Sharing Data Between the Microwave Design System and OmniSys/CDS

Product Note 85150-6

Introduction

This application note describes the tools and techniques that can be used to share nonlinear simulation data between the HP Microwave Design System (MDS), and the OmniSys Design Suite or the HP Communication Design Suite (OmniSys/CDS). There are several different paths that may be taken to transfer data. The correct path depends on the type of data to be shared and the type of simulator used.

Background

MDS Release 7.0 contains a new type of simulator. Called the Circuit Envelope simulator, this tool efficiently handles complex modulation waveforms in the time domain. To do so, it decomposes signals into a complex I and Q envelope and a carrier signal. Special mathematical algorithms then operate on these signal components to yield a fast, accurate simulation of circuits driven by even the most complicated modulated waveforms.

In addition, MDS includes a state-of-the-art harmonic balance simulator for finding steady-state circuit solutions in the frequency domain.

OmniSys and CDS, Release 5.0 and above, contain a Discrete Time simulator for use with block-level system analyses. Like the Circuit Envelope simulator, the Discrete Time simulator decomposes signals into a complex I and Q envelope and a carrier. The algorithms in the Discrete Time simulator are specialized for use in system-level simulation applications, yielding fast, accurate simulations of complete systems.

In addition to the Discrete Time simulator, CDS also includes Series IV's harmonic balance circuit simulator. In this product, the Discrete Time simulator is capable of automatically invoking the harmonic balance simulator to find solutions for component-level subcircuits. Using power-dependent S-parameter data generated by the harmonic balance simulator, the Discrete Time simulator can include component-level circuits within block-level system simulations.

Designers of RF circuits can run into problems trying to simulate complex circuits, particularly those found in today's wireless communications products. MDS's new Circuit Envelope simulator can analyze these complex designs at the circuit level, but the OmniSys/CDS Discrete Time simulator is needed to analyze the designs at the system level. Furthermore, sometimes it would be convenient to use an MDS harmonic balance simulator output in OmniSys or CDS. The solution to these problems is the basis of this application note; namely, sharing data between MDS and OmniSys or CDS.

Example

Consider the block diagram of a simple communications system, as shown in Figure 1.

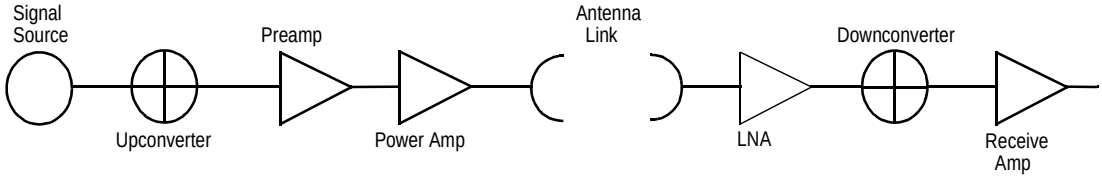


Figure 1. *Block diagram of the analog portion of a simple communications system. Some or all of the components in this block diagram can be simulated by OmniSys/CDS, MDS, or both.*

Typically, a baseband signal is generated, upconverted, amplified, transmitted, and received by a chain of components similar to those in Figure 1. Variations on this basic block diagram can be either simple or extremely complex, but the fundamental structure remains constant.

When a system designer uses OmniSys/CDS to simulate a system like the one in Figure 1, the power amplifier's compression characteristics can be specified using a number of different parameters, such as third-order intercept point and 1-dB compression point. The entire system can be simulated to verify acceptable performance. However, adjacent channel interference is not necessarily correlated with the amplifier's third-order intercept or with its 1-dB compression point because the input signal may be very complicated.

One solution to this problem is to simulate the power amplifier at the circuit level rather than with a black-box approximation. By using a circuit simulator to determine the power amplifier's exact operating characteristics under realistic signal conditions, good overall system performance is assured. Figure 2 shows a flow chart representation of this type of design flow.

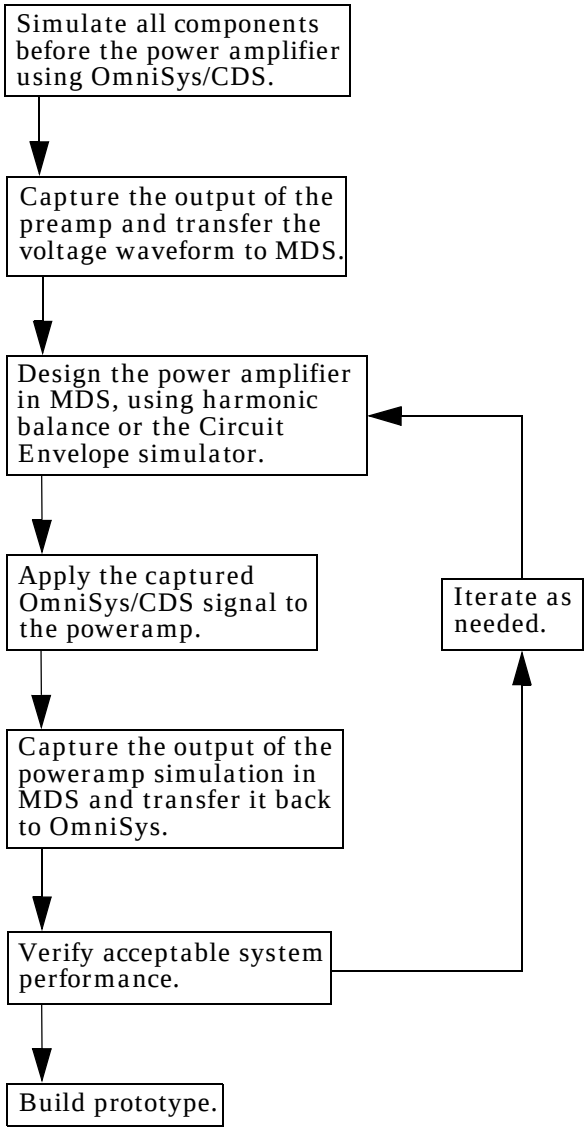


Figure 2. *A design flow using both OmniSys/CDS and the MDS Circuit Envelope simulator.*

With the techniques that are shown in Figure 2 and described in the remainder of this application note, system and circuit designers can:

- Verify subsystem performance.
- Have confidence that the completed system will function properly.
- Avoid subsystem rework and unnecessary design cycles.

The remainder of this application note details the steps necessary to transfer signal waveform data between OmniSys/CDS and MDS.

Using MDS Harmonic Balance Data in OmniSys/CDS

One important capability of the Communications Design Suite is its ability to handle circuit-level simulation data within a system-level simulation. This is accomplished with the use of power-dependent S-parameters. When the system simulator encounters a circuit-level component model, such as an amplifier, it automatically invokes the Series IV harmonic balance simulator. In turn, the harmonic balance simulator calculates a set of power-dependent S-parameters that it returns to the system simulator. System simulation then continues, using the power-dependent S-parameters as a model for the circuit-level component.

With Release 7.0, MDS can also create power-dependent S-parameters and save them in a file. That file can then be used by the Communications Design Suite in a larger system-level simulation.

Using MDS to Create Power-Dependent S-Parameters

Figure 3 shows an amplifier circuit from MDS with a simulation setup that will create a power-dependent S-parameter file.

