Reading Meter Data Exchange format files with Excel

Dr Martin Gill
Director OPPM Pty Ltd

The analysis of meter data provides insights into how electricity is generated and consumed. Microsoft Excel provides a useful platform to develop and refine the required analysis techniques. In Australia meter data is sometimes provided in files using Itron's Meter Data Exchange (MDE) format. This article discusses how to develop a module enabling Excel to process MDE files.

Introduction

The analysis of electricity consumption and generation can lead to a greater understanding of how electricity is generated and used. This analysis requires access to suitable meter data. In Australia Itron's Meter Data Management solutions are widely used. As a consequence meter data is sometimes provided in files using Itron's Meter Data Exchange (MDE) format.

Microsoft Excel provides a comprehensive platform upon which to test, develop and refine analysis techniques. Unfortunately Excel does not directly support Itron's MDE file format. Instead a module must be used to import the meter data.

This article describes the development of a module to import MDE files. A feature of Microsoft's Excel Visual Basic Environment appears to have contributed to intermittent problems with the first implementation. The article concludes by presenting a modification enabling Excel to reliably read MDE files.

Meter Data Exchange format

Itron's Meter Data Exchange (MDE) format[1] supports the exchange of metering data between their Meter Data Management solutions and other utility systems. These files use a binary format which is not directly supported by Excel.

An MDE file comprises four different record types allowing the file to store the meter data and information about each meter installation:

- Meter Site Record
- Meter Channel Record
- Channel Data Record
- Trailer Record

Example Meter Site Record

The Meter Site Record provides details about each meter installation. This includes customer details and the amount of meter data in following records. The format of the Meter Site Record is shown below:

Field Name	Bytes	Description
RLEN	01-02	Record Length
RCODE	03-04	Record Code
CM_CUSTID	05-24	Customer ID
CM_NAME	25-44	Customer Name
CM_ADDR1	45-64	Customer Address Line 1
CM_ADDR2	65-84	Customer Address Line 2
CM_ACCOUNT	85-104	Customer Account Number
	105-111	Reserved
CM_LOGChanS	112-115	Total Channels for Customer
	116-119	Reserved
TA_START	120-131	Start Time of data
TA_STOP	132-143	Stop Time of data
DST_FLAG	144	Daylight Savings Time Flag
	145-216	Reserved

Bytes 3 to 4 are used in all record types to specify the Record Code, for example if the RCODE = 1 then the record is a Meter Site Record.

Using Excel to Read MDE Files

The chosen approach was to create a user defined "Type" able to store each of the MDE records. For example the following user defined "Type" stores the Meter Site Record:

```
Type Meter_Site_Record_Type
   CM_CUSTID    As String * 20
   CM_NAME      As String * 20
   CM_ADDR1     As String * 20
   CM_ADDR2     As String * 20
   CM_ACCOUNT   As String * 20
   Reserved1    As String * 7
   CM_LOGChanS  As Long
   Reserved2    As String * 4
   TA_START     As String * 12
   TA_STOP      As String * 12
   DST_FLAG     As String * 1
   Reserved3    As String * 72
End Type
```

[1] Refer to link in the References for more details

Comparing the user defined "Type" to the table shows that a separate entry is created for each of the Fields listed in the MDE record (including the Reserved fields). To reduce ambiguity each entry is identified using the Field Name specified in Itron's specification.

Four different user defined "Types" are required, one for each of the record types used in the MDE file.

Observant readers may notice that the user defined "Type" does not contain the Record Length (RLEN) or Record Code (RCODE). These values are read separately and then used to identify the appropriate user defined Type to store the MDE record as demonstrated in the following snippet:

```
Get #FileNum, , RLEN
Get #FileNum, , RCODE
' Use RCODE to decide which Type to use
Select Case RCODE
  Case e_Site_Header_Record
    Get #FileNum, , Site_Record
    ' Process Site Record
  Case e_Channel_Record
    Get #FileNum, , Channel_Record
    ' Process Channel Record
  Case e_Trailer_Record
    Get #FileNum, , Trailer_Record
    ' Process the Trailer Record
  Case Else
    Get #FileNum, , Data_Record
    ' Process the (interval) Data Record
 End Select
```

A module based on the above approach successfully read numerous MDE files.

Intermittent Problem

The developed module worked reliably for weeks at a time. Then for no discernible reason the data being read from one of the processed MDE files would be subtly corrupted. Once the corruption appeared, the module would consistently read the MDE incorrectly.

A temporary solution involved adding a module to perform a dummy read of the corrupted MDE file one byte at a time. Reading the MDE file in this way often restored correct operation of the Excel Workbook.

This technique only restored correct operation around 90% of the time. When the technique did not work it was necessary to return to an archive version of the Excel Workbook. This took considerably more time since it also required the user to manually copy any enhancements implemented in the corrupted Excel Workbook into the archived version.

While unsatisfactory the two solutions allowed development to continue.

