Exporting Data from the Cmpware CMP-DK

Cmpware, Inc.

Introduction

The *Cmpware Configurable Multiprocessor Development Kit (CMP-DK)* is a multiprocessor simulation and software development environment. Its goal is to provide fast and efficient modeling of multiprocessor architectures and to provide support for architectural design and software development of such systems. The goal of supporting software development is achieved by providing a display-rich environment that permits large amnounts of information to be displayed in a fast, simple and uncluttered format. Such capabilities are essential in analyzing the behavior of multiprocessor systems.

While the Cmpware environment is display-rich and provides a large number of useful views of system data, it is impossible to anticipate all of the ways data may be displayed. In addition, often simulation data must be transferred to another system, typically one that is used for further analysis or design documentation and presentation.

To support these needs, the Cmpware CMP-DK provides a facility for exporting simulation data for off-line use. The goal of this export facility was, like other components of the Cmpware CMP-DK, was ease of use and flexibility.

Exporting Data

The approach used in the Cmpware CMP-DK is to provide a single button in the main Cmpware toolbar to toggle data exporting on and off. Pressing the **Export** button () the first time brings up a standard file dialog requesting the name and location of the file to which the data is written. Once a file is specified, data is written to this file after every simulation step. The data which is written is the current value of each of the outport ports of the currently selected processor. In addition, new data is written to the file only if values have changed since the last simulation step. This keeps file sizes managable, while improving simulation performance during data export.

The choice of exporting only output ports was made both for simplicity and for practical reasons. This provides an easy to use and easy to learn interface. A single button

controls the exporting of data and can quickly be turned on and off, with data easily being sent to various files. The alternative of exporting state data from within the processor simulation models would have been complex in most cases, and impossible in others. Given that the simulation models can specify arbitrarily complex and often hidden state, it was decided to export the data that was always clearly exposed: the output ports.

This is less of a restriction than it may seem, since it is a simple matter to add a 'dummy' output port to a processor and have the application, or even the model itself send dat to this port at required.

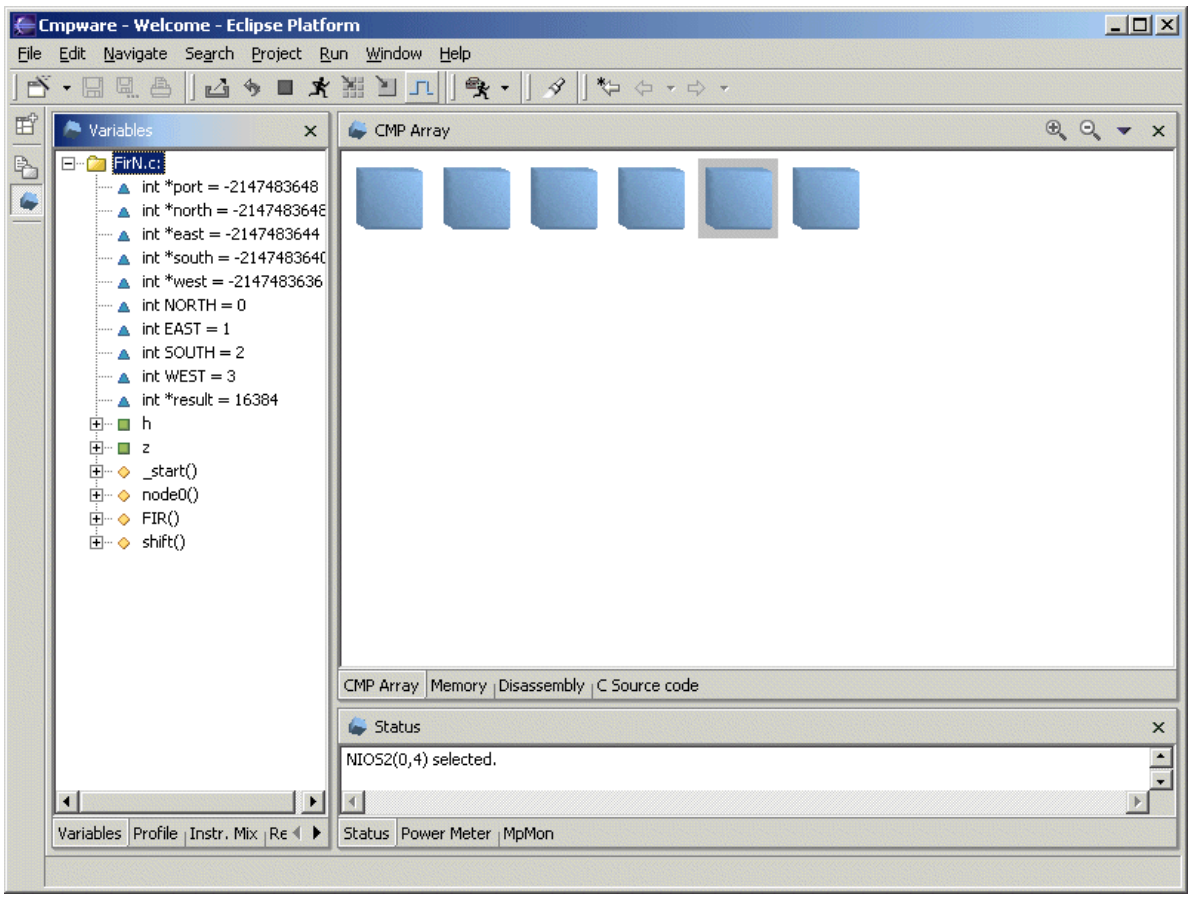

Figure 1: The Cmpware CMP-DK running a 4-processor FIR filter.

Finally, the exported data is writen as a simple comma delimited text file. This allows the data to be easily read and analyzed, as well as providing a fairly universal format for import into other display and analysis systems.

The following sections will give a brief example of data being exported in a Cmpware application, and bthen being imported and graphed in a external application.

