SCoPE

User manual

Index

Overview

SCoPE tool provides the technology to perform MPSoC HW/SW co-simulation with NoC (Network on Chip). It gets results to explore the design space to choose the right processors and HW/SW partition for embedded systems. It also allows the simulation of different nodes connected throw a NoC to analize the behaviour of large systems.

Commonly, this kind of tools are based on slow ISSs. The main difference with this technique is that SCoPE gets the performance estimations at source code level. This level of abstraction allows to decrease several orders of magnitude the simulation time with a good accuracy.

Features

SCoPE is a C++ library that extends, without modifying, the standard language SystemC to perform the co-simulation. On one side, it simulates C/C++ software code based on two different operating system interfaces (POSIX and MicroC/OS). On the other hand, it co-simulates these pieces of code with hardware described on SystemC language.

Software simulation

Estimations

An engineer with this tool can simulate a specific software over a custom platform and obtain estimations of:

- Number of switches
  - Thread
  - Context
- Time
  - Running time
  - Use of CPU (%)
- Instructions
  - Executed instructions
  - Cache misses
- Energy and power
  - Core
  - Instruction cache

```
processor_0_rtos_0
        Number of thread switches: 3000
        Number of context switches: 0
        Running time: 27167551026 ns
        Use of cpu: 90.5585%
        Instructions executed: 3667562959
        Instruction cache misses: 1224440
        Core Energy: 4.62113e+09 nJ
        Core Power: 154.038 mW
        Instruction Cache Energy: 5.5303e+09 nJ
        Instruction Cache Power: 184.343
```

The modelled estimations cover not only the core but also the processor's instruction cache.

RTOS

This library models the detailed behavior of the RTOS including concurrency (among tasks in the same processor), parallelism (among tasks in different processors), scheduling and synchronization.

Although the SystemC kernel executes processes following a non-preemptive scheduling policy without priorities, SCoPE models preemption under different scheduling policies based on priorities.

Operating system interfaces

SCoPE integrates a POSIX based API that allows the execution of a large number of software applications that follows this standard.

POSIX is the main operating system interface nowadays, but it is not the unique. Thus, SCoPE has been improved to support extensions for other types of interfaces. An example is the integration with the MicroC/OS interface. This is a demonstration of the scalability of the tool, in terms of software support.

Drivers

The design of embedded systems require not only software handling but also hardware communication. For this reason SCoPE includes a set of more of a hundred of driver facilities to implement this communication. One of the most extended operative systems in this sector is Linux, thus this driver facilities are based on the Linux kernel version 2.6.

Furthermore, SCoPE is able to simulate the loading of kernel modules and the handling of hardware interruptions and its correspondent scheduling.

Hardware simulation

SystemC is the language used for the modelling of the hardware platform due to the easiness of implementation (C++ extension) and its simulation kernel. For the purpose of simulate different platforms SCoPE incorporates some generic hardware modules.

- Bus based on TLM2 used for the communication with peripherals and the transmission of hardware interruptions.
- DMA for coping large amount of data.
- Simple memory for the simulation of cache and DMA traffic.
- Hardware interface for an easy custom hardware connection.
- Network interface that work as a net card for the NoC.
- External network simulator to implement the NoC connected to SCoPE.

System simulation

- Multi-computation: One of the advantages of this tool is the possibility of interconnection among independent nodes and simulate the interaction among them.
- Modular structure: Each RTOS component is an independent object that does not share any data with the others. Furthermore, each process is isolated from the rest of the system, thus, a process with global variables can be replicated in many nodes without data collision problems. That is, each process has a separated memory space.

Configuration

To make easy for users the configuration of the processor models, SCoPE integrates a graphical user interface that provides a structured framework to set the needed parameters.

SCoPE incorporates a simple template for platform configuration that covers the main use cases.

License

Requirements

- Linux distribution: tested on Fedora Core 7
- SystemC 2.2
- bison
- flex
- qt4 and qt4-devel: for the graphic interface
- qwt and qwt-devel: for demonstration with the graphic interface

Installation

Note: For this manual, the bash shell is supposed to be used. Change the commands for the right ones on your shell.

The first part of the installation is to set the environment variables:

export SCOPE_HOME=/SCoPE_path

export SYSTEMC=/SystemC_path

export CXX=g++

An especial variable is SCOPE_CPU. This variable represents the processor in which the code will be executed. The default processor is the arm926t, that we have characterized.

export SCOPE_CPU=arm926t

They are needed for application compilations, thus it is advisable to include it in the .bashrc file for automatic loading.

Some little configurations for compilation are needed. The script *configure* performs this changes. Next, for the tool compilation, it is only used the make command.

Command	**Explanation**
make	Prints the Makefile help
make all	Compiles all tools
make test	Compiles the tests

The users without QT4 library will get an error at the end of the global compilation with *make all*. Every part of the tool can be compiled separately.

