BETSRG Report 1999-1

May 1999

Anand Iyer

Oklahoma State University
School of Mechanical and Aerospace Engineering
ENHANCEMENT OF THE GENERIC DATABASE LIBRARY ENVIRONMENT FOR ENGINEERING APPLICATIONS AND ITS IMPLEMENTATION FOR HVAC LOAD EXPLORER

By

IYER, ANAND

Bachelor of Engineering

Victoria Jubilee Technical Institute

Mumbai, India

1996

Submitted to the Faculty of Mechanical and Aerospace Engineering Department Of Oklahoma State University in Partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May, 1999
ACKNOWLEDGEMENTS

I wish to express my sincere thanks and gratitude to my advisor, Dr. Ron D. Delahoussaye for his ideas, inspiration and guidance. I am very thankful for all his valuable inputs, his suggestions that have helped me to develop a good product. My sincere appreciation extends to Dr. Jeffrey D. Spitler, for giving me the opportunity to work on this project. It’s been an honor to work under one of the leading persons in the field of HVAC. I am very thankful to them, for their faith and confidence in me, during the course of the project and the valuable advice given to me from time to time.

I am very much thankful to Dr. Arena for agreeing to be a part of my Graduate committee at such a short notice.

My sincere thanks also extend to Vivek, my long time friend and also my partner for this project. Vivek, its been fun working with you. Your opinions and suggestions during the development of this product have been very useful to me.

I would like to thank all the faculty members of the Mechanical and Aerospace Engineering Department for their contribution towards my knowledge and education here at Oklahoma State University.

I would like to thank Sarah, the Receptionist at MAE Department, for her patience in coordinating meetings with my advisors and the coffee vending machine at Math Science department, for feeding me during those cold, long lonely nights.

Finally, thanks to my family members for their blessings, and without whose support I would not be here doing my Master’s and also to my friends and well-wishers, for their constant encouragement throughout my career.
TABLE OF CONTENTS

1. OVERVIEW

  1.1 Introduction .......................... 1
  1.2 Expectations from a Generic Database ...... 1
  1.3 Overview of the Components of the Database Library Environment 2

2. INSTRUCTIONS TO USE THE GENERIC DATABASE LIBRARY ENVIRONMENT IN AN APPLICATION

  2.1 Details on Creating the Database ............. 8
  2.2 Details on Creating the Tables ................ 8
    2.2.1 The SetUp Table ...................... 8
       • The First Record ..................... 11
       • Column details for remaining records .. 13
       • Title .................................. 14
       • Alignment ................................ 14
       • Width .................................. 15
       • UnitSI and UnitIP ....................... 16
       • Unit-Scale and Unit-Offset ............. 17
       • Data Type .............................. 17
       • Lower Limit and Upper Limit ........... 18
       • DecimalsSI and DecimalsIP ............. 18
    2.2.2 The Std and Usr Tables .................. 19
  2.3 Creating the User Interface for the Library .... 21
    2.3.1 Deciding the Library Editing Interface .... 22
    2.3.2 Adding the User Interface ............... 23
       2.3.2.1 Using the ApplicationDataInput Form .. 25
       2.3.2.2 Using the Custom Form ............... 28
       2.3.2.3 Using the Automatic Form ............. 29
# TABLE OF CONTENTS (Cont’d..)

3. CONCLUSION AND RECOMMENDATIONS 31

4. APPENDICES

   Appendix A: Interface Types 32
   Appendix B: Customization Routines 47
   Appendix C: Examples from HVAC Load Explorer 49
   Appendix C: Functions and Subroutines in the Modules 52

5. REFERENCES 48
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Table Types required for a library</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Field names with their data types for SetUp tables</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Setting up the precision of the Number data type in Field Size</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Records in the SetUp table</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>First Record</td>
<td>11</td>
</tr>
<tr>
<td>2.6</td>
<td>Form frmG1Table</td>
<td>12</td>
</tr>
<tr>
<td>2.7</td>
<td>Remaining records</td>
<td>13</td>
</tr>
<tr>
<td>2.8</td>
<td>SetUp table (1st half)</td>
<td>14</td>
</tr>
<tr>
<td>2.9</td>
<td>Starting form for the Library interface</td>
<td>16</td>
</tr>
<tr>
<td>2.10</td>
<td>SetUp table (2nd half)</td>
<td>17</td>
</tr>
<tr>
<td>2.11</td>
<td>Relation between field entry under DecimalsIP in SetUp table And display in frmG1Table</td>
<td>19</td>
</tr>
<tr>
<td>2.12</td>
<td>Fields in the MatlStd table</td>
<td>20</td>
</tr>
<tr>
<td>2.13</td>
<td>Prompt for Primary key declaration</td>
<td>20</td>
</tr>
<tr>
<td>2.14</td>
<td>Project\ References box</td>
<td>22</td>
</tr>
<tr>
<td>2.15</td>
<td>User interface elements for using the library</td>
<td>23</td>
</tr>
<tr>
<td>A.1</td>
<td>ApplicationDataInput form</td>
<td>32</td>
</tr>
<tr>
<td>A.2</td>
<td>ApplicationDataInput form as the LibEditing form</td>
<td>33</td>
</tr>
<tr>
<td>A.3</td>
<td>ApplicationDataInput form as the LibEditing form</td>
<td>38</td>
</tr>
<tr>
<td>A.4</td>
<td>ApplicationDataInput form as the LibEditing form</td>
<td>44</td>
</tr>
<tr>
<td>C.1</td>
<td>ApplicationDataInput form, that connects to Layers library</td>
<td>49</td>
</tr>
<tr>
<td>C.2</td>
<td>Interface for editing layer library</td>
<td>49</td>
</tr>
<tr>
<td>C.3</td>
<td>ApplicationDataInput form that connects to Windows library</td>
<td>50</td>
</tr>
<tr>
<td>C.4</td>
<td>Interface form for editing records</td>
<td>37</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>47</td>
</tr>
<tr>
<td>3</td>
<td>47</td>
</tr>
<tr>
<td>4</td>
<td>47</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>52</td>
</tr>
<tr>
<td>7</td>
<td>53</td>
</tr>
<tr>
<td>8</td>
<td>53</td>
</tr>
<tr>
<td>9</td>
<td>53</td>
</tr>
<tr>
<td>10</td>
<td>54</td>
</tr>
<tr>
<td>11</td>
<td>54</td>
</tr>
<tr>
<td>12</td>
<td>54</td>
</tr>
<tr>
<td>13</td>
<td>55</td>
</tr>
<tr>
<td>14</td>
<td>55</td>
</tr>
<tr>
<td>15</td>
<td>55</td>
</tr>
<tr>
<td>16</td>
<td>56</td>
</tr>
<tr>
<td>17</td>
<td>56</td>
</tr>
<tr>
<td>18</td>
<td>56</td>
</tr>
<tr>
<td>19</td>
<td>57</td>
</tr>
<tr>
<td>20</td>
<td>57</td>
</tr>
<tr>
<td>21</td>
<td>57</td>
</tr>
<tr>
<td>22</td>
<td>57</td>
</tr>
<tr>
<td>23</td>
<td>58</td>
</tr>
<tr>
<td>24</td>
<td>58</td>
</tr>
<tr>
<td>25</td>
<td>58</td>
</tr>
<tr>
<td>26</td>
<td>59</td>
</tr>
<tr>
<td>27</td>
<td>59</td>
</tr>
<tr>
<td>28</td>
<td>59</td>
</tr>
<tr>
<td>Table</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>29</td>
<td>60</td>
</tr>
<tr>
<td>30</td>
<td>60</td>
</tr>
<tr>
<td>31</td>
<td>60</td>
</tr>
<tr>
<td>32</td>
<td>61</td>
</tr>
<tr>
<td>33</td>
<td>61</td>
</tr>
</tbody>
</table>

LIST OF TABLES (Cont’d..)
CHAPTER 1: OVERVIEW

1.1 Introduction

An engineering application makes use of data available from different sources. It would be very useful if such related data is collected and stored in a database. The user can then just select the required record from the database. This collection of related data is what is referred to as a Library, in Engineering parlance. The records stored in the library may be standard and based on highly reliable sources or may be user defined.

A generic database library environment is an application with a database for storing data, and an interface for viewing and editing records. This environment can be used as a part of any application that has libraries, with minor modifications. The idea behind using a generic database is its wide applicability with many engineering applications.