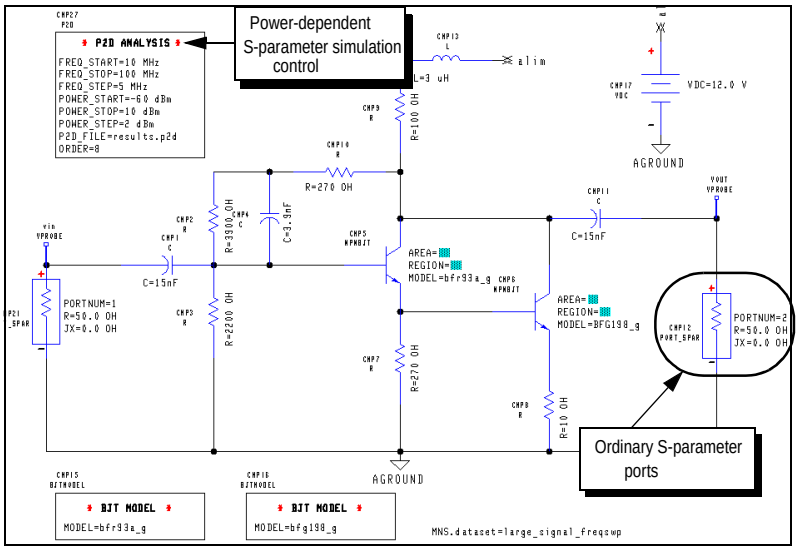


Figure 3. *An amplifier circuit in MDS, with a simulation setup that will create power-dependent S-parameters and place them into a file called results.p2d.*

It is easy to create power-dependent S-parameters in MDS:

- Except for the simulation control block, circuits are entered as if a normal small-signal S-parameter simulation is to be done. S-parameter port impedance components are used at the inputs and outputs. (Note that the default impedance for components in OmniSys is 50 ohms, which matches the default impedance of S-parameter ports in MDS. These impedances can be changed in both systems. If a non-50 ohm simulation is to be performed in OmniSys using data from MDS, then the impedance of the S-parameter ports in MDS should be adjusted to match the impedance used in OmniSys.)
- Instead of an S-parameter simulation control block, a power-dependent S-parameter (**P2D**) simulation block is used. The **P2D** simulation block can be found using the **INSERT/MDS CONTROL/ANALYSIS/S PARAMETER/P2D FILE - START/STOP** menu command. This simulation control block includes entries for a frequency range as well as a power range. Power-dependent S-parameters will be generated that span these ranges, and placed into the specified file.

Note that the final parameter shown on the **P2D** simulation control block in Figure 3 is **ORDER**. This parameter specifies the number of harmonics used in the harmonic balance simulation and is identical to the **ORDER** parameter for any other harmonic balance simulation. The power-dependent S-parameters only include data for the fundamental

frequency because OmniSys does not need data for harmonics. However, the **ORDER** parameter must be set correctly to obtain accurate answers.

When creating power-dependent S-parameters, it is important to specify a wide enough bandwidth and power range to enable OmniSys to accurately simulate signals in the system-level simulation. To accomplish this, set the frequency span so that it contains the entire bandwidth of the signal (including spurious signals) that OmniSys will use. Set the power range so that the highest power level is at least as high as the strongest signal that will be included in the OmniSys simulation, and the lowest power level is small enough so it does not generate significant nonlinearities in the circuit.

Using MDS Power-Dependent S-Parameter Data in OmniSys

Once a power-dependent S-parameter file has been generated with MDS, it is easy to use the data in a system simulation with OmniSys or the Communications Design Suite. In OmniSys, a **P2D** component is inserted, and the name of the data file is entered as a parameter. Figure 4 illustrates.

By default, the **P2D** component assumes that the power-dependent S-parameter data file is located in the **data** subdirectory of the current OmniSys/CDS project directory. This can be overridden by entering a full pathname on the **P2D** component. In general, however, it is more convenient to simply move the data file into the project directory so that everything associated with the OmniSys project stays together.

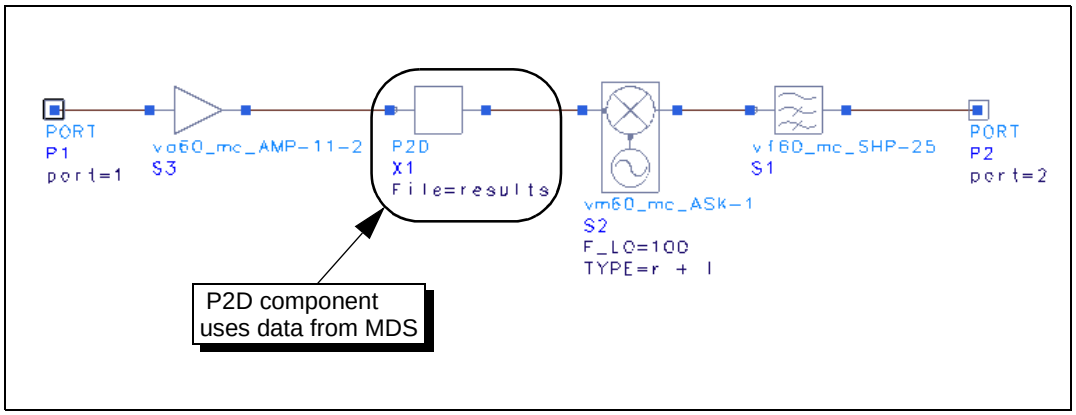


Figure 4. *A simple system schematic in OmniSys/CDS that uses previously calculated power-dependent S-parameters to model an amplifier. The power-dependent S-parameter data could have been generated by either MDS or by a separate OmniSys/CDS simulation.*

Using OmniSys Envelope Data with the MDS Circuit Envelope Simulator

Since the MDS Circuit Envelope Simulator and the OmniSys Discrete Time Simulator both work with complex (I and Q) signal envelopes, it is conceptually easy to transfer data between the two simulators. Because both simulators yield time-domain data using a time step that is determined by the signal envelope rather than the carrier frequency, it is important to transfer only baseband envelope data. If a modulated signal is to be transferred, the correct approach is to capture only the baseband envelope data in OmniSys, transfer it to MDS, and then re-modulate the captured envelope in MDS. OmniSys and MDS can accomplish the de-modulation and re-modulation steps as mathematically perfect operations, so the signal is not degraded.

To transfer this data, there are only two difficulties that must be overcome:

- File format differences.
- Handling imperfect impedance matches.

The solutions to these problems, along with the step-by-step procedure for data transfer, is outlined in this section.

Creating Envelope Data in OmniSys

Signal envelope data transfer between OmniSys and MDS can be accomplished via time-domain data files. In OmniSys, these files are known as TIM files and they have a filename extension of **`.tim`**. TIM files are created using **`OUTTIM`** components, which are found in the **`Output Data Files`** component palette in the OmniSys test bench window.

Figure 5 illustrates the use of **`OUTTIM`** components. In this simple OmniSys test bench, a $\pi/4$ DQPSK signal at 1 GHz drives an amplifier whose output I and Q envelope waveforms are captured by **`IV`** and **`QV`** measurement components. The two **`OUTTIM`** components instruct the simulator to create TIM files containing the signal envelope data.

After simulation, the test bench in Figure 5 creates files called **`i.tim`** and **`q.tim`**. These files will be located in the **`data`** subdirectory within the OmniSys project directory.

Note that the IV and QV measurements are applied to a *modulated* signal in this example, even though only unmodulated, baseband data should be transferred to the Circuit Envelope simulator. This works because the output of the **`IV`** and **`QV`** measurement components only includes baseband envelope data, whether their input signal is modulated or not.

