

Data Analysis and Mining With DataSpy and IPA (Integrated Post Analysis)

ICS TechDay 2019

Presented by John Mitchell

1



April 30, 2019



INTREPID
CONTROL SYSTEMS
www.intrepidcs.com

Data Analysis and Mining with DataSpy and IPA

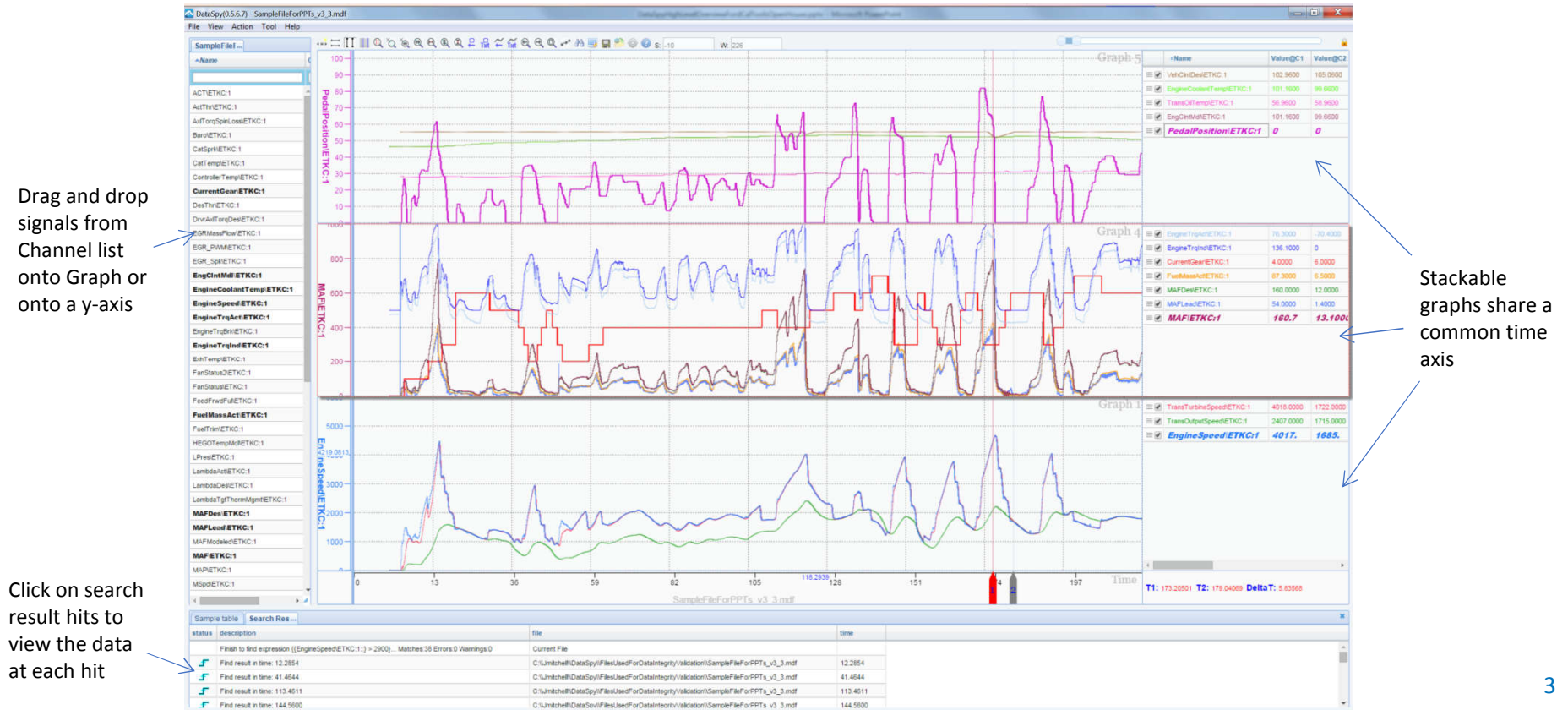
Topics Covered in Presentation:

- Review of PC Version of DataSpy
- Review of DataSpy running on Wireless NeoVI
- Definition of Integrated Post Analysis (IPA)
- Review architecture of IPA and operating modes
- Demo of FindInFiles and HistogramGenerator scripts on PC
- Review script configuration files for these scripts
- Review code in scripts
- Review main functions of IPA library
- Demo of FindInFiles and HistogramGenerator scripts on Wivi

DataSpy – Data plotting tool that can run on PC or Web

Powerful plotting tool that provides accurate graphical and numerical display of large mdx data files.

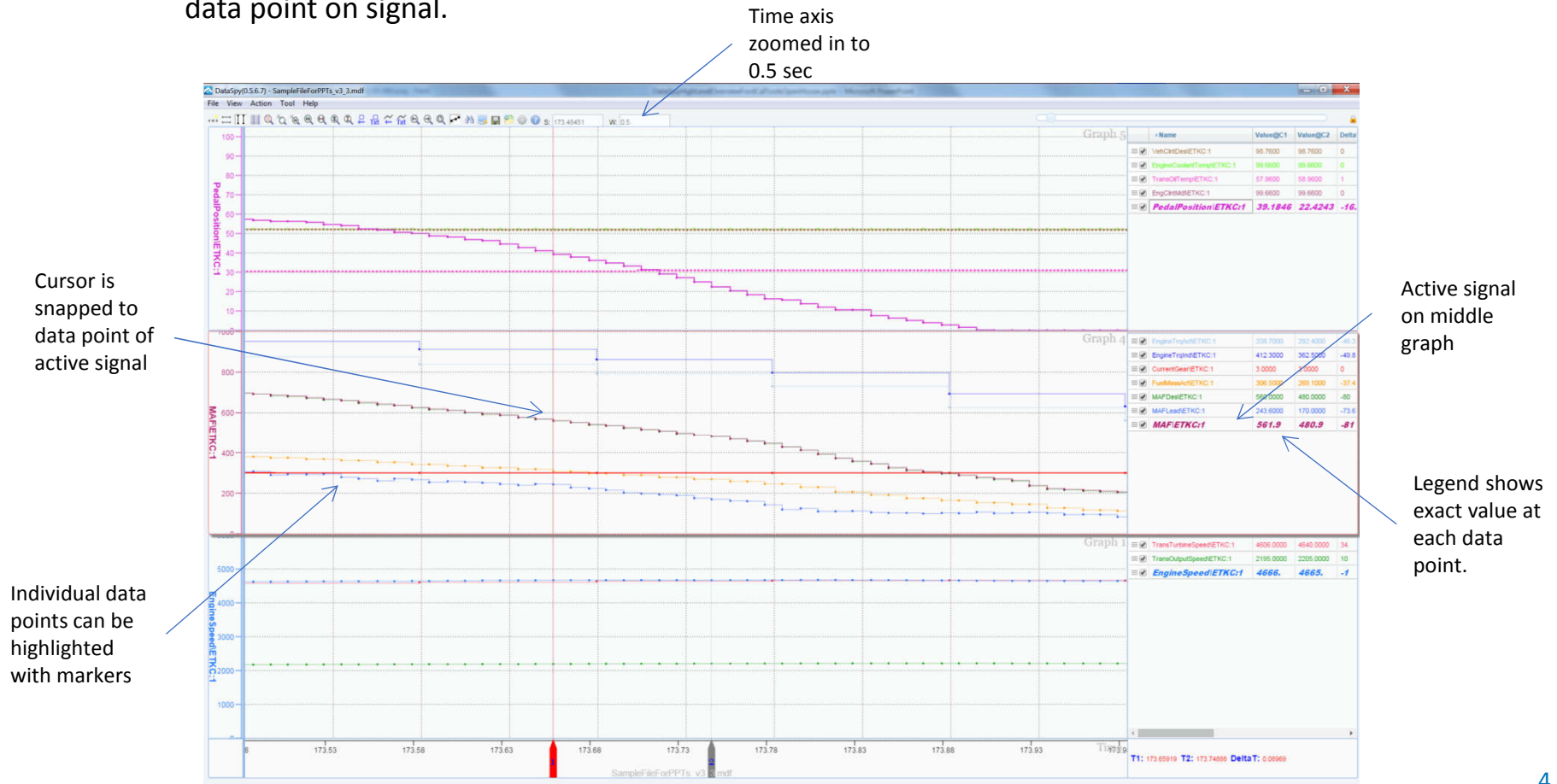
- Easily handle files that are several Gb in size.
- Verified to provide accurate results using a range of data files logged using ICS products as well as those from ETAS.
- Create custom views of key signals and save configurations for use with other data files.



