Future Technology Devices International Ltd.

Application Note

AN_169

Vinculum-II RTOS

Using the Kernel Diagnostic Service

Document Reference No.: FT_000399

Version 1.0

Issue Date: 2011-02-17

This application note provides information on how to use the FTDI Vinculum-II (VNC2) RTOS Kernel Diagnostic Service. Sample source code is included.

Future Technology Devices International Limited (FTDI)

Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom
Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758
E-Mail (Support): **support1@ftdichip.com** Web: http://www.ftdichip.com

Table of Contents

1 Introduction

Multi-threaded application development is a complex undertaking that demands the synchronisation of multiple threads of execution while maintaining the integrity of the individual components of the system. During application development the need will inevitably arise to balance resource use for optimal performance, so the developer must have access to run-time diagnostics.

The VOS kernel diagnostic service is a programming interface that allows a VNC2 application to access diagnostic information on a per-thread basis. It is available from VNC2 Development tools v1.4.0 onwards. The purpose of this application note is to introduce this API and present code examples of its use.

The sample source code contained in this application note is provided as an example and is neither guaranteed nor supported by FTDI.

1.1 Overview

Two types of diagnostic information can be obtained from a VNC2 application at run-time:

- CPU usage, an indication of the amount of time a thread spends running.
- Stack usage, a measure of the use of a thread's local stack.

This information can be used during system development to achieve better performance. For example, relatively high CPU usage could be an indication that a thread's priority is too high; or a low proportion of used bytes in a thread's stack could be an indication that the stack is too big for a particular application.

In addition to user-defined threads, an application has an idle thread that is created by VOS. By definition, the idle thread is the lowest priority thread in the system, having a priority of zero, and it is always able to run. As a consequence, the idle thread runs when all other threads are blocked or delayed. The idle thread is just like any other thread in the system: it has a local stack, and local storage for diagnostic information.

The idle thread should be included in any consideration of VOS kernel diagnostics. CPU usage in particular is a measure of a thread's running time, and the idle thread must be included in any analysis of the relative time a thread spends running. The inclusion of the idle thread in the examples in this application note stresses its importance.

For detailed information regarding VNC2 application development, see [1].

1.2 Thread Manager

The thread manager is a plug-in for the VNC2 IDE, available from v1.2.4 onwards. Thread manager effectively provides the same information as the kernel diagnostic service, displaying it in a graphical way that is likely to be easy for customers to digest. It is easy to use too - the user does not need to include code in the application, since thread manager enables the diagnostics via the debugger. This could be a useful aid during development, but since it relies on the debugger interface (to patch internal structures), it can't be used in a released system.

Figure 1 shows a typical screen shot of Thread Manager in operation.

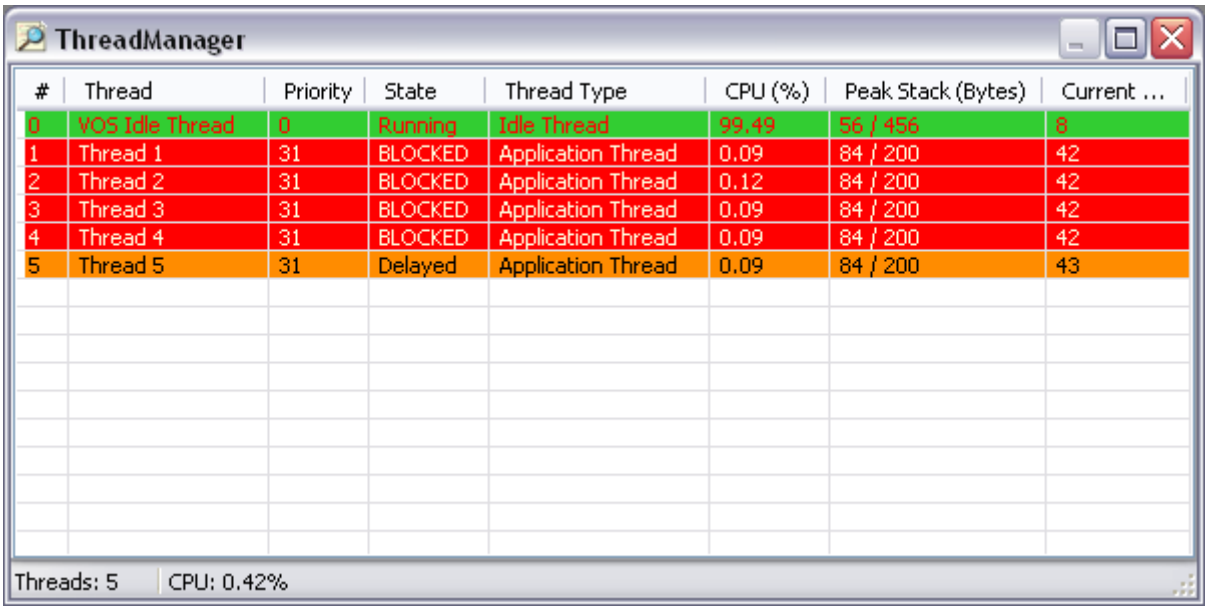

Figure 1: Thread Manager

2 Diagnostics

This section describes the functions which comprise the kernel diagnostic service API. Function definitions are found in the header file VOS.h, which is included with VNC2 Development Tools [2].

2.1 CPU Usage

Information about CPU usage is returned from the system profiler. When the profiler is enabled, the system maintains a record of the time each thread spends running. Thus the relative time spent in each thread can be calculated by comparing the running times of each thread in the system.