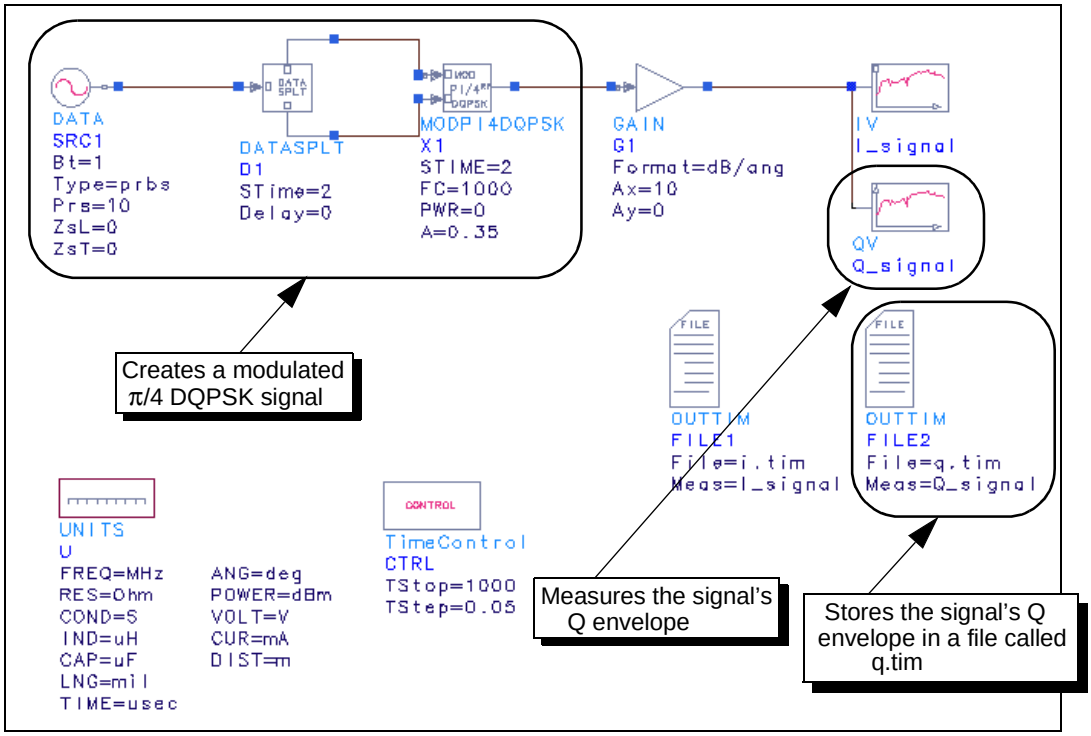


Figure 5. *An OmniSys test bench designed to save signal I and Q envelope waveforms in TIM files.*

Translating TIM Files into CITIfiles

MDS cannot directly read OmniSys TIM files. Instead, the TIM files must be translated into CITIfile format. Appendix A shows a UNIX shell script that can be used to perform the translation. This script is invoked from the UNIX command line by typing

```
tim2citi i.tim q.tim iq.citi
```

Where **`tim2citi`** = the name of the file containing the script from Appendix A.

`i.tim` = the name of the TIM file containing the I envelope data. If necessary, a complete pathname can be specified.

`q.tim` = the name (or complete pathname) of the TIM file containing the Q envelope data.

`iq.citi` = the name (or complete pathname) of a CITIfile that will be created. If a file with this name already exists, it will be overwritten.

The CITIfile format supports the ability to include more than one dependent variable in a single file, so the script in Appendix A combines the data in both TIM files into a single CITIfile. While not strictly necessary, this makes the data somewhat easier to handle and results in the creation of only a single MDS dataset in the next step of the data transfer process.

Reading CITIFILE Data into MDS

Once a CITIfile has been created, it must be read into an MDS dataset. Then the data can be used to define a signal source that can be used in the Circuit Envelope simulator.

To read the CITIfile data into an MDS dataset, open an MDS collector window (a file, workbench, or library icon) and use the **INSERT/DATASET** command from the pop-up menus to create a new (empty) dataset, as illustrated in Figure 6. Place the dataset's icon anywhere in the collector window, and give it the name of your choice. (In the example that follows, the dataset has simply been named **Dataset**.) Then double-click on the dataset icon to open it. The resulting window will display the dataset icon's index page, as shown in Figure 7.

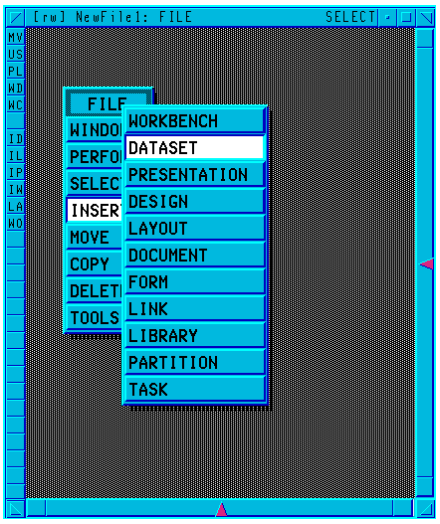


Figure 6. *Use the INSERT/DATASET menu command to create a new, empty dataset in MDS.*

Within the new dataset's index page, use the pop-up menu command **PERFORM/READ/CITIFILE** to read the signal envelope data into MDS. When MDS asks for the name of a CITIfile, enter the name (or complete pathname) of the CITIfile that was previously created by the shell script in Appendix A.

When the CITIfile data has been read into the dataset, close the dataset's window. It is also a good idea to give the dataset a descriptive name.

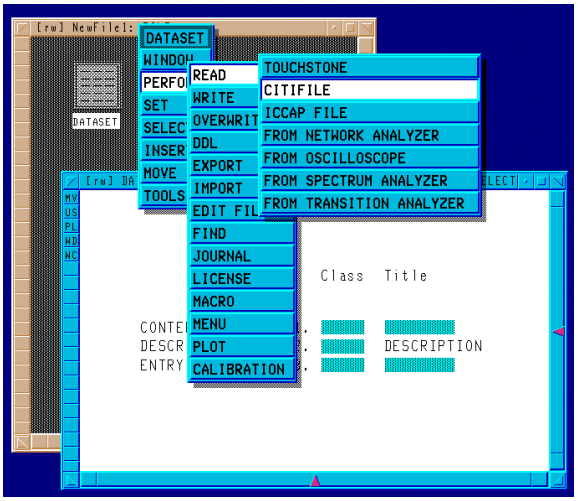


Figure 7. *Use the PERFORM/READ/CITIFILE menu command to read CITIfile data into an MDS dataset.*

Creating a Signal Source Component in MDS

Although the envelope waveform is now contained in a dataset, a small circuit must be constructed that extracts the data and uses it as a voltage in a circuit simulation. This is accomplished through the use of a **VTDATA** component in MDS, as shown in Figure 8. (The **VTDATA** component is accessed with the **INSERT/MDS SOURCES/INDEPENDENT VOLTAGE/TIME DOMAIN WAVEFORM/DATA-BASED** menu item.)

The **VTDATA** source component simultaneously reads the waveform data from the MDS dataset and re-modulates it to a specified carrier frequency. Figure 8 shows how to configure a **VTDATA** variable.

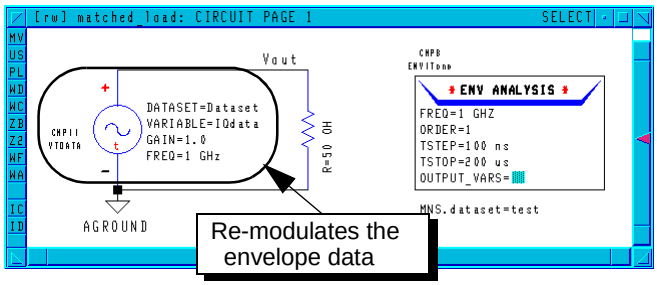


Figure 8. *A single VTDATA voltage source component is used to re-modulate the OmniSys signal envelope data in MDS.*

There are four parameters that must be supplied by the user:

- *The dataset name*. In Figure 8, the dataset has been named **Dataset**.
- *The name of the variable in the dataset*. A CITIfile can hold any number of dependent variables, each with its own unique name. This name is also stored in the MDS dataset. The script in Appendix A gives the name **IQdata** to the contents of the OmniSys TIM files, and Figure 8 shows the use of that name.
- *A gain parameter*, which is normally set to 1 but may be set to a different value to account for impedance mismatches. See the next section for a discussion on handling impedances.
- *A frequency parameter*, which specifies the carrier frequency used for modulating the envelope waveform. To create a baseband (unmodulated) source, set this parameter to zero. The example in Figure 8 uses a carrier frequency of 1 GHz.