Application simulation

Basic structure

sc_main.cpp

```
#include <iostream>
using namespace std;

#include "sc_scope.h" // platform headers

// user main functions
void * user_code_1(void *);
void * user_code_2(void *);

/***************/
/*** sc_main ***/
/***************/
#define NUM_NODES 2 // number of nodes
int sc_main(int argc, char **argv) {
      UC_rtos_class *rtos;
      vector<UC_rtos_class *> rtos_list;
      UC_HAL_hw_class *hal;

      // NoC creation
      UC_NoC_Interface *simulator = new UC_NoC_Interface("simulator", 1, true);
      int numbers[2][1][1];
      numbers[0][0][0] = 0;
      numbers[1][0][0] = 1;
      simulator->set_structure((int ***)numbers,2,1,1);

      for (int node = 0; node < NUM_NODES; node++) {
            // each node is controlled by a single RTOS:
            rtos = new UC_rtos_class(1);  // the rtos with the number of cpus
            // register new processes
            rtos->get_processor(0)->new_process(user_code_1, NULL);
            rtos->get_processor(0)->new_process(user_code_2, NULL);
            rtos_list.push_back(rtos);

            // creates the Hardware Abstraction Layer:
            hal = new UC_HAL_hw_class(sc_gen_unique_name("HAL"), 1, 2, 100);

            // Processor to HAL binding
            rtos->get_processor(0)->m_port(*hal);

            // HAL's irq to processor binding
            (*hal->get_irq_port(0))(*rtos->get_processor(0));

            // HW creation and binding
            UC_hw_NoC *eth = new UC_hw_NoC(sc_gen_unique_name("eth"),
                        NET_IRQ/*irq*/, 10/*resp.time(ns)*/, simulator);
            (*hal->get_hw_port(0))(*eth->get_target_port());
            (*eth->get_irq_port())(*hal);
            hal->memory_map->add_peripheral(0/*id*/, NET_START,
                        NET_START + 0x3F, 0/*port*/, "eth");

            UC_hw_memory *mem = new UC_hw_memory(sc_gen_unique_name("mem"),
                        10/*irq*/, 100/*resp.time(ns)*/);
            (*hal->get_hw_port(1))(*mem->get_target_port());
```

```
            (*mem->get_irq_port())(*hal);
            hal->memory_map->add_peripheral(1/*id*/, RAM_START,
                        RAM_START + RAM_SIZE - 1, 1/*port*/, "mem");

            insmod(rtos, tun_init); // loads the tun module driver
      }

      sc_start(500, SC_MS); // simulation time

      // Simulation finished
      cout << "Main finish" << endl;
      cout << "Simulated time: " << sc_time_stamp() << endl;

      simulator->end_simulation(); // deletes the NoC simulator for statistics
      cout << endl;

      // deletes all RTOS for statistics
      for (int i = 0; i < NUM_NODES; i++) {
            delete rtos_list[i];
      }

      return 0;
}
```

Detailed description

```
#include <iostream>
using namespace std;

#include "sc_scope.h" // platform headers
```

The system header for printing messages and a SCoPE header. *sc_scope.h* is a header that includes all SCoPE platform headers needed for the configuration of this file.

```
// user main functions
void * user_code_1(void *);
void * user_code_2(void *);
```

Declaration of user main functions. Each of these functions is a main function of a process that will be executed when the system starts. The current format of this kind of functions is POSIX thread like.

```
#define NUM_NODES 2 // number of nodes
```

A define to easily change the number of nodes of the system.

```
int sc_main(int argc, char **argv) {
      UC_rtos_class *rtos;
```

```
vector<UC_rtos_class *> rtos_list;
UC_HAL_hw_class *hal;
```

sc_main is the main function of the SystemC framework. The platform description is defined here and starts the simulation.

The next three lines declares some variables needed for building the platform, following described.

```
// NoC creation
UC_NoC_Interface *simulator = new UC_NoC_Interface("simulator", 1, true);
int numbers[2][1][1];
numbers[0][0][0] = 0;
numbers[1][0][0] = 1;
simulator->set_structure((int ***)numbers,2,1,1);
```

This example convers the main areas of platform configuration, thus some hardware devices are connected to the system. The first one is the network on chip.

At this point, the creation and configuration of the NoC hardware is performed. The NoC needs the *numbers* array, which sets the nodes structure up, to configure the net.

```
for (int node = 0; node < NUM_NODES; node++) {
      // each node is controlled by a single RTOS:
      rtos = new UC_rtos_class(1);  // the rtos with the number of cpus
      // register new processes
      rtos->get_processor(0)->new_process(user_code_1, NULL);
      rtos->get_processor(0)->new_process(user_code_2, NULL);
      rtos_list.push_back(rtos);
```

The current code create two exactly equal nodes, so a *for* loop is executed for this purpose.

