WSim: Worldsens Platform Simulator

User Documentation
May 2007

Antoine Fraboulet

This manual is for WSim, version 0.8cvs.

Table of Contents

1 Introduction

WSim is part of the Worldsens development environment. WSim is a full platform simulator that can run the target platform object code without modification. The primary goals of WSim are debugging, profiling and performance evaluation for deeply embedded wireless sensor node/network (WSN) distributed applications.

Writing distributed WSN applications using the worldsens toolkit is far more easier than using prototype boards due to the fast compile/debug/estimate development cycle. This performance estimation is made possible using a full platform simulation at a cycle precise level. Peripherals that deal with real time constraints (timers, interrupts, serial ports, radio interface, etc.) are included in the WSim platform model. Time is given in a cycle accurate manner using application instruction timing.

WSim provides a simulation environment in which you can get

- Full instruction set support for the target micro-controller
- Cycle-accurate simulation
- Simulation of peripherals such as timers
- Interrupts
- Cycle-accurate simulation of other peripherals (e.g. UART)
- External peripherals (radio modules, LCD display, ...)
- A **full system** debug and performance analysis framework

This manual describes the user interface of the WSim simulator. The internals of the simulator, including how to write the description of a new platform (a new simulation target) and some information about how to write hardware component models, are documented in a separate manual.

WSim should compile and run on any Un*x-like platform. It has been tested on Linux/x86, Linux/x86_64, Linux/ppc32, Linux/Alpha, Solaris9/Sparc, Solaris10/Sparc, MacOSX/ppc32, MacOSX/x86 and Windows XP (using cygwin). Please refer to the internal documentation for more information about supported platforms and porting to new operating systems.

2 WSim Command Line Options

WSim uses both general options to control the simulation behaviour as well as platform specific options. These options are enabled only when the associated peripherals are included in the simulation platform. Platform specific options are listed and explained in their appropriate sections.

This section presents the general options that are common to all platforms.

```
usage: wsim [options] [binary_file]
```

'`--help`' Print command line help.

'`--version`'
Print WSim and exit.

'`--ui`' Enables a user interface that includes target dependent output peripherals (leds, 7 segments, LCD display, buttons, ...)

'`--mode`' Starts WSim in gdb mode waiting on default port 2159/TCP

'`--modearg=int`'
Starts WSim in gdb mode waiting on the specified TCP port

'`--verbose=value`'
Set the level of debug and warning messages.

'`--logfile=filename`'
Set an output file name for WSim log system. Default is set to `stdout`

'`--trace[=tracefile]`'
Enables trace output. The tracer module records numbered events during the simulation. By default, WSim currently records interrupts and power modes. These traces can be extended in the platform configuration in order to record any event during the simulation. Traces can be read using the WTracer tool provided in the Worldsens toolsuite.

3 Description of Use

WSim is a cycle accurate platform simulator that takes an ELF (Executable and Linkable Format) file compiled for the target processor architecture as input. The executable file should be an executable statically linked object file that is a standalone application.

WSim uses an instruction based timing model. The main processing unit is thus responsible for simulation timing according to its frequency and instruction set performance numbers. When the micro-controller is in a low power state which inhibits the instruction fetch - decode - execute loop, an arbitrary delay loop is introduced to drive the simulation. The micro-controller unit model is fully responsible for time keeping within the simulation. For more information, see the WSim internal documentation.

The current version of WSim as only been tested using GCC generated ELF files produced using the GNU binutils tools. WSim should be compliant with other compilers and tools as long as they produce standard ELF files.

3.1 Running a program using WSim

WSim takes as arguments several command line options that are used to drive the simulation and an ELF program targeted for the simulated platform.

```
wsim --ui --trace=wsim.trc --mode=time --modearg=4000000000 program.elf
```

This command line starts a simulation with a user interface for peripherals display such as leds and LCD screen. The simulation will last for 4 seconds (time is expressed in nano seconds on the command line). The simulation will output trace files that can be read by the `wtracer` tool.

Simulation time is unlimited unless otherwise specified using `--mode=steps` or `--mode=time`.

