Getting Started with PiSnoop

This is a one-page summary of what PiSnoop is about and how to start using it.

Users of Pi Innovo Ltd's OpenECU product can find an application note on OpenECU and PiSnoop here, which explains setup tasks and operations which will also be relevant to non-OpenECU users including licence management, hardware configuration, flash reprogramming and handling calibration parameters.

Purpose of PiSnoop

PiSnoop is a tool which allows you to interact with software while it runs in real time on an embedded system or electronic control unit (ECU). It uses a communications link to gain access to the memory of the ECU for read and write operations. It also loads information about the program running on the ECU so that it is aware of the memory layout, functions and variables present in the ECU software. Typical uses for PiSnoop are:

- Software debugging: by inspecting and altering the values of internal variables and memory during ECU program execution, you can determine if the embedded program is running as expected and influence its behaviour.
- Testing: PiSnoop can be used as part of test procedures which exercise embedded software, where results may be obtained via program variables and tests run by setting internal values. ASAP3 control is supported.
- Flash reprogramming: you can use PiSnoop to download a new program version or block of constant data to the ECU.
- Calibration editing: where the ECU software behaviour depends on "constant" data which is actually in RAM, PiSnoop can be used to alter those values to tune the ECU behaviour for a particular application.
- CAN bus work: PiSnoop can generate, monitor, and log CAN bus messages.
- Diagnostics: PiSnoop is able to interact with ECUs via additional protocols, for example to emulate an automotive OBD scan tool. It can therefore be used to inspect ECU diagnostic information and perform tests of diagnostic support.

Overall PiSnoop is intended to aid the developers of embedded systems with the debugging, testing and configuration of their ECU software.

Important Safety and Liability Information

Many embedded systems have the potential to present safety hazards, cause commercially expensive damage to physical equipment, or make important services unavailable if they do not operate correctly, etc. Such systems should be developed with an appropriate degree of rigour considering the consequencs of their failure, which are entirely specific to your application.

PiSnoop itself is intended only as an informal software development tool and is not itself developed to safety-related standards.

PiSnoop makes direct access to the memory of embedded systems via communication protocols. By using PiSnoop you accept that this can have unexpected or undesirable results and you accept all responsibility for the outcome, even if this program behaved incorrectly.

How to Start

Set Up Communications Hardware

PiSnoop needs to use a physical communications interface such as a CAN card to interact with the ECU. The device should be supplied with device driver software from the manufacturer, which you must install before you can use it with PiSnoop.

1. Do File... New Hardware Comms Channel to open a hardware window.
2. Select the type of interface you have in the list of supported types, e.g. Vector CAN Interface.
3. Adjust the settings if necessary to select the correct hardware interface (if you have more than one) and communication parameters suitable for your ECU.
4. If required you may define multiple hardware channels and you may mix hardware from different manufacturers.

See Hardware window help for more details.

Set Up the ECU

PiSnoop is supports multiple ECUs in parallel if required, but typically is used with just one.

1. Do File... New ECU to open an ECU window.
2. Use the ECU Add... button to add the linker output file (e.g. program.elf) or ASAP2 file (e.g. build.a2l) generated by your software build process. PiSnoop loads symbols from this, i.e. the list of variable and function names used in your program, together with their memory addresses. This is required if you want to access variable values by name.
3. Use the ECU Add... button to add the image or hex file (e.g. program.s37) generated by your software build process. PiSnoop uses this to display the "original" content of memory. In many systems this file should be used for flash reprogramming and not the linker output file, because it may have additional data inserted (e.g. validation checksums) which the program needs to run correctly.
4. Click the ECU Load button to load the linker symbols and image data into the PC memory. You should do this again each time you rebuild your software project.
5. Ensure the correct hardware interface is selected for this ECU.
6. If necessary change the memory access communication protocol and its settings to match the protocol settings which your ECU implements.
7. Seed-key security may be used by some protocols.

See ECU window help for more details.

Saving the Workspace

PiSnoop uses the concept of a workspace to save all of the settings for your current project. The workspace remembers the windows you have open and all of the settings they contain.

1. Do File... Save Workspace to save your current workspace as a file, e.g. my_project.snw. Usually this should be saved somewhere close to the output of your build process, because the ECU files are referenced by relative location if possible, and the workspace file can then be "moved" along with your build.
2. Do File... Open Workspace or use the list of recent files at the bottom of the File menu to reload this workspace in future.
3. The workspace file is human-readable XML format.
4. Save the workspace as a SNoop teMplate (*.snm) file to create your own template for future reuse.
5. See also File... Workspace Properties to store your own data.
6. Do File... Import from Other Workspace to pull in items like Watch variable lists from a previous workspace into your current project.

Flash Reprogramming

PiSnoop can be used to load a new program version or other block of data into the ECU, if the ECU supports this operation via the communication protocol.

1. You need at least one image or hex file specified in the ECU window with the "Flash?" option selected. Alternatively, a linker file may be used directly on some ECUs.
2. Click the Flash or Load and Flash button on the ECU window.
3. PiSnoop will show the progress of flash reprogramming and report any errors which occur during the process.
4. Depending on your ECU it may be necessary to cycle the power so that the new program starts running.

See ECU window help for more details.

Inspect Memory

PiSnoop has a memory viewing/editing window, similar to that in software debuggers. You can use this as a quick check that the program is communicating successfully with the ECU. It shows the hex value of each byte in memory.