Each node includes a RTOS that manages one processor. Moreover, the two declared user functions are connected to this processor to be executed when the system boots.

The last line inserts the RTOS instance in a list that will be used later.

In SCoPE v1.0.0, the unique way to create a process is to create it at startup. The execution of new processes during the simulation is not allowed. This includes the use of the POSIX *fork* function, not implemented.

```
      // creates the Hardware Abstraction Layer:
      hal = new UC_HAL_hw_class(sc_gen_unique_name("HAL"), 1, 2, 100);

      // Processor to HAL binding
      rtos->get_processor(0)->m_port(*hal);

      // HAL's irq to processor binding
      (*hal->get_irq_port(0))(*rtos->get_processor(0));
```

Once the processors are created, the next step is to connect them to a bus. For this purpose, the HAL module has a bus API, completely transparent for the user.

For the HAL module, it is needed to configure the number of masters and slaves and the bus bandwidth in bytes/nanosecond.

After the creation of the bus, it must be connected with the processors. First, from the actual *rtos* we get the processor '0' with *rtos->get_processor(0)*. From this structure, the *m_port* variable is needed to perform the connection, *rtos->get_processor(0)->m_port(*hal)*.

From the other side, the interruption ports of the bus must be connected to its correspondent processor.

The two last lines of code must be replicated until every processor and bus interruption port of the system are interconnected.

```
            // HW creation and binding
            UC_hw_NoC *eth = new UC_hw_NoC(sc_gen_unique_name("eth"),
                        NET_IRQ/*irq*/, 10/*resp.time(ns)*/, simulator);
            (*hal->get_hw_port(0))(*eth->get_target_port());
            (*eth->get_irq_port())(*hal);
            hal->memory_map->add_peripheral(0/*id*/, NET_START,
                        NET_START + 0x3F, 0/*port*/, "eth");

            UC_hw_memory *mem = new UC_hw_memory(sc_gen_unique_name("mem"),
                        10/*irq*/, 100/*resp.time(ns)*/);
            (*hal->get_hw_port(1))(*mem->get_target_port());
            (*mem->get_irq_port())(*hal);
            hal->memory_map->add_peripheral(1/*id*/, RAM_START,
                        RAM_START + RAM_SIZE - 1, 1/*port*/, "mem");
```

The last components to connect to the platform are the HW modules. In this example, there are two different devices, a NoC interface and a memory.

The creation and the connection of both elements are exactly the same but the constructor. The NoC interface needs a name, the interruption number, the delay time and the simulator object created at the beginning of this function. After the creation, the device has to be connected to the bus. As the processor, we need to connect the bus port to the HW

*(*hal->get_hw_port(0))(*eth->get_target_port())*

and the irq HW port to the bus

*(*eth->get_irq_port())(*hal)*.

The last function to execute is to include the peripheral address range in the bus's memory map.

The main difference between this device connections that has to be taken into account is that the HW and bus ports must be different. In this example, the *eth* device is the HW number '0' and the *mem* device is the '1'.

```
            insmod(rtos, tun_init); // loads the tun module driver
      }
```

insmod is the function used to load driver modules, but it is not the same interface that in Linux kernel. This new function needs the RTOS object to know the operative system in which load the module and the driver init function. This init function is loaded at the simulation startup.

In this case, the *tun_init* function initialize the connection with the NoC interface.

```
	sc_start(500, SC_MS); // simulation time

	// Simulation finished
	cout << "Main finish" << endl;
	cout << "Simulated time: " << sc_time_stamp() << endl;
```

The configuration of the platform has been finished, thus we only have to start the simulation.

The time of simulation is set up with the SystemC *sc_start* function.

```
	simulator->end_simulation(); // deletes the NoC simulator for statistics
	cout << endl;

	// deletes all RTOS for statistics
	for (int i = 0; i < NUM_NODES; i++) {
		delete rtos_list[i];
	}

	return 0;
}
```

To finish the simulation we need to kill the NoC simulator with the function *end_simulation*. In the same way, the RTOS variables must be deleted.

These destructions are done to provoke the simulation statistics.

Global variables

At this moment, the result of the compilation of all user software programs is an unique executable file. In consequence, every global variable is shared by all software code in the simulation. If the process will not be executed several times, in the same or different RTOS, you only have to create the global variables with different names. However, if the process is going to be executed two or more times, its global variables must fulfill two extra requirements:

- To have the modifier *__thread*. (e.g. __thread int foo;)
- To be POD (Plain Old Data) types
  - Accepted: basic types like int, float, double, pointers...
  - Declined: arrays, structures, classes...

These requirements ensure the correct operation of the simulation and the possibility of executing several times the same software process on the simulated platform.