DataSpy – Data plotting tool that can run on PC or Web

Users can zoom way out on large data files to view signal ranges or can zoom way in to view individual data points.

- User can select signal in legend to make it the active signal, then use arrow keys to step through each data point on signal.



DataSpy – Data plotting tool that can run on PC or Web

Search for complex expressions in data file using Advanced Search feature.

- Provide a boolean expression of signals and then click on Find All
- Example below looks for WOT pedal launches from rest
- Expression parser can also be used to create calculated signals

Find in files

Search In Chart Advanced Search

Find What:

`((693.VehicleSpeed@FILE1) < 1) && ((512.ActualPedalPos@FILE1) > 90)`

Find All Save

Look in: Current File

☒ Current File ☐ Current Directory ☐ Selected Directories

ID Directory

Sub-Folders Add Delete

Look at file type: MDF Files

Expression Editor Existed Expression

| ID | Name |
|-----|--------------------------|
| | Actual |
| 86 | ACTUAL_SPK_CYL2 |
| 312 | RrAxleCouplingTrq_Actual |
| 314 | Ft_RrCouplingTrq_Actual |
| 424 | ActualGear |
| 512 | ActualPedalPos |
| 853 | ActualGearForDisplay |

sin asin log abs
cos acos log10 sqrt
exp ^ && ||
+ - * /
< > <= >=
== != ()

Enter expression and click on Find All to execute search

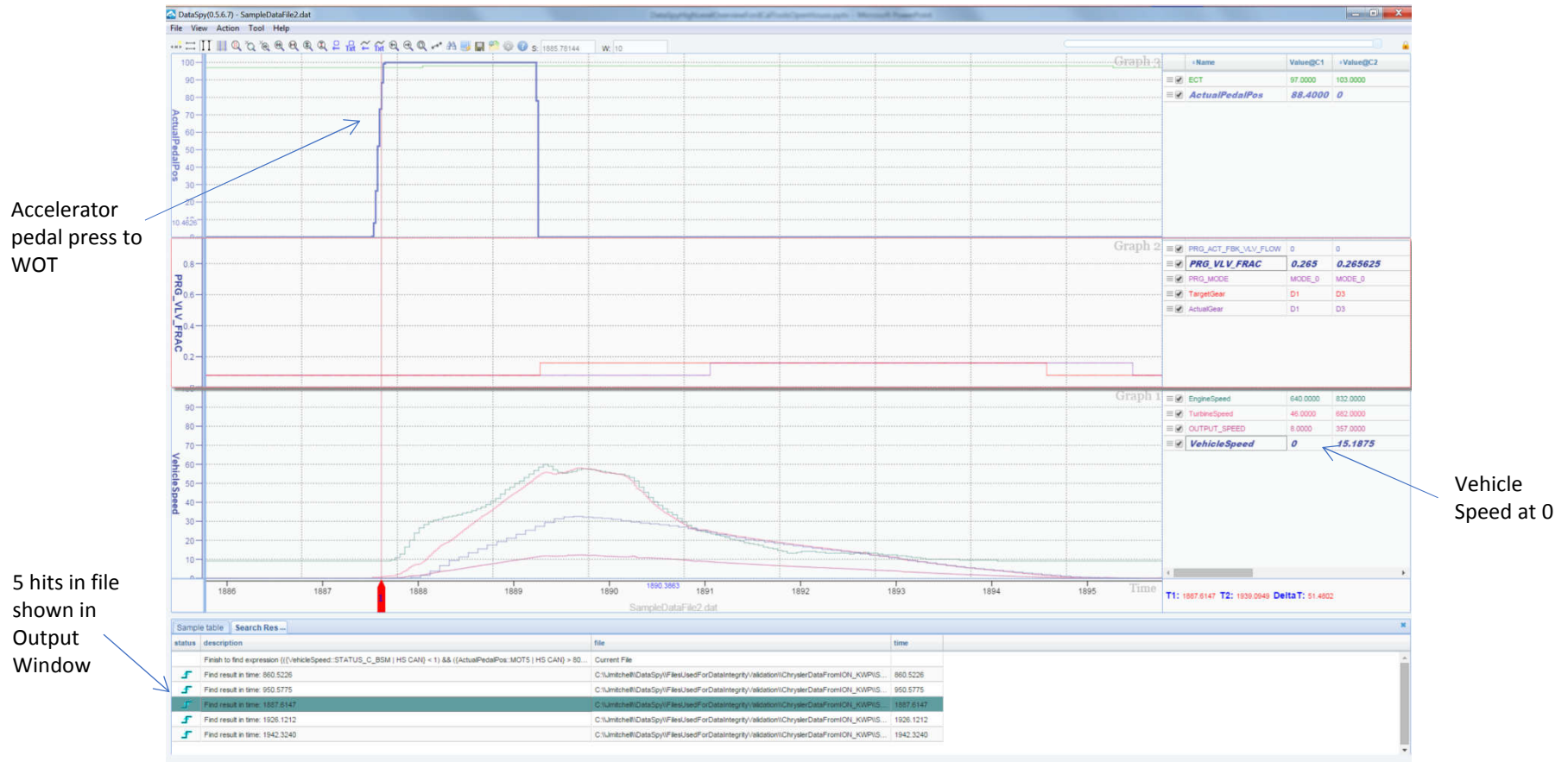
Search expressions can be built using channel list or can be pasted into "Find What" text box

Boolean operators currently supported in expression parser

DataSpy – Data plotting tool that can run on PC or Web

Search results shown in output window below graphical view.