1.2 Expectations From A Generic Database

The following are expected from a Generic Database:

- Ease of implementation i.e. easy to customize with little or no modification.
- Easy to use i.e. it should have a good user interface, preferably, a Graphical User Interface, for easy access to the database
- Easy to maintain. It should be easy to carry out standard database operations like:
  - adding records
  - modifying records
  - deleting records
• viewing record details
• searching through the database
• porting data in and out of the database, so that different users can share the data, and viewing records.
• The database should be normalized to prevent data redundancy while adding, modifying or deleting records.
• This generic database for engineering applications should incorporate a Standard library and a User library.
• The Standard library contains data from highly reliable sources like ASHRAE and cannot be modified directly by the user.
• The User library contains data of interest to the user and can be edited by the user.

1.3 Overview Of The Components Of The Database Library Environment

The generic database, to be used for your application, is comprised of:

• 3 tables for each library
• 3 interface forms
• The associated code and modules, for accessing and editing the data.

1. Specifications for the 3 Tables

1. SetUp table contains the following information, that define the structure of the Std and Usr tables, and also how data is stored in these tables and displayed to the user:

• The order in which the fields are stored in the data table
• The caption of the various fields in the data table
• The units of the various fields in SI and IP
• The current system of units in which data is stored in the table
• The conversion factors to convert the various fields from one unit to the other
• The data type of the various fields, whether Single, Integer or Text.
• The permissible range of values for various fields (numeric fields alone)
• The accuracy i.e. the number of significant digits in which data has to be stored in
  the table or displayed to the user, for each field, for both the units.
• The size of the grid in which data is to be displayed.
• The alignment details for the fields. i.e. left, right or center.

2. Standard (Std) Table

This is the standard table, which stores data from reliable sources like ASHRAE. The
data from this table can be read-only by the user. The user cannot add, modify or delete
records from this table.

3. User (Usr) Table

This is the user table, which also stores data but is maintained by the user. Data can be
added, modified, deleted from this table.

The following checks are made when the user enters data in the table:

• The “Std” and “Usr” tables have the same order of fields and the same primary key. A
  check is done to see if the primary key of the data entered by the user already exists in the
  “Std” table. If it does, a message box appears. The user cannot duplicate a record that
  already exists in the “Std” library.

• If it duplicates a record that is already there in the User library, the user will be asked for
  confirmation before overwriting.
• Validation is performed on all fields i.e. if a field is numeric, check is performed to ensure that the value entered is within bounds specified in the “Setup” table or if a field is a text field, a check is performed to make sure that it is not empty. If the data entered does not pass the validation, an appropriate error message is displayed and that particular field, which fails the validation, gets the focus.

2. Information about Interfaces

The generic database basically uses three interfaces:

• An interface to display and select the records (frmG1Table.frm).

This interface has a grid that displays all the records from both, the Std as well as the Usr libraries. The records from the Std library are displayed in black color, and cannot be edited by the user. The records in the Usr library are editable, and are displayed in red color. A status label beneath the grid displays a message that indicates whether the selected record is from Std library or the Usr library. The buttons for editing the records are disabled if a record from the Std library is selected.

• An interface to input the data selected from the library (ApplicationDataInput form)

The data selected from the library is input in the corresponding fields of this form. This is the form that has the Library button on it and is the interface to the library.

• An interface for editing the records

The interface for editing the records can be any of the following three types:

1. Automatic form

This interface is automatically created by the program based on the information given in the Setup table about the number of fields. However, the fields generated for editing are vertically aligned. This may be a problem
when the number of fields is large, in which case, all the fields cannot be displayed and therefore, difficult to edit. This is the easiest interface to add to an application.

2. Custom form

This interface has to be specially created for editing the records. Thus, it is customized for editing purposes. It solves the problem of Automatic form, as here the fields are arranged as per the user’s requirements.

3. ApplicationDataInput form (using the Instance of the ApplicationDataInput form)

This interface uses an instance of the ApplicationDataInput form for editing records. So the appearance is the same as the ApplicationDataInput form, which means, that the user will not have to adjust to a different look. Hence, this is desirable.

Chapter 2 will describe these components in more detail with the help of the example given in the disk.
CHAPTER 2: INSTRUCTIONS TO USE THE GENERIC DATABASE LIBRARY ENVIRONMENT IN AN APPLICATION

Instructions on how to use the generic database in your application are explained with the help of an example given in the accompanying disk. Please open the file “Demo.vbp”. The example shows a library of alloys.

To create your own application, it is much easier to copy some of the existing code rather than write them yourself. This is what the generic database aims to achieve. It provides code that can be reused, customized with minor modifications.

This is a brief explanation on how to integrate the database part to your application. It will be explained in more detail later. Basically there are three steps involved:

1. Creating the database and the tables
   - Create a database for your application, using MSAccess. If the database in which you need to store the tables already exists, go to next step.
   - Create the following tables in the database:
     1. SetUp table
     2. Usr table
     3. Std table
2. Creating the forms, modules and classes for the user interface.

   Copy the following from the disk and paste it to the application directory:

   - **Forms(*.frm):**
     1. frmDialog
     2. frmG1Table
     3. frmG1AddModDetSearch
     4. frmHelpForSearch.

   - **Modules(*.bas):**
     1. Defn1Tbl
     2. FileUtil
     3. G1Table
     4. GUtils

   - **Classes(*.cls):**
     1. Dialog

3. Copy and Paste frmStart/ frmCustom depending on the type of interface to be used, so that it will be easier to copy and paste their events and routines (given below) that are required for customization.

   Customizing your ApplicationDataInput form (in our example, frmStart) and/or your Custom form (in our example, frmCustom) by copying all or some of the following routines and events from frmStart and/or frmCustom into your Application’s ApplicationDataInput form and/or Custom form respectively.
• Events
  1. Library_Click
  2. Form_Load
  3. Ok_Click
  4. Cancel_Click

• Routines
  1. TransferDataIn
  2. TransferDataOut
  3. GetFocusOnFields
  4. DisableControls

2.1 Details on Creating the Database

To create a new database for your application:

1. Open MS Access.
2. Go to File
3. Select New Database
4. A window pops up with Blank Database highlighted. Click OK
5. Select the directory and enter a name for your database. Click OK.

A .mdb file is created. This is your database that will hold the tables.

2.2 Details on Creating the Tables

2.2.1 The SetUp Table

Step1: First create the Setup table in the above database, which holds the tables representing the libraries. Create a new table “MatlSetup” (refer Fig. 2.1) with the details as shown in Fig. 2.2
Fig. 2.1 Table Types required for a library

Fig. 2.2 Field names with their data types for SetUp tables.
Fig. 2.3 Setting up the precision of the Number data type in Field Size

Alternatively, you can import this table structure alone from another database file, which has got a table with the same structure.

Step 2: Enter the precision for the field names that have Number as their data type by entering the Field Size field in the General tab. Fig. 2.3 shows that the highlighted field name “Table Column” has Number as the Data Type and Field Size as “Integer”.

The field names (that have Number as the Data type) and their precision are as given below:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Column</td>
<td>Integer</td>
</tr>
<tr>
<td>Align</td>
<td>Integer</td>
</tr>
<tr>
<td>Width</td>
<td>Integer</td>
</tr>
<tr>
<td>Unit-Scale</td>
<td>Single</td>
</tr>
<tr>
<td>Unit-Offset</td>
<td>Single</td>
</tr>
<tr>
<td>Lower Limit</td>
<td>Single</td>
</tr>
<tr>
<td>Upper Limit</td>
<td>Single</td>
</tr>
</tbody>
</table>

Table 1 Field names and their Precision
Step 3: Now the setup table has to be filled with the correct data, which describe the library. Refer to Fig. 2.4. Start numbering from the first column of the SetUp table, from 0 to the number of fields in the table representing the library. In our example, the library table contains five rows: UNS Designation, Carbon, Nitrogen, Density, Melt Temperature.

Step 4: So open the MatlSetUp table and enter numbers 0 through 5 in the first column, one number each in a row in ascending order.

The first record i.e. the record with value 0 in the Table Column is different from the remaining records.

**The First Record**

<table>
<thead>
<tr>
<th>Table Column</th>
<th>Title</th>
<th>Align</th>
<th>Width</th>
<th>UnitLP</th>
<th>UnitSL</th>
<th>Unit - Scale</th>
<th>Unit - Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alloy Library</td>
<td>5370</td>
<td>7825</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data Type | Lower Limit | Upper Limit | DecimalsLP | DecimalsSL |
-----------|-------------|-------------|-------------|------------|

Fig. 2.4 Records in the SetUp table

Fig. 2.5 First Record
The first record in the SetUp table describes the details about the grid (Height & Width), the caption of the form that pops up with this grid and the units in which the data is stored, whether IP or SI.