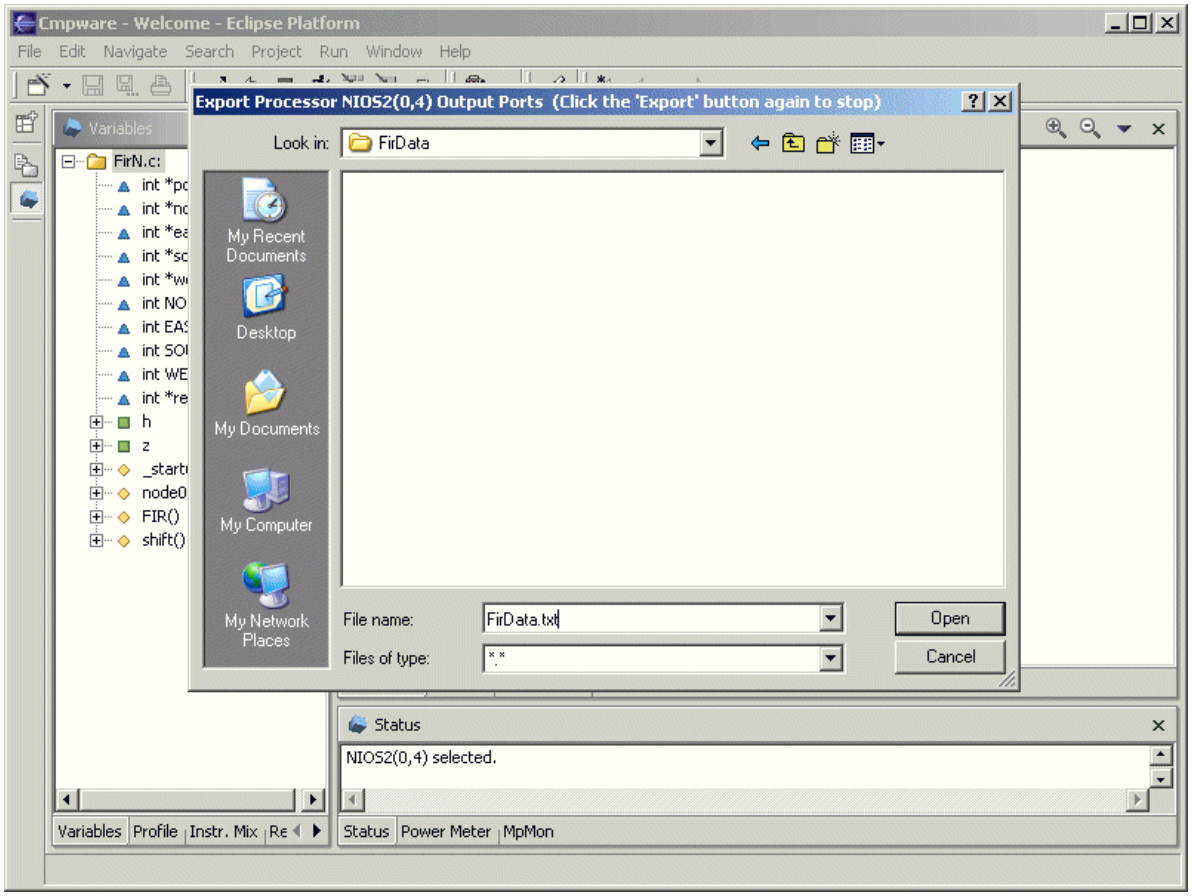

Figure 2: The file selection dialog from the **Export** button.

Example: An FIR Filter

Figure 1 shows the Cmpware CMP-DK set up to run a four-node Finite Impulse Response (FIR) filter. This filter uses six processing nodes, with the first node being used simply to supply data to the filter and the last node being used to store the final results to memory for later analysis. The processor array has already been allocated and the executable code already loaded into the processing nodes. The implementation and set-up for the FIR filter application is beyond the scope of this

document and is not discussed here. See the related documents from Cmpware for more information on implementing and running multiprocessor designs.

Since the fifth node is the last node to perform the FIR filter functions, this node will be outputting the final FIR data over it's link to the last processor. In order to exoprt the final FIR filter result, this fifth processing node should be selected. Figure 1 shows this node highlighted. Once the processing node has been selected, the **Export** button ( ) must be pressed. This brings up the File Selection Dialog box as shown in Figure 2. This is the standard system file dialog. In this case the system is Microsoft Windows XP; other systems will display a similar dialog box.