- User can click on each hit to jump to that point in data file.
- Tool places cursor 1 at the point the expression toggles true



DataSpy – Web Version Launches in Browser from Wivi

Some basic integration with Wireless NeoVI currently available

- Allows users to view data files directly from the server by clicking on link
- No need to install any software, works in any browser as long as user is logged into Wivi
- User authentication controls allow administrators to control which users can see which data

The screenshot shows the Wivi DataSpy web interface. At the top, there's a navigation bar with 'Dashboard', 'Devices', 'Organizations', 'Server', and a user profile 'John Mitchell'. Below this is a breadcrumb trail: 'Dashboard / ICS TEST / Vehicle 400815 (2016 neo VI) / Data'. The main section is titled 'Search Filters' and includes a 'Date Range' section with 'Start Date' and 'End Date' pickers, a 'Name' search box, and a 'Collections' list with items like 'Chrysler200EntireBus_', 'ChryslerOneShotData_', 'Chrysler200C_BusQuery', and 'Chrysler200_AliBusAndKWP2K_'. There are also 'File Types' (CSV, DAT, DB, MAT, VSB, XML) and a 'Search' button. Below the filters, a status bar shows 'no data processing, no data pending' and '0 files, 0 B selected'. The main table has columns for 'Start', 'End', 'Name', 'View', and 'Data Type'. It lists several data entries, including 'DemoDataAllBus' and 'Chrysler200C_KWP2K_ColdStartParams03-14-17_v1'. A tooltip is visible over the 'View' column, showing options like 'Multiple Data Views', 'View DAT in DataSpy', and 'DB Viewer'. A blue arrow points from the text 'Clicking on DataSpy link will show data file in DataSpy viewer' to the 'View DAT in DataSpy' option in the tooltip.

| Start | End | Name | View | Data Type |
|---------------------|---------------------|---|---------|---|
| 2017-04-16 22:59:53 | 2017-04-16 23:38:45 | DemoDataAllBus | [Icons] | DAT 118.65 MB, DB 278.42 MB, VSB 63.15 MB |
| 2017-04-16 21:24:27 | 2017-04-16 21:24:50 | DemoDataAllBus | [Icons] | DAT 111.21 MB, DB 261.04 MB, VSB 59.44 MB |
| 2017-04-16 19:05:45 | 2017-04-16 19:42:10 | DemoDataAllBus | [Icons] | DAT 111.21 MB, DB 261.04 MB, VSB 59.44 MB |
| 2017-04-16 00:52:34 | 2017-04-16 00:52:48 | DemoDataAllBus | [Icons] | DAT 111.21 MB, DB 261.04 MB, VSB 59.44 MB |
| 2017-04-15 23:52:03 | 2017-04-15 23:57:08 | DemoDataAllBus | [Icons] | DAT 111.21 MB, DB 261.04 MB, VSB 59.44 MB |
| 2017-04-14 19:54:57 | 2017-04-14 19:55:01 | DemoDataAllBus | [Icons] | DAT 111.21 MB, DB 261.04 MB, VSB 59.44 MB |
| 2017-03-15 12:46:34 | 2017-03-15 12:48:48 | Chrysler200C_KWP2K_ColdStartParams03-14-17_v1 | [Icons] | DAT 2.34 KB, DB 44 KB, VSB 311 B |
| 2017-03-14 21:27:44 | 2017-03-14 21:58:43 | Chrysler200C_KWP2K_ColdStartParams03-14-17_v1 | [Icons] | DAT 2.39 MB, DB 28.91 MB, VSB 7.9 MB |
| 2017-03-14 21:27:41 | 2017-03-14 21:27:41 | Chrysler200C_KWP2K_ColdStartParams03-14-17_v1 | [Icons] | DAT 345 B, DB 28 KB, VSB 325 B |
| 2017-03-14 20:06:57 | 2017-03-14 20:07:55 | Chrysler200C_KWP2K_ColdStartParams03-13-17_v2 | [Icons] | DAT 345 B, DB 28 KB, VSB 342 B |
| 2017-03-14 19:11:54 | 2017-03-14 19:20:01 | Chrysler200C_KWP2K_ColdStartParams03-13-17_v2 | [Icons] | DAT 345 B, DB 28 KB, VSB 336 B |
| 2017-03-14 16:52:09 | 2017-03-14 16:53:27 | Chrysler200C_KWP2K_ColdStartParams03-13-17_v2 | [Icons] | DAT 345 B, DB 28 KB, VSB 336 B |

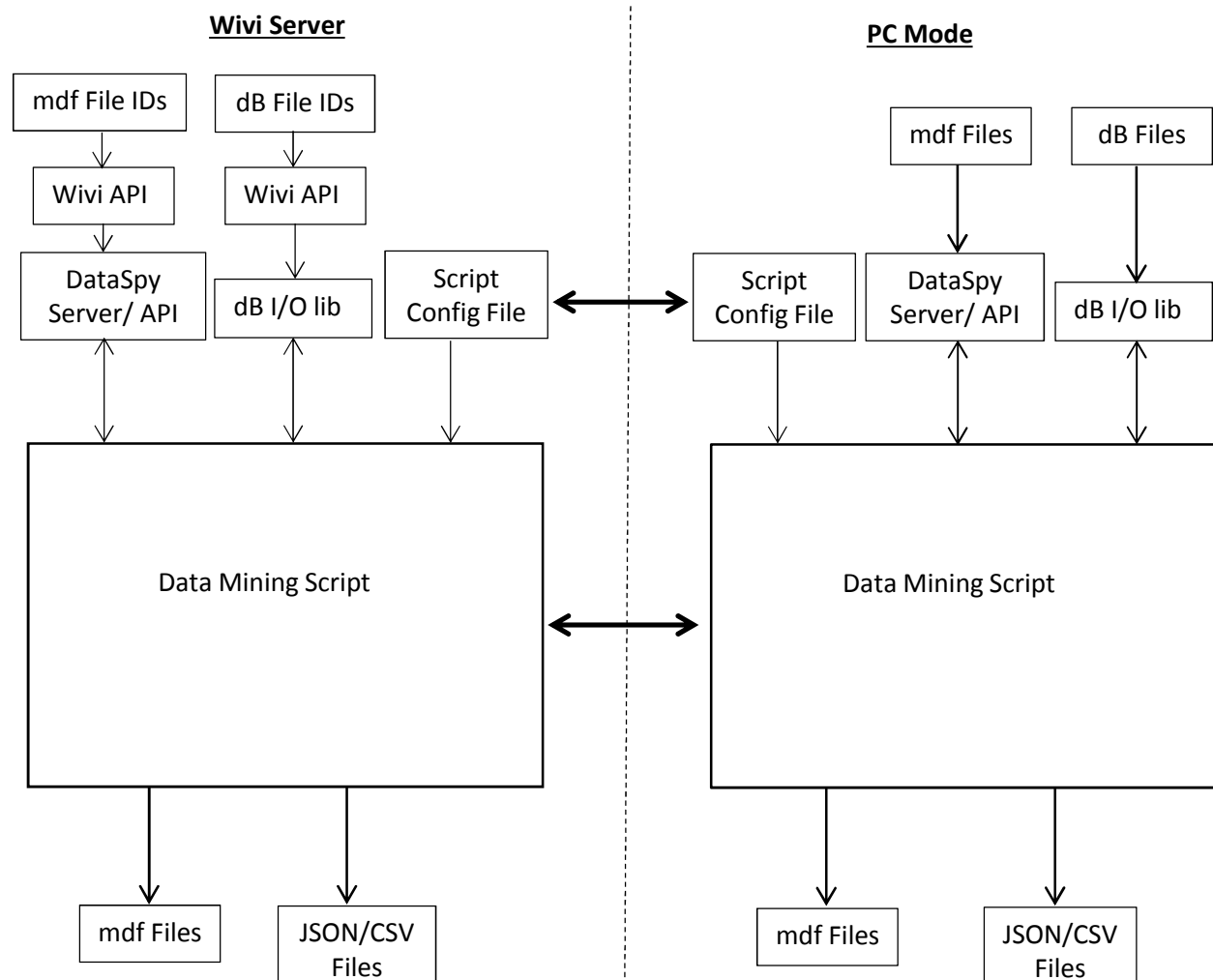
Clicking on DataSpy link will show data file in DataSpy viewer

IPA – Data mining tool that can run on PC or Web

Product Definition:

- Library of mdf and db file i/o functions allowing users to develop data mining scripts to generate reports based on time based vehicle network data.
- Scripts can be used to mine through data on users PC and can be uploaded to our WirelessNeoVI server and have the server run the scripts on files directly on the server.
- Library of functions can be used with any language in PC mode. We have example scripts in Excel VBA, MATLAB, C#, and Python.
- Python scripts can be uploaded and executed by WirelessNeoVI.
- Other cross platform languages like R may be supported in the future.