Step1: Enter the caption of the form (frmG1Table) that displays the records under Column “Title”. In our example, the caption of the pop up form is “Alloy Library”.

Step2: Enter the height of the grid under the column heading “Align”. The height depends on the number of records displayed at a time. In our example, we have set the height as 5370 twips to display 22 records. If more records are required to be displayed, increase the height (e.g. Height = 5600 for 23 records). The minimum height required is 5370. If the height is decreased further, frmG1Table wont appear neat because, now the height of the grid will become less than the position of the bottom surface of the last button (“Details”) in the grid. (Refer to Fig. 2.6)

![Alloy Library](image)

Fig. 2.6 Form frmG1Table (The middle portion has been removed)
Step 3: Enter the width of the grid under column heading “Width”. It is dependent on the number of columns in the library table and the width of each and every column. In our example, the width is 7825 twips. The minimum width required is 4000 twips, this being the length of the message (Current selected record is from the Standard Library) displayed in the label beneath the grid. The sum of the widths of all the remaining records (1 through 5) should be approximately equal to or slightly lesser than the grid width. The minimum width for each column can be calculated by taking a value of 100 twips per character. So, for a column heading like Carbon (6 characters), the minimum width is 100 x 6 = 600 twips.

Step 4: Enter the unit in which the data are to be stored in the library table. If the storage unit is SI, enter 1 under the “UnitSI” column heading and 0 under the “UnitIP” heading of the first record else vice-versa. Even if the data has no units, enter 1 in any of the above two fields. In our example, we are storing the data in IP units. So enter 1 in the UnitIP field and 0 in the UnitSI field.

- Column Details For Remaining Records

<table>
<thead>
<tr>
<th></th>
<th>UNS Designation</th>
<th>1</th>
<th>2800</th>
<th></th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Carbon</td>
<td>1</td>
<td>1100%</td>
<td>%</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Nitrogen</td>
<td>1</td>
<td>1100%</td>
<td>%</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Density</td>
<td>1</td>
<td>1125</td>
<td>lb/ft³</td>
<td>kg/m³</td>
</tr>
<tr>
<td>5</td>
<td>Melt Temperature</td>
<td>1</td>
<td>1700</td>
<td>degF</td>
<td>degC</td>
</tr>
</tbody>
</table>

Fig. 2.7 Remaining Records (1 to 5)

The section contains details, about how, for remaining records (1 through 5), each and every column (field) of the library table should be stored in rows 1 to 5. Starting from the second row in the SetUp table, all the information entered will describe the various
fields of the library table. Start with the second row, i.e. the row whose first column has a value 1 stored in it. Refer to Fig. 2.7

- **Title** (Refer to Fig. 2.8)

Enter the name of the first column of the library table into “Title” field (Second column) or enter any other name, which you would use to refer to the first column of the library table. In our example, enter “UNS Designation” in the “Title” field. Do the same for all rows. Enter the names of the fields in the same order they were stored in the library table.

<table>
<thead>
<tr>
<th>Table Column</th>
<th>Title</th>
<th>Align</th>
<th>Width</th>
<th>UnitP</th>
<th>UnitSI</th>
<th>Unit - Scale</th>
<th>Unit - Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UNS Designation</td>
<td>1</td>
<td>2600</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Carbon</td>
<td>1</td>
<td>1100</td>
<td>%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Nitrogen</td>
<td>1</td>
<td>1100</td>
<td>%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Density</td>
<td>1</td>
<td>1125</td>
<td>lb/ft³</td>
<td>kg/m³</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Melt Temperature</td>
<td>1</td>
<td>1700</td>
<td>degF</td>
<td>degC</td>
<td>0.5555556</td>
<td>-17.7778</td>
</tr>
</tbody>
</table>

Fig. 2.8 Setup Table (1st half)

In our example, the library table has fields: UNS, Carbon, Nickel, Density, and Melt Temperature. So the “Title” field of the SetUp table has data “UNS Designation”, “Carbon”, “Nickel”, “Density” and “Melt Temperature” which would act as titles for these fields in the grid.

- **Alignment** (Refer to Fig. 2.8)

Next the alignment details need to be specified i.e. how the data need to be aligned within the grid cell. This is specified in the field “Align” of the SetUp table.
• Left Align, enter 1
• Center Align, enter 4
• Right Align, enter 7

If the column displaying “UNS Designation” needs to be left aligned, enter 1 in the “Align” field of the row, which has its “Title” field as “UNS Designation”. In our example, we have left aligned all the fields. Ignore the first record as it contains information about the display (width of the grid) and not about the fields of the library table. This alignment is applied only to cells of the grid displaying data not to the ones displaying titles. Always left align the last column as a horizontal scroll bar may pop up if the data displayed does not fit within the viewable area of the grid.

• **Width** (Refer to Fig. 2.8)

The width of the columns should also be stored in the SetUp table. Enter the column width of the various columns in the “Width” field. The width of the columns may be found by trial and error. Start with a width of 1800. If the grid does not display the column name completely, increase the width by 100. Check again. Keep trying till you get proper width. The width of the column that will display “UNS Designation” needs to be 2800. So enter 2800 in the “Width” field of the row which has its “Title” field as “UNS Designation”. Similarly, enter the widths of other columns.
• **UnitSI and UnitIP** (Refer to Fig. 2.8)

Next the units in SI and IP are to be specified. These details should be entered in “UnitIP” and “UnitSI” fields of the SetUp table. These units are displayed in the labels in the forms (Fig. 2.9) and also in the column heading of the grid (Fig. 2.6).

![Fig. 2.9 Starting form for the library interface](image)

Enter the IP units in the “UnitIP” field of the corresponding row in the SetUp table and the SI units in the “UnitSI” field. If a field of the library table is not numeric, leave the “UnitIP” and “UnitSI” fields of the corresponding record empty. If it’s a unitless field, like the one representing percentage composition, you might want to display some characters like “%” in the grid, irrespective of the working units. In that case, enter the character to be displayed in both the fields, viz. “UnitIP” and “UnitSI” fields, for all the remaining fields in the corresponding rows of the SetUp table. If the field does not have any units, then leave that field empty. Don’t enter anything. In our example, Density has units “kg/m^3” in SI and “lb/ft^3” in IP.
• **Unit-Scale and Unit-Offset** (Refer to Fig. 2.8)

The conversion factors for converting the numeric fields from the storage units to the other one has to be entered in the SetUp table. The conversion is implemented with two constants, “Unit-Scale” and “Unit-Offset” as follows:

\[
(\text{Qty}) \text{ Other Units} = (\text{Qty}) \text{ Storage Units} \times (\text{“Unit-Scale”}) + \text{“Unit-Offset”} \quad \text{(1)}
\]

These constants are stored in “Unit-Scale” and “Unit-Offset” fields of the corresponding record in the SetUp table. Leave these fields empty if the corresponding library data field is not numeric. If it’s a dimensionless numeric field, enter 1 in “Unit-Scale” and 0 in “Unit-Offset” field. In our example, leave these two fields empty for the record representing “UNS Designation” field. The records “Carbon” and “Nickel”, being dimensionless, enter 1 in “Unit-Scale” field and 0 in “Unit-Offset” field. For the record “Temperature”, the relation between DegC and DegF is given by:

\[
\text{DegC} = \frac{5}{9}(\text{DegF} – 32) = 0.555556\times \text{DegF} – 17.77778 \quad \text{-----------------(2)}
\]

Since the units in which data is stored in the table has been selected as IP, therefore, in this case, “Storage Units” is DegF. So, here, Unit-Scale is 0.555556 and Unit-Offset is –17.77778.

• **Data Type** (Refer to Fig. 2.10)

<table>
<thead>
<tr>
<th>Table Column</th>
<th>Data Type</th>
<th>Lower Limit</th>
<th>Upper Limit</th>
<th>DecimalsIP</th>
<th>DecimalsSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>0</td>
<td>100</td>
<td>#.00</td>
<td>#000</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>0</td>
<td>100</td>
<td>#.00</td>
<td>#000</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>0</td>
<td>1000</td>
<td>#.00</td>
<td>#000</td>
</tr>
<tr>
<td>4</td>
<td>S</td>
<td>0</td>
<td>3000</td>
<td>#.0</td>
<td>#0</td>
</tr>
</tbody>
</table>

Fig. 2.10 Setup Table (2\textsuperscript{nd} half)
In “Data Type” field of the SetUp table, enter “T”, “S” or “I” specifying that the corresponding field of the library table is of type “Text”, “Single” or “Integer”. In our example, the field “UNS Designation” alone is a text field, the rest are numbers with single precision. So enter “T” in the record corresponding to the field “UNS Designation” and “S” in all the other records.