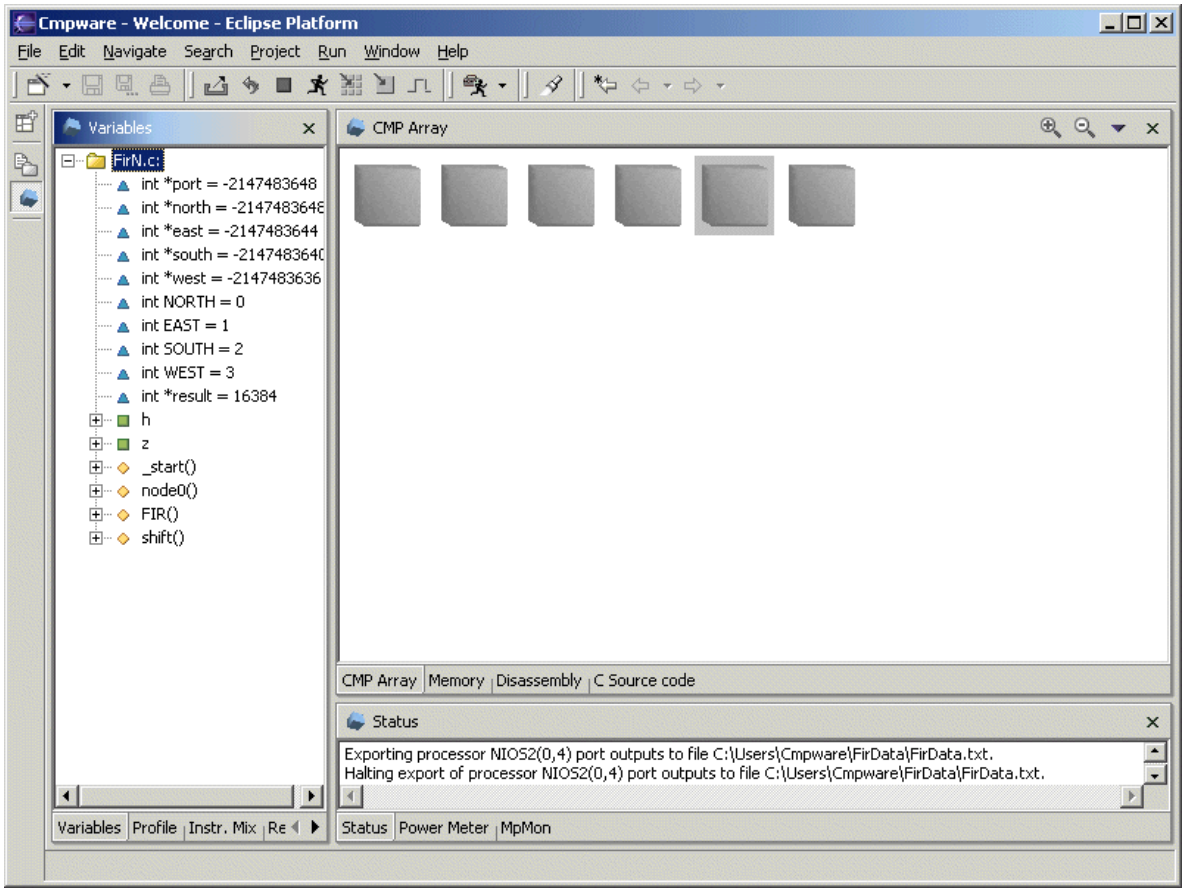

Figure 3: Halting the export with the **Export** button.

Use this File Selection Dialog box to set the name and directory location of the output file to contain the exported data. In this case, a file named *FirData.txt* is created in a directory called *FirData*. If the file is successfully created, the Cmpware Status Window

will display a short message indicating that exporting of data is ready to begin. Now at the end of every simulation step, any changes in the values of the output ports will be written to the FirData.txt file. Now the simulation may be stepped using either the **Step** button ( ) or the **Run** button ( ). With each simulation cycle, the output port data from processor (0, 4) will be updated to the *FirData.txt* file. If the Run button was selected, the simulation will continue, with all of the displays updating, until the **Stop** button ( ) is pressed. Once the simulation is run up to or even past the point where the necessary data has been acquired, the Exporting of data may be stopped. This is done by again pressing the **Export** button ( ). This will display another message in the Cmpware Status window indicating that data exporting has stopped and the FirData.txt file has been closed. This message can be seen in Figure 3.

The Exported Data

The data exported from this procedure is shown in Figure 4. The lines at the top of the file that begin with the double dash (--) comment character contain information describing the data in each column. The first column is the clock cycle, with the following columns containing the addresses of the ouput ports exported. In this example, there are four output ports at addresses: 0x80000000, 0x80000004, 0x80000008 and 0x8000000C. These correspond to the *north*, *south*, *east* and *west* ports in the *Torus* network used by this appilcation. But note that all communication occurs in a horizontal direction, so only the *east* node is of interest.

```
--
-- cycles, 0x80000004, 0x80000008, 0x8000000c, 0x80000000,
--
10, 0,0,0,0,
60, 2,0,0,0,
290, 0,0,0,0,
540, 1,0,0,0,
590, 0,0,0,0,
660, 3,0,0,0,
710, 0,0,0,0,
780, 6,0,0,0,
840, 0,0,0,0,
900, 10,0,0,0,
960, 0,0,0,0,
1030, 15,0,0,0,
1080, 0,0,0,0,
1150, 21,0,0,0,
...
```

Figure 4: Exported data file.

From Figure 4, it is clear that the second column, corresponding to the port address of 0x80000004 is the one of interest. All of the other outputs are always set to zero. A second point of interest is that this application sends pairs of data over this link, in this case corresponding to the FIR *shift* value, which is the original source data being processesd, as well as the filter output for that stage of the filter. While it is possible to graph these values in this interleaved fashion, it may be desitable to separate these pairs of values. This can be done in a variety of ways using various post-processing techniques. Perhaps the simplest is via one of the many scripting languages commonly used for this purpose.

While there are many such alternatives, a small *AWK* script is perhaps the simplest and most widely available. In this case, the *AWK* command line:

```
$ awk '((NR % 2) == 0) {print}' FirData.txt > FirDataEven.txt
```

will give all the even numbered lines and send them to the file *FirDataEven.txt*. In this case, the even numbered lines are the 'shifted' input values. Figure 4 shows that alternate values are the unfiltered inputs, and for the first several cycles, zero. The actual data ramps up from zero to seven, than back down to zero and repeats. This gives a 'sawtooth' input to be filtered. The odd numbered lines can be produced with an *AWK* command similar to the one above. Except that insteading of looking for a modulus (%) equal to zero, a modulus of one is used as below:

```
$ awk '((NR % 2) == 1) {print}' FirData.txt > FirDataOdd.txt
```

This command produces the file *FirDataOdd.txt* which contains all of the filtered output values. Again, note that there are many methods which may be used to massage this data into a more useful format, this is just one simple and commonly available tool. In fact, *AWK* is quite handy at doing other things, including stripping off the unused ports. But for simplicity, this approach has not been used. Instead we will eliminate or ignore these other fields later.

Graphing the Data Using a Spreadsheet

Once the data has been exported to a file and put into the required format, it may be consumed by many types of display and analysis software. As an example, this data will be imported into a spreadsheet and graphed. This graph can then be used in various documentation and presentation applications.

The spreadsheet used in this example is from *OpenOffice* version 1.1. *OpenOffice* is a portable, Java-based set of office software, including word processing, spreadsheet,

and drawing packages, similar to other integrated business tools available for various computing platforms. In this example, the *FirDataOdd.txt* will be first imported into the spreadsheet, then used to produce a graph.