Makefile template

Makefile

```
# Makefile for bubble sort example

# SCoPE variables
export SCOPE_CPU = arm926t
CXX = g++
SW_CXX = $(SCOPE_HOME)/compiler/sw_g++

# Compiler flags
DEBUG = -g
CFLAGS = $(DEBUG) -c

# Executable
OUT = bubble.x

# Libraries and their directories
LIB_DIR = -L. -L$(SCOPE_HOME)/scope -L$(SYSTEMC)/lib-linux \
          -L$(SCOPE_HOME)/tinyxml

LIB = -lscope -ltinyxml -lsystemc -lpthread -lrt

# Include directories
SCOPE_INC_DIRS =  $(SCOPE_HOME)/scope                      \
                  $(SCOPE_HOME)/scope/hal                  \
                  $(SCOPE_HOME)/scope/rtos/api/posix       \
                  $(SCOPE_HOME)/scope/rtos/api/ucos        \
                  $(SCOPE_HOME)/scope/rtos/drivers         \
                  $(SCOPE_HOME)/scope/rtos/kernel          \
                  $(SCOPE_HOME)/scope/rtos/low_level       \
                  $(SCOPE_HOME)/scope/rtos/qt_interface    \
                  $(SCOPE_HOME)/scope/rtos/utils

INC_DIRS =  \
                  $(SCOPE_HOME)/scope/sicosys/SC_Simul     \
                  $(SCOPE_HOME)/TLM2/tlm                   \
                  $(SYSTEMC)/include                       \
                  $(SCOPE_INC_DIRS)


INC_DIR = $(addprefix -I,$(INC_DIRS))

# Objects
OBJS =  bubble.o

.PHONY: $(OUT)

# Link
$(OUT): sc_main.o $(OBJS)
      $(CXX) $(LIB_DIR) $^ -o $@ $(LIB)

# Parse and compile software application files
.cpp.o:
      $(SW_CXX) $(CFLAGS) $(INC_DIR) $< -o $@

# Compile no SW files with standard g++
sc_main.o : sc_main.cpp
      $(CXX) $(CFLAGS) $(INC_DIR) $< -o $@
```

```
# Clean
.PHONY: clean
.PHONY: distclean
clean:
	rm -rf $(OBJS) sc_main.o

distclean: clean
	rm -rf $(OUT)

# File dependencies
bubble.o: bubble.cpp
```

The most important points for making a Makefile for SCoPE are:

- SCOPE_CPU: Environment variable for getting the target processor.
- CXX: The project compiler.
- SW_CXX: The SCoPE compiler for SW code.
- Compile only the SW with SW_CXX and the rest of the application and the linking with CXX.
- Insert all include and library directories and libraries shown above.

Operating system interfaces

POSIX

This is a list of POSIX functions included in SCoPE.

```
int pthread_create(pthread_t *__thread_arg, const pthread_attr_t *__attr, void
*(*__start_routine) (void *), void *__arg, const char *func_name);

int pthread_attr_init(pthread_attr_t *attr);
int pthread_attr_destroy(pthread_attr_t *attr);
int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate);
int pthread_attr_getdetachstate(const pthread_attr_t *attr, int *detachstate);
int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy);
int pthread_attr_getschedpolicy(const pthread_attr_t *attr, int *policy);
int pthread_attr_setschedparam(pthread_attr_t *attr,const struct sched_param
*param);
int pthread_attr_getschedparam(const pthread_attr_t *attr,struct sched_param
*param);
int pthread_attr_setinheritsched(pthread_attr_t *attr, int inherit);
int pthread_attr_getinheritsched(const pthread_attr_t *attr, int *inherit);
int pthread_attr_setscope(pthread_attr_t *attr, int scope);
int pthread_attr_getscope(const pthread_attr_t *attr, int *scope);
int pthread_cancel(pthread_t thread);
void pthread_cleanup_push(void (*routine)(void*), void *arg);
void pthread_cleanup_pop (int execute);
int pthread_cond_init(pthread_cond_t *cond, pthread_condattr_t *cond_attr);
int pthread_cond_destroy(pthread_cond_t *cond);
int pthread_cond_signal(pthread_cond_t *cond);
int pthread_cond_broadcast(pthread_cond_t *cond);
int pthread_cond_timedwait(pthread_cond_t *cond, pthread_mutex_t *mutex,const
struct timespec *abstime);
int pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex);
int pthread_condattr_init(pthread_condattr_t *attr);
int pthread_condattr_destroy(pthread_condattr_t *attr);
int pthread_condattr_getpshared(const pthread_condattr_t *attr, int *pshared);
int pthread_condattr_setpshared(pthread_condattr_t *attr, int pshared);
int pthread_condattr_getclock(const pthread_condattr_t *attr,clockid_t
*clock_id);
int pthread_condattr_setclock(pthread_condattr_t *attr,clockid_t clock_id);
int pthread_detach (pthread_t __th);
int pthread_equal (pthread_t __thread1, pthread_t __thread2);
void pthread_exit (void *__retval);
int pthread_join (pthread_t __th, void **__thread_return);
int pthread_setschedparam(pthread_t target_thread, int policy, const struct
sched_param *param);
int pthread_getschedparam(pthread_t target_thread, int *policy, struct
sched_param *param);
int pthread_setschedprio(pthread_t target_thread, int priority);
int pthread_getschedprio(pthread_t target_thread, int *priority);
int pthread_key_create(pthread_key_t *key, void (*destructor)(void*));
int pthread_key_delete(pthread_key_t key);
int pthread_setspecific(pthread_key_t key, void *value);
void *pthread_getspecific(pthread_key_t key);
int pthread_getcpuclockid(pthread_t thread_id, clockid_t *clock_id);
int pthread_mutex_init(pthread_mutex_t *mutex,const pthread_mutexattr_t
*mutexattr);
```