The simple circuit shown in Figure 8 can be simulated, and it is useful for verifying that all steps in the data transfer process worked correctly. The simulated data at the **Vout** wire label should exactly match the envelope data that was originally created by OmniSys. More specifically, *real(Vout)* should exactly match the I component of the envelope, and *imag(Vout)* should exactly match the Q component. However, circuits such as power amplifiers are sensitive to the impedances of other circuit components, so these other impedances need to be taken into account.

Note that a similar MDS source component, **VTCDATA**, can be used instead of the **VTDATA** component. The **VTCDATA** component is identical to the **VTDATA** component except that it allows the time axis to be offset and scaled.

Handling Impedances

OmniSys' default behavior is to assume that all components are matched perfectly in a 50-ohm system. For non-50-ohm circuits, lumped-element circuit components and terminations are available. If OmniSys data is to be used in circuit simulations, it is important to account for imperfect impedances.

As a simple example, consider Figure 5. Assume that the simple system shown in the figure models everything except for a final power amplifier. Since the final power amplifier design is extremely nonlinear and critical to the performance of the system, the power amplifier will be modeled using the Circuit Envelope simulator. To do this correctly, we need to know the output impedance of the preamp.

If the output impedance of the preamp is close to 50 ohms and the input impedance of the power amplifier is close to 50 ohms, then it can be treated as a 50-ohm signal source in MDS. Figure 9 illustrates.

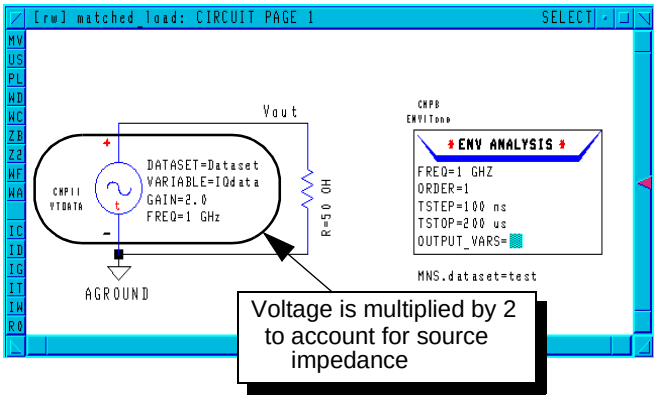


Figure 9. *A single VTDATA voltage source component is used to re-modulate the OmniSys signal envelope data in MDS.*

Figure 9 is similar to Figure 8, except the voltage of the source is multiplied by two, which is necessary to correctly account for the source impedance. The original OmniSys simulation assumed that the output impedance of the preamp was 50 ohms and the input impedance of both measurement

components was 50 ohms. The data that OmniSys stored in the original TIM files, then, corresponds to the voltage at the connector labeled **Vout** in Figure 9. In order for the voltage at the connector to match the OmniSys data, the envelope data at the source must be multiplied by two.

If either the preamp or the power amplifier is a non-50-ohm device, then more care must be taken in the OmniSys simulation, *before* the data is transferred to MDS, to ensure that the Circuit Envelope simulator receives the correct driving voltage. This is accomplished through the use of lumped-element components to model non-50-ohm impedances and the **SPLIT2** element to capture the desired voltage. (**SPLIT2** components are found in the Electrical Miscellaneous Elements component library in OmniSys/CDS.)

Figure 10 shows an OmniSys test bench that models a preamp with a complex output impedance and a power amplifier with a 30 ohm input impedance. In addition, a **SPLIT2** element has been added. One of the outputs of the **SPLIT2** element is used to capture a voltage waveform that will be sent to MDS.

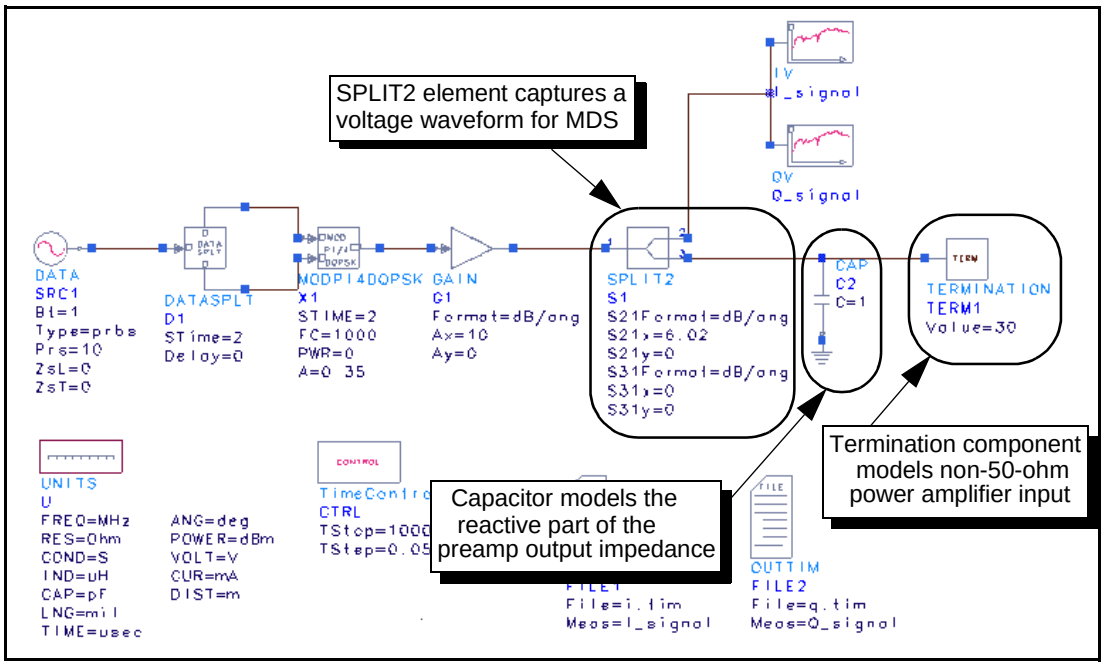


Figure 10. *An OmniSys test bench that could be used to model a preamp with a complex output impedance and a power amplifier with a 30-ohm input impedance.*

The **SPLIT2** element shown in Figure 10 is placed in the circuit *before* the capacitor that models the complex part of the preamp's output impedance. The reason for this can be understood by considering the forward-travelling and reverse-travelling parts of the signal at the output of the preamp. The simple schematic in Figure 11 illustrates.

In Figure 11, the voltage (*V*) is to be measured and transferred to MDS. Consider the forward-travelling voltage waveform that is created by this source. After passing through the 50-ohm resistor, the signal encounters an ideal splitter. The top branch of the splitter, which has a voltage gain of 2, terminates the signal in an ideal 50-ohm impedance (the IV and QV measurement components provide this ideal termination, despite the fact that there are two measurement components). Therefore, no signal is reflected from the upper port of the splitter back into the amplifier. The voltage at the upper splitter port is multiplied by 2 to account for the voltage-dividing action of the two 50-ohm resistors. (Note that the **SPLIT2** component's parameters are specified in terms of power gain in dB,

so the S_{21} parameter is specified as 6.02 dB.) The lower port of the splitter has no voltage gain and transmits the forward-travelling voltage unchanged. The forward-travelling signal is reflected back through the lower splitter port and is "seen" by the preamp component.