The `--verbose` output can display various runtime information about program execution. An example of simulation output is presented below

```
== MSP430 creation
  model: msp430f1611
==
PTTY1: local fifo is [/dev/pts/6]
PTTY1: remote fifo is [unknown]
== Machine description
mcu   : msp430
model : msp430f1611
device list:
  dev 00 : m25p80 flash memory
  dev 01 : Led display
  dev 02 : Led display
  dev 03 : Led display
  dev 04 : ds2411 silicon serial number
  dev 05 : ptty serial I/O
  dev 06 : 7 segments display
  dev 07 : 7 segments display
==

==
== simulation time :  2.1 seconds
== simulated time  :    6 seconds
== speedup    : 2.9
== backtracks : 0
```

==

The simulation time is given using the "wall clock" processor time used by the simulation process (this may vary on some systems). The speedup time is the ratio between the simulation time and the simulated time. The speedup depends on the simulated activity (serial port communication, ui widget drawing, ...)

3.2 Error reporting and Warnings

WSim reports dynamic common errors and warnings that occur during the simulation. Examples of common errors and warnings are missed interrupts, writing to flash memory, serial port buffer overrun, etc.

The errors and warning messages begin with a prefix that can be used to filter the output. This prefix is made of the component's name and its subsystem name separated by semi-columns.

The following example is a missed interrupt that has occured while GIE (General Interrupt Enable) was equal to zero. The current PC information can be used to determine where this interrupt has occurred in the program. In this particular example, this address is in a non-reentrant interrupt handler.

```
msp430:intr: Interrupt 6 received but GIE = 0 (current PC = 0x40f6)
```

3.3 Debugging using WSim

WSim understands the GDB remote protocol to drive the simulation engine. The simulator can be started using the `--mode=gdb` parameter to use this functionality. The following example starts a simulation in debug mode and waits on the default port (TCP 2159) until a connexion is made from a remote debugger that implements the same target architecture (`GDB` must be compiled as a cross-debugger for the target architecture).

```
wsim --ui --verbose=1 --mode=gdb prog.elf
```

While WSim is used in gdb mode, the simulation is stopped just after machine creation. The program (ELF file) can be preloaded using the command line or loaded through the GDB interface using standard `load` commands.

4 WSim Platform Support

WSim is composed of a set of libraries that have to be configured and linked together to build a simulator dedicated to a specific platform. This "compile time" configuration creates an efficient simulation engine tailored to a specific node mode. The main drawback to this strategy is that there is no single "WSim executable" that can be used for all platforms.

A specific platform configuration has the global options explained above and adds a set of options from the components included in the design. These options are specific to the components and can be set in the command line when running WSim. The available WSim components are listed in the following sections and are presented with their corresponding options.

4.1 Wsn430 platform example

As an example of platform description, we present `wsn430` platform that can be simulated using WSim. This platform owns the following main components

- Ti MSP430f1611 micro controller
- 1MB Flash memory STMicroelectronics STM25P80
- Chipcon CC1100 radio interface
- Silicon serial number DS2411
- External access to the MCU serial port
- External crystals with specific frequencies

This example illustrates the type of platforms that can implemented within the simulator. One of the main restriction in the platform design is the assumption that a node owns only one micro-controller. This restriction comes from the timing model used within the simulation process.

4.2 Ti ez430 USB key

The Ti USB ez430 is very simple device that is made to test and develop very small application on an MSP430f2013 micro-controller. The only peripheral included on the platform is a led connected to port 1, bit 0 (P1.0)

5 WSim Processing Units

WSim currently support the Ti MSP430 micro-controller family. Other micro-controllers will be available in a short timeframe to extand the capabilities of the simulator.

5.1 Ti MSP430 micro-controllers

5.1.1 Introduction

The MSP430 micro-controller is an integrated circuit that implements its own instruction set architecture and that has many on-chip mixed analog and digital peripherals.

Although the models implemented in Wsim cover the full MSP430 instruction set and memory mapping they do not cover the full range of available msp430 models from Texas Instruments. The following sections presents the peripherals that can be used within the simulation. Peripherals that are not fully implemented are replaced by stub devices that only report reads and writes to them.

5.1.2 MSP430 Model Current State

The following list of internal peripherals are included in the current version of the Wsim software. This list will be modified and updated has new peripheral implementation are included in the MSP430 model.