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
int pthread_mutex_trylock(pthread_mutex_t *mutex);
int pthread_mutex_destroy(pthread_mutex_t *mutex);
int pthread_mutex_getprioceiling(const pthread_mutex_t *mutex, int *restrict);
int pthread_mutex_setprioceiling(const pthread_mutex_t *mutex, int prioceiling,
int *restrict);
int pthread_mutex_timedlock(pthread_mutex_t *mutex,const struct timespec*
timelock);
int pthread_mutexattr_init(pthread_mutexattr_t *attr);
int pthread_mutexattr_destroy(pthread_mutexattr_t *attr);
int pthread_mutexattr_getprioceiling(const pthread_mutexattr_t *attr, int*
restrict);
int pthread_mutexattr_getprotocol(const pthread_mutexattr_t *attr, int*
restrict);
int pthread_mutexattr_getpshared(const pthread_mutexattr_t *attr, int*
restrict);
int pthread_mutexattr_gettype(const pthread_mutexattr_t *attr, int* restrict);
int pthread_mutexattr_setprioceiling( pthread_mutexattr_t *attr, int value);
int pthread_mutexattr_setprotocol( pthread_mutexattr_t *attr, int value);
int pthread_mutexattr_setpshared( pthread_mutexattr_t *attr, int value);
int pthread_mutexattr_settype( pthread_mutexattr_t *attr, int value);
int pthread_once(pthread_once_t* once_control, void (*init_routine)(void));
pthread_t pthread_self();
int pthread_setcancelstate(int state, int *oldstate);
int pthread_setcanceltype(int type, int *oldtype);
void pthread_testcancel();

int pthread_rwlockattr_init(pthread_rwlockattr_t *attr);
int pthread_rwlockattr_destroy(pthread_rwlockattr_t *attr);
int pthread_rwlockattr_getpshared(const pthread_rwlockattr_t *attr, int *num);
int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *attr, int num);
int pthread_rwlock_init(num_pthread_rwlock_t *id_lock, const
pthread_rwlockattr_t *attr);
int pthread_rwlock_destroy(num_pthread_rwlock_t *id_lock);
int pthread_rwlock_rdlock(num_pthread_rwlock_t *id_lock);
int pthread_rwlock_tryrdlock(num_pthread_rwlock_t *id_lock);
int pthread_rwlock_timedrdlock(num_pthread_rwlock_t *id_lock, const struct
timespec *abs_time);
int pthread_rwlock_wrlock(num_pthread_rwlock_t *id_lock);
int pthread_rwlock_trywrlock(num_pthread_rwlock_t *id_lock);
int pthread_rwlock_timedwrlock(num_pthread_rwlock_t *id_lock, const struct
timespec *abs_time);
int pthread_rwlock_unlock(num_pthread_rwlock_t *id_lock);

int pthread_spin_lock(pthread_spinlock_t *lock);
int pthread_spin_trylock(pthread_spinlock_t *lock);
int pthread_spin_destroy(pthread_spinlock_t *lock);
int pthread_spin_init(pthread_spinlock_t *lock, int pshared);
int pthread_spin_unlock(pthread_spinlock_t *lock);

mqd_t mq_open(const char *, int);
mqd_t mq_open(const char *, int, int, mq_attr *);
int mq_close(mqd_t);
int mq_notify(mqd_t, const struct sigevent *);
int mq_getattr(mqd_t, struct mq_attr *);
int mq_setattr(mqd_t, const struct mq_attr *, struct mq_attr *);
ssize_t mq_receive(mqd_t, char *, size_t, void *);
ssize_t mq_timedreceive(mqd_t, char *, size_t,void *, const struct timespec *);
int mq_send(mqd_t, const char *, size_t, int );
int mq_timedsend(mqd_t, const char *, size_t, int , const struct timespec *);
```