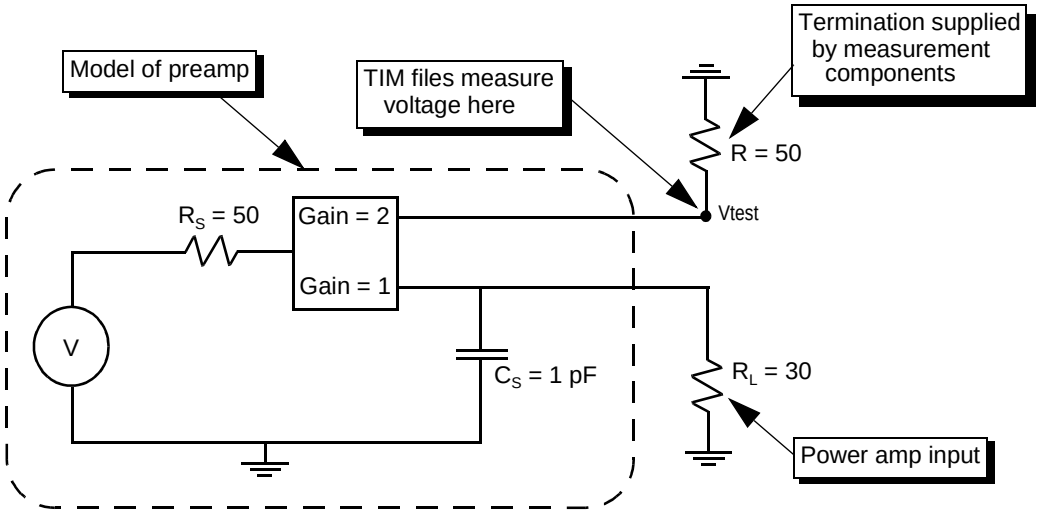
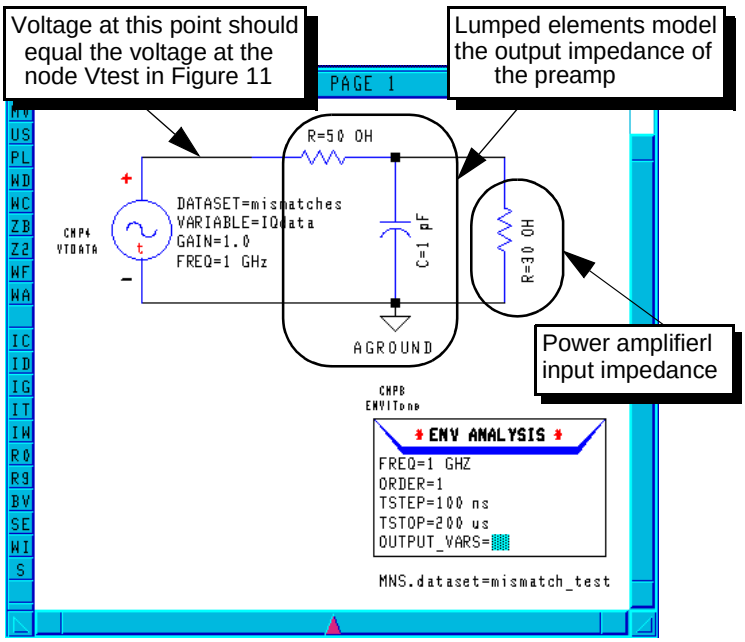


Figure 11. *Equivalent circuit of the output stages of the OmniSys simulation in Figure 10.*

When the data created by simulating Figure 10 is transferred to MDS, a few lumped-element components are used to re-create the non-50-ohm impedances that were modeled in OmniSys/CDS. Figure 12 shows a simple model that re-creates the impedances used in the OmniSys/CDS simulation in Figure 10. In Figure 12, the input impedance of the power amplifier is modeled as a single resistor. For a complete circuit analysis of a power amplifier design, this resistor can be replaced by a complete power amplifier design.

Figure 12. *An MDS circuit that models the output impedance of the OmniSys preamp. In this circuit, the power amplifier's input impedance is modeled as a single resistor. For more advanced simulations this resistor is replaced with the complete power amplifier circuit.*



Using MDS Circuit Envelope Data in OmniSys

Transferring waveform data from the MDS Circuit Envelope simulator into the OmniSys/CDS environment requires the exact opposite of the procedure described in the previous section. Simply put, the steps involved are:

- Translate an MDS dataset into a CITIfile.
- Translate the CITIfile into a pair of TIM files.
- Use the **`.tim`** files as sources in OmniSys/CDS.

Creating Waveform Envelope Data in MDS

Figure 13 shows a simple BJT amplifier in MDS that illustrates a simulation setup that can be used to capture a signal waveform for later use in OmniSys/CDS.

The simulation setup shown in figure 13 creates an MDS dataset named **`DataOut`**. This dataset will contain, among other things, the complex envelope waveform at the point labeled **`Vout`** on the schematic. The data is in real-imaginary (I and Q) format, and covers all specified harmonics of the 1 GHz fundamental frequency.

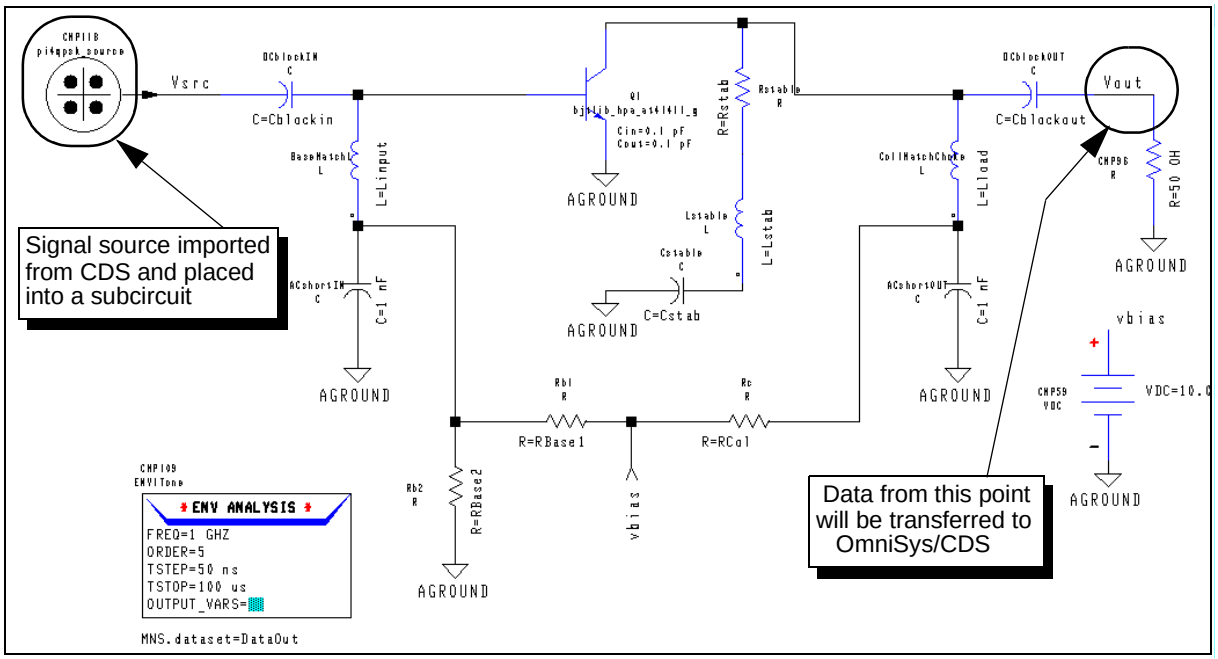


Figure 13. *A BJT amplifier analysis in MDS, using the Circuit Envelope simulator. The signal at the point labeled Vout will be transferred to OmniSys/CDS. The signal source at the upper left is a subcircuit that contains the signal source created in the previous section.*

Translating MDS Datasets into CITIfile Format

Once an MDS dataset has been created, translating it into CITIfile format is simple:

1. Find the dataset's icon, and double-click on it to open it.
2. Normally, the dataset's Index page will be visible. If not, use the menu command **`WINDOW/CHANGE PAGE/INDEX PAGE`**.
3. Use the menu command **`PERFORM/WRITE/CITIFILE`**. The system will ask for a file name. Any file name, including complete pathnames, may be entered.

Translating a CITIfile into TIM Files

Typically, CITIfiles created by MDS contain several variables. For a simple Circuit Envelope simulation like the one shown in Figure 13, the resulting CITIfile will contain two independent variables (time and frequency) and at least as many dependent variables as there are wire labels on the schematic. Finding the correct data within the CITIfile and transferring it into TIM files can be complicated.