- **Lower Limit and Upper Limit** (Refer to Fig. 2.10)

If a field of the library table is numeric, then the permissible range of values need to be entered, **for the storage units**, in the SetUp table. These values are specified in the fields “Lower Limit” and “Upper Limit”. If the user enters a new record, the validation routines will check to see that the value of the numeric field lies within the bounds specified in these two fields. Enter the lowest permissible value for that field in the field “Lower Limit” and the highest permissible value in “Upper Limit” field. These fields should be left empty if the corresponding field of the library table is not numeric. In our example, these fields will be left empty in the record corresponding to “UNS Designation” as it is a text field. The limits for remaining records (2 to 5) have been filled.

- **DecimalsSI and DecimalsIP** (Refer to Fig. 2.11)

The number of significant digits to be displayed can be controlled, for each numeric field, for each unit (SI or IP), through the fields “DecimalsSI” and “DecimalsIP” in the SetUp table. In our example, we set the units for all the fields to two significant digits. The format in which the number to be displayed is desired, in entered in the fields. E.g. the format can be “#0.0” for 1 decimal place, “#0.00” for two decimal places. Fig.
2.11 shows that for Melt Temperature, the DecimalsIP has been entered as #.0 whereas for remaining fields it is #.00. This reflects in frmG1Table, which shows, for record G301276, Melt Temperature is 1200.0 (i.e. one decimal precision) whereas others have 2 decimal places precision in their display. If a field has a value between 0 and 1, say 0.7, then entering DecimalsIP as #.0 will display it as .7, whereas, if #0.0 is entered, it will be displayed as 0.7.

![Table](image)

<table>
<thead>
<tr>
<th>Title</th>
<th>DecimalsIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alloy Library</td>
<td></td>
</tr>
<tr>
<td>UNS Designation</td>
<td>#.00</td>
</tr>
<tr>
<td>Carbon</td>
<td>#.00</td>
</tr>
<tr>
<td>Nitrogen</td>
<td>#.00</td>
</tr>
<tr>
<td>Density</td>
<td>#.00</td>
</tr>
<tr>
<td>Melt Temperature</td>
<td>#.0</td>
</tr>
</tbody>
</table>

**Fig 2.11 Relation between the field entry under DecimalsIP in SetUp table and the display in frmG1table**

### 2.2 The Std and Usr Tables

Step1: Create the Std and Usr tables in the same database, where you created the SetUp tables but with the structure as shown in Fig. 2.12. The Std and Usr tables (Refer to Fig. 2.1), in our example, MatlStd and MatlUsr tables are similar in their structure. The fields in these tables are corresponding to records 1 to 5 of the SetUp table (Fig 2.12). i.e. from MatlSetUp table, its UNS Designation, Carbon, Nitrogen, Density and Melt Temperature.

<table>
<thead>
<tr>
<th>UNS Designation</th>
<th>Carbon</th>
<th>Nitrogen</th>
<th>Density</th>
<th>Melt Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>G301276</td>
<td>30.00</td>
<td>10.00</td>
<td>230.00</td>
<td>1200.0</td>
</tr>
<tr>
<td>G301576</td>
<td>30.00</td>
<td>15.00</td>
<td>250.00</td>
<td>1500.0</td>
</tr>
</tbody>
</table>
Step 2: A primary key is a field or a combination of fields whose values cannot be repeated. **Set the field or the combination of fields that can’t have duplicate values, as the primary key** (here UNS). **It is very important that you set a field as the Primary key**, to avoid problems later on.

To set a field as the Primary key, select that particular field and then click on the primary key icon in the toolbar. Then save the tables. If no primary key is selected, Access issues a warning as shown in Fig 2.13. **Click the No button.** If you click, Yes, Access creates an Id field automatically. This will result in runtime error, as this field is not recognized in the SetUp table.

![Fig. 2.13 Prompt for Primary Key declaration.](image)

Step 3: Set the precision (Refer to Fig 2.3) of these fields, as per your requirements.
2.3 Creating the User Interface For The Library

The next step after adding the database is to add forms and code to access and edit it.

Step1: Copy the following from the disk into your project directory.

- **Forms(*.frm):**
  1. frmDialog
  2. frmG1Table
  3. frmG1AddModDetSearch
  4. frmHelpForSearch.

- **Modules(*.bas):**
  1. Def1Tbl
  2. FileUtil
  3. G1Table
  4. GUtils

- **Classes(*.cls):**
  1. Dialog

Step2: Add the above to your project in the project workspace.

Step3: Go to Project menu and select References. It shows up a list of Available References as in Fig. 2.14.

Step4: Select the check box for Microsoft DAO 3.51 Object Library (highlighted in Fig.2.14) and press OK.
2.3.1 Deciding the Library Editing Interface

This section briefly explains which form is to be selected for editing.

In most of the cases, it is desired that the interface resemble that of the ApplicationDataInput form as close as possible. Since the user works with the ApplicationDataInput form, the user is familiar with it and hence may find it more comfortable. So the first preference would be for Type 1 i.e. ApplicationDataInput form. Sometimes, it’s not possible to use the ApplicationDataInput form, for e.g. if the ApplicationDataInput form contains a DBGrid control, this control has to be connected to a data source and hence not possible to use for our interface.

The Automatic form is the easiest of all forms to customize. It generates all fields in a vertical alignment with the “OK”, “Cancel” and “Clear” button at the bottom (Fig. A.4).
However, if the number of fields is more, then these tend to go outside the visible range of the screen and hence not easy to work on.

The Custom form (Fig A.3), as the name implies, is customized for editing purposes. Its more easy to use compared to the Parent form. This is more aesthetically appealing than the Automatic form but has the overhead of extra space because of addition of an extra form in the workspace.

### 2.3.2 Adding the User Interface (Refer to Fig 2.15)

![Diagram](image-url)

**Inclusion of Routines\Events in a form is indicated as:**

- **Form name**
- **Included in**
- **Routines\Events**

*Fig 2.15 User interface elements for using the Library*
Fig 2.15 shows the user interface elements for using the Library and the customization event and routines along with the forms in which they have to be included. As the above figure shows, the ApplicationDataInput form will always contain the routines TransferDataIn and TransferDataOut and the event Library_Click, irrespective of the Interface used for editing the library records. **The routines are always included under the General\Declarations section and the code within the events will be included under the corresponding events on the ApplicationDataInput and/or Custom form.**

To summarize:

1. When Automatic form is used, add:
   - TransferDataIn, TransferDataOut from frmStart to ApplicationDataInput form
   - Library_Click from frmStart to ApplicationDataInput form

2. When Custom form is used, add:
   - TransferDataIn, TransferDataOut from frmStart to ApplicationDataInput Form
   - Library_Click from frmStart to ApplicationDataInput form
   - TransferDataIn, TransferDataOut, GetFocusOnFields, DisableControls from frmCustom to Custom form.
   - OK_Click, Cancel_Click, Form_Load from frmCustom to Custom form

3. When ApplicationDataInput form (Instance of) is used, add:
   - TransferDataIn, TransferDataOut, GetFocusOnFields, DisableControls from frmStart to ApplicationDataInput form.
   - Library_Click from frmStart to ApplicationDataInput form
• OK_Click, Cancel_Click, Form_Load from frmStart to ApplicationDataInput form

The TransferDataIn and TransferDataOut routines transfer data to and from the controls (text boxes, labels, combo box, list box, check box, etc.), that are used to display the data from the library. The GetFocusOnFields routine shifts the focus on the appropriate control (text box, list box or combo box) that fails the validation. The DisableControls routine is used to disable appropriate controls when in library mode.

2.3.2.1 Using the ApplicationDataInput form (Refer to Appendix A)

Step1: Add frmStart to your project workspace. This is done to facilitate easy transfer of code to your form.

Step2: Copy the code routines TransferDataIn, TransferDataOut, GetFocusOnFields, and DisableControls from the General/Declarations Section of frmStart and paste it in the General/Declarations section of your ApplicationDataInput form.

Step3: Make appropriate changes to these routines.

Step4: Copy the code within the Library_Click event of frmStart and paste it within the corresponding Click event of your control.

Step5: Make changes to the Library_Click routine as explained below, with reference to the example in the disk. (The words in Bold Italic are the only ones to be changed by the programmer to integrate it to the database).