```
int mq_timedsend_front(mqd_t mqd_id, const char *buffer, size_t size, int
msg_prio, const struct timespec *abs_time);
int mq_unlink(const char *);

int sched_setparam(int pid, const struct sched_param *param);
int sched_getparam(int pid, struct sched_param *param);
int sched_getscheduler(int pid);
int sched_setscheduler(int pid, int policy, const struct sched_param *param);
int sched_yield();
int sched_get_priority_max(int policy);
int sched_get_priority_min(int policy);
int sched_rr_get_interval(int pid, struct timespec *tp);
int sched_setaffinity(int pid, unsigned int len, unsigned long *mask);
int sched_getaffinity(int pid, unsigned int len, unsigned long *mask);
struct sched_param * schedparam_init();

int sem_close(sem_t *);
int sem_destroy(sem_t *);
int sem_getvalue(sem_t *, int *);
int sem_init(sem_t *, int, int);
sem_t *sem_open(const char *, int);
sem_t *sem_open(const char *,int , int, int );
int sem_post(sem_t *);
int sem_timedwait(sem_t *, const struct timespec *);
int sem_trywait(sem_t *);
int sem_unlink(const char *);
int sem_wait(sem_t *);

int shm_open(const char *name, int oflag, mode_t mode);
int ftruncate(int fildes, off_t length);
int shm_unlink(const char *name);
void *mmap(void *addr, size_t len, int prot, int flags, int fildes, off_t off);
int munmap(void *addr, size_t len);

int kill(pid_t, int);
int killpg(pid_t, int);
int pthread_kill(pthread_t, int);
int pthread_sigmask(int, const sigset_t *, sigset_t *);
int raise(int);
int sigaction(int, const struct sigaction *restrict, struct sigaction
*restrict);
int sigaddset(sigset_t *, int);
int sigdelset(sigset_t *, int);
int sigemptyset(sigset_t *);
int sigfillset(sigset_t *);
int sigismember(const sigset_t *, int);
sighandler_t signal(int sig, sighandler_t handler);
int sigpending(sigset_t *);
int sigprocmask(int, const sigset_t *restrict, sigset_t *restrict);
int sigqueue(pid_t, int, const union sigval);
int sigsuspend(const sigset_t *);
int sigtimedwait(const sigset_t *restrict, siginfo_t *restrict, const struct
timespec *restrict);
int sigwait(const sigset_t *, int *restrict);
int sigwaitinfo(const sigset_t *restrict, siginfo_t *restrict);
void (*bsd_signal(int sig, void (*handler)(int)))(int);
void (*sigset(int sig, void (*disp)(int)))(int);
int sighold(int);
int sigignore(int);
int sigpause(int);
int sigrelse(int);
```

```
int siginterrupt(int, int);

clock_t clock();
time_t time(time_t *);
double difftime(time_t time1, time_t time0);
int nanosleep(const struct timespec *nseconds, struct timespec *rem);
int clock_getcpuclockid(int pid, clockid_t *clockid);
int clock_getres(clockid_t clock_id, struct timespec *rqtp);
int clock_gettime(clockid_t clock_id, struct timespec *rqtp);
int clock_settime(clockid_t clock_id, const struct timespec *rqtp);
int clock_nanosleep(clockid_t clock_id, int flags, const struct timespec *rqtp,
struct timespec *rmtp);
int timer_create(clockid_t clockid, struct sigevent *s_ev, timer_t *timer);
int timer_delete(timer_t timer);
int timer_gettime(timer_t time, struct itimerspec * spec);
int timer_getoverrun(timer_t timer);
int timer_settime(timer_t timer, int flags, const struct itimerspec *i_spec,
struct itimerspec *o_spec);

int mkstemp(char *tmp_char);
void abort();
int system(const char* buf);

pid_t getpid();
gid_t getgid();
pid_t getppid();
pid_t getpgid(pid_t pid);
int setpgid(pid_t pid, pid_t pgid);
pid_t getpgrp();
int setpgrp();
int setuid(uid_t uid);
int getuid();
gid_t getegid();
gid_t setegid(gid_t ee);
uid_t geteuid();
uid_t seteuid(uid_t ee);
long gethostid();
int open(const char *camino, int flags);
int open(const char *camino, int flags, mode_t modo);
int creat(const char *camino, mode_t modo);
int pipe(int filedes[2]);
int close(int fd);
int write(int fd, const void *buf, size_t len);
int write(int fd, const void *buf, size_t len, off_t offset);
int write_base(int fd, void *buf_i, size_t len);
int read(int fd, void *buf, size_t len);
int pread(int fd, void *buf, size_t count, off_t offset);
int read_base(int fd, void *buf_i, size_t len);
int read_base_std(int fd,void* buf_i,size_t len);
int pread_base(int fd, void *buf_i, size_t len, off_t offset);
int pwrite(int fd, const void *buf, size_t count, off_t offset);
int pwrite_base(int fd, void* buf_i, size_t len, off_t offset);
int write_base_std(file *filedes, void *buf_i, size_t len);
int nice(int inc);
int usleep(unsigned long int useconds);
unsigned int sleep(unsigned int seconds);
unsigned int alarm(unsigned int seconds);
int pause(void);
void _exit(int status);
void exit(int status);
```

uCOS

This is a list of MicroC/OS II functions included in SCoPE. The version used for this interface is 2.85.

