ACCELERATING SOFTWARE DEVELOPMENT FOR RESPONSIVE AND RELIABLE IOT M2M SOLUTIONS

DEVELOP FAST AND STABLE SYSTEM SOFTWARE FASTER WITH INTEL® SYSTEM STUDIO, A COMPREHENSIVE SOFTWARE BUILD, DEBUG, ANALYSIS, AND PERFORMANCE TUNING SOLUTION FOR WIND RIVER® IDP OR YOCTO PROJECT*-BASED IOT GATEWAYS.

Challenges Developing for Connected Devices

Distributed software workloads running on connected devices in Internet of Things (IoT) applications, such as those supporting industrial, transportation, or infrastructure use cases, can be very time sensitive. Connected devices must process data in a multitude of formats and with different priorities, which creates significant demands with respect to managing I/O streams, responsiveness, and continuous reliability – notwithstanding temporary workload spikes or changing environmental conditions. This makes it necessary to have development tools that can streamline IoT gateway development with a strong focus on performance analysis to reduce stress on memory and CPU, and debug to resolve device driver, I/O, and system software issues before they show up in the field.
Performance and Analysis Studio for your IoT Gateway

Intel® System Studio provides a one-stop software development suite solution, tying closely into the Wind River® Intelligent Device Platform Workbench® IDE, as illustrated in Figure 1. In doing so, it allows developers to customize, optimize, and enhance the reliability of the system software stack running on Intel® IoT Gateway.

Intel System Studio (Figure 2) covers the three central pillars of the software design cycle: Build, Analysis, and Debug. Its components are shown in Figure 3 and described in the following:

- **DEBUGGERS & TRACING**
  - System
  - Application
- **ANALYZERS**
  - Power & Performance
  - CPU / Graphics
  - Memory & Threading
- **COMPILERS & LIBRARIES**
  - C/C++ Compiler
  - Image, Signal, Media, Data & Math Processing
- **JTAG Interface**
  - System & Application Code Running on Linux®, Android®, or Windows®
  - Intel® Quark™ SoC
  - Intel® Atom™ Processor
  - Intel® Core™ i7, Intel® Core™ i5, Intel® Core™ i3 Processors
  - Intel® Xeon® Processor

1. Linux®, Embedded Linux, Wind River® Linux, Yocto Project®, Tizen®

Figure 2. Intel® System Studio Overview
If an EFI Development Kit (EDK II) compliant debug agent is part of the firmware, BIOS debug without a JTAG device is also available.

The System Visible Event Nexus (SVEN) Technology SDK provides a static code-instrumentation-based approach that supports ultra-low overhead data event tracing, which is useful for debugging I/O timing and data hand-shake issues between device driver and application space.

- The Intel-enhanced GNU* Project Debugger (GDB*) provides the ability to interactively debug data race conditions in Intel® Compiler generated code. It also provides process-specific Branch Trace Storage (BTS)-based instruction tracing in User Space, permitting debugging of the execution history even in scenarios where reverse execution is not possible or the callstack is corrupted.

Analyzers
- The Intel® VTune™ Amplifier for Systems provides remote performance analysis and energy profiling from a Linux* or Windows*-based host development system via SSH protocol. Performance bottlenecks and their root causes can be pinpointed quickly and efficiently.

On Intel® Atom™ processor-based platforms, the tool can additionally use the processor’s performance monitoring unit (PMU) for system-level performance analysis and general exploration of memory bandwidth, caching, and instruction latency concerns. In addition to its own performance analysis data format, it also can visualize Linux Ftrace and Perf data.

Compilers and Libraries
- The Intel® C++ Compiler is a highly-optimized compiler with state-of-the-art architectural optimizations for Intel Atom processors and Intel Quark SoC as well as advanced vectorization and loop optimizations. It seamlessly plugs into a Wind River Linux cross-build environment and allows developers to mix and match GNU* GCC* and Intel C Compiler-generated binaries as it reuses the GNU linker and binutils. Whenever data parallelism and fast compute are of the essence, the Intel C++ Compiler can be used to drive the responsiveness of gateways forward.

- The Intel® Integrated Performance Primitives provide a rich set of software building blocks that enable developers to write highly-optimized and easy-to-maintain signal and media processing codecs.
Using the Intel® C++ Compiler from within Wind River® IDP Workbench*

Developers can use Intel C++ Compiler in the Wind River Workbench to build and optimize their code. To do this, create and build a Wind River Linux platform project in the workbench, or import a Wind River Linux SDK into Wind River workbench. The following steps describe how to use the Intel C++ Compiler inside the Workbench:

1. Create a new project in the Wind River workbench by selecting File > New > Example > WR Linux Application Sample Project, select “The Hello World Demonstration Program”, click Finish.

2. Select Project > Properties > Build Properties > Build Support and Specs. Select an Active build spec that supports the Intel C++ compiler, the one that has the –icc extension in the name. For example, intel-x86-64-glibc_std-x86-64-wrl6_x86_64_prj-icc, as shown in Figure 4.

3. Build the application.

4. Modify the ICC (Intel C++ Compiler) build flags and linker flags, if needed. Select Project > Properties > Build Properties > Tools; select Build tool C++ Compiler or other tool, if needed; check the build flags and modify, if needed.

Figure 4. Using the Intel® C++ Compiler Inside the Wind River® IDP Build Environment
Grant write access to the proper /dev/bus/usb entry every time a device is connected in order to run OpenOCD using a non-root account. The process can be automated by adding a udev rule. Simply create a text file in the rules directory:

```
$ sudo vim /etc/udev/rules.d/99-openocd.rules
```

The product ID codes depend on the JTAG device. For example, for the Flyswatter2* and the Olimex* device, the rules file must have the following content:

```
SUBSYSTEM=="usb", ATTR{idVendor}=="0403", ATTR{idProduct}=="6010", MODE="0666"
SUBSYSTEM=="usb", ATTR{idVendor}=="15ba", ATTR{idProduct}=="002b", MODE="0666"
```

Starting a Debug Session

1. Launch OpenOCD*

   - Go to the place where the OpenOCD component was installed
   - Default location:
     `<install-dir>/system_studio_2015.0.xxx/debugger/openocd/bin`

2. Launch OpenOCD:

   - `$ cd <installdir>/system_studio_2015.0.xxx/debugger/openocd/
   - `$ ./bin/openocd -f scripts/interface/ftdi/olimex-armusb-ocd- h.cfg -f scripts/board/quark_x10xx_board.cfg`

2. Launch the Intel System Debugger

   - After you power on the target and launch OpenOCD
   - Start "start_xdb_gdb_remote.bat"
   - Click on the "connect icon" at the upper left corner of the Intel System Debugger graphical user interface.

   It will automatically try to connect to the OpenOCD's local host default settings (127.0.0.1:3333), as shown in Figure 5.
Next Steps

The Intel System Studio provides a complete build, analysis, optimization, and debug solution that enables developers to customize, optimize, and maximize the reliability of system software and C/C++ application software stack running on an Intel IoT Gateway. Its close and seamless integration with the Intelligent Device Platform build environment from Wind River enables developers to have full control over the gateway software stack.

For more information about Intel solutions for the Internet of Things, visit [www.intel.com/iot](http://www.intel.com/iot).

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel® microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimizations on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804