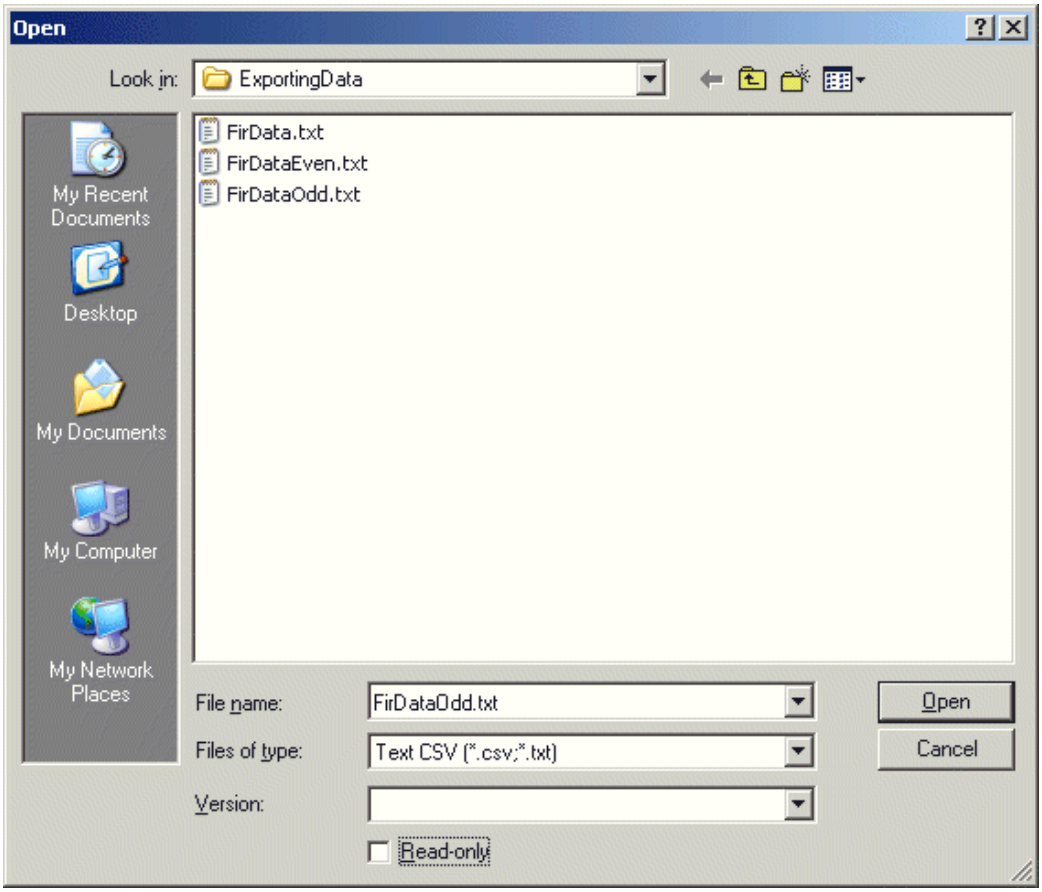

Figure 5: Importing the data file into the OpenOffice spreadsheet.

Using the **File --> Open ...** menu item in the spreadsheet, open the *FirDataOdd.txt* file. In the **File of Type** field, be sure to select the **Text CSV (*.csv, *.txt)** item, as in Figure 5. This tells the Open Office spreadsheet that this these are 'comma separated values' for consumption by the spreadsheet, and not just a standard text file to be displayed.

After this file has been opened, another dialog box will be displayed by the *OpenOffice* spredsheet. This is used to specify the way in which the values are interpreted. In this case, be sure to check the **Comma** box in the **Separated by** area, since the values are separated by commas. It is also useful to change the **From Row** near the top to '3'. This avoids importing the comments at the top of the file. These could also be deleted

from the spreadsheet after importation. Figure 6 shows this dialog box. Note that the **Fields** displayed at the bottom of the dialog box show the data in the expected spreadsheet columns. Pressing the **[Ok]** button imports the data into the spreadsheet as expected. Figure 7 shows the spreadsheet with this data imported.

