

# Installing $\mu$ C/OS-II Operating System on the Ide68k Integrated Development Environment

by: [Peter J. Fondse](#)

## 1. Introduction to IDE68K

The 68000 Integrated Development Environment or IDE68K combines program editor, 68000/68020 assembler and C compiler in one program. A simulator is provided to run 680x0 programs in a simulated environment on Windows® personal computers. Its main purpose is to provide hands-on experience for students learning to program in assembly and in a combined C language/assembly environment with interfaces to virtual (simulated) I/O devices.

Although essentially a simulator, benchmarks (such as the Dhrystone benchmark) have shown that on contemporary PC's the 68000 virtual machine runs at 10 – 15 MIPS (Million Instructions Per Second) which is about 20 times faster than a real 68000 or 68020 with clock frequency of 10 – 12.5 MHz (typical for the 1980's).

### 1.1 About the MC68000 microprocessor family

The M68000 microprocessor family introduced in 1978 was one of the most popular architectures in its time. This architecture provided solutions to various markets including automotive, consumer, and communications. The M68000 family is based on a complex instruction set computing (CISC) architecture that is easy to use, but there were needs for higher performance devices based on the same architecture. Because of its popularity, Freescale Semiconductor (formerly Motorola Semiconductors) decided to continue the M680x0 legacy by offering customers the same architecture but with greater performance and other functional enhancements. The solution was to offer customers a reduced instruction set computing (RISC) version based on the M68000 architecture. The ColdFire core was introduced to address issues needed by today's demanding complex applications. The ColdFire architecture and instruction set is for 95% compatible with the older 680x0 architecture and instruction set.

## 2. Introduction to $\mu$ C/OS-II

[MicroC/OS-II](#) is the second generation of a kernel originally published in a two-part 1992 article in *Embedded Systems Programming* magazine and the book  *$\mu$ C/OS-II The Real-Time Kernel* (ISBN 978-1-57820-103-7) by Jean J. Labrosse. The author intended at first to simply describe the internals of a portable operating system he had developed for his own use, but later developed the OS as a commercial product. MicroC/OS-II (commonly termed  $\mu$ C/OS-II or uC/OS-II), is the acronym for Micro-Controller Operating Systems Version 2. It is a priority-based pre-emptive real-time multitasking operating system kernel for microprocessors, written mainly in the C programming language. It is widely used in many applications, such as cameras, medical instruments, musical instruments, engine controls, network adapters, ATM machines, industrial robots, etc. all over the world and since short even on Mars, in the Mars Curiosity Rover. Numerous colleges and universities use  $\mu$ C/OS-II to teach students about real-time systems.

$\mu$ C/OS-II is currently maintained by [Micrium Inc.](#) and can be licensed per product or per product line. Use of the operating system however is free for educational, non-commercial use.

### 2.1 Features

$\mu$ C/OS-II, the real time kernel is a portable, ROMable, preemptive real-time, multitasking kernel for microcontrollers and microprocessors. It can manage up to 64 tasks.

- **Portable**

Most of  $\mu$ C/OS-II is written in ANSI C, with target microprocessor specific code written in assembly language. Assembly language is kept to a minimum to make  $\mu$ C/OS-II easy to be ported.  $\mu$ C/OS-II can be ported to a large number of microprocessors as long as the microprocessor provides a stack pointer and the CPU registers can be pushed onto and popped from the stack. Also, the C compiler must provide language extensions to enable and disable interrupts from C.

- **Scalable**

The  $\mu$ C/OS-II is designed scalable so that customers can only use the features needed in the applications. This means that a product can have a few of  $\mu$ C/OS-II's features while another product can have the full set of features. It reduces the

amount of memory (RAM and ROM) needed by  $\mu$ C/OS-II on a per-product basis.

Scalability is accomplished by the use of conditional compilation. Users only specify (through *#define* constants) which features are needed for the application.

- **Preemptive**

$\mu$ C/OS-II is a fully-preemptive real-time kernel. It always runs the task with highest priority ready. Most commercial kernels are preemptive and  $\mu$ C/OS-II is comparable in performance with many of them.

- **Multi-tasking**

$\mu$ C/OS-II can manage up to 64 tasks. However, the current version of the software reserves eight of these tasks for system use. This leaves the application with up to 56 tasks.  $\mu$ C/OS-II cannot do round-robin scheduling because each task has a unique priority assigned which doubles as task ID. There are 64 priority levels.

- **Deterministic**

Execution time of all  $\mu$ C/OS-II functions and services is deterministic. You can calculate how much time it takes  $\mu$ C/OS-II to execute a function or a service.

Execution time of all  $\mu$ C/OS-II services do not depend on the number of tasks running in the application.

- **Stacks**

Each task requires its own stack. However,  $\mu$ C/OS-II allows each task to have a different size. This allows it to reduce the amount of RAM needed in the application. With  $\mu$ C/OS-II's stack-checking feature, the stack space each task actually requires can be calculated.

- **Services**

The  $\mu$ C/OS-II provides a number of system services as follows:

- Semaphores
- Event flags
- Mutual exclusion semaphores
- Message mailboxes
- Message queues
- Timers
- Task management
- Fixed-sized memory block management
- Time management

- **Interrupt Management**

Interrupts can suspend the execution of a task. If a higher priority task is awakened as a result of the interrupt, the task with the highest priority runs as soon as all the nested interrupts complete. Interrupts can be nested up to 255 levels.

- **Robust and reliable**

μC/OS-II is used in hundreds of commercial applications.

