Quantum® LeaPs

innovating embedded systems

Application Note

µC/OS-II and Turbo C++ 1.01

Document Version 1.0

April 2005

Quantum LeaPs™, LLC

www.quantum-leaps.com

µC/OS-II

1 Introduction

This Application Note describes the port of µC/OS-II to the 80x86 CPU with the Borland Turbo C++ 1.01 compiler. The port is very similar to the original µC/OS-II port to 80x86 published in the book *MicroC/OS-II The Real-Time Kernel* by Jean Labrosse [Labrosse 02]. The book version requires Borland C++ 4.51, which is not free. Borland Turbo C++ 1.01, on the other hand, is available for a free download from the Borland "Museum" (http://bdn.borland.com/article/0,1410,21751,00.html). The port has been tested with µC/OS-II v2.52 and 2.60 available on the CD-ROM accompanying the book. No changes in the port are required to work with either version.

2 Installation

The archive `ucos-ii_tcpp101.zip` contains all the elements of the port. The archive is designed to "plug into" the directory structure used in the book distribution of µC/OS-II, and contains the following directories and files:

```
software
  |
  +-blocks/
  | +-PC/              - utilities for running µC/OS-II on a DOS PC
  |   +-tcpp101        - Borland Turbo C++ 1.01 version of the utilities
  |     +-pc.h         - PC utilities interface
  |     +-pc.c         - PC utilities implementation
  |
  +-ucos-ii/
  | +-EX1_x86L/        - example #1, as described in the "MicroC/OS-II" book
  | | +-tcpp101/       - Borland Turbo C++ 1.01 version of the example
  | |   +-SOURCE
  | |   | +-includes.h
  | |   | +-os_cfg.h
  | |   | +-test.c
  | |   | +-test.rsp
  | |   |
  | |   +-TEST
  | |   | +-MAKETEST.BAT
  | |   | +-TEST.EXE
  | |   | +-TEST.MAK
  | |   | +-TEST.MAP
  | |
  | +-Ix86L/           - µC/OS-II port to 80x86 CPU, Large memory model
  | | +-tcpp101/       - Borland Turbo C++ 1.01 version of the port
  | |   | +-os_cpu.h
  | |   | +-os_cpu_a.asm
  | |   | +-OS_CPU_A.OBJ
  | |   | +-os_cpu_c.c
```

Listing 1 Contents of the port ZIP file

NOTE: this port does not contain the uC/OS-II source files or even the example source files. These files are available from the CD-ROM accompanying the µC/OS-II book [Labrosse 02].

All the files in the port have standard names for the µC/OS-II distribution and have been described in the book. To install the port you need to unzip the archive in the root directory on the drive where you've installed µC/OS-II. For example, if you installed µC/OS-II on drive C: then you need to unzip the `ucos-ii_tcpp101.zip` port at the root of drive C:.

2.1 Downloading and Installing Borland Turbo C++ 1.01

The legacy Borland Turbo C++ 1.01 compiler is available for free downloads from the Borland "Museum" (http://bdn.borland.com/article/0,1410,21751,00.html). In addition, Borland provides a scanned image of the original "Turbo C++ User's Guide" documentation in PDF format (http://bdn.borland.com/cbuilder/tsuite). The User's Guide is for Borland C++ v3.0, but still largely applies to version 1.01.

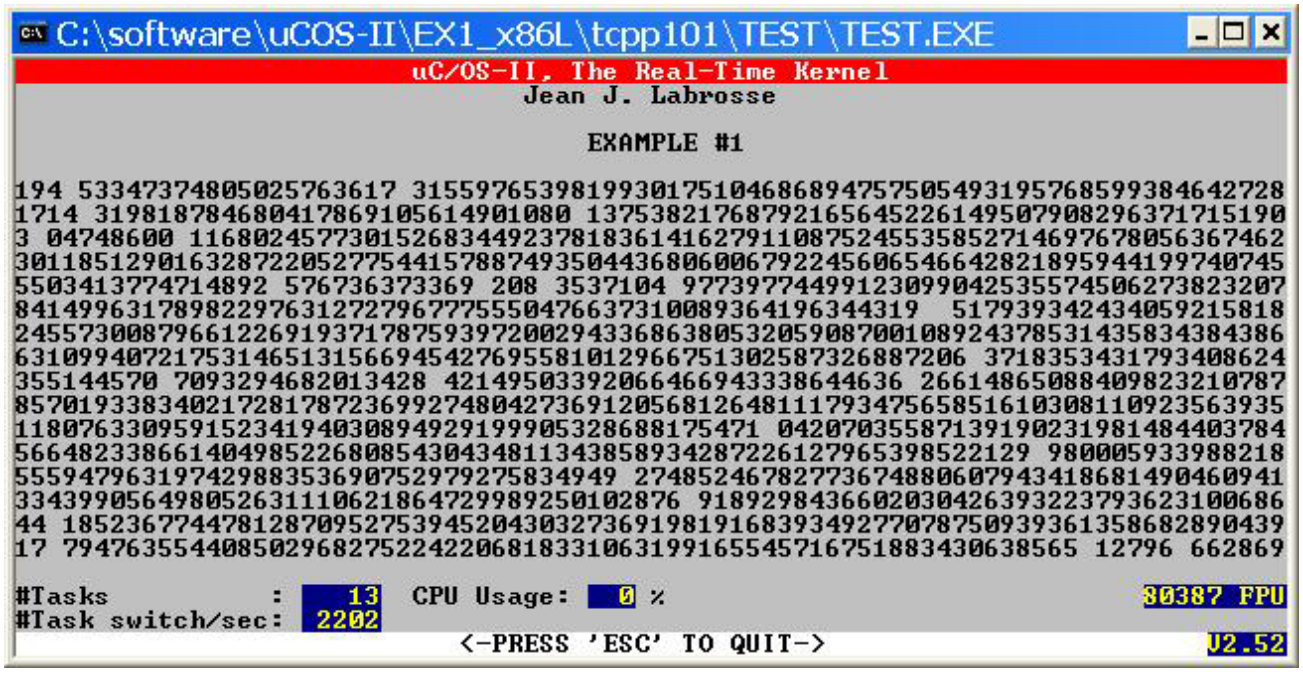
To install Borland Turbo C++ 1.01, unzip the `TCPP101.ZIP` archive from the Borland "Museum" onto your hard drive. Run the `INSTALL.EXE` program and follow the installation instructions to install the software. In this document I assume that Turbo C++ 1.01 has been installed in the directory `C:\tools\tcpp101`. If you choose a different directory, you'll need to modify the make file and the linker response file provided in this port.