The UNIX shell script in Appendix B can be used to find data in a CITIfile and create a pair of TIM files. To use the script, type the following command:

`citi2tim` *CITIfile Data Freq Ifile Qfile*

where **`citi2tim`** = the name of the script in Appendix B.

CITIfile = the name of the CITIfile to translate.

Data = the name of the data (from an MDS wire label).

Freq = the carrier frequency for which you want to extract data.

Ifile = the name (or pathname) of the TIM file to create for the I channel data.

Qfile = the name (or pathname) of the TIM file to create for the Q channel data.

For example, if the CITIfile's name is **`data.citi`**, and name of an MDS wire label is **`Vout`**, and the desired waveform's carrier frequency is 1 GHz, type:

```
citi2tim data.citi Vout 1E9 i.tim q.tim
```

The script will extract the I and Q signal envelope data from the CITIfile at the specified frequency and create two TIM files, which can then be used as source components in OmniSys/CDS.

Note that the script in Appendix B makes a few assumptions about the contents of the CITIfile. In particular, the script only works for straightforward Circuit Envelope simulations like the one shown in Figure 13. This simulation creates two independent variables, time and frequency. *If any other circuit parameter is swept, the script will fail.*

The two TIM files that are created by the script in Appendix B will be used later in an OmniSys/CDS simulation. Because OmniSys/CDS assumes (by default) that TIM files reside in the **`data`** subdirectory of the current project directory, it is most convenient to store the TIM files there.

Using TIM Files in an OmniSys/CDS Simulation

Given a pair of TIM files that contain I and Q waveform data, the signal from MDS can be re-created in OmniSys/CDS through the use of a single component, as illustrated in Figure 14.

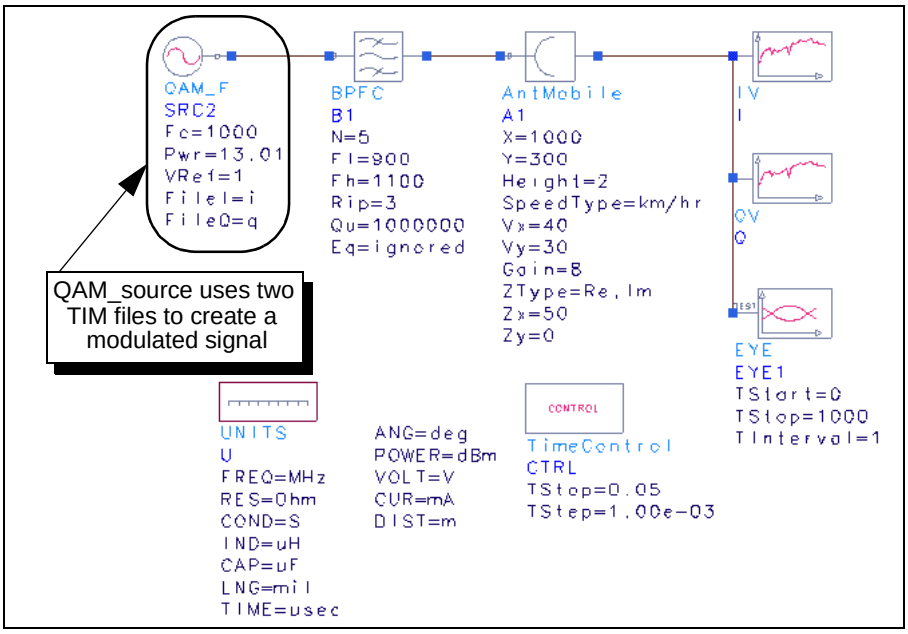


Figure 14. *An OmniSys/CDS test bench that uses a QAM_F source to create a modulated signal from a pair of TIM files.*

In Figure 14, a **QAM_F** component is used to create a modulated signal. (**QAM_F** stands for QAM with input by file, and the component can be found in the Sources and Terminations library in an OmniSys/CDS test bench.) On the **QAM_F** component, the first parameter (**Fc**) is set to equal the carrier frequency that was used in the MDS Circuit Envelope simulation. The second parameter (**Pwr**) should be set so that the carrier amplitude is 1 volt into whatever load impedance is used. The final two parameters (**File1** and **File2**) specify the names of the TIM files that were created using the script in Appendix B.

The **QAM_F** component can be used with several different file formats, and the default format is *not* TIM. To use **QAM_F** components with TIM files, it is necessary to edit the component parameters as follows:

1. Double-click on the **QAM_F** component to edit the parameters.
2. In the dialog box that appear, click on the **FileI** parameter
3. Change the **value type** setting (in the upper right portion of the dialog box) from the default **SPE filename** to **TIM filename**.
4. Repeat steps 2 and 3 for the **FileQ** parameter.

Note that the values of the **FileI** and **FileQ** parameters shown in Figure 14 are **i** and **q**, respectively. Series IV automatically appends a **.tim** extension to these file names, so the actual UNIX file names of these files are **i.tim** and **q.tim**. The default location for these files is the **data** subdirectory of the Series IV project directory.

The output of the **QAM_F** component can then be applied to other OmniSys/CDS components for further overall system simulation and measurements. In illustration, Figure 14 shows the signal being filtered and then transmitted over a mobile antenna link, with several measurements applied at the receiver.

Handling Impedance Mismatches

Because the signal data in this scenario originates in the MDS circuit simulator, impedance mismatch problems are easy to handle. The impedance of an OmniSys/CDS component can be modeled in the MDS simulation, and an MDS wire label can be placed at the correct point in the circuit to capture the waveform at the input to the OmniSys device.

For example, consider the OmniSys mobile antenna model. This component includes parameters for specifying the input impedance of the antenna. For the sake of example, assume that the antenna impedance can be modeled as a series resistor-inductor pair. The MDS model that could be used for this is shown in Figure 15.

By modeling the input impedance of the OmniSys/CDS component in the MDS environment, the captured waveform data can be used in OmniSys/CDS and applied directly to the input of a non-50-ohm component for further system simulation.