Specifically, μC/OS-II is currently implemented in a wide spectrum of high level safety-critical devices, including:

- Those certified for Avionics DO-178B
- Medical FDA pre-market notification (510(k)) and pre-market approval (PMA) devices
- SIL3/SIL4 IEC for transportation and nuclear systems, 99% compliant with the Motor Industry Software Reliability Association (MISRA-C:1998) C Coding Standards

## **2.2 Licensing**

No licensing fee is required for μC/OS-II for educational use. However, before downloading the software, you are required to register with [Micrium](#).

Contact Micrium for a proper license to use μC/OS-II in commercial products.

## **3. Downloading and installation**

### **3.1 IDE68K installation**

The architecture-specific files for μC/OS-II (associated with IDE68K), together with the IDE68K and μC/OS-II example programs and project files, are included in the download version 3.0 of IDE68K. A copy of IDE68K version 3.0 can be obtained from the [68000 Integrated Development Environment Homepage](#).

The program is freeware.

Delivery of the program code should take the form of a .ZIP file, which could be installed to the C:\ directory.

Normally this gives the following directory structure:

```
C:\Ide68k\Benchmarks
    \Custom
    \Examples
    \Include
    \Lib
    \OS Examples
    \uCOSII
```

The programs are installed in C:\Program Files\Ide68k, or for 64-bit Windows, in C:\Program Files (X86)\Ide68k.

Architecture-specific files for  $\mu$ C/OS-II are in the directory C:\Ide68k\uCOSII.  
(Distributed with IDE68K version 3.0 download)

These are

os_cpu.h	680x0 specific header file
os_cfg.h	$\mu$ C/OS-II configuration header, application dependent
app_cfg.h	Application specific header file
os_cpu_c.c	680x0 specific C code
os_fcpx_c.c	680x0 specific C code with 68881 FPU support
os_cpu_a.asm	680x0 specific assembly code
os_fcpx_a.asm	680x0 specific assembly code with 68881 FPU support
os_boot.asm	IDE68K specific assembly code (vectors, basic IO)
files.txt	List of files for $\mu$ C/OS-II.

Source code and project files for a number of example programs are in the directory C:\Ide68k\OS Examples. These programs can be compiled and run after the  $\mu$ C/OS-II kernel is installed.

### 3.2 $\mu$ C/OS-II Installation

The files comprising the architecture-independent files for  $\mu$ C/OS-II (not associated with any specific microprocessor) can be obtained from Micrium Inc.

The  $\mu$ C/OS-II operating system software is neither freeware nor Open Source software. A licensed copy of version 2.91 of the operating system should be obtained directly from [Micrium Inc.](#)

#### LICENSING TERMS:

μC/OS-II is provided in source form for FREE evaluation, for educational use or for peaceful research. If you plan on using μC/OS-II in a commercial product you need to contact Micrium to properly license its use in your product. We provide ALL the source code for your convenience and to help you experience μC/OS-II. The fact that the source is provided does NOT mean that you can use it without paying a licensing fee.

Delivery of the source code should take the form of a .ZIP file.

When extracted to the C:\ directory gives the following directory structure:

```
C:\Micrium\Software\uCOS-II\Doc\QuickRefChart-Color.PDF
      \QuickRefChart-Color.XLS
      \README.TXT
      \ReleaseNotes.PDF
      \TaskAssignmentWorksheet.PDF
      \TaskAssignmentWorksheet.XLS
      \uCOS-II-CfgMan.PDF
      \uCOS-II-RAM-CALC.XLS
      \uCOS-II-RefMan.PDF
      \WhatsNewSince-V200.PDF
```

and

```
C:\Micrium\Software\uCOS-II\Source\os_cfg_r.h
      \os_core.c
      \os_dbg_r.c
      \os_flag.c
      \os_mbox.c
      \os_mem.c
      \os_mutex.c
      \os_q.c
      \os_sem.c
      \os_task.c
      \os_time.c
      \os_tmr.c
      \ucos_ii.c
      \ucos_ii.h
```

All the files in the \Source directory must be copied to the directory C:\Ide68k\uCOSII

Make the following modifications: (to prevent a warning message during compilation)

File OS\_FLAG.C: (This file can be read-only, remove this flag first)

Line 124 change:

```
" wait_type &= ~OS_FLAG_CONSUME; "
```

to

```
" wait_type &= (INT8U)~OS_FLAG_CONSUME; "
```

## 4. References

### 4.1 68000 Integrated Development Environment

The program IDE68K can be downloaded from the 68000 IDE homepage,

[68000 Integrated Development Environment Homepage](#).

Version 3.0 comprises the IDE68K and 680x0 specific files to run  $\mu$ C/OS-II. In addition, several example programs are provided which can be quickly compiled and run.

See also [Getting started with IDE68K](#) and [Programming with  \$\mu\$ C/OS-II on IDE68K](#) for more information on how to run those programs.

### 4.2 MicoC/OS-II, The Real-Time Kernel

The architecture-independent files for  $\mu$ C/OS-II can be downloaded from [Micrium Inc.](#) However, prior to downloading, registration with Micrium is required. In addition to the source code files, the download also contains a configuration manual and a reference manual for  $\mu$ C/OS-II.

Excellent and extensive information about  $\mu$ C/OS-II can be found in the book

**MicroC/OS-II, The Real-Time Kernel, second edition**

Jean J. Labrosse

San Francisco, CPM Books

ISBN 978-1-57820-103-7

This book can also be ordered directly from Micrium Inc.

<http://micrium.com/books/older-books>