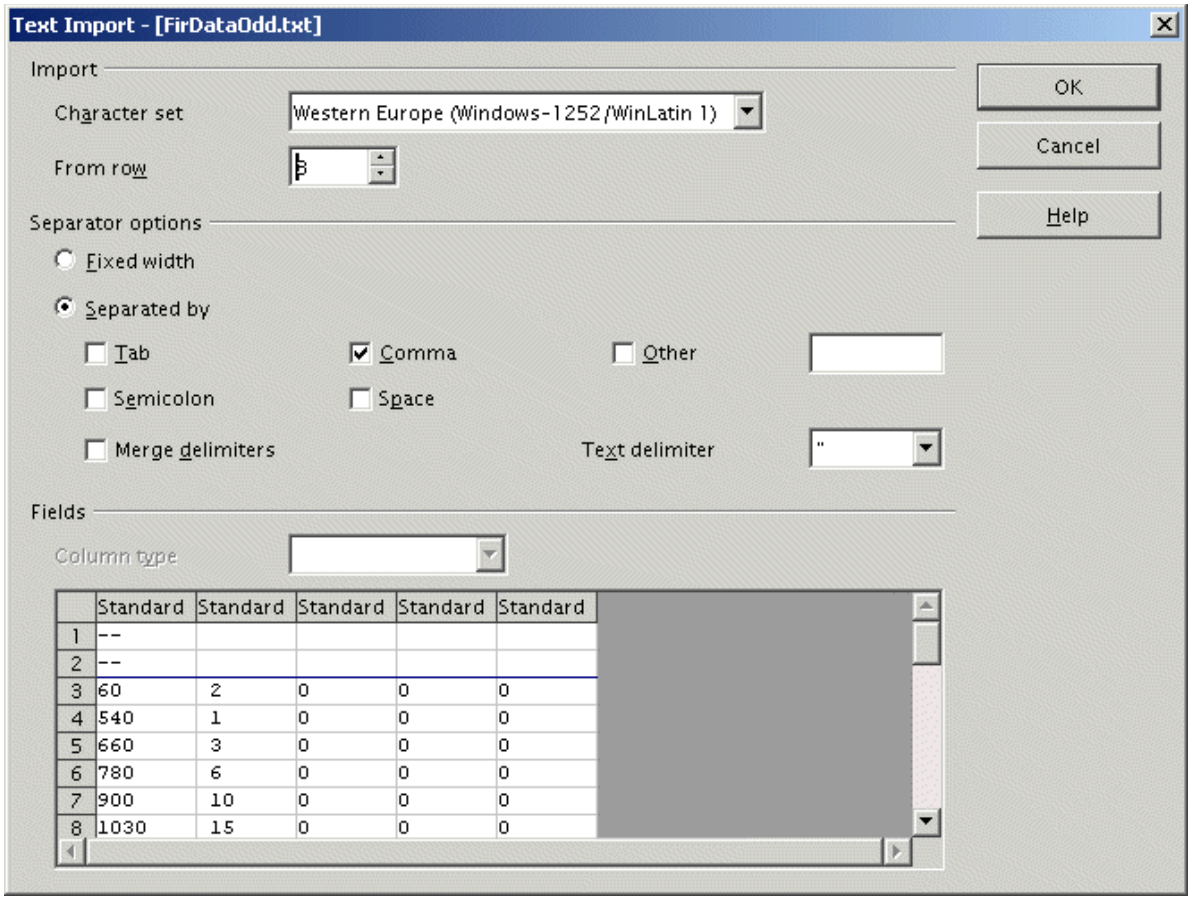

Figure 6: Importing the data file into the OpenOffice spreadsheet.

Because the last three columns do not contain useful data, they may be deleted. This is most easily done by selecting each column by clicking on the column headers ('C', 'D', and 'E', in this case), right clicking the mouse and selecting the **Delete Column** item. This will remove these extra columns. Figure 7 shows this process. Alternatively, these columns could also be ignored in the graphing step. But deleting them requires minimal effort and ensures that this data will not accidentally be used at some point in the future.

At this point all of the data of interest has been imported into the spreadsheet. All that

is required is to generate a graph. In the Open Office spreadsheet, this is accomplished by selecting the spreadsheet cells of interest, in this case columns A and B, and selecting the **Insert --> Chart ...** menu item. This will bring up the dialog box in Figure 8.

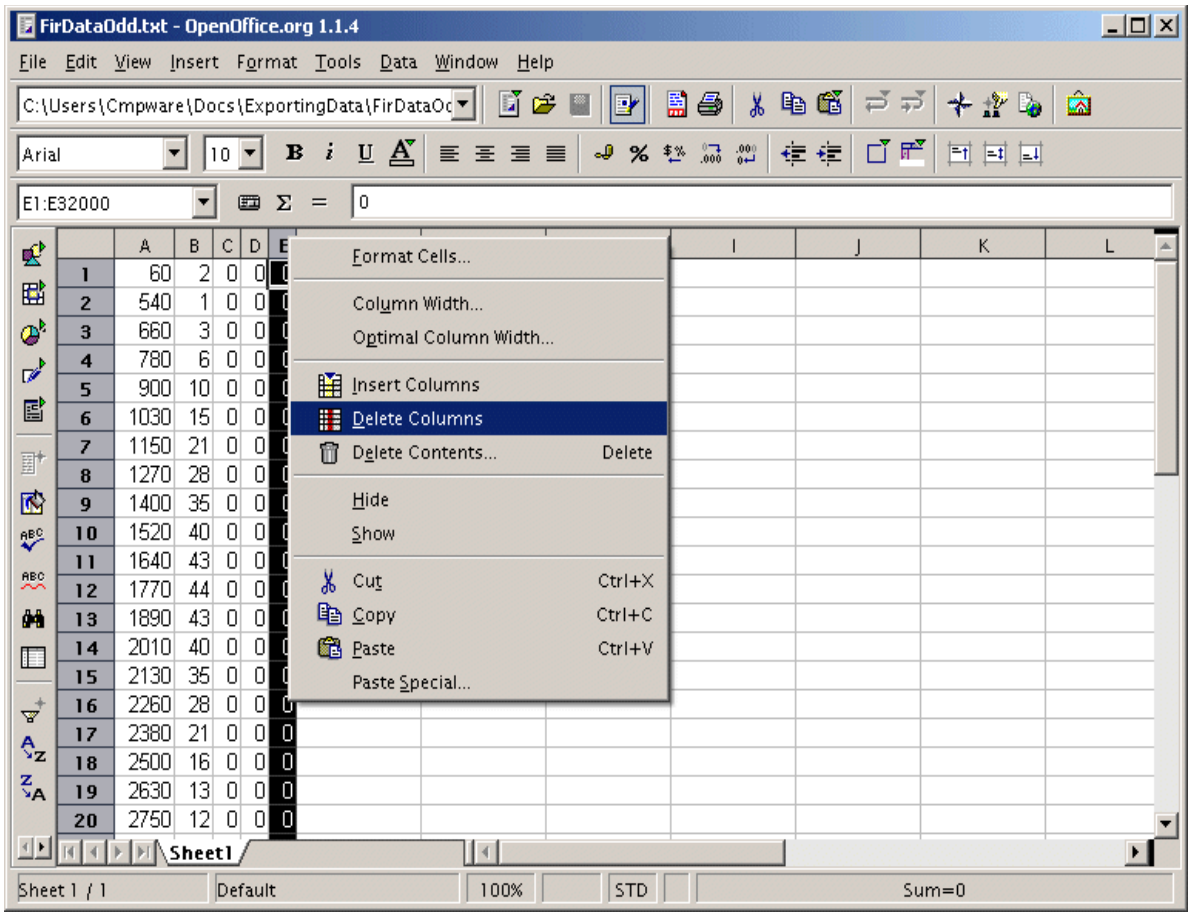

Figure 7: Deleting unused columns of data in the OpenOffice spreadsheet.

As indicated in Figure 8, the spreadsheet cells from A1 to B31 are selected for graphing. This dialog box is the first in a series of dialog boxes that make up a graphing 'wizard'. These will be used to produce the basic graph, which can then be edited through various menu items and clickable objects in the graph itself. Clicking on the **[Next >>]** button moves on to the next dialog box in the wizard. This next dialog box is used to select the basic chart type and is shown in Figure 9.