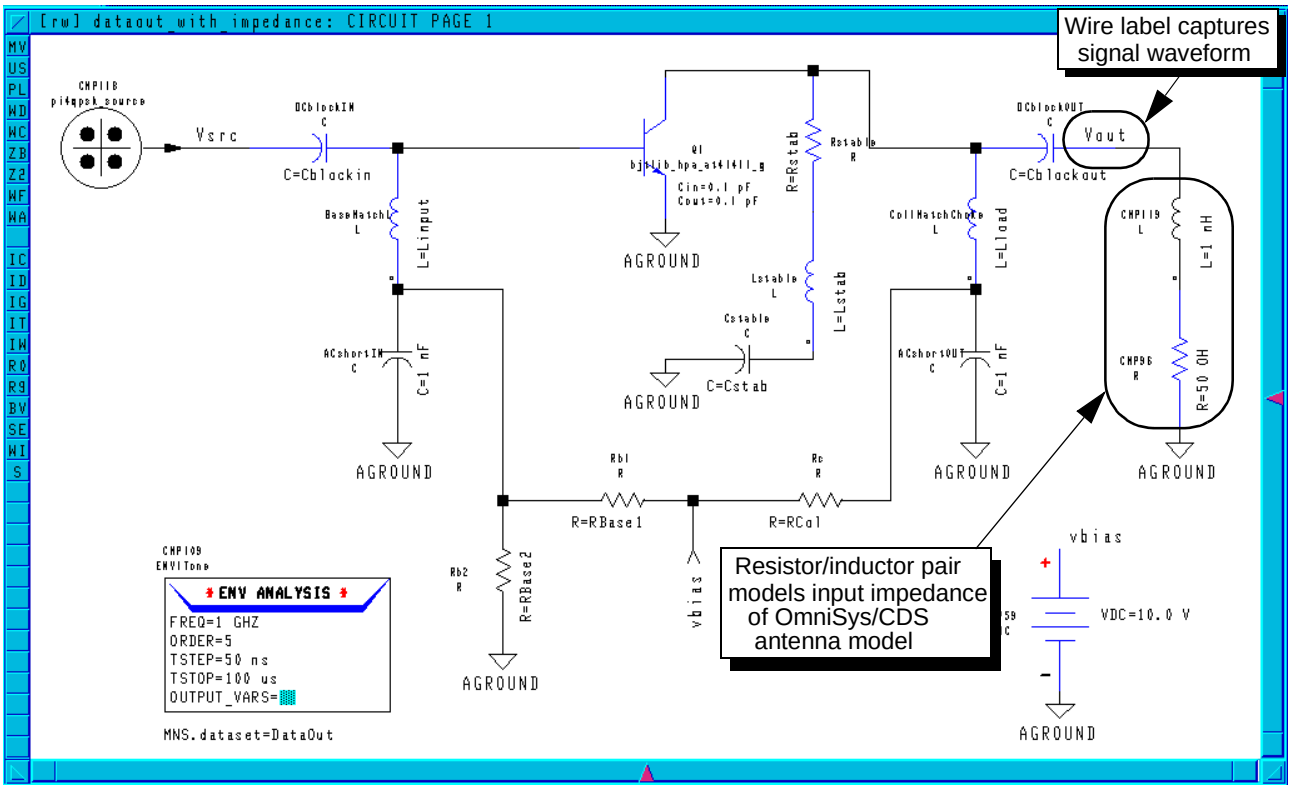


Figure 15. *An MDS simulation that models a non-50-ohm input impedance of an OmniSys/CDS antenna component. The voltage at the Vout wire label can be transferred to OmniSys/CDS and applied to the input of an antenna component for further system-level simulations.*

Appendix A - UNIX Shell Script to Translate TIM Files to CITIfile

This UNIX shell script can be used to translate TIM files from OmniSys/CDS into CITIfile format. This script uses the Korn shell, and uses only common UNIX utilities like **awk** and **grep**. However, it has only been minimally tested on the HP-UX 9.05 operating system. For other operating systems, some modifications may be necessary.

This script is available electronically via anonymous ftp at *hpeesof.external.hp.com* (192.6.21.2) in the **distribution/app_note** directory, or from the HP EEsof worldwide web site (*http://www.hp.com/go/hpeesof*) in the Applications area.

```
#! /bin/ksh

# This script uses a temporary file.  Set the
# file name and path here...
TEMPFILE=/tmp/cititemp

# Check for the correct number of parameters...
if [ "$#" -ne 3 ] ; then
        echo "Usage: $0 Ifile Qfile CITIfile"
        echo "    Ifile = the .tim file containing the I channel data"
        echo "    Qfile = the .tim file containing the Q channel data"
        echo "    CITIfile = the name of the CITIfile to create"
        echo ""
        exit
fi

# Insert CITIFILE header information...
echo "CITIFILE A.01.01" > $3
echo "COMMENT I/Q data from "$1" and "$2 >> $3
echo "NAME I_Q_DATA" >> $3
echo "#MDS DATATYPE TRAN" >> $3
echo "#MDS VARTABLE IVARDATA1" >> $3

# Find out how many data points are in the .TIM files.  This script
# assumes that there are the same number of points in both .TIM
# files, and doesn't do any checking of that assumption.  The
# first 'grep' command filters out any blank lines, and the second
# counts the lines that only contain numbers.
NUMDATA=`grep -v '^[ \t]*$' $1 | grep -c -i -x '[- \t0-9e.]*'`

# more CITIFILE header information...
echo "VAR TIME MAG "$NUMDATA >> $3
echo "DATA IQdata RI" >> $3
echo "" >> $3

# Figure out that the time scale is.  The time scale is given on the second
# line of the .TIM file, and it can be 'sec', 'msec', or 'usec'.  Knowing
# that, we can scale the independent data (time) correctly, because MDS
# always assumes that time data is in seconds.  Once again, this script
# assumes that the I and Q data have the same scale factor, and doesn't
# check the assumption.
# The second line in the .TIM file should begin with "# T(", so the
# following grep command finds it.  Then the awk command picks out
# the scale factor, which is the third field.
TIMESCALE=`grep '^#[ \t]*[tT](' $1 | awk '{print $3}' -`

# Figure out what the voltage scale is.  The voltage scale factor can only
# be volts, millivolts, or "data" (which means that the data has abrupt
# transitions rather than piecewise-linear transistions).  If the
# scale is "data", this script treats it like "volts".
VOLTSCALE=`grep '^#[ \t]*[tT](' $1 | awk '{print $4}' -`

# The independent variable data (time) goes here, followed by the I channel
# data and the Q channel data, in that order.  The grep commands filter
# out any blank lines, and the awk commands filter out the .TIM file
# header and trailer lines.  We use the scale factor, found above,
# to re-scale the time data here.
echo "VAR_LIST_BEGIN" >> $3
case $TIMESCALE in
        psec | PSEC) grep -v '^[ \t]*$' $1 | awk '$1 !~ /[abcdf-zABCDF-Z#%!]/ {OFMT="%.14g"; \
print($1*1E-12)}' - >> $3 ;;
        nsec | NSEC) grep -v '^[ \t]*$' $1 | awk '$1 !~ /[abcdf-zABCDF-Z#%!]/ {OFMT="%.14g"; \
print($1*1E-9)}' - >> $3 ;;
        msec | MSEC) grep -v '^[ \t]*$' $1 | awk '$1 !~ /[abcdf-zABCDF-Z#%!]/ {OFMT="%.14g"; \
```

```
print($1*1E-3)}' - >> $3 ;;
        usec | USEC) grep -v '^[ \t]*$' $1 | awk '$1 !~ /[abcdf-zABCDF-Z#%!]/ {OFMT="%.14g"; \
print($1*1E-6)}' - >> $3 ;;
        *) grep -v '^[ \t]*$' $1 | awk '$1 !~ /[abcdf-zABCDF-Z#%!]/ {OFMT="%.14g"; print $1}' - \
>> $3 ;;
        esac
echo "VAR_LIST_END" >> $3
echo "" >> $3

# Output the data here.  Use the VOLTSCALE scale factor from above
# to properly scale the voltage data.  Here, all of the data goes
# into a single temporary file.  Later, this file will be re-read
# so the data can be formatted correctly as real-imaginary pairs
# in the CITIfile.

# I channel data...
echo "BEGIN" >> $3
case $VOLTSCALE in
        mv | MV) grep -v '^[ \t]*$' $1 | awk '$1 !~ /[abcdf-zABCDF-Z#%!]/ {OFMT="%.14g"; \
print($2*1E-3)}' - >> $TEMPFILE ;;
        *) grep -v '^[ \t]*$' $1 | awk '$1 !~ /[abcdf-zABCDF-Z#%!]/ {print $2}' - >> $TEMPFILE ;;
        esac

# Q channel data...
case $VOLTSCALE in
        mv | MV) grep -v '^[ \t]*$' $2 | awk '$1 !~ /[abcdf-zABCDF-Z#%!]/ {OFMT="%.14g"; \
print($2*1E-3)}' - >> $TEMPFILE ;;
        *) grep -v '^[ \t]*$' $2 | awk '$1 !~ /[abcdf-zABCDF-Z#%!]/ {print $2}' - >> $TEMPFILE ;;
        esac

# Read the temporary file all at once, and output the data
# into the CITIfile with the correct formatting...
awk 'BEGIN {i = 0;} {data[i++] = $1;} END {i /= 2; for (j = 0; j < i; j++) \
print(data[j],",",data[i+j]);}' $TEMPFILE >> $3
echo "END" >> $3

# Clean up...
rm $TEMPFILE
```

Appendix B - UNIX Shell Script to Translate CITIfiles to TIM Files

This UNIX shell script can be used to translate CITIfiles into OmniSys/CDS TIM files. This script uses the Korn shell, and uses only common UNIX utilities like **awk** and **grep**. However, it has only been minimally tested on the HP-UX 9.05 operating system. For other operating systems, some modifications may be necessary. Note that this script only works with the results of the Circuit Envelope simulator, and it will not work if any MDS variable (other than time) is swept.