2.2 Building the Example

To compile the example (EXAMPLE1) you need to follow exactly the instructions from Chapter 1 of the "MicroC/OS-II" book, except that you work in directory `\software\EX1_x86L\tcpp101\test`.

```
C:\software\uCOS-II\EX1_x86L\tcpp101\TEST>MAKETEST.BAT
```

Among others, this batch file creates the executable `TEST.EXE` in the `TEST` directory. You can run this executable on any DOS machine or in a DOS console on any Windows PC.



3 About The Port

One of the limitations of Turbo C++ 1.01 is the lack of the Turbo Assembler in the free distribution. To work around this shortcoming, this port avoids using in-line assembly, because inlining assembly in Turbo C++ 1.01 causes compilation via assembly, which cannot complete without the Turbo Assembler.

The only assembly language part of the port (`OS_CPU_A.ASM`) is available in source form and is discussed here, but this module has been pre-compiled with Turbo Assembler 3.0 and is included as an object file in this port.

3.1 OS_CPU.H Header File

Listing 2 shows the choice of the critical section method (`OS_CRITICAL_METHOD`), and task-to-task context switching method (`OS_TASK_SW()`).

```
#define  OS_CRITICAL_METHOD    3

. . .

#if      OS_CRITICAL_METHOD == 3
OS_CPU_SR OSCPUSaveSR(void);
void OSCPURestoreSR(OS_CPU_SR cpu_sr);

#define  OS_ENTER_CRITICAL()  (cpu_sr = OSCPUSaveSR())      /* Disable interrupts */
#define  OS_EXIT_CRITICAL()   (OSCPURestoreSR(cpu_sr))      /* Enable  interrupts */
#endif

. . .

#define  uCOS            0x80    /* Interrupt vector # used for context switch    */
#define  OS_TASK_SW()    geninterrupt(uCOS)
. . .
```

Listing 2 Critical Section Method and Task-to-Task Context Switching.

This port demonstrates the most advanced Critical Section Method 3. Also, to avoid inlining assembly, the `OS_TASK_SW()` macro has been defined as the `geninterrupt(uCOS)` function call.

3.2 OS_CPU_A.ASM Assembly Language Module

Listing 3 shows the assembly implementation of the functions referred in OS_CPU.H header file. The file provides also Critical Section method 1 functions (`OSCPUEnable/ OSCPUDisable`) as well as the Critical Section #3 functions (`OSCPUSaveSR()/OSCPURestoreSR`). Turbo C++ 1.01 would allow for Critical Section #2 implementation, which in fact would be more efficient, but for pedagogical reasons I wanted to demonstrate the use of Critical Section #3.

```
. . .
            PUBLIC _OSCPUTaskSW
            PUBLIC _OSCPUDisable
            PUBLIC _OSCPUEnable
            PUBLIC _OSCPUSaveSR
            PUBLIC _OSCPURestoreSR
. .
;*********************************************************************************************
;
```

```
;   void OSCPUDisable(void)
;
;*********************************************************************************************

_OSCPUDisable PROC FAR
            CLI               ; clear the I flag
            RET               ; return to the caller
_OSCPUDisable ENDP

;*********************************************************************************************
;
;   void OSCPUDisable(void)
;
;*********************************************************************************************

_OSCPUEnable PROC FAR
            STI               ; set the I flag
            RET               ; return to the caller
_OSCPUEnable ENDP

;*********************************************************************************************
;
;   int OSCPUSaveSR(void)
;
;*********************************************************************************************

_OSCPUSaveSR PROC FAR
            PUSHF             ; push the flags
            POP AX            ; pop the flags into the return value AX
            CLI               ; clear the I flag
            RET               ; return to the caller
_OSCPUSaveSR ENDP

;*********************************************************************************************
;
;    void OSCPURestoreSR(int key)
;
;*********************************************************************************************

_OSCPURestoreSR PROC FAR
            PUSH BP
            MOV  BP,SP        ; establish frame pointer
            MOV  AX, WORD PTR[BP+6] ; get the CPU_SR into AX
            PUSH AX           ; push the CPU_SR on the stack
            POPF              ; restore the CPU_SR from the stack
            POP  BP           ; cleanup the frame pointer
            RET               ; return to the caller
_OSCPURestoreSR  ENDP
. . .
```

Listing 3 Added assembly functions to OS_CPU_A.ASM

4 References

[Labrosse 02]	Labrosse, Jean, J., "MicroC/OS-II The Real-Time Kernel, Second Edition", CMP Books 2002.
[Borland 92]	Borland, Inc. "Turbo C++ 3.0 User's Guide"