Figure 9 shows the variety of graph types available. In this example, the the type of

graph required is one that graphs 'XY' values. This indicates the values in column 'A' represent the X axis of the graph and that the values in column 'B' represent the Y axis of the graph. Other graphs types typically graph each value in each successive row as an independent value, with the X axis simply indicating the row number. This is suitable for graphs of data in single independent columns, but not in the situation in this example, where it is desirable to plot the FIR filter output value versus cycles (time).

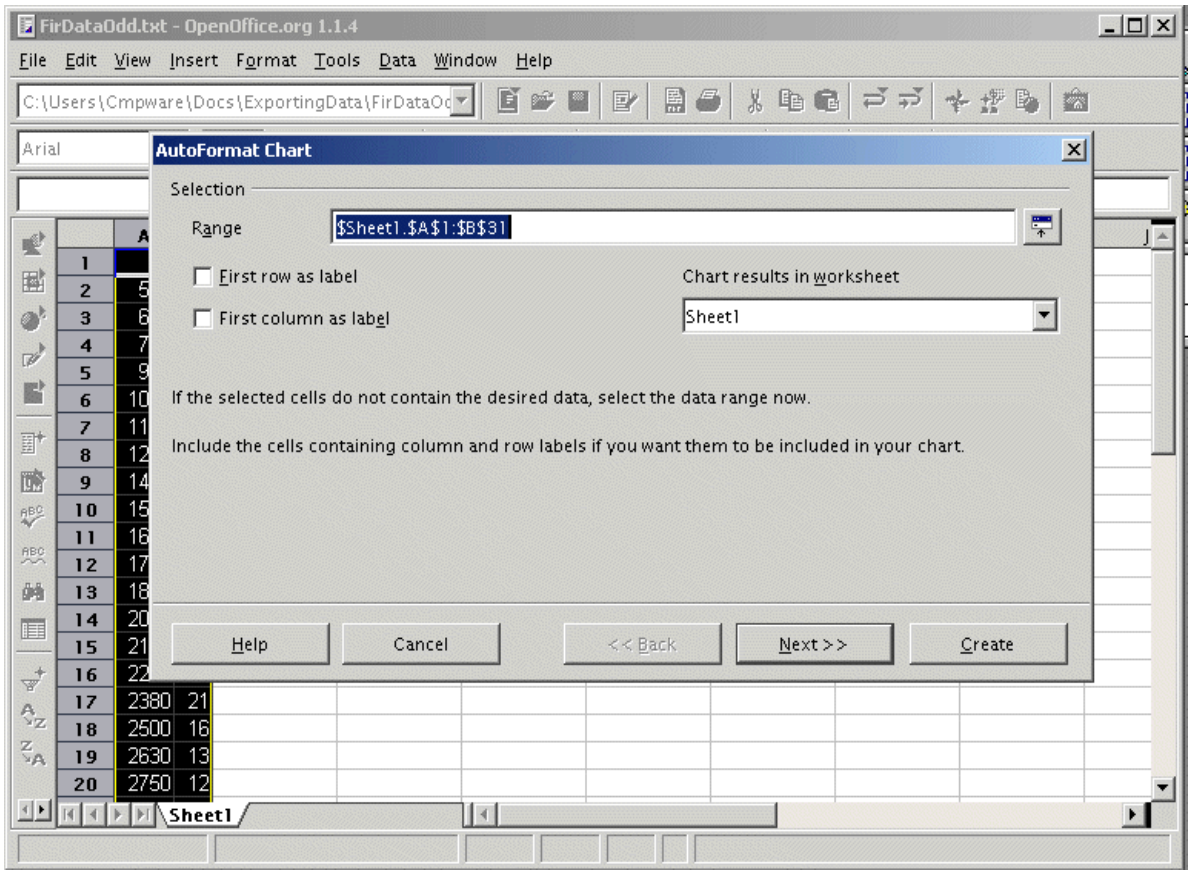

Figure 8: Deleting unused columns of data in the OpenOffice spreadsheet.

Once the **[Next >>]** button is pressed, the dialog box in Figure 10 is brought up by the wizard. This is used to select the presentation of the graphical data in the XY Chart. The data can be shown in the graph as a simple scatter plot of dots, each indicating one data point, or these points may be connected by lines. In addition, the lines connecting the data points may be various types of splines, to provide a smooth curve.

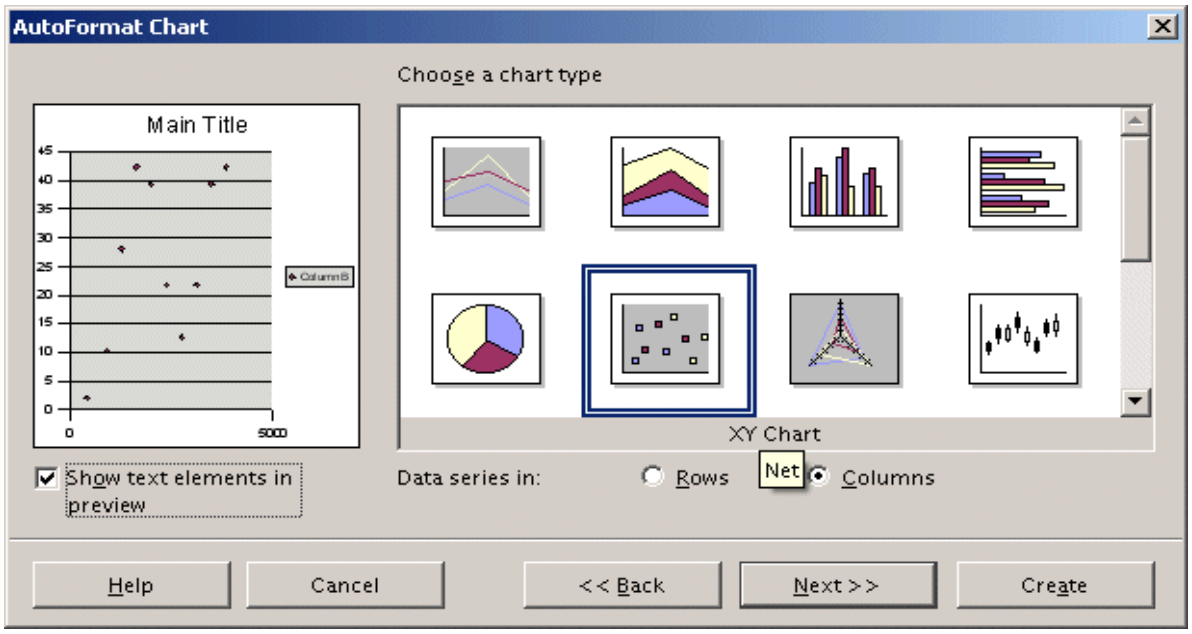

Figure 9: Selecting the *XY Chart* type in the OpenOffice spreadsheet.

