**AuraM6 LWP**

**(Findings of Implementation LWP in AuraM6)**

**Introduction**

**AuraM6 GUI** based on **DOS** is currently runs on non multi-threading mode because **DOS** does not support multi-threading process. Its kernel has not designed to perform multi-thread process. In this paper, I will share my findings through the process that I have applied to implement an open source **LWP** designed for **DOS**.

**Implementation**

**Testing AuraM6**

When I applied in **AuraM6** over several days and found that it does not continue all the threads. The process was stopped after one thread. To find out the cause, I have written two diagnostic C codes as describe in the following.

**Testing Example1.c**

In the **example1.c**, it initializes the **LWP** with **RTC1024** which usage **IRQ8** to use the system real time clock and then it creates threads and scheduled the desired function pointer and finally runs the main loop.

The source **example1.c** has been compiled by using **GCC** **DJGPP** tool chain and ran in the **DosBox** and it was printing only the main threads “**MAIN**” not scheduled threads 1, 2, 3, 4 which will print “**PROC 1**”, “**PROC2**”, “**PROC3**” and “**PROC4**”.

After lots of work, I have found that the initialization function of **LWP** was successful as its return value but actually not. Because the timer did not fire the scheduled function that’s why it only prints the main thread not scheduled threads.

lwpInit(RTC1024, 1);

But the schedule function was successful by its return value and has been proved by writing diagnostic C code name **eg-diag1.c** and **eg-diag2.c**. It was also tested by using system timer which is **RTC0** usages **IRQ0**.

ids[0] = lwpSpawn(proc, &a, 4096,1, TRUE);

**Example1.c** usages **mutex** functions of the **LWP** to use **printf()** function.

lwpCreateMutex(printf);

lwpLockMutex(printf);

lwpReleaseMutex(printf);

lwpDeleteMutex(printf);

**Testing eg-diag1.c**

In the **eg-diag1.c** file has been updated with returned error messages and used **critical section** functions of **LWP** instead of **mutex** functions of **LWP** and usage system timer **RTC0** not real time clock **RTC1024**. To safe printing, here usages **cprintf()** function instead of **printf().** And increased the stack memory to **8192** instead of **1024** to avoid crashing.

ids[i] = lwpSpawn(proc, &thread\_ids[i], 8192, 1, TRUE);

In the main thread usages…

        lwpEnterCriticalSection();

        cprintf("MAIN thread active\r\n");

        lwpLeaveCriticalSection();

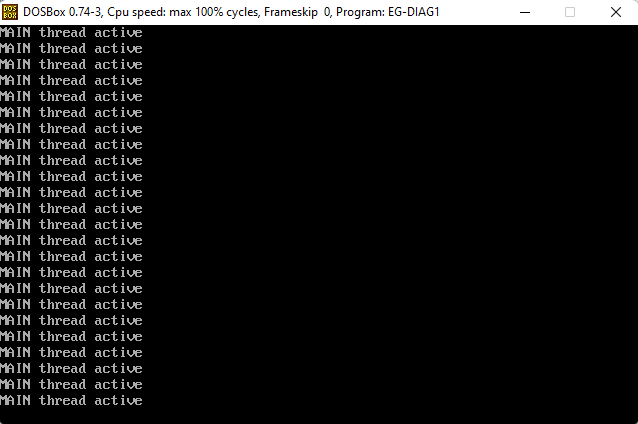
In the scheduled function usages…

        lwpEnterCriticalSection();

        cprintf("Thread %d running\r\n", id);

        lwpLeaveCriticalSection();

Finally, compiled in the **DosBox** and the test result was same. It only prints the main thread “**MAIN thread active**”.



It means that the system timer which is **RTC0** on **IRQ0** was not firing the scheduled function pointer.

