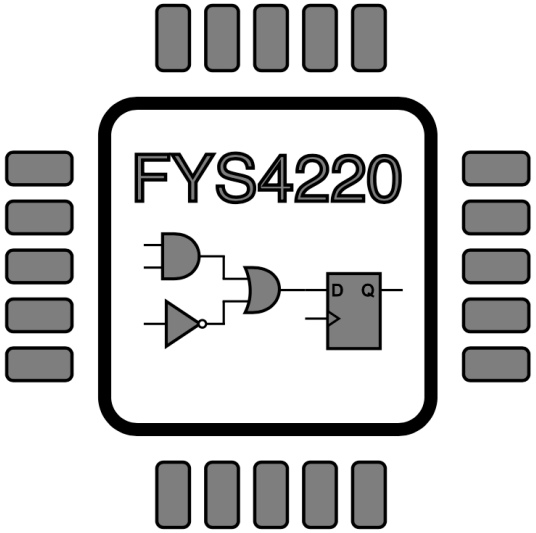


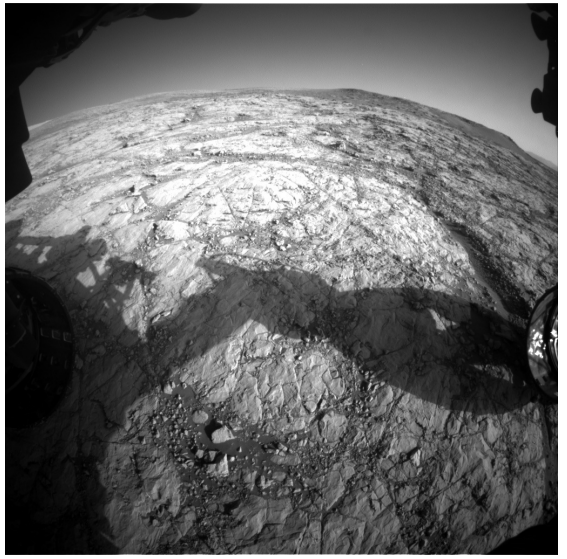
μC/OS-II

- Micro-controller Operating Systems version 2 (1998)
- Based on µC/OS published in 1992
- Maintained by Micrium Inc.
  - Acquired by Silicon Labs in 2016
- Used in all kinds of applications such as cameras, avionics, high-end audio, medical instrumentation, musical instruments, engine control, industrial robots and many more.
- Used in many universities to teach students about real-time systems

uC/OS-II

- Controls one of the analytical labs called SAM (Sample Analysis at MARS)
- Investigate the chemical and isotopic composition of the Martian atmosphere and soil.
- 20000 lines of C code
- Runs on top of µC/OS-II platform
  - µC/OS-II is off the-shelf except for adoptation of Coldfire BSP
- SW resides in nonvolatile memory
- On-board computer is all custom built electronics
  - Radiation tolerant ColdFire CPU

http://micrium.com/curiosity/

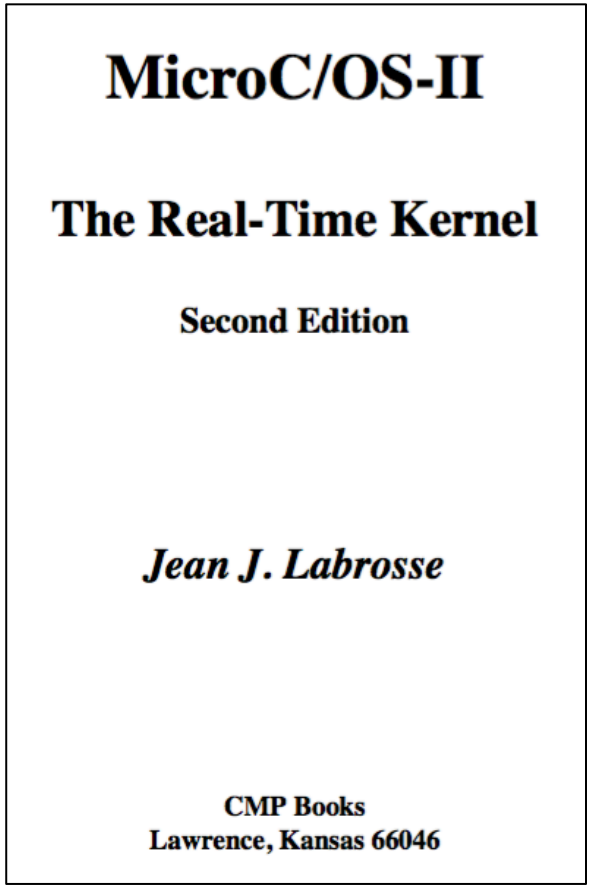


This image was taken by Front Hazcam: Left B (FHAZ_LEFT_B) onboard NASA's Mars rover Curiosity on Sol 1859 (2017-10-29 08:57:28 UTC).