Integrated Post Analysis – Product Architecture



Parallel scripting language set up so that users can develop scripts and script configurations on their PC and upload them to the Wivi server so that they can be run by the server. Example applications include:

- Generate reports based on data automatically on server without downloading any raw data.
- Generate complex histograms (ie: absorbed clutch power per shift type as a function of clutch volume) from user defined data sets
- Pull out chunks of data from multiple mdf files and append to a single file.

Integrated Post Analysis – Defining File Set

File Sets are named collections of files used to define what files to run script on.

- Use filters in WirelessNeoVI Data Archive then select files by clicking on file icon or you can select all file matching filter criteria
- After selecting desired list of file click on dropdown and header and select Add Data to File Set

Use filters in DataArchive as first step in file selection

Each row represents a data file.

The screenshot shows the WirelessNeoVI Data Archive web interface. The top navigation bar includes links for Dashboard, Data Archive, Reports, Devices, Organizations, and Server. The main content area is titled 'Data Archive' and contains a 'Search Criteria' section. This section has a 'Fleets/Vehicles' filter with a dropdown menu showing 'Vehicle: 42M768'. There are also 'Date Range' filters for 'Start Date', 'End Date', 'Start Time', and 'End Time'. Below these are 'Collection or File Name' and 'File Types' (CSV, DB) filters. A 'Search' button is at the bottom right of the search criteria section. The results section shows a table with columns: Start, End, Vehicle, and Name. The table contains 15 rows of data, each representing a data file. A dropdown menu is open over the table, showing options: 'or 4 files (244.52 MB) selected', 'Select All', 'CSV', 'DB', 'Create Download', 'Add Data To File Set', 'Export Selections', 'Clear Selections', and 'Delete Selections'. The 'Add Data To File Set' option is highlighted.

| Start | End | Vehicle | Name |
|---------------------|---------------------|---------|-----------------------------------|
| 2018-04-12 16:10:32 | 2018-04-12 16:10:32 | 42M768 | DataSpySampleDataFileAllSignals1 |
| 2018-06-23 16:14:39 | 2018-06-23 16:32:58 | 42M768 | DataSpySampleDataFileAllSignals17 |
| 2018-06-23 16:14:39 | 2018-06-23 16:32:58 | 42M768 | DataSpySampleDataFileAllSignals7 |
| 2018-06-23 16:14:39 | 2018-06-23 16:32:58 | 42M768 | DataSpySampleDataFileAllSignals13 |
| 2018-06-23 16:14:39 | 2018-06-23 16:32:58 | 42M768 | DataSpySampleDataFileAllSignals3 |
| 2018-06-23 16:14:39 | 2018-06-23 16:32:58 | 42M768 | DataSpySampleDataFileAllSignals18 |
| 2018-06-23 16:14:39 | 2018-06-23 16:32:58 | 42M768 | DataSpySampleDataFileAllSignals8 |
| 2018-06-23 16:14:39 | 2018-06-23 16:32:58 | 42M768 | DataSpySampleDataFileAllSignals14 |
| 2018-06-23 16:14:39 | 2018-06-23 16:32:58 | 42M768 | DataSpySampleDataFileAllSignals4 |
| 2018-06-23 16:14:39 | 2018-06-23 16:32:58 | 42M768 | DataSpySampleDataFileAllSignals9 |
| 2018-06-23 16:14:39 | 2018-06-23 16:32:58 | 42M768 | DataSpySampleDataFileAllSignals15 |

Use these buttons to select all files of a certain type. After selecting files click on dropdown and select Add Data To File Set.

Select individual files by clicking on button next to each file.

10

Integrated Post Analysis – Creating Report on WirelessNeoVI

Reports are created from the Reports page on WirelessNeoVI

- Click on + New Report button
- Define report inputs and click on Create (see next slide)

The screenshot displays the 'Reports' page in the WirelessNeoVI interface. The page has a navigation bar at the top with links to 'Dashboard', 'Data Archive', 'Reports', 'Devices', 'Organizations', and 'Server'. A '+ New Report' button is located in the top right corner. Below the navigation bar, there are two report entries:

- TechDaysDemoFindInFilesReport**
 - Created: 2018-04-29 09:22:13
 - Script: FindInFiles_ExplnConfig
 - Config: ConfigForFindInFilesExplnConf
 - File Set: TechDaysSampleFileSet
 - Processing completed at: 2018-04-29 09:23:14
 - Output files: FindInFiles_04-29-18_13-23-14.dsr, IPA.log
 - Buttons: Re-run, Delete
- TechDaysDemoHistogramReport**
 - Created: 2018-04-29 09:20:26
 - Script: HistogramGenDemoFiles_032218
 - Config: IPAHistConfig_DemoData
 - File Set: TechDaysSampleFileSet
 - Processing completed at: 2018-04-29 09:20:47
 - Output files: HistogramGen_04-29-18_13-20-47.xlsx, IPA.log
 - Buttons: Re-run, Delete

Click on + New Report button to create a new report.

Some output file types are recognized by Wivi and they launch views from link.

All output files from a given report are listed on window of report.

11

Integrated Post Analysis – Creating Report on WirelessNeoVI

Each report has 4 inputs

- Report Name
- Script file to use to generate outputs from report
- Config file lists signal names and other script configuration parameters referenced in the script
- File Set – Named set of data files that the report is based on

