****

**Student ID: C00131026**

**Student Name: Guanting Su**

**Supervisor: Joseph Kehoe**

**Date of submission: 8.4.2011**

**User Manual**

**<STM Lua>**

**(CW228)**

Content

[1. Introduction 1](#_Toc290277741)

[2. System Requirement 1](#_Toc290277742)

[3. Install 1](#_Toc290277743)

[4. Simple Sample 1](#_Toc290277744)

[5. Function Description 2](#_Toc290277745)

[i. CreateTransacton 2](#_Toc290277746)

[ii. Add 3](#_Toc290277747)

[iii. Data Process 3](#_Toc290277748)

[iv. Commit 3](#_Toc290277749)

[v. Callfunc 3](#_Toc290277750)

[vi. CreateT 4](#_Toc290277751)

# Introduction

This application is a library of Lua. It helps users achieve parallel program by STM (Software Transaction Memory) control. But this library only work for Lua code. Therefore my product can help you create multithread program. Following is introducing about which system my product will work well, how to install, how to use it and the function description.

# System Requirement

One of the most important problems the user should pay more attention is the user’s running environment. So far the application only can run in the Windows system. And not support older version of Windows, for example Windows NT, 95 and 98. Also require setup Lua environment. Every computer which wants to run this software should have this environment.

# Install

Install my application; you just need copy my application files into the same folder as your Lua code. My application includes stm.lua, stmLock.dll and some test codes.

# Simple Sample

If you want to achieve STM control by my application, please type

Require “stm”

In your Lua code. And then you can define function to create transaction and run transaction.

function op1()

 local done=false

 --create transaction also create table y

 y=createTransaction()

 --add value and variables into table y

y.add("a",4)

 y.add("b",5)

 y.add("c",7)

 --stm begin

 while done==false do

 y.a=y.a+y.b --y.a=4+5

 y.b=y.c-y.b

 y.c=y.c+1

--transaction commit, if fail return false keep looping and if successful, return true, end transaction

 done=y.commit()

 end

end

After that you need run the function, get the result. Application support

--run function without thread and the data type of function\_name is string

callfunc(function\_name)

Example:

callfunc(“op1”)

--run function with thread and the data type of function\_name is string

createT(function\_name)

Example:

createT(“op1”)

At last you can call function printGtable() to print out the all elements in the table and their value.

# Function Description

## CreateTransacton

If you want to use the application to achieve parallel control, you need create transaction first. The application support function createTransaction(). In this function, it will create a table x, in this table there are functions to achieve STM control.

For example:

y=createTransaction()

Now you get table as table x. Therefore you can call some functions in table x.

## Add

After create transaction, the application help you create an empty table, so you can add variables whatever you want to add and values into table.

Example:

At the beginning the table y={}

y.add("a",4)

y.add("b",5)

y.add("c",7)

After that the table

 y={a={value=4,version=1},b={ value=4,version=1},

c={ value=4,version=1}}

The first parameter must be string and the second one must be number. And the different transactions don’t share variables. But they share variables that are in global table.

## Data Process

So far you have added variables and value into table; you can do some data calculating.

For example:

y.a=y.a+y.b

y.b=y.c-y.b

y.c=y.c+1

## Commit

Before end the transaction, you need call commit function to update variable to global table. In commit, it will help you make sure you update the value is right value.

Example:

 local done=false

 ………………..

 while done==false do

 ……………

 done=z.commit()

 end

## Callfunc

After finish writing transaction, you can call function callfunc() to run transactions. But this doesn’t create thread.

Example:

callfunc("op1")

The parameter must be string.

## CreateT

If you need to create a thread to run transactions, you can call function createT(). Support multithreads.

Example:

createT("op1")

The parameter must be string.

Thank you for using my product. If you meet errors or have suggestions, please contact me, I will try my best to maintain and modify it become well.