Image Credit: NASA/JPL-Caltech

uC/OS-II

- Priority-based pre-emptive real-time multitasking operating system kernel
  - Always runs the highest priority task that is ready
- Written mainly in ANSI C and intended for use in embedded systems
  - Source code of about 5500 lines.
- Very small real-time kernel
  - Memory footprint of about 20kB for a fully functional kernel
- It can manage up to 64 tasks (56 user tasks)
  - Reserve 8 for uC/OS-II
  - Each task has a unique priority assigned to it, which means that round-robin scheduling is not supported
- Ported to more than 100 different µP and µC. (Link)
- Supports all type of processors from 8-bit to 64-bit



http://micrium.com/downloadcenter/download-results/?searchterm=ft-book&supported=true

uC/OS-II

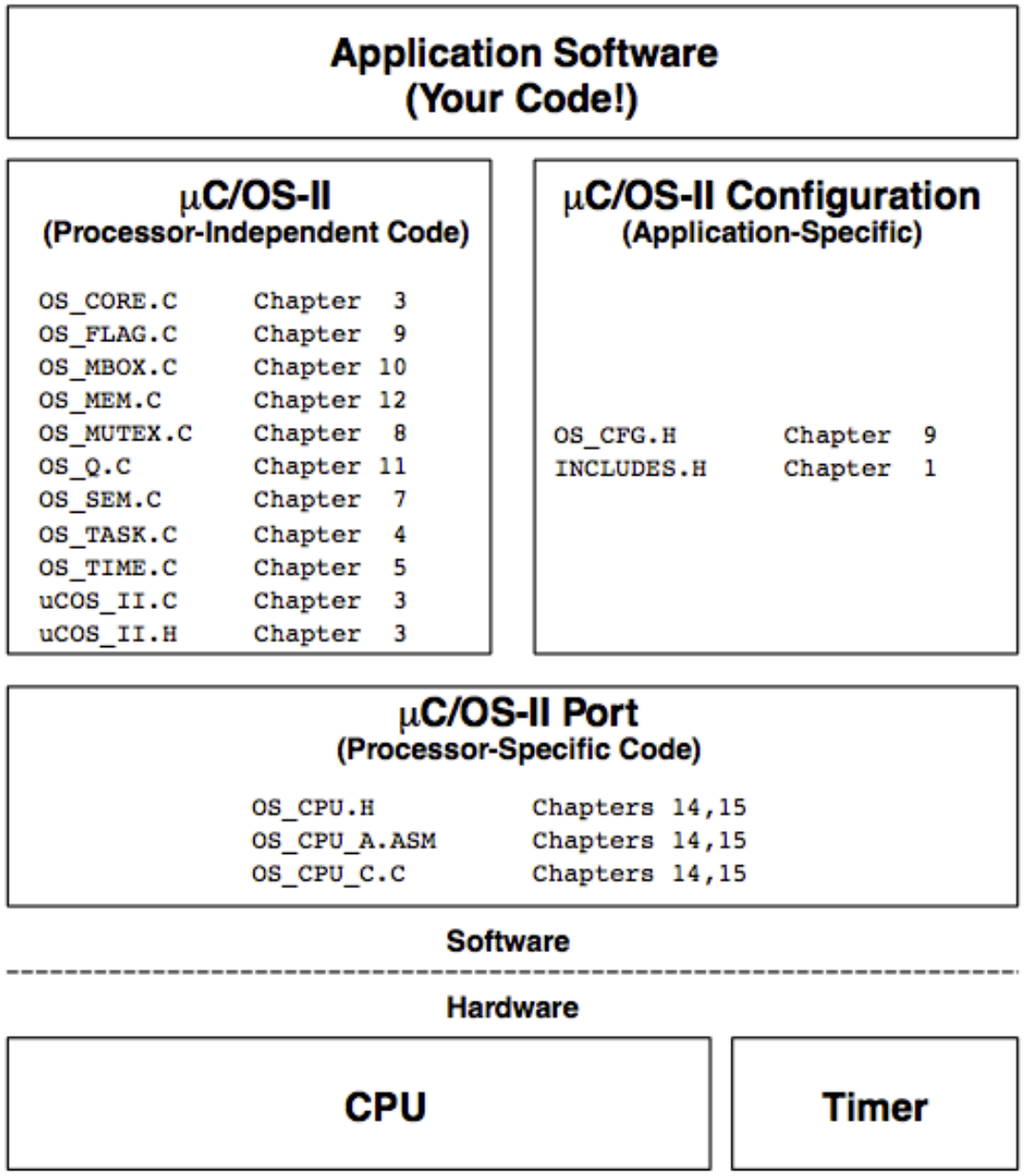
- Certified for use in avionic products (DO-178B)
  - "*Every feature, function and line of code has been examined and tested to demonstrate that it is safe and robust enough to be used in safety-critical systems where human life is on the line*", J.J.Labrosse MicroC/OS-II The Real-Time Kernel
- Permissive, open-source license model, as of February 28, 2020 (Link)
  - Previously: Not freeware nor open source, but free for educational non-commercial use
- On March 24, 2009, Micrium released an enhanced product, µC/OS-III
  - Unlimited number of tasks and priorities
  - Round robin scheduling
- Provides other middleware software products such as µC/CAN, µC/GUI, µC/FS, µC/TCP-IP, µC/USB, µC/Probe etc.