- 8bit blocks
  - SFR registers
  - FLL+ Clock
  - Basic Clock Module
  - USART with UART and SPI mode
- 16bits blocks
  - Watchdog
  - Hardware multiplier
  - Timer A3
  - Timer B3
  - Timer B7

All other peripherals are present (a read or a write access to their control registers does not produce an error during the simulation) but they only act as stubs that produce warnings during execution.

The available list of integrated peripherals can be combined to implement the digital part of the following MSP430 models:

- MSP430f135
- MSP430f449
- MSP430f1611

Although the missing peripherals models can be easily implemented in the Wsim msp430 library, they will not be done until a request is made to the authors through the WorldSens website.

5.1.3 Simulation Options

The MSP430 model adds two options to the set of available Wsim command line options. These options are

‘`--xt1=arg`’
: `XT1` crystal frequency (Hz).

‘`--xt2=arg`’
: `XT2` crystal frequency (Hz)

The default values for these options are fixed in platform configurations.

6 Available External Peripherals

WSim includes several external peripherals that can be used to design platforms. These devices are modeled using finite state machines and use the simulator interface and timing model to preserve their precise behavior during the simulation. The presented models include drivers for several embedded protocols (1-wire, UART in both asynchronous and SPI modes) that can be reused for other devices.

6.1 Led

The led device in WSim is a very simple one that can be connected directly to a general MCU IO port. The led device does not have any option. Its colour can only be set at compile time. The display is activated using the `--ui` option.

6.2 7 Segments Display

The 7 segmenst model is an 8 bits display that can be connected to general IO ports. The 7 segments display device does not have any option. The display is activated using the `--ui` option.

6.3 GDM1602a LCD display

The GDM1602a is an external LCD display that uses the KS0066u LCD controller. The controller uses only the 8 bits addressing mode (4bit is not implemented in the WSim device). The gdl1602a does not have any option.

6.4 Maxim DS2411 Silicon Serial Number

The Maxim DS2411 (`http://www.maxim-ic.com/quick_view2.cfm/qv_pk/3711`) uses the 1-wire protocol to communicate with the micro-controller using only one external digital I/O pin.

The model proposed with the WSim simulator can be parameterized at run time with the following options:

‘`--ds2411=xx:s1:s2:s3:s4:s5:s6:id`’
: The serial number is composed of 8 bytes in hexadecimal notation. xx is the serial number crc, s1 to s6 are the 6 bytes serial ID and id is the chip family (fixed to 01). s1 is the most significant byte (MSB) and s6 is the least significant byte (LSB) of the serial number.

The default serial number is fixed to 0f:07:06:05:04:03:02:01 in the simulator.

6.5 STMicro ST25P80 1MB Flash Memory Module

The STMicroelectronic M25P80 (`http://www.st.com/stonline/products/literature/ds/8495.htm`) Flash memory device is an 8Mb module driven by SPI serial communications. Standard memory operations are supported including low power modes.

‘`--flash_init=arg`’
: Flash init image (binary file)

‘`--flash_dump=arg`’
: Flash dump image (binary file)

The default behavior is to start the simulation with a factiry default Flash for which all bits are set to 1.

6.6 Pseudo Serial Port Emulation

The Ptty (Pseudo TTY) peripheral model is a special peripheral that can be used to connect a platform to a RS232 serial port. The model has the capability to open a full duplex communication stream with the operating system in order to connect the simulator to external tools such as the console interface provided in the Worldsens environment.

The `ptty` device adds a command line option to the WSim simulator. This option is used to give the name of the peripheral fifo that will be used for communication with the outside operating system world. When the option is not present on the command line the `ptty` device is not activated and the output sent from the micro-controller to its serial port is lost.

Naming conventions for operating system bi-directional fifos is operating system dependant.

'`--serial=[name]`'

'`stdout`' The special `stdout` name can be used as an option to the serial I/O device output. Input is disabled while running in this mode.

If more than one `ptty` device are present on a platform, the WSim options are extended and are names `--serial0`, `--serial1`, ...