New Report

Report Name

TechDaysDemoHistogramReport ✓

Post Analysis Script

HistogramGenDemoFiles_032218 ▼

Config

IPAHistConfig_DemoData ▼

File Set

TechDaysSampleFileSet ▼

← Cancel

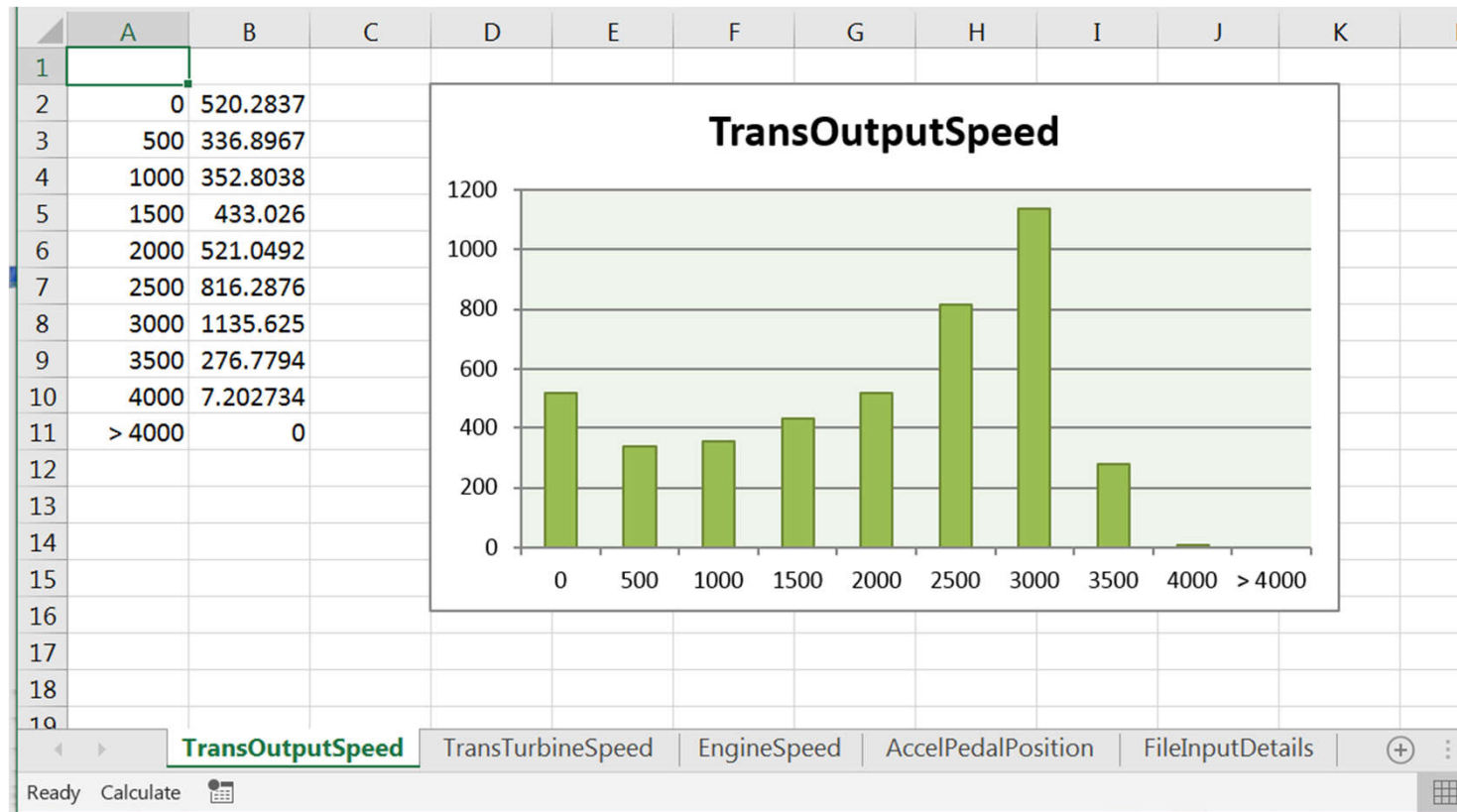
+ Create

12

Integrated Post Analysis – GenerateHistogram Script Output File

Script generates a histogram of each signal specified in the config using the bins specified in the config.

- Output file is *.xlsx
- Script uses Python xlswriter library to generate xlsx output
- Last tab lists all of the input files used for the report



Each signal is explicitly identified using channel_name, message_name, network name.

Config for histogram script generator also defines histogram bins for each signal.

Integrated Post Analysis – FindInFiles Script Output File (*.dsr)

FindInFiles script outputs a file with extension *.dsr which stands for DataSpy report

- DSR file is JSON file that lists all of the hits on events found in search
- Each hit has a Description, StartTime, and EndTime describing a period of interest in a data file

```
"HitList": [
  {
    "FilenameAndPath": "C:/Jmitchell/DataSpy/SourceCode/ICS_IPA_FORK/SampleDataFiles/DataSpySampleDataFi
    "Hits": [
      {
        "Description": "EngineStateVariableChangeFrom9To10",
        "EndTime": 37.81936073303223,
        "StartTime": 35.80622589588165
      },
      {
        "Description": "WOTLaunchFromRest",
        "EndTime": 39.58238410949707,
        "StartTime": 38.056864619255066
      },
      {
        "Description": "EngineStateVariableChangeFrom9To10",
        "EndTime": 39.94499200582504,
        "StartTime": 37.931904673576355
      },
      {
        "Description": "EngineStateVariableChangeFrom9To10",
        "EndTime": 48.49747771024704,
```

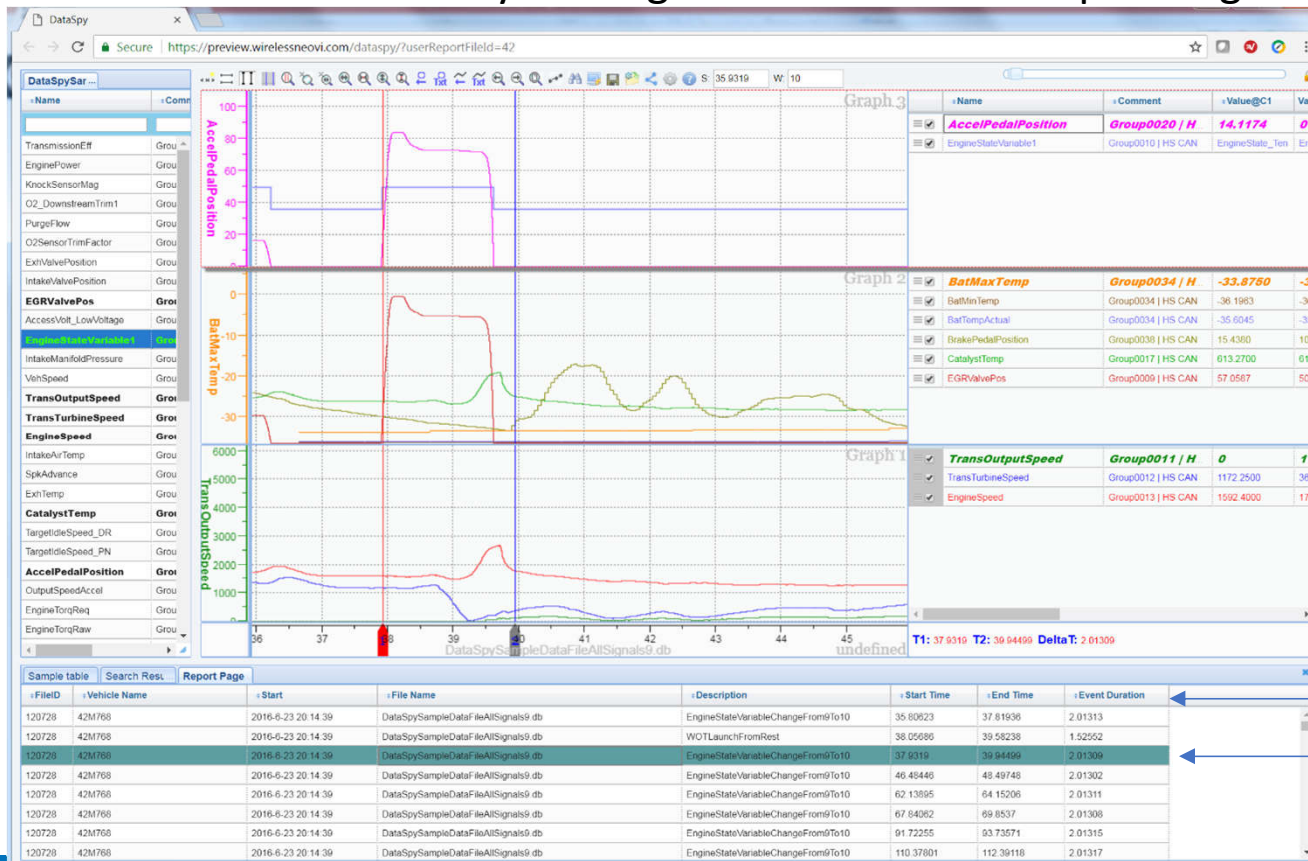
Filename is listed first, then all hits related to that file. Unlimited number of files and hits are supported.