https://www.micrium.com/micrium-expands-rtos-family-with-µcos-iii/

uC/OS-II Services

- Time management
- Task management
- Semaphores (Binary and counting)
- Mutual exclusion semaphores (mutexes)
- Message mailboxes
- Message queues
- Event flags
- Fixed-size memory block manager

uC/OS-II file structure



A basic uC/OS-II example

```c
#include <stdio.h>
#include "includes.h"

/* Definition of Task Stacks */
#define   TASK_STACKSIZE       2048
OS_STK    task1_stk[TASK_STACKSIZE];
OS_STK    task2_stk[TASK_STACKSIZE];

/* Definition of Task Priorities */

#define TASK1_PRIORITY      1
#define TASK2_PRIORITY      2

/* Prints "Hello World" and sleeps for three seconds */
void task1(void* pdata)
{
  while (1)
  {
    printf("Hello from task1\n");
    OSTimeDlyHMSM(0, 0, 3, 0);
  }
}
/* Prints "Hello World" and sleeps for three seconds */
void task2(void* pdata)
{
  while (1)
  {
    printf("Hello from task2\n");
    OSTimeDlyHMSM(0, 0, 3, 0);
  }
}
```

```c
/* The main function creates two task and starts multi-tasking */
int main(void)
{

  OSTaskCreateExt(task1,
                  NULL,
                  (void *)&task1_stk[TASK_STACKSIZE],
                  TASK1_PRIORITY,
                  TASK1_PRIORITY,
                  task1_stk,
                  TASK_STACKSIZE,
                  NULL,
                  0);


  OSTaskCreateExt(task2,
                  NULL,
                  (void *)&task2_stk[TASK_STACKSIZE],
                  TASK2_PRIORITY,
                  TASK2_PRIORITY,
                  task2_stk,
                  TASK_STACKSIZE,
                  NULL,
                  0);
  OSStart();
  return 0;
}
```

A basic FreeRTOS example

```c
#include <stdio.h>
#include "FreeRTOS.h"
#include "task.h"
#include "string.h"

#define TASK_STACKSIZE 2048


void task1( void *p){
  while(1){
    printf("Hello from task1\n");
    vTaskDelay(20);
  }
}

void task2(void *p){
while(1){
  printf("Hello from task2\n");
  vTaskDelay(4);
 }
}


int main()
{
  xTaskCreate(task1, "task1", TASK_STACKSIZE, NULL, 1, NULL);
  xTaskCreate(task2, "task2", TASK_STACKSIZE, NULL, 2, NULL);

  vTaskStartScheduler();
  return 0;

}
```

https://www.freertos.org/index.html