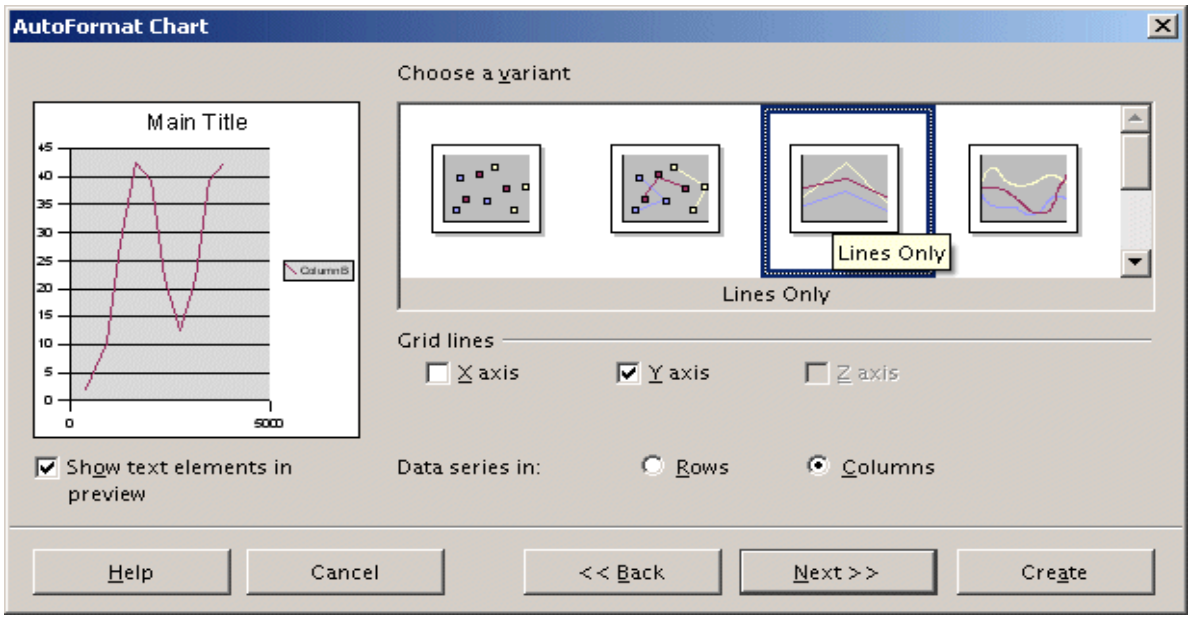

Figure 10: Selecting the *Lines Only* data display in the OpenOffice spreadsheet.

In this example, the *'Lines Only'* choice has been specified. This means that lines connect each point to the next point, and that no smoothing is performed. Additioanlly, the individual data points are not marked, so the graph appears as a continuous line. In this wizard, an optional preview check box gives an indication of how the final graph will appear.

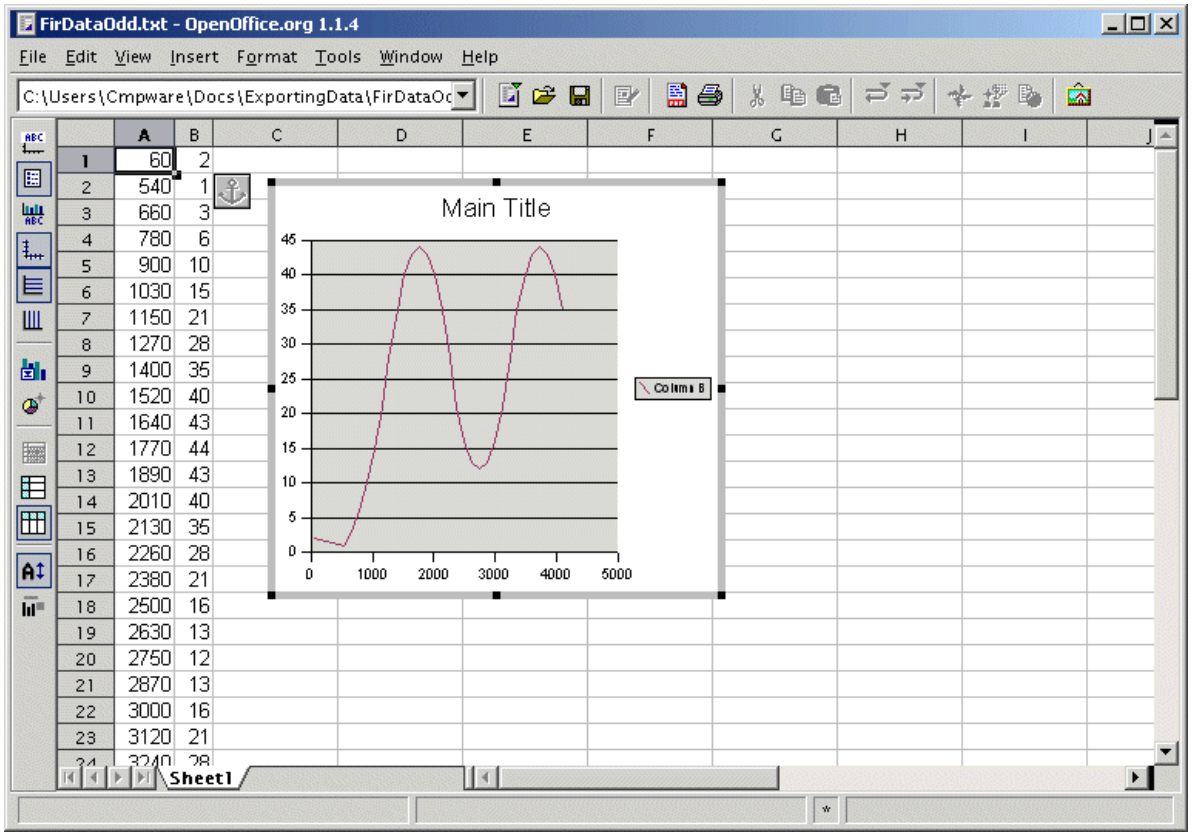

Figure 11: The graph of the exported FIR filter data.