Each hit, has a Description, StartTime, and EndTime

Integrated Post Analysis – FindInFiles Script Output File (*.dsr)

DSR files can be loaded into DataSpy allowing convenient organization and navigation of search results.

- Hits are displayed in report page on bottom of DataSpy window
- Users can click or use arrows to jump from hit to hit in DataSpy
- User can sort hits by clicking on header row of Report Page



Click on any column header to sort hits by values in column

Click on any hit in the report page to jump to that hit in the plot window.

15

Integrated Post Analysis – GenerateHistogram Script Config File

Script config file defines parameters used by the script.

- Most scripts will have input channels lists define in the config file
- Other parameters are based on design of script
- Allows users to control script performance without having to edit code.

name_in_script
is used to refer
to a signals
value within the
script

Config supports
optional list of
signals to use
among input
files.

```
{
  "Channels": [
    [
      {
        "name_in_script": "TransTurbineSpeed",
        "OptionalList": [
          {
            "channel_name": "TransTurbineSpeed",
            "message_name": "TransSignals_2",
            "network_name": "HSCAN"
          }
        ],
        "bins": [0, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500, 6000]
      },
      {
        "name_in_script": "AccelPedalPosition",
        "OptionalList": [
          {
            "channel_name": "AccelPedalPosition",
            "message_name": "DriverInputs05",
            "network_name": "HSCAN"
          },
          {
            "channel_name": "Pd1Pos",
            "message_name": "DriverStat1",
            "network_name": "MSCAN"
          }
        ],
        "bins": [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 99]
      },
      {
        "name_in_script": "EngineSpeed",
        "OptionalList": [
          {
            "channel_name": "EngineSpeed",
            "message_name": "EngineSignals_1",
            "network_name": "HSCAN"
          }
        ]
      }
    ]
  ]
}
```

Each signal is
explicitly
identified using
channel_name,
message_name,
network name.

Config for
histogram script
generator also
defines histogram
bins for each signal.

16

Integrated Post Analysis – FindInFiles Config File

In addition to the signal list, the FindInFiles script also allows users to provide a list of EventDefinitions to search for. Each event has the following parameters:

- **Description** – a name for the event which will show up in DataSpy
- **StartExpression** – a Boolean expression using signal name and any valid Python operators describing the start of the event
- **EndExpression** - a Boolean expression using signal name and any valid Python operators describing the end of the event. Once the StartExpression is found the script looks for the EndExpression. EndExpression can be not of StartExpression
- Putting a **“Prev__”** before a signal name in an expression refers to the value of the signal name in the previous loop
- **TimeFromExpStart** keyword can be used in the EndExpression to refer to the time since the current event started

```
"EventDefinitions": [{
  "Description": "WOTLaunchFromRest",
  "StartExpression": "(AccelPedalPosition > 80) and (TransOutputSpeed < 100)",
  "EndExpression": "(AccelPedalPosition < 50) or (TransOutputSpeed > 1000)"
},{
  "Description": "TipOutUpshifts",
  "StartExpression": "(AccelPedalPosition < 40) and (EngineSpeed > 4000)",
  "EndExpression": "(EngineSpeed < 2000)"
},{
  "Description": "EngineStateVariableChangeFrom9To10",
  "StartExpression": "(EngineStateVariable1 == 9) and (Prev__EngineStateVariable1 == 8)",
  "EndExpression": "(TimeFromExpStart > 2)"
}]
}
```

This event always
lasts 2 seconds due to
TimeFromExpStart > 2

This expression
triggers when
EngineStateVariable1
changes from 8 to 9

17

Integrated Post Analysis – GenerateHistogram example script

Sample GenerateHistogram script loops through list of files from start to end and creates a *.xlsx file that has a time based histogram.

- Does not require Excel
- Histogram bins are defined in script config.
- Uses Python numpy data analysis library

```
for dbFilePath in dbFilePaths:
    with icsFI.ICSDDataFile(dbFilePath, slFilePath) as db:
        ActiveMaskResult = db.SetActiveMask(MaskString)
        #db.SetActiveMask('1'*n_signals) # capture all signals
        prevTimestamp = 0.0
        curTimestamp = db.JumpAfterTimestamp(0)
        dataPoints = db.GetPoints()
        while curTimestamp != sys.float_info.max:
            for sig_num in range(n_signals):
                datum = dataPoints[sig_num]
                # find which bin datum belongs to
                bin_num = np.digitize([datum], bins_list[sig_num], right=True)[0]
                delta_time = curTimestamp - prevTimestamp
                time_tallys[sig_num][bin_num] += delta_time
                prevTimestamp = curTimestamp
            curTimestamp = db.GetNextRecord()

# Create a workbook and insert one worksheet for each signal.
xlHistWorkbook = ExcelHistogramReport()
for sig_num in range(n_signals):
    sig_info = signals[sig_num]
    xlHistWorkbook.add_sig_worksheet(sig_info, time_tallys[sig_num])

xlHistWorkbook.AddFileInfoListSheet(dbFilePaths, IPAInterfaceLibrary.is_running_on_wivi_server())
xlHistWorkbook.CloseWorkbook()
```

Top section of script calculates 2D time_tallys[] list that has the numbers for the histogram for each signal.

Bottom section writes histogram data to an Excel worksheet.

18

Integrated Post Analysis – FindInFiles example script

Sample FindInFiles script loops through list of files from start to end looking for events:

- Generates a *.dsr (DataSpy report) file that lists the hits for events in config file
- A few dozen lines of script support the FindInFiles functionality
- Python eval function used to evaluate text from config as code at runtime

```
for dbFilePath in dbFilePaths:
    try:
        with icsFI.ICSDDataFile(dbFilePath, slFilePath) as data:
            curTimestamp = data.JumpBeforeTimestamp(0)
            dataPoints = data.GetPoints()
            dsr.StartDSR(data)
            dataPointsPrev = dataPoints.copy()
            ActiveMaskResult = data.SetActiveMask("00010")
            while curTimestamp != sys.float_info.max:
                for i, expressionStart in enumerate(StartExpressionEval):
                    if EventActive[i] == False:
                        SearchExpState[i] = eval(expressionStart)
                    else:
                        SearchExpState[i] = eval(EndExpressionEval[i])
                    if EventActive[i] == False and SearchExpState[i] == True:
                        SearchExpStartTime[i] = curTimestamp
                        EventActive[i] = True
                        TimeFromExpressionStart[i] = 0
                    if EventActive[i] and EventActivePrev[i]:
                        if SearchExpState[i] == True:
                            SearchExpEndTime[i] = curTimestamp
                            dsr.LogHit(EventDescription_list[i], SearchExpStartTime[i], SearchExpEndTime[i])
                            EventActive[i] = False
                            TimeFromExpressionStart[i] = -1
                        else:
                            TimeFromExpressionStart[i] = curTimestamp - SearchExpStartTime[i]
                    EventActivePrev[i] = EventActive[i]
            dataPointsPrev = dataPoints.copy() # copy previous loops record array to new array
            curTimestamp = data.GetNextRecord()
```

Script converts user defined events into executable code then uses the “eval” function in Python to run the code at runtime

GetNextRecord function advances the virtual cursor to next record; event conditions are checked for each record throughout file.¹⁹

Integrated Post Analysis Review – Main Script Functions

For the Python scripts, the IPA functions are wrapped in a Python class called ICSDDataFile. A data filename and config filename are provided when the script creates an instance of this class.