This script is available electronically via anonymous ftp at *hpeesof.external.hp.com* (192.6.21.2) in the **distribution/app_note** directory, or from the HP EEsof worldwide web site (*http://www.hp.com/go/hpeesof*) in the Applications area.

```
#! /bin/ksh

# Check for the correct number of parameters...
if [ "$#" -ne 5 ] ; then
        echo "Usage: $0 CITIfile Data Freq Ifile Qfile"
        echo "   CITIfile = name of the CITIfile to translate"
        echo "   Data = name of the data (from an MDS wire label)"
        echo "   Freq = carrier frequency for which you want to extract data"
        echo "   Ifile = name (or pathname) of the .tim file to create for the I channel data"
        echo "   Qfile = name (or pathname) of the .tim file to create for the Q channel data"
        echo ""
        exit
fi

# It can be hard to find the data you want in the CITIfile, because a CITIfile can contain
# many different variables, both dependent and independent.  Independent variables are defined
# by VAR statements, and dependent variables are defined by DATA statements.  We have to find
the
# relative locations of the DATA and VAR statements and remember them in order to sort out the
data
# we want.  This part of the script finds values for TIME_LOCATION and FREQ_LOCATION, which are
# integer values indicating the location of the data in the CITIfile.  For instance, if "time"
# is the first independent variable in the file, then TIME_LOCATION will be 0.
#
# Note that this script assumes that there are only two independent variables: time and
# frequency.  This will always be true of a Circuit Envelope simulation unless the user
# sets up a simulation with another swept parameter.  If that happens, this script
# will not work.
LOWEST_VAR_LOCATION=‘grep -n -i "^VAR[ \t]*" $1 | awk -F: ’{print $1;exit}’‘
TIME_LOCATION=‘grep -n -i "^VAR[ \t]*time[ \t]*MAG" $1 | awk -F: ’{print $1;exit}’‘
FREQ_LOCATION=‘grep -n -i "^VAR[ \t]*freq[ \t]*MAG" $1 | awk -F: ’{print $1;exit}’‘
let TIME_LOCATION=TIME_LOCATION-LOWEST_VAR_LOCATION
let FREQ_LOCATION=FREQ_LOCATION-LOWEST_VAR_LOCATION
if [ $TIME_LOCATION -ne 0 ] ; then
        echo "Sorry, I can’t handle this CITIfile."
        echo "Either there are too many swept variables, or"
        echo "the data is not in the order I expect."
        exit
fi
if [ $FREQ_LOCATION -ne 1 ] ; then
        FREQ_LOCATION=0
fi

# Find the number of points in the time data, and the number
# of frequency points...
TIME_POINTS=‘grep -i "^VAR[ \t]*time[ \t]*MAG" $1 | awk ’{print $4;exit}’‘
FREQ_POINTS=‘grep -i "^VAR[ \t]*freq[ \t]*MAG" $1 | awk ’{print $4;exit}’‘

# Find the location of the data we want.  As with the independent variables, there can be
# many dependent variables in a CITIfile.  Here, DATA_LOCATION is an integer
# like TIME_LOCATION and FREQ_LOCATION.
LOWEST_LOCATION=‘grep -n -i "^DATA" $1 | awk -F: ’{print $1;exit}’‘
DATA_LOCATION=‘grep -n -i -x "^DATA[ \t]*$2[ \t]*RI" $1 | awk -F: ’{print $1;exit}’‘

# Now, find the index of the frequency
# that we want.  The frequency list is stored in
# the CITIfile as a single list.  This
# ’awk’ command finds the frequency data, and
# then finds the desired frequency within
# that list.  FREQ_INDEX becomes an integer
# that is the index of the desired frequency
```

```
# within the list.
FREQ_INDEX=‘awk ’BEGIN {e = 1e9; i = 0; found = 0; best = 0;} \
        /^VAR_LIST_BEGIN/, /^VAR_LIST_END/ { \
            if ($1 == "VAR_LIST_BEGIN") found++; \
            else if ((found == 2) && ($1 != "VAR_LIST_END")) { \
                if ($1 >= desired) { \
                    if ($1 - desired < e) {best = i; e = $1 - desired;} \
                    } \
                else if (desired - $1 < e) {best = i; e = desired - $1;} \
                i++; \
                } \
            } \
        END {print best}’ \
        desired=$3 $1‘

# Make sure that the data names, as entered on the command line,
# are correct and the data actually does exist in the CITIfile.
# Note that the data type has to be "RI" (for real/imaginary),
# or this script will not work.
if [ ${#DATA_LOCATION} -eq "" ] ; then
        echo "Sorry, but there is no data called "$2" in the CITIfile."
        echo "(Or if it does exist, it’s the wrong kind of data.)"
        exit

fi
let DATA_LOCATION=DATA_LOCATION-LOWEST_LOCATION

# Get, and output, the I channel data.  Since the CITIfile contains the independent
# data and dependent data in a single column of text, the awk command below reads all
# the data into a single (large) array, and then prints it back out in two columns.
# The values that are calculated above are used here to index into the
# data array and find the subset of the data that we are looking for.
echo "BEGIN TIMEDATA" > $4
echo ’# T( SEC V R 50.0 )’ >> $4
echo ’% T V’ >> $4
awk ’BEGIN {OFMT = "%.14g"; i = 0;} \
        /^VAR_LIST_BEGIN/, /^VAR_LIST_END/ {if (i <= time_points) time[i++] = $1;} \
        /^BEGIN/, /^END/ {data[j++] = $1;} \
        END {data_points = 2 + (time_points * freq_points); \
        offset = freq_index + 1 + (i_offset * data_points); \
        for (i = 0; i < time_points; i++) print time[i+1],",",data[(i*freq_points)+offset];}’ \
        i_offset=$DATA_LOCATION time_points=$TIME_POINTS \
        freq_points=$FREQ_POINTS freq_index=$FREQ_INDEX \
        $1 | awk -F, ’{print $1,"   ",$2}’ >> $4
echo "END" >> $4

# Output the Q channel data...
echo "BEGIN TIMEDATA" > $5
echo ’# T( SEC V R 50.0 )’ >> $5
echo ’% T V’ >> $5
awk ’BEGIN {OFMT = "%.14g"; i = 0;} \
        /^VAR_LIST_BEGIN/, /^VAR_LIST_END/ {if (i <= time_points) time[i++] = $1;} \
        /^BEGIN/, /^END/ {data[j++] = $1;} \
        END {data_points = 2 + (time_points * freq_points); \
        offset = freq_index + 1 + (i_offset * data_points); \
        for (i = 0; i < time_points; i++) print time[i+1],",",data[(i*freq_points)+offset];}’ \
        i_offset=$DATA_LOCATION time_points=$TIME_POINTS \
        freq_points=$FREQ_POINTS freq_index=$FREQ_INDEX \
        $1 | awk -F, ’{print $1,"   ",$3}’ >> $5
echo "END" >> $5
```

HEWLETT® PACKARD

For more information, contact a regional HP office listed below, or check your telephone directory for a local HP sales office.

United States
(800) 452 4844

Canada
(905) 206 4725

Europe (Amsterdam)
Hewlett-Packard
European Marketing Centre
P.O. Box 999
1180 AZ Amstelveen
The Netherlands

Japan
(81) 426 48 0722

Latin America (Miami, Florida)
(305) 267 4245

Australia/New Zealand
(13) 1347 Ext. 2902

Asia Pacific (Hong Kong)
(8522) 599 7070

Data subject to change

Printed in USA PN 85150-6 5M 4/96

5965-1211E