1. Do Window... New Memory Explorer to open a new memory window.
2. Go to an address which corresponds to valid memory in your ECU.
3. See Understanding Memory for the colour coding used.
4. Places in memory which belong to variables or functions are highlighted if you move the mouse pointer over them (if you have loaded a linker file). This allows you to see which objects are adjacent in memory and identify variables which are changing value.
5. You may edit byte values directly. Click on a byte to edit it. You can use the arrow or TAB keys to move to adjacent bytes.
6. You may open many memory windows, but bear in mind that their continual update may use a significant amount of the bandwidth available for ECU memory access.

Access Variables, Logging and Oscilloscope

PiSnoop has a watch window, similar to that in software debuggers. You can use this to inspect or edit the current values of variables used in your ECU program.

Additionally, a live graphical oscilloscope-like view of variables can be viewed, and logged data saved at high speed. Later, logged data can be re-opened for inspection and analysis in the scope.

1. Do Window... New Watch Window to open a new watch window.
2. Ensure symbols are loaded from the ECU linker file.
3. Click on the Symbols... button to browse for variables to add to the watch window. You may select many variables at once.
4. Alternatively, you may type variable names directly into the Name column (or copy and paste them from your code editor).
5. Arrays and structures can be expanded as a tree, and pointers followed to a certain depth.
6. Click "Scope" to open the graphical view. Running the scope consumes more communications bandwidth.
7. You may open many watch windows, but all the data currently visible will take some bandwidth to keep updated.
8. DBC-defined CAN signals can be watched, logged and scoped in Watch windows directly alongside internal ECU values.

See Watch Window help and calibration help for more.

Use Lookup Tables

Embedded systems often use lookup tables or "maps" as functions which map input values to output values.

1. Do Window... New Lookup Editor to open a lookup window.
2. Ensure symbols are loaded from a linker file.
3. The array containing the output values looked up is always known as Z. Click the "..." button next to "Z-out array" to choose the array.
4. For a 1-dimensional lookup of the form z = f(x), specify the 1-D array of breakpoints as the X-array if you want the table presented horizontally or as the Y-array to display it vertically.
5. For a 2-dimensional lookup of the form z = f(x,y), specify the X-array as the breakpoint values which index the columns and the Y-array for the values which index the rows.
6. Optionally, you may specify the scalar variables in ECU memory which contain the value(s) being used as inputs to the lookup, and the value obtained as its output. If so, PiSnoop displays their current values as the current operating point.
7. Right-click on the map data for more options, including interpolation (automatic filling of a range of cells based on the end or corner values).

For more see Lookup Editor help and calibration help.

CAN Monitor

PiSnoop provides a general-purpose CAN monitoring window with a logging facility.

1. Do Window... New CAN Monitor to open this kind of window.
2. By default all CAN channels start with monitoring enabled except for CCP messages, which otherwise tend to swamp what you see.
3. Note that the "By ID" tab shows the latest message received for each CAN ID, while the "Chronological" tab shows all messages listed in time order.
4. Load a DBC file to see named messages and human-readable signal values, including multiplexed signals.

For more see CAN Monitor help.

CAN Sender

PiSnoop provides a general-purpose CAN sending window.

1. Do Window... New CAN Sender to open this kind of window.
2. Add messages you wish to send and click "Tx" to send a message once.
3. Use the periodic settings to send a message repeatedly.
4. Load a DBC file to browse for named messages and set named signal values, including multiplexed signals.

For more see CAN Sender help.

ISO 15765-2 Diagnostics

For automotive ECUs, on-board diagnostic (OBD) access over the CAN bus is supported for ISO 15031-5/SAE J1979, Keyword 2000-3 and UDS. PiSnoop simply allows you to send any string of bytes over the ISO15765-2 protocol, so you can use it to send messages according to any of those standards.

1. Do Window... New ISO Diagnostics to open this kind of window.
2. If necessary, change the CAN identifiers to match those recognised by your ECU.
3. To send a message, type some bytes and click Transmit.
4. Optionally you can define a name for a message you have prepared and save it to the list on the right for future re-use. A few examples are provided to get you started.
5. The list of messages for this window is saved as part of the workspace. If you close the window however, its settings are lost. You may minimise it without losing its settings.
6. Seed-key security may be used.

For more see ISO 15765-2 Window help.

J1939 Messaging and Diagnostics

SAE-J1939 is a set of standards defining a protocol for transferring data around vehicles over CAN, especially in heavy-duty vehicles in North America.

1. Do Window... New J1939 Window.
2. Set the DA that your ECU uses as its node address.
3. Set the SA that you wish PiSnoop to use as its node address. (Note: it will not use the address resolution protocol to claim this address.)
4. To send a PGN, type in some bytes and press Transmit.
5. To request a PGN, type its identifier in the Request PGN box and the other fields and transmit data will be set automatically to request it. Then press Transmit.
6. You can optionally save a message you have prepared for later re-use in the list to the right, which is saved as part of the workspace.

For more see J1939 Window help.

Extending PiSnoop with your own plug-ins

PiSnoop uses plug-in DLLs to handle interactions with hardware interfaces, different ECU protocols and custom user functions. It can also be invoked by other programs that need to make use of its ability to interact with embedded systems. By writing your own plug-ins, you can customise PiSnoop to work in new ways.

For more see Developer help.