The profiler API is as follows:

void vos_start_profiler(void)

Initialise the profiler variables for all threads in the system and enable the profiler.

void vos_stop_profiler(void)

Disable the profiler.

uint32 vos_get_profile(vos_tcb_t *tcb)

Return the running time for the given thread. A pointer to the thread control block is passed in the *tcb* argument.

2.2 Stack Usage

A thread has its own local stack, and information about stack usage for a thread is available to the application. This information can be obtained for a system during execution, and used to optimize the thread's stack size.

uint16 vos_stack_usage(vos_tcb_t *tcb)

Return the amount of bytes used in the stack area of the given thread. A pointer to the thread control block is passed in the *tcb* argument.

2.3 Idle Thread TCB

In order to obtain diagnostic information for the idle thread, an application must first get a pointer to the idle thread's thread control block. This pointer is used in subsequent calls to the CPU and stack usage functions.

vos_tcb_t *vos_get_idle_thread_tcb(void)

Return a pointer to the thread control block of the idle thread.

3 Examples

Strategies for obtaining system diagnostics will vary. Diagnostic service API calls could be implemented in existing threads. Alternatively, and perhaps neater, all calls could be encapsulated in single, diagnostic thread. This section makes no assumption as to the diagnostic strategy chosen, but it does provide code fragments that demonstrate the use of the diagnostic service API.

3.1 CPU Usage

This example shows how to obtain the profiler *count* values for all threads in an application, and calculate the relative amount of time each thread spent running.

```
// number of application threads plus idle thread
#define NUMBER_OF_THREADS    (NUMBER_OF_APPLICATION_THREADS+1)

vos_tcb_t *tcbs[NUMBER_OF_THREADS];
uint32 running_time[NUMBER_OF_THREADS];
uint32 relative_time[NUMBER_OF_THREADS];
uint32 total_time;
uint8 i;


// create application threads and save pointers to tcbs
for (i=1; i<NUMBER_OF_THREADS; i++) {
       tcbs[i] = vos_create_thread(…);
}

…

// get pointer to idle thread tcb and save it
tcbs[0] = vos_get_idle_thread_tcb();

…

// start profiling

vos_start_profiler();

…


// calculate relative running times spent in threads

for (i=0; i<NUMBER_OF_THREADS; i++) {
      running_time[i] = vos_get_profile(tcbs[i]);
}

for (i=0, total_time=0; i<NUMBER_OF_THREADS; i++) {
      total_time += running_time[i];
}

for (i=0; i<NUMBER_OF_THREADS; i++) {
      relative_time[i] = (running_time[i] * 100) / total_time;
}
```

3.2 Stack Usage

These code fragments show how to obtain the stack usage for all threads in the system.

```
// number of application threads plus idle thread
#define NUMBER_OF_THREADS    (NUMBER_OF_APPLICATION_THREADS+1)

vos_tcb_t *tcbs[NUMBER_OF_THREADS];
uint16 stack_bytes_used[NUMBER_OF_THREADS];
uint8 i;


// create application threads and save pointers to tcbs
for (i=1; i<NUMBER_OF_THREADS; i++) {
     tcbs[i] = vos_create_thread(…);
}

…

// get pointer to idle thread tcband save it
tcbs[0] = vos_get_idle_thread_tcb();

…

// get stack usage for all threads
for (i=0; i<NUMBER_OF_THREADS; i++) {
       stack_bytes_used[i] = vos_stack_usage(tcbs[i]);
}
```

4 Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales) sales1@ftdichip.com
E-mail (Support) support1@ftdichip.com
E-mail (General Enquiries) admin1@ftdichip.com
Web Site URL http://www.ftdichip.com
Web Shop URL http://www.ftdichip.com

Branch Office – Taipei, Taiwan

Future Technology Devices International Limited (Taiwan)
2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan , R.O.C.
Tel: +886 (0) 2 8791 3570
Fax: +886 (0) 2 8791 3576

E-mail (Sales) tw.sales1@ftdichip.com
E-mail (Support) tw.support1@ftdichip.com
E-mail (General Enquiries) tw.admin1@ftdichip.com
Web Site URL http://www.ftdichip.com

Branch Office – Hillsboro, Oregon, USA

Future Technology Devices International Limited (USA)
7235 NW Evergreen Parkway, Suite 600
Hillsboro, OR 97123-5803
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-Mail (Sales) us.sales@ftdichip.com
E-Mail (Support) us.support@ftdichip.com
E-Mail (General Enquiries) us.admin@ftdichip.com
Web Site URL http://www.ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited (China)
Room 408, 317 Xianxia Road,
Shanghai, 200051
China
Tel: +86 21 62351596
Fax: +86 21 62351595

E-mail (Sales) cn.sales@ftdichip.com
E-mail (Support) cn.support@ftdichip.com
E-mail (General Enquiries) cn.admin@ftdichip.com
Web Site URL http://www.ftdichip.com

Distributor and Sales Representatives

Please visit the Sales Network page of the FTDI Web site for the contact details of our distributor(s) and sales representative(s) in your country.

5 Appendix A – References

Document References

[1] FTDI Application Note 151, *Vinculum-II User Guide*, FTDI, 2010.

[2] VNC2 Development Tools, latest version available from:

http://www.ftdichip.com/Firmware/VNC2tools.htm#VNC2Toolchain

Acronyms and Abbreviations

Terms	Description
VNC2	Vinculum II
VOS	Vinculum Operating System
RTOS	Real-Time Operating System

6 Appendix B – Revision History

Revision	Changes	Date
draft	Initial Draft	2011-01-31
1.0	Initial Release	2011-02-17