Researching the issue

The Microsoft Development Network (MSDN) provides detailed documentation for the numerous Microsoft products, including Excel. The MSDN also provides discussion forums where developers are able to seek help resolving issues they have encountered.

Searching the MSDN forums revealed other users occasionally report apparently valid modules inexplicitly producing vastly different results. Several users even posted their modules, however unsurprisingly (since the modules work correctly most of the time) other forum users failed to identify issues.

Interestingly one user also described the solution proposed above. Specifically copying the module to a new Excel Workbook restored correct operation. This user suggested the issue lay with the Visual Basic Environment (VBE).

To allow Excel to run on a wide range of different computers all Excel code modules actually run within the VBE. The VBE provides the interface with the computer. Critically this includes the ability to read files stored on the computer.

Digging Deeper

An insight into the issue was provided by a slight modification to the first solution discussed above. The modification listed the number of times different values appeared in the MDE file. This revealed that even when Excel was corrupting the MDE file, the file contents were identical. The conclusion was that the fault did not lie in how Excel was reading the file but in how it was subsequently interpreting the values after they were read.

Returning to the MSDN documentation a likely cause was identified:

> Storage Assignment. The common language runtime can assign storage based on the current characteristics of the platform on which your application is executing.

- If memory is nearly full, it might pack your declared elements as closely together as possible.
- In other cases it might align their memory addresses to natural hardware boundaries to optimize performance.

The translation is that Excel's VBE can change how it decides to store the user defined "Types". Depending on how the VBE decides to store the user defined Types determines if the module works correctly or corrupts the values.

Fixing the issue

Formal development environments allow users to control how user defined Types are stored. Microsoft Excel's VBE does not provide this level of control.

An alternative solution ensures each field of the MDE record is interpreted correctly regardless of how the VBE stores the data. This is achieved by modifying the module to read each field individually, as demonstrated in the following snippet.

```
Get #FileNum, , RLEN
Get #FileNum, , RCODE
' Use RCODE to decide which Type to use
Select Case RCODE
  Case e_Site_Header_Record
    With Site_Record
      Get #FileNum, , .CM_CUSTID
      Get #FileNum, , .CM_NAME
      Get #FileNum, , .CM_ADDR1
      Get #FileNum, , .CM_ADDR2
      Get #FileNum, , .CM_ACCOUNT
      Get #FileNum, , .Reserved1
      Get #FileNum, , .CM_LOGChanS
      Get #FileNum, , .Reserved2
      Get #FileNum, , .TA_START
      Get #FileNum, , .TA_STOP
      Get #FileNum, , .DST_FLAG
      Get #FileNum, , .Reserved3
    End With
    ' Process Site Record
  etc
 End Select
```

The snippet only shows the modification applied to the Site Record, but a similar modification is required to read each record type.

To date the modified module has been used to reliably read MDE files on a range of computers.

Conclusion

Analysis of meter data can provide useful insights into how electricity is generated and consumed. Microsoft Excel provides a convenient and easy to use platform upon which to test and develop the necessary algorithms. The wide use of Itron's Meter Data Management solutions in Australia means that meter data may be provided in Meter Data Exchange (MDE) format. When this is occurs a separate module is required to enable Excel to read the MDE formatted files.

Intermittent problems were encountered when using an elegant solution to read the files. While not conclusively proved evidence suggests the problem is related to Excel's Visual Basic Environment (VBE), specifically the lack of control over how the VBE stores user defined binary data. An implementation reading each element individually proved successful.

Citation

Comments or Questions?

The author is happy to receive comments or questions about this article. He can be contacted at martin@drmartingill.com.au.

Clarification

The examples provided in this article are only intended to give an overview of the approach.

Any implementation should include a comprehensive check of the success (or otherwise) of the file read. This is possible using the various fields contained in the MDE file. These checks have been omitted from this article.

The article assumes only one of the four possible Channel Data Records is required. If the module must process *any* MDE file then it becomes necessary to include all four possible binary formats.

References

Itron's Meter Data Exchange Format (www.fairoakslabs.com/meterdoc/mdef.pdf)

Microsoft Development Network Forums (social.msdn.microsoft.com/Forums/en-US/home)

Storage Assignment in the Visual Basic Environment (msdn.microsoft.com/en-us/library/47zceaw7.aspx)

About Dr Martin Gill

Dr Gill specialises in the provision of advice and data analysis to the energy industry. As a consultant he has prepared advice for government regulators, distributors, retailers, consumers, asset operators and equipment vendors.

Dr Gill has lead teams researching and developing new products across a broad range of industries, including advanced communication modems, burglar alarms, high voltage fault monitors and power quality analysers. One of his teams developed the first in home display and web-portal providing Australian customers the ability to view their electricity use. This innovation was recognised with the Green Globe Award, NSW Government's Premier's Award and Best New Product by the Australian Electrical and Electronics Manufacturers Association.

Version History

Version	Date of publication	Comments
V01	10/9/2015	Original Version