Finally, the **[Create]** button completes the graph. In the *OpenOffice* spreadsheet, the graph appears as a selectable object near the center of the spreadsheet cell array as in Figure 11. At this point, the graph may be edited using various menus and clicking on various components in the graph. Describing this editing process is beyond the scope of this document, but the documentation on the *OpenOffice* spreadsheet can assist in this process.

This graph has been created and edited to put in proper heading and axis labels, as well as changing the graph line color and thickness. This was accomplished in a few

minutes by an inexperienced user of this software. The final graph is now suitable for use in documentation or presentations. In fact, this graph has been cut and pasted into this document using the standard operating system supported cut and paste operation and is shown in Figure 12.

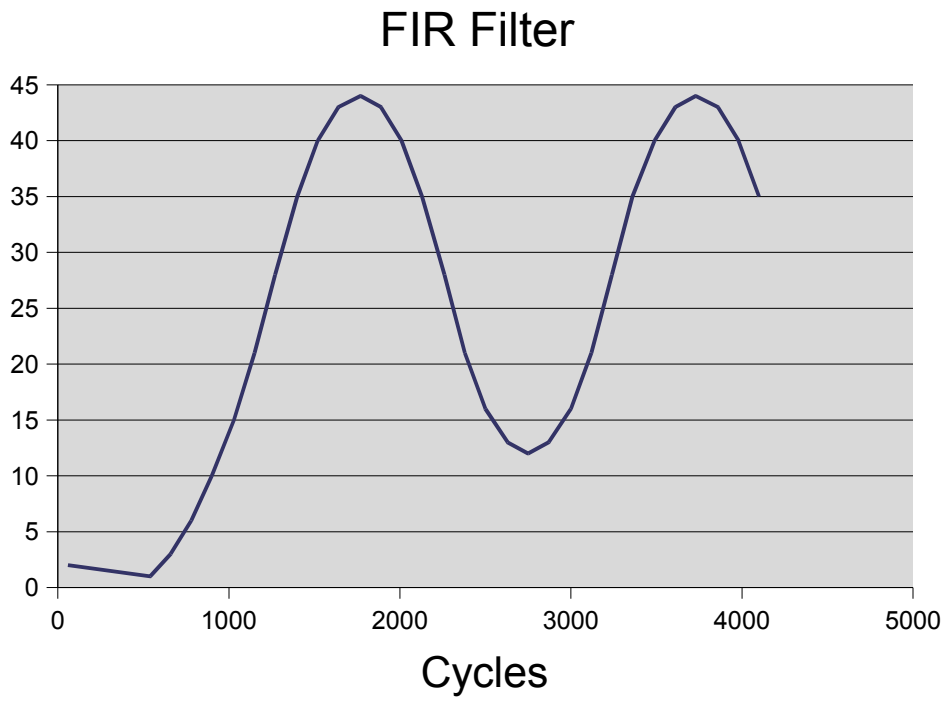

Figure 12: The edited graph, pasted into this document.

Conclusions

The *Cmpware CMP-DK* can be used to quickly capture data being output from a processing node and send that data to a file. This data may be imported into a variety of post-processing environments for use in further anaylsis and display. In this example, data was massaged using a simple *AWK* script, imported into a spreadsheet and graphed. This entire process should take no more than a few minutes, even for users unfamiliar with any part of the system.

The export utility is a simple, but powerful feature of the Cmpware CMP-DK. It provides a generic interface to other downstream tools with a minimal learning curve and effort.

For more information on the commercial version of the *Cmpware CMP-DK* see out web site at:

http://www.cmpware.com/

or send an email to:

info@cmpware.com

Resources

- The *Cmpware CMP-DK:* *http://www.cmpware.com/*
- The OpenOffice suite*:* *http://www.openoffice.org/*
- The Gnu AWK User's Guide*:* *http://www.gnu.org/software/gawk/manual/gawk.html*

Appendix A: FirData.txt (partial)

```
--
-- cycles, 0x80000004, 0x80000008, 0x8000000c, 0x80000000,
--
10, 0,0,0,0,
60, 2,0,0,0,
290, 0,0,0,0,
540, 1,0,0,0,
590, 0,0,0,0,
660, 3,0,0,0,
710, 0,0,0,0,
780, 6,0,0,0,
840, 0,0,0,0,
900, 10,0,0,0,
960, 0,0,0,0,
1030, 15,0,0,0,
1080, 0,0,0,0,
1150, 21,0,0,0,
1210, 0,0,0,0,
1270, 28,0,0,0,
1330, 0,0,0,0,
1400, 35,0,0,0,
1450, 1,0,0,0,
1520, 40,0,0,0,
1570, 2,0,0,0,
1640, 43,0,0,0,
1700, 3,0,0,0,
1770, 44,0,0,0,
1820, 4,0,0,0,
1890, 43,0,0,0,
1940, 5,0,0,0,
2010, 40,0,0,0,
2070, 6,0,0,0,
2130, 35,0,0,0,
2190, 7,0,0,0,
2260, 28,0,0,0,
2310, 7,0,0,0,
2380, 21,0,0,0,
2440, 6,0,0,0,
2500, 16,0,0,0,
2560, 5,0,0,0,
2630, 13,0,0,0,
2680, 4,0,0,0,
...
```