April 2005.
"Website Analysis: NIST Chemistry Workbook", School of Informatics, Indiana University Purdue University Indianapolis, April 2005.
"URL Spoofing", School of Informatics, Indiana University Purdue University Indianapolis, February 2005.
"Next Generation Search Tools", School of Informatics, Indiana University Purdue University Indianapolis, February 2005.
"Design of a green house for off season growth", Department of Chemical Engineering and Technology, Panjab University, March 2003.
"Expanded Foam Molding", Department of Chemical Engineering and Technology, Panjab University, November 2002.

SKILLS
LabWare, LabVantage Sapphire
eLab Notebook: Waters eLab Notebook, LabTrack, CyberLab
Molecular Modelling: Spartan, SYBYL
Software: Spotfire, ChemDraw, Total Chrom
Computer Languages: LabVIEW

MEMBERSHIPS
Member of Indian Institute of Chemical Engineers
Member of Indian Society of Plastics

REFERENCES
Available on request