**Testing** **eg-diag2.c**

After that, I had decided to write another diagnostic C code **eg-diag2.c** which is same as **eg-diag1.c**. The only different is that I had added a **manual firing function** of the **LWP** which is call…

lwpYield();

This function fires the scheduled function pointer if it has been scheduled successfully. Just added the function as bellow…

In the main thread usages…

        lwpEnterCriticalSection();

        cprintf("MAIN thread active\r\n");

        lwpLeaveCriticalSection();

        lwpYield();

In the scheduled function usages…

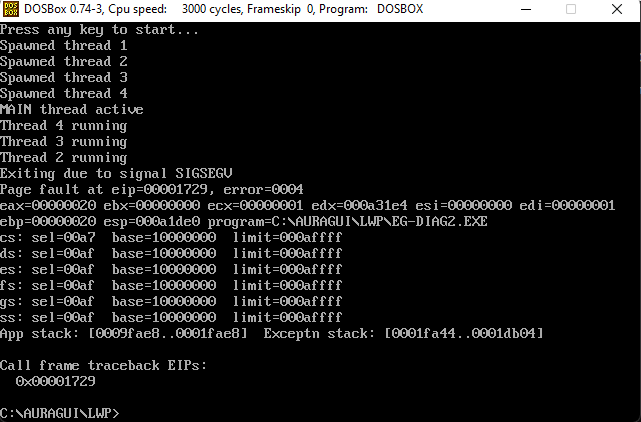
        lwpEnterCriticalSection();

        cprintf("Thread %d running\r\n", id);

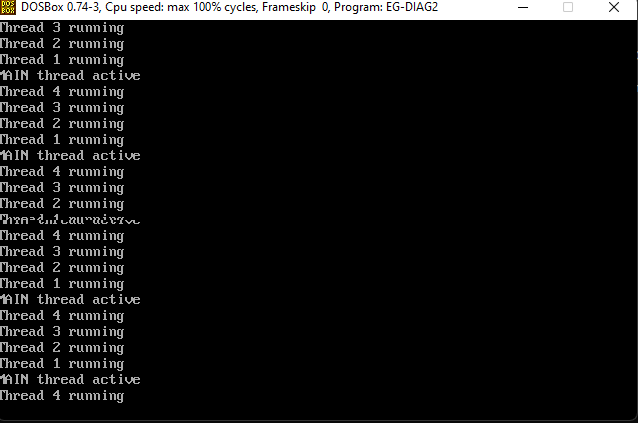
        lwpLeaveCriticalSection();

        lwpYield();

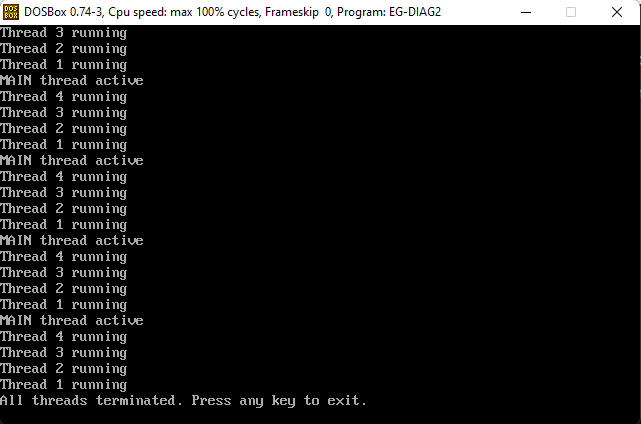
After compiled in **DosBox** and ran the result was good. It had been printed all the threads including main thread but crashed after few lines printing…



To avoid crashing, I have increased the stack memory to **8192** instead of **1024** and compiled and ran and this time it was continue with all the threads including main thread…



After pressing any key… to stop



The finding is that **GCC** **DJGPP** tool chain cannot initiate system timer or real time clock. It should be compiled in **real mode**. May be, it will work if we compile it in real mode because **LWP** has been designed to use in real mode.

**Conclusion**

It is not a good idea to fire the threads manually instead of hardware **RTC** interrupt in **AuraM6**. The output result will be not good as hardware interrupt. It is now has to be decided that will we try to build the **LWP** and **AuraM6** in real mode? If so, then we need to find out a way to compile both in real mode. There is a compiler call **Turbo C** (tcc) which compiles in real mode.

**NOTE**

Programs have been compiled and ran in the **Real DOS** in a **LPC** by using **FreeDOS USB bootable**. The result was same.

**SPARKtech**

Ashif Mahmud Chowdhury

[www.freelancer.com/u/SPARKtechBD](http://www.freelancer.com/u/SPARKtechBD)

**END OF REPORT**