(Here enter the name of your application e.g. Demo)

• Eg1.G1_strAppName = "Demo"

(Here enter the name of the database containing the library e.g. Library)

• Eg1.G1_strDBName1 = App.path & "Mail.mdb"
(Here enter the name of the Setup table e.g. MatlSetUp)

• Eg1.G1_strSetupTblName1 = "MatlSetUp"

(Here enter the name of the table representing the standard library e.g. MatlStd)

• Eg1.G1_strStdLib1 = "MatlStd"

(Here enter the name of the table representing the user library e.g. MatlUsr)

• Eg1.G1_strUsrLib1 = "MatlUsr"

(Here enter specify a directory where *.lib files need to be stored e.g.".\")

• Eg1.G1_DataDirectory = ".\"

(Set unique string to the name of the record, which you want to be highlighted when the grid with records is displayed, for the first time. For e.g. if it is desired that, when all the records are displayed in the grid, a particular record be highlighted, say, the alloy with UNS Designation G3012762, then instead of UnqStr_DB1 enter “G3012762”. If UnqStr_DB1 is not replaced by any particular record string name, then the first record in the grid is highlighted.)

• Eg1.G1_UnqString = UnqStr_DB1

(Here enter the name of the form which calls the library e.g. frmStart)

• Set Eg1.G1_ApplicationDataInputForm = frmStart

(Here enter the name of the form you wish to use for editing the library. In case, it will be the same as Eg1.G1_ApplicationDataInputForm with the New keyword in front of it, as this is an instance.)

• Set Eg1.G1_LibModificationForm = New frmStart

(Here set the names of the OK, Cancel and Library buttons of your interface form. e.g If you are using the ApplicationDataInput form, and it has its OK button named as buOK then the statement will be:
Set Eg1.G1_OK = Eg1.G1_LibModificationForm.buOK

Thus depending on the interface form, set the right hand side. The same thing holds good for the Cancel and Library buttons)

- Set Eg1.G1_OK = Eg1.G1_LibModificationForm.OK
- Set Eg1.G1_Cancel = Eg1.G1_LibModificationForm.Cancel
- Set Eg1.G1_Library = Eg1.G1_LibModificationForm.Library

(Here, enter the Boolean variable used in your application to indicate units. If the right hand side is true, it indicates SI units else its IP units. If the application does not make use of any variables for indicating units, enter True or False in the right side for the appropriate units.)

- Eg1.G1_UnitSI = True

Step6: Copy the code within the events Ok_Click, Cancel_Click and Form_Load to the corresponding events in your ApplicationDataInput form. These events are of form

If G1LibOperation = "" Then

----- your code comes here -------

Else

--- generic database code----

End If

G1LibOperation is a global variable used to indicate that the application is in the Library mode and the editing operation is taking place. When G1LibOperation = "", it means it is in the non-Library mode. So enter your code within the If-Else part only.

**Library** is the name of the button that triggers the Library_Click routine. Replace it by the name of your button that accesses the library. The **Library** acts as the HELP button during the Search operation. The caption gets changed to “HELP” during Form_Load (as seen above) and pops up frmHelpForSearch on clicking, when in the Search mode.
2.3.2.2 **Using the Custom form** (Refer to Appendix A)

The Custom form should have the OK and Cancel button. A button named “Help” is also desired on the Custom form. The Click event for this button Help_Click pops up the form frmHelpForSearch. This form shows the kind of Search operations that are possible to be carried on the library.

Step1: Add frmStart and frmCustom to your project workspace. This is done to facilitate any transfer of code to your form.

Step2: Copy the code routines TransferDataIn and TransferDataOut from the General/Declarations Section of frmStart and paste it in the General/Declarations section of your ApplicationDataInput form.

Step3: Make appropriate changes to the TransferDataIn and TransferDataOut routines.

Step4: Copy the code within the Library_Click event of frmStart and paste it within the corresponding Click event of your control.

Step5: Repeat Step5 given for ApplicationDataInput form with the only changes as given below:

(In this case, we don’t need the New keyword, as this is a different form and not an instance of the ApplicationDataInput form)

- Set Eg1.G1_LibModificationForm = **frmCustom**

(Here enter the name of the OK, Cancel and Library buttons of your Library editing form, here frmCustom, in place of the bold, italicized text.)

- Set Eg1.G1_OK = Eg1.G1_LibModificationForm.**OK**
- Set Eg1.G1_Cancel = Eg1.G1_LibModificationForm.**Cancel**
- Set Eg1.G1_Library = Eg1.G1_LibModificationForm.**Library**
Step 6: Now, from the frmCustom form, copy the entire code-routines: TransferDataIn, TransferDataOut, GetFocusOnFields, DisableControls from the General/Declarations section to the General/Declarations section of your custom form.

Step 7: Make appropriate changes to these routines.

Step 8: Copy the code within the events: OK_Click, Cancel_Click, Form_Load, Help_Click (if you are using the Help button).

Step 9: Remove frmStart and frmCustom from your workspace.

2.3.2.3 Using the Automatic form (Refer Appendix A)

Step 1: Add frmStart to the project workspace. This is done to facilitate easy transfer of code to your form.

Step 2: Copy the routines TransferDataIn, TransferDataOut from the General/Declarations Section from frmStart and paste it in the General/Declarations section of your ApplicationDataInput form.

Step 3: Make appropriate changes to the TransferDataIn and TransferDataOut routines.

Step 4: Copy the code under Library_Click of frmStart to the corresponding Click Event of your ApplicationDataInput form.

Step 5: Do the same things as Step 5 of your ApplicationDataInput form making a few changes as given below:
(Here enter the name of the form you wish to use for editing the library. For Automatic form, set the right hand side to Nothing)

- Set Eg1.G1_LibModificationForm = Nothing
(Set the entire right hand side to Nothing, for all the three buttons)

- Set Eg1.G1_OK = Nothing
- Set Eg1.G1_Cancel = Nothing
- Set Eg1.G1_Library = Nothing

Step 6: Remove frmStart from your workspace.

This marks the end of the Programmer’s manual.
CHAPTER 3: CONCLUSION & RECOMMENDATIONS

3.1 CONCLUSION

New functionalities were successfully added to the generic database. The generic database can now have three types of interfaces for editing records. Control over the number of significant digits to be displayed has been implemented. The search routines have been made more robust by adding a tolerance value that accommodates for accuracy loss while converting and storing data. This generic database has been successfully used for implementing all the libraries in HVAC Load Explorer with minimum modifications. The generic routines take care of implementing all the qualities expected from an Engineering Database Application.

3.2 RECOMMENDATIONS

1. A context sensitive Help would be helpful.

2. The above generic database was for 1 Table libraries. It should also be implemented for 2 table libraries (Master-Detail type).
APPENDIX A: INTERFACE TYPES

Refer to the example given in the accompanying disk. This chapter shows the different types of interfaces possible in a generic database application:

The ApplicationDataInput form (frmStart) is as shown in Fig. A.1

![ApplicationDataInput Form](image)

Clicking the Library button access both the standard and user libraries of Alloys.

Now, for editing operations, any of the following three forms can be used

1. The same starting form (ApplicationDataInput Form).
2. Custom form
3. Automatic form.
1st Case: Using ApplicationDataInput form as LibEditing Form

To customize, following events and routines are included in the ApplicationDataInput form:

- **Events**
  1. Library_Click
  2. Form_Load
  3. OK_Click
  4. Cancel_Click

- **Routines**
  1. TransferDataIn
  2. TransferDataOut
  3. GetFocusOnFields
  4. DisableControls
Customization Code:

Events
Private Sub Library_Click()

    Dim Eg1 As G1Info
    Dim counter As Integer

    'Enter the application name
    Eg1.G1_strAppName = "Demo"
    'Enter the database name here
    Eg1.G1_strDBName1 = App.path & "\Matl.mdb"
    'Enter the Setup table name here
    Eg1.G1_strSetupTblName1 = "MatlSetUp"
    'Enter the name of the table representing the std library here
    Eg1.G1_strStdLib1 = "MatlStd"
    'Enter the name of the table representing the usr library here
    Eg1.G1_strUsrLib1 = "MatlUsr"
    'Specify a directory where *.lib files need to be stored
    Eg1.G1_DataDirectory = ".\"
    'Set the unique string
    Eg1.G1_UnqString = UnqStr_DB1
    'if u want the record with UNS desig G301276 be highlighted when the grid
    'is displayed, then set Eg1.G1_UnqString as shown below
    Eg1.G1_UnqString = "G301276"

    ' Here enter the name of the form which has the library button
    Set Eg1.G1_ApplicationDataInputForm = frmStart

    ' Enter the name of the LibModification form depending on which interface u
    ' choose to use
    Set Eg1.G1_LibModificationForm = New frmStart

    ' Enter the name of the buttons in the LibModification form
    Set Eg1.G1_OK = Eg1.G1_LibModificationForm.OK
    Set Eg1.G1_Cancel = Eg1.G1_LibModificationForm.Cancel
    Set Eg1.G1_Library = Eg1.G1_LibModificationForm.Library

    ' Enter your boolean variable on the right hand side or enter True/False.
    ' if RHS is true  ---> SI units
    ' if RHS is false  ---> IP units
    Eg1.G1_UnitSI = True

    'this routine calls the form G1table in modal form
    G1TableOperations Eg1

End Sub
Private Sub Form_Load()
    ' in the library mode
    If G1LibOperation = "" Then
        ' -------enter your code here-------
    Else ' the library mode
        ' loading the form
        G1LoadForm
    End If
End Sub

Private Sub OK_Click()
    If G1LibOperation = "" Then
        '------- enter your code here --------
    Else
        G1OK
    End If
End Sub

Private Sub Cancel_Click()
    If G1LibOperation = "" Then
        '------- enter your code here --------
        Unload Me
    Else
        G1Cancel
    End If
End Sub
**Routines**

Public Sub TransferDataOut()
    txtFl(0) = Me.txUNS.Text
    txtFl(1) = Format$(Me.txCarbon.Text, GetNumberFormat(2))
    txtFl(2) = Format$(Me.txNickel.Text, GetNumberFormat(3))
    txtFl(3) = Format$(Me.txDensity.Text, GetNumberFormat(4))
    txtFl(4) = Format$(Me.txTemp.Text, GetNumberFormat(5))
End Sub

Public Sub TransferDataIn()
    Me.txUNS.Text = txtFl(0)
    Me.txCarbon.Text = txtFl(1)
    Me.txNickel.Text = txtFl(2)
    Me.txDensity.Text = txtFl(3)
    Me.txTemp.Text = txtFl(4)
End Sub
Public Sub GetFocusOnField(Index As Integer)
    Select Case Index
        Case 0:
            txUNS.SetFocus
        Case 1:
            txCarbon.SetFocus
        Case 2:
            txNickel.SetFocus
        Case 3:
            txDensity.SetFocus
        Case 4:
            txTemp.SetFocus
    End Select
End Sub

Public Sub DisableControls (textstate As Boolean)
    ' disable all the extra text here
    txAirplane.Enabled = False
    ' disable all the extra buttons here
    Notes.Enabled = False

    ' Enter all fields in your database that appear in the ApplicationDataInput form here
    txUNS.Enabled = textstate
    txCarbon.Enabled = textstate
    txNickel.Enabled = textstate
    txDensity.Enabled = textstate
    txTemp.Enabled = textstate
End Sub
2nd case: Using the Custom form as LibEditing Form

To customize, following events and routines are included:

In the Custom form:

- Events
  1. Form_Load
  2. OK_Click
  2. Cancel_Click

- Routines
  1. TransferDataIn
  2. TransferDataOut
  3. GetFocusOnFields
  4. DisableControls
In the ApplicationDataInput Form:

- **Events**
  1. Library_Click

- **Routines**
  1. TransferDataIn
  2. TransferDataOut
Customization Code:

Events

Private Sub Form_Load()
    G1LoadForm
End Sub

Private Sub OK_Click()
    G1OK
End Sub

Private Sub Cancel_Click()
    G1Cancel
End Sub
Private Sub Library_Click()

    Dim Eg1 As G1Info
    Dim counter As Integer

    'Enter the application name
    Eg1.G1_strAppName = "Demo"
    'Enter the database name here
    Eg1.G1_strDBName1 = App.path & "\Matl.mdb"
    'Enter the Setup table name here
    Eg1.G1_strSetupTblName1 = "MatlSetUp"
    'Enter the name of the table representing the std library here
    Eg1.G1_strStdLib1 = "MatlStd"
    'Enter the name of the table representing the usr library here
    Eg1.G1_strUsrLib1 = "MatlUsr"
    'Specify a directory where *.lib files need to be stored
    Eg1.G1_DataDirectory = ".\"
    'Set the unique string
    Eg1.G1_UnqString = UnqStr_DB1
    'if u want the record with UNS desig G301276 be highlighted when the grid
    ' is displayed, then set Eg1.G1_UnqString as shown below
    Eg1.G1_UnqString = "G301276"

    ' Here enter the name of the form which has the library button
    Set Eg1.G1_ApplicationDataInputForm = frmStart
    ' Enter the name of the LibModification form depending on which interface u
    ' choose to use
    Set Eg1.G1_LibModificationForm = frmCustom
    ' Enter the name of the buttons in the LibModification form
    Set Eg1.G1_OK = Eg1.G1_LibModificationForm.OK
    Set Eg1.G1_Cancel = Eg1.G1_LibModificationForm.Cancel
    Set Eg1.G1_Library = Eg1.G1_LibModificationForm.Library
    ' Enter your boolean variable on the right hand side or enter True/False.
    ' if RHS is true    ---> SI units
    ' if RHS is false   ---> IP units
    Eg1.G1_UnitSI = True

    'this routine calls the form G1table in modal form
    G1TableOperations Eg1
End Sub
Routines

Public Sub TransferDataOut()
    txtFl(0) = Me.txUNS.Text
    txtFl(1) = Format$(Me.txCarbon.Text, GetNumberFormat(2))
    txtFl(2) = Format$(Me.txNickel.Text, GetNumberFormat(3))
    txtFl(3) = Format$(Me.txDensity.Text, GetNumberFormat(4))
    txtFl(4) = Format$(Me.txTemp.Text, GetNumberFormat(5))
End Sub

Public Sub TransferDataIn()
    Me.txUNS.Text = txtFl(0)
    Me.txCarbon.Text = txtFl(1)
    Me.txNickel.Text = txtFl(2)
    Me.txDensity.Text = txtFl(3)
    Me.txTemp.Text = txtFl(4)
End Sub
Public Sub GetFocusOnField(Index As Integer)
' Enter all fields in your database that appear in the Custom form here
  Select Case Index
    Case 0:
      txUNS.SetFocus
    Case 1:
      txCarbon.SetFocus
    Case 2:
      txNickel.SetFocus
    Case 3:
      txDensity.SetFocus
    Case 4:
      txTemp.SetFocus
  End Select
End Sub

Public Sub DisableControls(textstate As Boolean)
' Enter all fields in your database that appear in the Custom form here
  txUNS.Enabled = textstate
  txCarbon.Enabled = textstate
  txNickel.Enabled = textstate
  txDensity.Enabled = textstate
  txTemp.Enabled = textstate
End Sub
3rd case: Using the Automatic form as LibEditing Form

Fig: A.3 Automatic form as the LibEditing Form

To customize, following events and routines are included:

In the ApplicationDataInput Form:

- **Events**
  1. Library_Click

- **Routines**
  1. TransferDataIn
  2. TransferDataOut

The Automatic form contains the routines for TransferDataIn, TransferDataOut, GetFocusOnFields, DisableControls and the Events Form_Load, OK_Click, Cancel_Click. So nothing additional needs to be added.
Customization Code

Events

Private Sub Library_Click()

    Dim Eg1 As G1Info
    Dim counter As Integer

    'Enter the application name
    Eg1.G1_strAppName = "Demo"
    'Enter the database name here
    Eg1.G1_strDBName1 = App.path & "\Matl.mdb"
    'Enter the Setup table name here
    Eg1.G1_strSetupTblName1 = "MatlSetUp"
    'Enter the name of the table representing the std library here
    Eg1.G1_strStdLib1 = "MatlStd"
    'Enter the name of the table representing the usr library here
    Eg1.G1_strUsrLib1 = "MatlUsr"
    'Specify a directory where *.lib files need to be stored
    Eg1.G1_DataDirectory = ".";
    'Set the unique string
    Eg1.G1_UnqString = UnqStr_DB1
    ' if u want the record with UNS desig G301276 be highlighted when the grid
    ' is displayed, then set Eg1.G1_UnqString as shown below
    Eg1.G1_UnqString = "G301276"

    ' Here enter the name of the form which has the library button
    Set Eg1.G1_ApplicationDataInputForm = frmStart

    ' Set the Right Hand Side to Nothing
    Set Eg1.G1_LibModificationForm = Nothing

    ' Set the Right Hand Side for all buttons to Nothing
    Set Eg1.G1_OK = Nothing
    Set Eg1.G1_Cancel = Nothing
    Set Eg1.G1_Library = Nothing

    ' Enter your boolean variable on the right hand side or enter True/False.
    ' if RHS is true ---> SI units
    ' if RHS is false ---> IP units
    Eg1.G1_UnitSI = True

    'this routine calls the form G1table in modal form
    G1TableOperations Eg1

End Sub
Routines

Public Sub TransferDataOut()
    txtFl(0) = Me.txUNS.Text
    txtFl(1) = Format$(Me.txCarbon.Text, GetNumberFormat(2))
    txtFl(2) = Format$(Me.txNickel.Text, GetNumberFormat(3))
    txtFl(3) = Format$(Me.txDensity.Text, GetNumberFormat(4))
    txtFl(4) = Format$(Me.txTemp.Text, GetNumberFormat(5))
End Sub

Public Sub TransferDataIn()
    Me.txUNS.Text = txtFl(0)
    Me.txCarbon.Text = txtFl(1)
    Me.txNickel.Text = txtFl(2)
    Me.txDensity.Text = txtFl(3)
    Me.txTemp.Text = txtFl(4)
End Sub
APPENDIX B: CUSTOMIZATION ROUTINES

The following routines (either some or all) in the ApplicationDataInput form or the Custom form are required and need to be customized for each library interface.

<table>
<thead>
<tr>
<th>Name</th>
<th>TransferDataOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub TransferDataOut</td>
</tr>
<tr>
<td>Called By</td>
<td>G1Table.bas\G1LoadForm, G1OK, G1Table.frm\Form_Load</td>
</tr>
<tr>
<td>Purpose</td>
<td>This routine stores the data present in the editing/ApplicationDataInput form to an array of text fields txtFl().</td>
</tr>
</tbody>
</table>

Table 2. TransferDataOut

<table>
<thead>
<tr>
<th>Name</th>
<th>TransferDataIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub TransferDataIn</td>
</tr>
<tr>
<td>Called By</td>
<td>G1Table.bas\G1LoadForm, G1OK</td>
</tr>
<tr>
<td>Purpose</td>
<td>This routine transfers the data present in the array txtFl() to the editing/ApplicationDataInput form.</td>
</tr>
</tbody>
</table>

Table 3. TransferDataIn

<table>
<thead>
<tr>
<th>Name</th>
<th>GetFocusOnFields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub GetFocusOnFields (integer)</td>
</tr>
<tr>
<td>Called By</td>
<td>G1Table.bas\G1OK</td>
</tr>
<tr>
<td>Purpose</td>
<td>This routine is called when the validation on a field fails. It gets the information about that field from the argument passed to it and shifts the focus on that field based on a set of Select Case statements.</td>
</tr>
</tbody>
</table>

Table 4. GetFocusOnFields

47
<table>
<thead>
<tr>
<th>Name</th>
<th>DisableControls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Private Sub DisableControls (Boolean)</td>
</tr>
<tr>
<td>Called By</td>
<td>G1Table.bas\G1LoadForm</td>
</tr>
</tbody>
</table>
| Purpose      | This routine is used to  
1. Disable any extra buttons (extra buttons are the buttons other than OK, Cancel and Library buttons) and extra text fields that are not used in that particular library operation.  
2. In the “View Details” mode, this disables all the fields. |

Table 5. DisableControls
APPENDIX C: EXAMPLES FROM HVAC LOAD EXPLORER

Example 1: Layer library with Custom Form as LibEditing form

Fig. C.1 shows the Layer information form that connects to the layer library.

![Layer Information Table]

Clicking on the Library button opens up the grid for displaying records. When any of the editing buttons/search buttons is clicked, it pops up the screen as shown in Fig. C.2

![Add: Layers Library]

Fig. C.2 Interface for editing layer library
This form is different from the main form. It has been customized for editing operations. In this case, the main form uses MSFlexgrid control for displaying data. This control does not permit direct editing in its fields. So it cannot be used for editing operations. Hence, the custom form has been used.

**Example 2: Window library with ApplicationDataInput form as LibEditing form**

The main form (ApplicationDataInput) is as shown in Fig C.3

![Window Details](image)

**Fig. C.3 ApplicationDataInput form that connects to Windows Library**

Clicking on the Library button opens up the grid for displaying the records. When any of the editing buttons/search button is clicked, it pops up the screen as show in Fig. C.4
Fig. C.4 Interface form for editing records

This form (Fig. C.4) is similar in appearance as the ApplicationDataInput form (Fig. C.3). It is an instance of the ApplicationDataInput form. Here, the fields: Area, Reveal, Overhang Width and Overhang Protrusion have been disabled, as they are not part of the Windows library.
APPENDIX D: FUNCTIONS AND SUBROUTINES IN THE MODULES

The Generic database makes use of functions and subroutines for managing the libraries.

These are listed below along with their purpose, module wise:

- Gutils.bas

  This module contains sub routines, which will be used by the generic utilities.

<table>
<thead>
<tr>
<th>Name</th>
<th>ConvForward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub ConvForward ( Single, Single )</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function is used to convert the quantity (number) passed to it from the storage units to the other one. Say the data is stored in the table in IP units and the user is working in SI units, this routine converts the data from the table to SI when data is retrieved from the table. The data displayed is also in the converted form (i.e. SI) so the user does not know the units in which data is stored inside the table.</td>
</tr>
<tr>
<td>Implementation</td>
<td>[(\text{qty})<em>{\text{SI}} = (\text{qty})</em>{\text{IP}} \times \text{conv}(1) + \text{conv}(2)]</td>
</tr>
<tr>
<td></td>
<td>where (\text{conv}(1)) = the data in the “Unit-Scale” field of the Setup Table</td>
</tr>
<tr>
<td></td>
<td>(\text{conv}(2)) = the data in the “Unit-Offset” field of the Setup Table</td>
</tr>
<tr>
<td></td>
<td>e.g. (deg C) = (deg F) * 0.55 + (– 17.78)</td>
</tr>
</tbody>
</table>

Table 6. ConvForward
<table>
<thead>
<tr>
<th>Name</th>
<th>ConvReverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub ConvReverse( Single, Single )</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function is used to convert the quantity (number) passed to it into the storage units from the other one. Say the data is stored in the table in IP units and the user is working in SI units, this routine converts the data from SI to IP when data is stored/ searched in the table.</td>
</tr>
</tbody>
</table>
| Implementation | ( qty)\_IP = ( ( qty)\_SI - \text{conv}(2) ) / \text{conv}(1)  
where  
\text{conv}(1) = the data in the “Unit-Scale” field of the Setup Table  
\text{conv}(2) = the data in the “Unit-Offset” field of the Setup Table  
e.g. (deg F) = ( (deg C) – (– 17.78)) / 0.55 |

Table 7. ConvReverse

<table>
<thead>
<tr>
<th>Name</th>
<th>ValidityCheck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub ValidityCheck</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function checks if the data entered by the user in the text field belong to the valid set of characters. If not, the GetFocusOnField routine is called and that field gets the focus.</td>
</tr>
</tbody>
</table>

Table 8. ValidityCheck

<table>
<thead>
<tr>
<th>Name</th>
<th>GetNextLine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Type</td>
<td>Integer</td>
</tr>
<tr>
<td>Syntax</td>
<td>Public function GetNextLine( Integer, String ) as Integer</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function is used to read the next non-empty line from a file passed to it as argument. This function considers lines that only have tabs and spaces as empty lines. When empty lines are encountered, they are ignored and the function tries to read the next non-empty line. The value that is returned by this function is the length of the non-empty line read. If the end of file is encountered, value returned by the function is zero.</td>
</tr>
</tbody>
</table>

Table 9. GetNextLine

53
### Table 10. GetSearchCriteria

<table>
<thead>
<tr>
<th>Name</th>
<th>GetSearchCriteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Type</td>
<td>String</td>
</tr>
<tr>
<td>Syntax</td>
<td>Public Function GetSearchCriteria(Form, String) As String</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function generates the search criteria used for searching the database.</td>
</tr>
</tbody>
</table>

### Table 11. GetTolerance

<table>
<thead>
<tr>
<th>Name</th>
<th>GetTolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Type</td>
<td>Single</td>
</tr>
<tr>
<td>Syntax</td>
<td>Private Function GetTolerance(number As Integer) As Single</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function returns the tolerance level within which the data is searched in the database.</td>
</tr>
</tbody>
</table>

### G1Table.bas

This module contains functions, which will be used for handling generic library data stored in a single table. G1 implies generic one table and hence this module is named as G1Table.bas

### Table 12. G1TableSetUp

<table>
<thead>
<tr>
<th>Name</th>
<th>G1TableSetUp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub G1TableSetUp(G1TableInfo, Single)</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function sets the properties of the grid in which the library is to be displayed. The title of the columns of the grid, the appropriate units, the alignment details (left, right, center), the height and width of the grid are all set by this function. The Setup Table information is also read in this routine.</td>
</tr>
<tr>
<td>Name</td>
<td>FillGenericGrid</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>Syntax</td>
<td>Public Sub FillGenericGrid ( G1TableInfo, Single, Recordset, Integer, Integer )</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function fills the grid with the data from the recordset passed to it as argument. The grid is filled with the data in the current working unit system. The current working units system may or may not be the same as the storage system. The data is filled in the grid with the color specified by the argument passed to it.</td>
</tr>
</tbody>
</table>

Table 13. FillGenericGrid

<table>
<thead>
<tr>
<th>Name</th>
<th>G1TableExport</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub G1TableExport ( G1TableInfo, Single, String, Recordset )</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function writes the library data stored in a single table, represented by recordset within the program, over to a text file. The data written over to the text file in a specific format in the unit specified by the variable “UnitSystem”. The recordset passed as argument represents the table representing the User Library or the records from the User Library, which match a specific search criterion.</td>
</tr>
</tbody>
</table>

Table 14. G1TableExport

<table>
<thead>
<tr>
<th>Name</th>
<th>G1TableImport</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub G1TableImport ( G1TableInfo, Single, String, Recordset)</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function reads a text file and stores the data from it in the User Library, which is represented by a single table in the database. The data is stored in the User Library in the storage unit system. The data from the text file may or may not be in the storage unit system.</td>
</tr>
</tbody>
</table>

Table 15. G1TableImport
Name | OpenG1TableDb
---|---
Syntax | Public Function OpenG1TableDb as Integer
Return Type | Integer
Purpose | This function opens the generic database containing the library data stored in a single table. The recordsets, which will represent the Standard Library and the User Library in the program, are also filled with data from the corresponding tables. This function returns 1 if it is successful in opening the database else it returns 0.

Table 16. OpenG1TableDb

Name | GetNumberFormat
---|---
Syntax | Public Function GetNumberFormat (integer) as String
Return Type | Integer
Purpose | This function returns a string that specifies the format in which data has to be displayed. It retrieves this data from the SetUp table, which stores this format.

Table 17. GetNumberFormat

Name | G1TableOperations
---|---
Syntax | Public Sub G1TableOperations (G1Info)
Purpose | This routine serves as the entry point into the Grid, where the editing operations on the records are carried out. The grid is modal in nature. Once the control comes back, with either the user selecting a record or canceling, the data transfer into the parent form takes place. Also, this shows up the Help screen during Search, when the user clicks the Help button. The Help button is the same as the Library button with caption changed and serves a different purpose during Search. For other editing operations the Library button will be disabled.

Table 18. G1TableOperations
### Table 19. G1Cancel

<table>
<thead>
<tr>
<th>Name</th>
<th>G1Cancel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub G1Cancel</td>
</tr>
<tr>
<td>Purpose</td>
<td>This routine is called when the user clicks on cancel button of the editing form. The user is prompted for saving the data and if the user says Ok its calls G1OK else it unloads the form.</td>
</tr>
</tbody>
</table>

### Table 20. G1OK

<table>
<thead>
<tr>
<th>Name</th>
<th>G1OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub G1OK</td>
</tr>
<tr>
<td>Purpose</td>
<td>This routine is called when the user clicks the OK button in the editing form. The record is Added, Modified, Searched or its Details are shown based on which button is pressed.</td>
</tr>
</tbody>
</table>

### Table 21. G1LoadForm

<table>
<thead>
<tr>
<th>Name</th>
<th>G1LoadForm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub G1LoadForm</td>
</tr>
<tr>
<td>Purpose</td>
<td>This routine is called in the Form load of the editing form. The fields are initialized to the selected row. If no row is selected then the fields are left blank. In case of Search, it puts an equal to sign (=) in front of the field value.</td>
</tr>
</tbody>
</table>

### Defn1Tbl.bas

### Table 22. G1SelectTable

<table>
<thead>
<tr>
<th>Name</th>
<th>G1SelectTable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub G1SelectTable (G1Info)</td>
</tr>
<tr>
<td>Purpose</td>
<td>This routine stores globally all the information related to library that uses the database such as the name of the application, database, SetUp table name, User and Std table names, the name of the parent form and the name of the instance form.</td>
</tr>
</tbody>
</table>
### FileUtil.bas

<table>
<thead>
<tr>
<th>Name</th>
<th>SafeMkDir</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub SafeMkDir ( String )</td>
</tr>
<tr>
<td>Purpose</td>
<td>This routine creates a new directory or folder. Only use when you know that the only error you can get is &quot;Directory already exists&quot;. If the directory is invalid or cannot be made for some other reason, you will NOT get any notification...</td>
</tr>
</tbody>
</table>

Table 23. SafeMkDir

<table>
<thead>
<tr>
<th>Name</th>
<th>SafeKill</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub SafeKill (String )</td>
</tr>
<tr>
<td>Purpose</td>
<td>This routine deletes files from a disk. Only use when you know that the only error you can get is &quot;No such file&quot; If any other error occurs, you will NOT get any notification.</td>
</tr>
</tbody>
</table>

Table 24. SafeKill

<table>
<thead>
<tr>
<th>Name</th>
<th>SplitFileExtension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub SplitFileExtension (String, String, String )</td>
</tr>
<tr>
<td>Purpose</td>
<td>This routine returns the file extension.</td>
</tr>
</tbody>
</table>

Table 25. SplitFileExtension
<table>
<thead>
<tr>
<th>Name</th>
<th>SplitFileSpec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Sub SplitFileSpec (String, String, String )</td>
</tr>
<tr>
<td>Purpose</td>
<td>This subroutine splits the file into its path title.</td>
</tr>
</tbody>
</table>

Table 26. SplitFileSpec

<table>
<thead>
<tr>
<th>Name</th>
<th>StripExtension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Function StripExtension (String) as String</td>
</tr>
<tr>
<td>Return Type</td>
<td>String</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function returns the file path.</td>
</tr>
</tbody>
</table>

Table 27. StripExtension

<table>
<thead>
<tr>
<th>Name</th>
<th>AddExtension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Function AddExtension (String, String) As String</td>
</tr>
<tr>
<td>Return Type</td>
<td>String</td>
</tr>
<tr>
<td>Purpose</td>
<td>If Extension is a file extension, it should include the period... (ie, .txt, not just txt). This makes the functions more generic, and allows for checking/adding any suffix, not just file extensions.</td>
</tr>
</tbody>
</table>

Table 28. AddExtension
<table>
<thead>
<tr>
<th>Name</th>
<th>ChangeExtension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Function ChangeExtension (String, String) As String</td>
</tr>
<tr>
<td>Return Type</td>
<td>String</td>
</tr>
<tr>
<td>Purpose</td>
<td>This changes the extension of the file.</td>
</tr>
</tbody>
</table>

Table 29. ChangeExtension

<table>
<thead>
<tr>
<th>Name</th>
<th>DontOverWrite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Function DontOverWrite (String, String) As Boolean</td>
</tr>
<tr>
<td>Return Type</td>
<td>String</td>
</tr>
<tr>
<td>Purpose</td>
<td>If the file name specified already exists, then this function gives the warning.</td>
</tr>
</tbody>
</table>

Table 30. DontOverWrite

<table>
<thead>
<tr>
<th>Name</th>
<th>FilePath</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Function FilePath (String) As String</td>
</tr>
<tr>
<td>Return Type</td>
<td>String</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function returns the file path of the file specified by the argument.</td>
</tr>
</tbody>
</table>

Table 31. FilePath
<table>
<thead>
<tr>
<th>Name</th>
<th>FileTitle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Function FileTitle (String) As String</td>
</tr>
<tr>
<td>Return Type</td>
<td>String</td>
</tr>
<tr>
<td>Purpose</td>
<td>This function returns the file Title of the file specified by the argument.</td>
</tr>
</tbody>
</table>

Table 32. FileTitle

<table>
<thead>
<tr>
<th>Name</th>
<th>SafeDir</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Public Function SafeDir (String) As String</td>
</tr>
<tr>
<td>Return Type</td>
<td>String</td>
</tr>
<tr>
<td>Purpose</td>
<td>This example uses the Dir function to check if certain files and directories exist.</td>
</tr>
</tbody>
</table>

Table 33. SafeDir