```
OS_ENTER_CRITICAL()
OS_EXIT_CRITICAL()
void   OSInit(void);
void   OSIntEnter(void);
void   OSIntExit(void);
void   OSSchedLock(void);
void   OSSchedUnlock(void);
void   OSStart(void);
void   OSStatInit(void);
INT16U OSVersion(void);

bool uc_FlagBlock(OS_FLAG_GRP *pgrp, OS_FLAGS *flags, INT8U wait_type, INT16U
timeout);
void uc_FlagUnblock(OS_FLAG_GRP *pgrp);
OS_FLAGS OSFlagAccept(OS_FLAG_GRP *pgrp, OS_FLAGS flags, INT8U wait_type, INT8U
*perr);
OS_FLAG_GRP *OSFlagCreate(OS_FLAGS flags, INT8U *perr);
OS_FLAG_GRP *OSFlagDel(OS_FLAG_GRP *pgrp, INT8U opt, INT8U *perr);
INT8U OSFlagNameGet(OS_FLAG_GRP *pgrp, INT8U *pname, INT8U *perr);
void  OSFlagNameSet(OS_FLAG_GRP *pgrp, INT8U *pname, INT8U *perr);
OS_FLAGS OSFlagPend(OS_FLAG_GRP *pgrp, OS_FLAGS flags, INT8U wait_type, INT16U
timeout, INT8U *perr);
OS_FLAGS OSFlagPendGetFlagsRdy(void);
OS_FLAGS OSFlagPost(OS_FLAG_GRP *pgrp, OS_FLAGS flags, INT8U opt, INT8U *perr);
OS_FLAGS OSFlagQuery(OS_FLAG_GRP *pgrp, INT8U *perr);

void *OSMboxAccept(OS_EVENT *pevent);
OS_EVENT *OSMboxCreate(void *pmsg);
OS_EVENT *OSMboxDel(OS_EVENT *pevent, INT8U opt, INT8U *perr);
void *OSMboxPend(OS_EVENT *pevent, INT16U timeout, INT8U *perr);
INT8U OSMboxPost(OS_EVENT *pevent, void *pmsg);
INT8U OSMboxQuery(OS_EVENT *pevent, OS_MBOX_DATA *p_mbox_data);

OS_MEM *OSMemCreate(void *addr, INT32U nblks, INT32U blksize, INT8U *perr);
void   *OSMemGet(OS_MEM *pmem, INT8U *perr);
INT8U   OSMemNameGet(OS_MEM *pmem, INT8U *pname, INT8U *perr);
void    OSMemNameSet(OS_MEM *pmem, INT8U *pname, INT8U *perr);
INT8U   OSMemPut(OS_MEM *pmem, void *pblk);
INT8U   OSMemQuery(OS_MEM *pmem, OS_MEM_DATA *p_mem_data);

BOOLEAN   OSMutexAccept(OS_EVENT *pevent, INT8U *perr);
OS_EVENT *OSMutexCreate(INT8U prio, INT8U *perr);
OS_EVENT *OSMutexDel(OS_EVENT *pevent, INT8U opt, INT8U *perr);
void  OSMutexPend(OS_EVENT *pevent, INT16U timeout, INT8U *perr);
INT8U OSMutexPost(OS_EVENT *pevent);
INT8U OSMutexQuery(OS_EVENT *pevent, OS_MUTEX_DATA *p_mutex_data);

void *OSQAccept(OS_EVENT *pevent, INT8U *perr);
OS_EVENT *OSQCreate(void **start, INT16U size);
OS_EVENT *OSQDel(OS_EVENT *pevent, INT8U opt, INT8U *perr);
INT8U OSQFlush(OS_EVENT *pevent);
void *OSQPend(OS_EVENT *pevent, INT16U timeout, INT8U *perr);
INT8U OSQPost(OS_EVENT *pevent, void *pmsg);
INT8U OSQPostFront(OS_EVENT *pevent, void *pmsg);
```

```
INT8U OSQQuery(OS_EVENT *pevent, OS_Q_DATA *p_q_data);