6.6.1 Worldsens serial console

The WorldSens serial console (WConsole) is a standalone application that can be used to drive the communication with the pseudo tty model. Its main use is to provide a full duplex interface that can output the octet stream from the simulator and use the host keyboard to communicate with the simulated application. The source code of the serial console can also be used to derive and test applications that will communicate through the serial port of the node in production use such as base station monitoring applications.

The screenshot of the WSim serial console presents a simple application that produces an echo for every character sent to its serial port 1. The platform used in this test has two serial ports and the WSim serial console is attached to serial port 1. As the option for serial port 0 is not present it is disabled during simulation.

```
$./serial
Local fifo is /dev/ptmx
Remote fifo is /dev/pts/7
$./wsim --ui --serial1=/dev/pts/7 uart.elf
PTTY0: ptty not activated
PTTY1: local fifo is [/dev/pts/7]
PTTY1: remote fifo is [unknown]
```

Top half of the user interface window presents the simulation output with grayed letters and bottom half is the local keyboard echo for the host machine.

6.7 Chipcon CC1100 Packet Radio Interface

The Chipcon CC1100 peripheral is a radio interface. As for the pseudo tty interface used for serial communications, this kind of peripherals is able to communicate with external tools. The primary purpose of this model is to be connected to the WSNet radio simulator included in the Worldsens environment.

A more detailed description of the Worldsens environment and its use with both WSim and WSNet is detailed in next section.

7 Using WSim with WSNet Radio Simulator

WSim can be attached to the WSNet simulator through radio interfaces such as the Chipcon CC1100 model. This allows to build a complete simulation framework that is able to debug and analyze a complete wireless sensor network. Thus WSim and WSNet allows to **design and refine** a full distributed application from the top level specifications to the final application binary within the same framework and application description: the **real application code**.

Communication between the simulators are done using IPv4 unicast and multicast in order to allow the distribution of the simulation load across a network of workstations. A typical middle range workstation can handle several dozens of concurrent WSN node simulators so that a small set of workstations can scale to hundreds of nodes. This allows the Worldsens environment to provide a performance analysis of real world applications using cycle accurate simulation tools.

WSim, when run in the Worldsens environment has the following list of options added to its basic option list:

‘`--node-id=[id]`’
: `id` is the configuration node id in the WSNet configuration file. Default node id is set to `1`.

‘`--server-ip=[ip]`’
: Specify the WSNet server port IP address. This corresponds to the WSNet control channel. Default IP address is `127.0.0.1`.

‘`--server-port=[port]`’
: WSNet Control channel port. Default port value is set to `9998`

‘`--broadcast-ip=[ip]`’
: WSNet multicast IP address. Default address is set to `224.0.0.1`

‘`--broadcast-port=[port]`’
: WSNet multicast port. Default port is set to `9999`.

8 External Resources

- Compilers and Tools
  - Worldsens web page
    `http://worldsens.citi.insa-lyon.fr/`
  - GCC port for Ti MSP430
    `http://mspgcc.sourceforge.net/`
  - GDB, the GNU Project Debugger
    `http://www.gnu.org/software/gdb/`
  - DDD, Data Display Debugger
    `http://www.gnu.org/software/ddd/`
- Profiling and Visualization Tools
  WSim performance evaluation output files can be used with the following tools:
  - Gnuplot
    `http://www.gnuplot.info/`
  - KCachegrind - Profiling Visualization
    `http://kcachegrind.sourceforge.net/cgi-bin/show.cgi`
  - GTKWave - Electronic Waveform Viewer
    `http://www.cs.manchester.ac.uk/apt/projects/tools/gtkwave/`
- Operating Systems links
  The following operating systems are known to work on the Worldsens environment:
  - The TinyOS Project
    `http://www.tinyos.net/`
  - The Contiki OS
    `http://www.sics.se/~adam/contiki/`
  - FreeRTOS OS
    `http://www.freertos.org/`
- WSens Hardware Components
  - Texas Instrument MSP430 microcontroller
    `http://www.ti.com/msp430`
  - Chipcon radio chipsets
    `http://www.chipcon.com/`
  - Maxim DS2411 onewire serial number
    `http://www.maxim-ic.com/DS2411`
  - STM 8Mb Flash Memory
    `http://www.st.com/stonline/products/literature/ds/8495.htm`