JumpBeforeTimestamp(TimeToJumpToInSeconds)

- Input is desired time in Datafile to place the virtual cursor
- Places virtual cursor at closest time prior to TimeToJumpToInSeconds where there is a complete record
- If TimeToJumpToInSeconds is lower than time corresponding to the first complete record it places the virtual cursor at the first complete record
- Updates array of signal values corresponding to location of virtual cursor
- Returns the exact position of the virtual cursor

GetNextRecord()

- Advances the virtual cursor to the next chronological time where any of the active signals have updated values
- Returns the exact position of the virtual cursor
- Updates array of signal values corresponding to location of virtual cursor
- If virtual cursor is already at the last record in the file it returns MaxDouble (10^{308})

Integrated Post Analysis – Main Script Functions (cont)

GetNextChangedRecord()

- Advances the virtual cursor to the next chronological time where any signal has a value that is different than the current records values
- Returns the exact position of the virtual cursor
- Updates array of signal values corresponding to location of virtual cursor
- If virtual cursor is already at the last record in the file it returns MaxDouble (10^{308})

SetActiveMask(SignalMaskString)

- Used to control which signals the GetNextRecord() function considers when stepping to the next record.
- If only one signal is active, then GetNextRecord() will step through each timestamp that corresponds to this signal.
- If all signals are active GetNextRecord() will step through every new timestamp in every signal in the config file.
- By default all signals are active.

Integrated Post Analysis – CurrentRecord Definition

One challenge in performing calculations on time based data is how to handle multiple asynchronous time vectors

- One common practice is to force the data to fit a fixed time rate. This method has detrimental effect on fidelity of data
- IPA method matches the way DataSpy displays values at cursor in legend. The legend reports the most recent values that precede the cursor.

View data points tool shows dots on each trace indicating where updates occurred

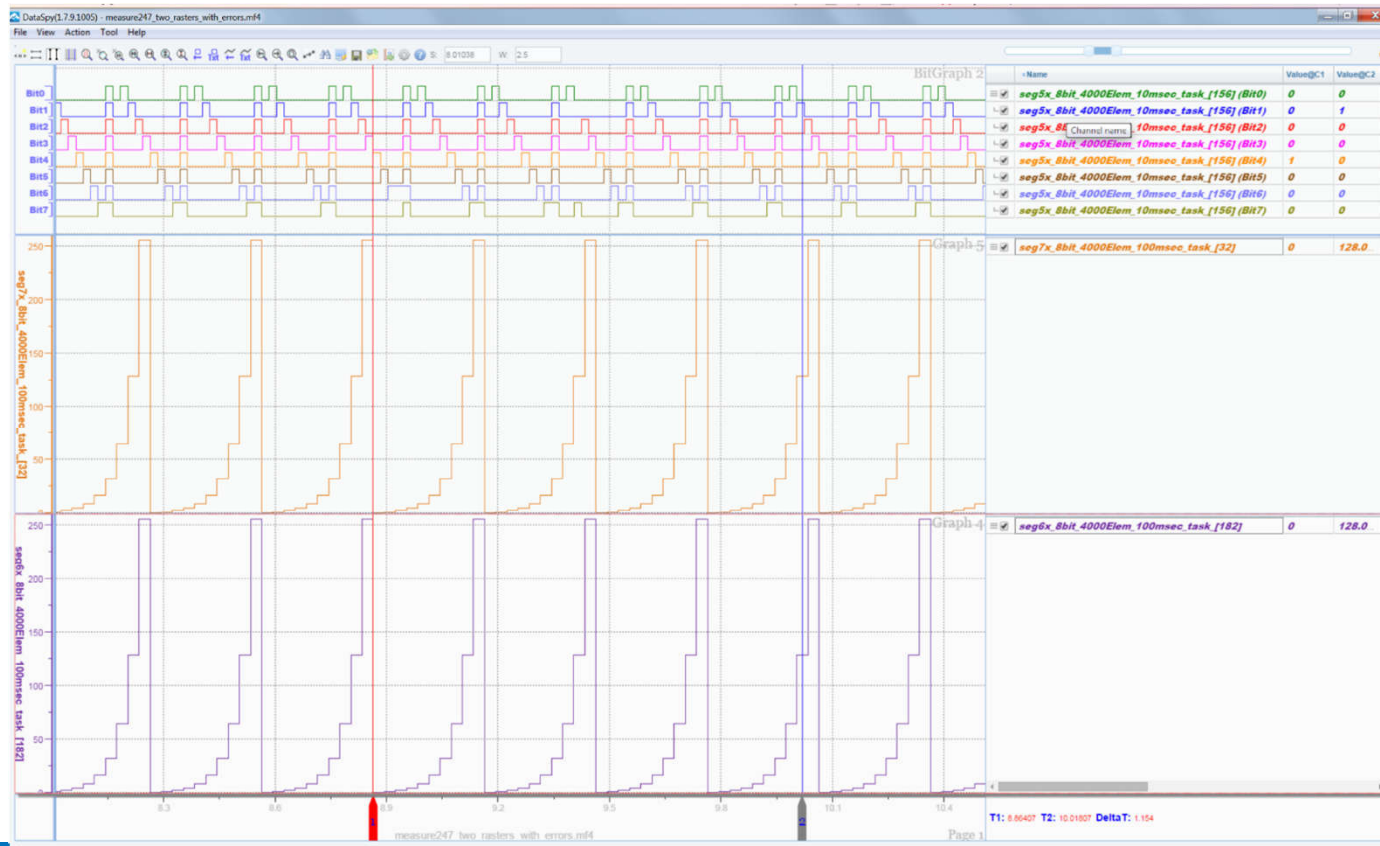


Value@C1 column in legend shows value of most recent point that precedes position of cursor 1.

Integrated Post Analysis – Used for cmProbe Validation

IPA Library used to validate cmProbe bench data on 40,000 simultaneous signals

- Bench ECU set up to generate 40,000 signals that follow known walking ones pattern
- Script written to search through large files and look for cases where recorded pattern did not match expected pattern like the one shown below



23

Integrated Post Analysis Review – Numpy vs Matlab

Indexing and accessing elements (Python: slicing)

| MATLAB/Octave | Python | Description |
|--|--|--------------------------------|
| <code>a = [11 12 13 14 ... 21 22 23 24 ... 31 32 33 34]</code> | <code>a = array([[11, 12, 13, 14], [21, 22, 23, 24], [31, 32, 33, 34]])</code> | Input is a 3,4 array |
| <code>a(2,3)</code> | <code>a[1,2]</code> | Element 2,3 (row,col) |
| <code>a(1,:)</code> | <code>a[0,]</code> | First row |
| <code>a(:,1)</code> | <code>a[:,0]</code> | First column |
| <code>a([1 3],[1 4]);</code> | <code>a.take([0,2]).take([0,3], axis=1)</code> | Array as indices |
| <code>a(2:end,:)</code> | <code>a[1:,]</code> | All, except first row |
| <code>a(end-1:end,:)</code> | <code>a[-2:,]</code> | Last two rows |
| <code>a(1:2:end,:)</code> | <code>a[:,2:,]</code> | Strides: Every other row |
| | <code>a[...,-2]</code> | Third in last dimension (axis) |
| <code>a(:,[1 3 4])</code> | <code>a.take([0,2,3],axis=1)</code> | Remove one column |
| | <code>a.diagonal(offset=0)</code> | Diagonal |

Assignment

| MATLAB/Octave | Python | Description |
|-----------------------------------|---|--|
| <code>a(:,1) = 99</code> | <code>a[:,0] = 99</code> | |
| <code>a(:,1) = [99 98 97]'</code> | <code>a[:,0] = array([99,98,97])</code> | |
| <code>a(a>90) = 90;</code> | <code>(a>90).choose(a,90)</code> | Clipping: Replace all elements over 90 |
| | <code>a.clip(min=None, max=90)</code> | |
| | <code>a.clip(min=2, max=5)</code> | Clip upper and lower values |

Transpose and inverse

| MATLAB/Octave | Python | Description |
|----------------------------------|-----------------------------------|-------------------------|
| <code>a'</code> | <code>a.conj().transpose()</code> | Transpose |
| <code>a.' or transpose(a)</code> | <code>a.transpose()</code> | Non-conjugate transpose |
| <code>det(a)</code> | <code>linalg.det(a) or</code> | Determinant |
| <code>inv(a)</code> | <code>linalg.inv(a) or</code> | Inverse |
| <code>pinv(a)</code> | <code>linalg.pinv(a)</code> | Pseudo-inverse |
| <code>norm(a)</code> | <code>norm(a)</code> | Norms |
| <code>eig(a)</code> | <code>linalg.eig(a)[0]</code> | Eigenvalues |
| <code>svd(a)</code> | <code>linalg.svd(a)</code> | Singular values |
| <code>chol(a)</code> | <code>linalg.cholesky(a)</code> | Cholesky factorization |
| <code>[v,1] = eig(a)</code> | <code>linalg.eig(a)[1]</code> | Eigenvectors |
| <code>rank(a)</code> | <code>rank(a)</code> | Rank |

NumPy is a Python library that supports large, multi-dimensional arrays and has many functions that mimic MATLAB.


NumPy includes ndarrays arrays which can be used as views into memory buffers allocated by C/C++ dlls so that data can be exchanged without the need to do a lot of copying which is slow.

“MATLAB is to Python what Encyclopedia Britannica is to Wikipedia” crowd based engineering

Integrated Post Analysis – Script Config File Generator

Excel tool used to generate signal list portion of config file

- Allows users to pick signals from files on Wireless NeoVI or on PC
- Supports prioritized optional list for each signal (multiple versions of Engine Speed)
- Excel tool uses functions from the IPA library



Wivi URL:

Username:

Password:

Start Date:

End Date:

Fleet Name:

Collection Name:

Update File List From Wireless NeoVI

Update Signal List Using Selected FileID

Update Signal List from File on Your PC

Export Signal List *.asl and *.py Files

After defining signals click on button to generate asl (aliased signal list)

| Vehicle | File Capture Date | File Size (KB) | FileID | Signal Name | Message Name | Network Name | Name In Script | Priority |
|---------------|-------------------|----------------|--------|--------------------------------|--------------|--------------|--------------------------------|----------|
| John Mitchell | 4/12/18 4:05 PM | 90272 | 26804 | ManufacturerSpecific8 | ProprietaryA | HS CAN | ManufacturerSpecific8 | |
| John Mitchell | 4/12/18 3:19 PM | 88924 | 26792 | ManufacturerSpecific7 | ProprietaryA | HS CAN | ManufacturerSpecific7 | |
| John Mitchell | 4/12/18 1:47 PM | 4944 | 26633 | ManufacturerSpecific6 | ProprietaryA | HS CAN | ManufacturerSpecific6 | |
| John Mitchell | 4/10/18 11:09 PM | 713556 | 25764 | ManufacturerSpecific5 | ProprietaryA | HS CAN | ManufacturerSpecific5 | |
| John Mitchell | 4/10/18 9:37 PM | 53580 | 25613 | ManufacturerSpecific4 | ProprietaryA | HS CAN | ManufacturerSpecific4 | 1 |
| John Mitchell | 3/24/18 6:05 PM | 291328 | 25615 | ManufacturerSpecific3 | ProprietaryA | HS CAN | ManufacturerSpecific3 | |
| John Mitchell | 3/24/18 12:52 AM | 165796 | 25611 | ManufacturerSpecific2 | ProprietaryA | HS CAN | ManufacturerSpecific2 | |
| John Mitchell | 3/23/18 10:54 PM | 129756 | 25609 | ManufacturerSpecific1 | ProprietaryA | HS CAN | ManufacturerSpecific1 | |
| John Mitchell | 3/23/18 10:07 PM | 298748 | 25607 | SourceAddress | ETC1 | HS CAN | SourceAddress | |
| John Mitchell | 3/23/18 9:28 PM | 193828 | 25605 | InputShaftSpeed | ETC1 | HS CAN | InputShaftSpeed | |
| John Mitchell | 3/23/18 4:45 PM | 125524 | 25603 | ProgressiveShiftDisable | ETC1 | HS CAN | ProgressiveShiftDisable | 1 |
| John Mitchell | 3/23/18 3:48 PM | 85120 | 25601 | MomentaryEngineOverspeedEnable | ETC1 | HS CAN | MomentaryEngineOverspeedEnable | |
| | | | | PercentClutchSlip | ETC1 | HS CAN | PercentClutchSlip | |
| | | | | OutputShaftSpeed | ETC1 | HS CAN | OutputShaftSpeed | |
| | | | | ShiftInProcess | ETC1 | HS CAN | ShiftInProcess | |
| | | | | TorqueConverterLockupEngaged | ETC1 | HS CAN | TorqueConverterLockupEngaged | |
| | | | | DrivelineEngaged | ETC1 | HS CAN | DrivelineEngaged | 1 |
| | | | | SourceAddress | EEC1 | HS CAN | SourceAddress | |
| | | | | EngineSpeed | EEC1 | HS CAN | EngineSpeed | |
| | | | | ActualEngine_PercTorque | EEC1 | HS CAN | ActualEngine_PercTorque | |
| | | | | DriversDemandEngine_PercTorque | EEC1 | HS CAN | DriversDemandEngine_PercTorque | |
| | | | | EngineRetarderTorqueMode | EEC1 | HS CAN | EngineRetarderTorqueMode | |
| | | | | RemoteAccelerator | EEC2 | HS CAN | RemoteAccelerator | |

Pick signals by putting priority number next to each signal referenced in the script. Use priority 1 if no optional list required.

25

Integrated Post Analysis – Road Map

Develop more example scripts

- Work with existing WirelessNeoVI customers to develop useful scripts to automate their current workflows
- Pull out key metrics from particular ECU feature and generate report
- Support more ways to visualize output data
 - View Excel output files directly on Wivi
 - Support other visualization utilities like ChartJS
- Multi-thread scripts to maximize performance
 - Currently scripts roughly process 1Gb/15sec
 - Multi-threading could speed that up to 1Gb/2sec
 - Custom C++ code could be written with multi-thread for maximized performance.

Several features to implement on WirelessNeoVI

- Use Docker to create VMs to run each script
- Change permissions so that all users in an organization can view reports and run scripts. In order to view or download a report, a given user would have to have read access to all files in FileSet
- Support Public and Private reports

26

THANK YOU

For any queries or questions please feel free to contact:

John Mitchell
Intrepid Control Systems, USA
Madison Heights, MI, USA 48071
+ 1 248 416 2848

Also, you can visit our website www.intrepidcs.com for more information

27



April 30, 2019



INTREPID
CONTROL SYSTEMS
www.intrepidcs.com