INT16U OSSemAccept(OS_EVENT *pevent);
OS_EVENT *OSSemCreate(INT16U cnt);
OS_EVENT *OSSemDel(OS_EVENT *pevent, INT8U opt, INT8U *perr);
void  OSSemPend(OS_EVENT *pevent, INT16U timeout, INT8U *perr);
INT8U OSSemPost(OS_EVENT *pevent);
INT8U OSSemQuery(OS_EVENT *pevent, OS_SEM_DATA *p_sem_data);
void  OSSemSet(OS_EVENT *pevent, INT16U cnt, INT8U *perr);

INT8U OSTaskChangePrio(INT8U oldprio, INT8U newprio);
INT8U OSTaskCreate(void (*task)(void *p_arg), void *p_arg, OS_STK *ptos, INT8U
prio);
INT8U OSTaskCreateExt(void (*task)(void *p_arg), void *p_arg, OS_STK *ptos,
INT8U prio, INT16U id, OS_STK *pbos, INT32U stk_size, void *pext, INT16U opt);
INT8U OSTaskDel(INT8U prio);
INT8U OSTaskDelReq(INT8U prio);
INT8U OSTaskNameGet(INT8U prio, INT8U *pname, INT8U *perr);
void  OSTaskNameSet(INT8U prio, INT8U *pname, INT8U *perr);
INT8U OSTaskResume(INT8U prio);
INT8U OSTaskSuspend(INT8U prio);

void OSTimeDly(INT16U ticks);
INT8U OSTimeDlyHMSM(INT8U hours, INT8U minutes, INT8U seconds, INT16U milli);
INT8U OSTimeDlyResume(INT8U prio);
INT32U OSTimeGet(void);
void OSTimeSet(INT32U ticks);

OS_TMR *OSTmrCreate(INT32U dly, INT32U period, INT8U opt, OS_TMR_CALLBACK
callback, void *callback_arg, INT8U *pname, INT8U *perr);
BOOLEAN OSTmrDel(OS_TMR *ptmr, INT8U *perr);
INT8U OSTmrNameGet(OS_TMR *ptmr, INT8U *pdest, INT8U *perr);
INT32U OSTmrRemainGet(OS_TMR *ptmr, INT8U *perr);
INT8U OSTmrStateGet(OS_TMR *ptmr, INT8U *perr);
BOOLEAN OSTmrStart(OS_TMR *ptmr, INT8U *perr);
BOOLEAN OSTmrStop(OS_TMR *ptmr, INT8U opt, void *callback_arg, INT8U *perr);
```

Hardware creation

The creation of a hardware requires to inherit from a SCoPE class, uc_generic_bus_if. This class contains the interface to connect any module to the bus. The new hardware has to implement the functions read and write to be accessible from the software.

This is a simple example of a register of 4 bytes, that may be the bus interface of a complex HW.

uc_hw_memory.h

```
#ifndef _UC_HW_MEMORY_H
#define _UC_HW_MEMORY_H

#include "uc_generic_bus_if.h"

// The new hardware inherits from uc_generic_bus_if
class UC_hw_memory : public uc_generic_bus_if
{
      // Internal variables
      sc_time response_time;
      int value;
public :
      //! Constructor
      /*!
       *  \param module_name module's name
       *  \param irq_num HW interruption
       *  \param ret HW delay time
       */
      UC_hw_memory(sc_module_name module_name,int irq_num, int ret);


      //! Write function
      /*!
       *  Inherited from uc_generic_bus_if
       *  \param data pointer to data to be written
       *  \param size bytes of data to be written
       *  \param addr HW's address to write to
       *  \param tlm_id writer processor id
       *  \return size of the written data
       */
      int write(DATA data,int size,ADDRESS addr,int tlm_id);
```

```
      //! Read function
      /*!
       *  Inherited from uc_generic_bus_if
       *  \param data double pointer to memory for writing in
       *  \param size bytes of data to be read
       *  \param addr HW's address to read from
       *  \param tlm_id reader processor id
       *  \return size of the read data
       */
      int read(DATA *data,int size,ADDRESS addr,int tlm_id);
};

#endif
```

uc_hw_memory.cpp

```
#include "uc_hw_memory.h"

UC_hw_memory::UC_hw_memory(sc_module_name module_name,int irq_num, int ret):
      uc_generic_bus_if(module_name, irq_num), response_time(ret,SC_NS)
{
      value = 0;
}

int UC_hw_memory::write(DATA data, int size, ADDRESS addr, int tlm_id) {
      wait(response_time * size / 4);
      if (size == 4) {
            memcpy(&value, data, 4);
      }
      return size;
}

int UC_hw_memory::read(DATA *data, int size, ADDRESS addr, int tlm_id) {
      wait(response_time * size / 4);
      if (size == 4) {
            memcpy(*data, &value, 4);
      }
      return size;
}
```

Appendix

GPL v3 license

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we

have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an

"aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation

into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For

purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not

excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY

GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS