# Application Programming Interfaces for FreeRTOS / MOS for Agon Light

Julian Rose

We primarily describe the Application Programming Interface (APIs) of FreeRTOS / MOS for the Agon Light and compatibles. These APIs enable user application programs to access the system services and various capabilities in the C programming language. We assume the reader is acquainted with Agon Light hardware [Agon], the MOS software [MOS], FreeRTOS, and the C programming language in general, so we do not describe them further herein. See the References for links to these subjects and more.

## Summary

We briefly summarise the FreeRTOS port to the eZ80-based Agon Light.

Targeting Agon as a Micro-controller, FreeRTOS port for the Zilog eZ80-based Agon Light (and compatibles) running MOS. The concept is very much FreeRTOS over MOS, reflected in the project name. FreeRTOS provides concurrency and time, and MOS provides the system services.

This Agon Light port integrates FreeRTOS version 20221201-LTS (10.5.1) with the Zilog ZDSII eZ80Acclaim! version 5.3.5 (Build 23020901) C language toolset. The choice of version 20221201-LTS prioritises stability over latest and greatest.

### MOS versions

FreeRTOS for Agon is built to MOS version 1.04. Version 1.03 may work, but has not been tested. Likewise newer Console8 versions of MOS 2.x will work, but have not been tested. Console8-specific MOS and VDP functions are not yet supported. There are a small number of tightly coupled dependencies in parts of the code (such as the keyboard read functions) which may fail if differences start to emerge between forks of the MOS code.

### What is FreeRTOS?

FreeRTOS is a real-time software development kernel and library. It provides an application development framework that allows a C language application to be arranged into a number of concurrent tasks. The core of FreeRTOS (and indeed any RTOS) is its kernel; refer to [FreeRTOS Kernel]. And to [FreeRTOS API] for the API.

FreeRTOS is not an operating system in the sense of MOS or CP/M, which provide a command console interpreter and long-term storage. Instead, FreeRTOS for Agon applications run on MOS to access its services. No re-Flashing is required - you just build and link a C language application together with the FreeRTOS software into a MOS executable in the usual way.

#### Why do we care about concurrency and time?

In a nutshell, because they are manifest in the real world. A micro-controller interacts with the real world, through sensors and actuators. To perform its functions well, we need to embrace time and the order of events (concurrency) into our embedded software.

## Capabilities

FreeRTOS / MOS for Agon Light is a highly configurable, multi-capability project.

|  |  |
| --- | --- |
| Capability | API |
| Alpha | FreeRTOS API, with a minimal MOS API, in eZ80 ADL mode |
| Beta | MOS API (MOS 1.04 subset of the FFS API), plus a DEV API, the hardware interfaces APIss |
| Gamma | VDP API (MOS 1.04 subset), the graphics API |
| Delta | Console8 MOS 2.x & VDP extended API |
| Epsilon | A real-time API inspired by posix-4 |
| Omega | A safer version with protected Z80-mode tasks and the ADL-mode kernel |

Each capability is configurable through user-settable definitions in application-specific config header files. The different capabilities are not mutually exclusive; a user-application can be configured to include the exact mix of capabilities it requires. In this way the built executable occupies the least RAM footprint possible.

### Alpha Capability API

The Alpha capability supports all the FreeRTOS functions; the Zilog Standard C library functions; and a minimal number of MOS functions.

|  |  |
| --- | --- |
| Header Files | Functions |
| <cio.h> | getch, putch – implemented in MOS/init.asm |
| <assert.h>, <ctype.h>, <errno.h>, <float.h>, <format.h>, <limits.h>, <math.h>, <setjmp.h>, <stdarg.h>, <stddef.h>, <stdio.h>, <stdlib.h>, <string.h> | Standard C library from Zilog, including printf, memcpy, strlen, etc. |
| <atomic.h>, <croutine.h>, <event\_groups.h>, <FreeRTOS.h>, <list.h>, <message\_buffer.h>, <portable.h>, <queue.h>, <semphr.h>, <stdint.h>, <stream\_buffer.h>, <task.h>, <timers.h> | FreeRTOS library, including xTaskCreate, vSemaphoreCreate, xEventGroupCreate, xMessageBufferCreate, etc. |
| FreeRTOSConfig.h | Application Alpha FreeRTOS configuration.  configUSE\_PREEMPTION  configTICK\_RATE\_HZ  configCHECK\_FOR\_STACK\_OVERFLOW, etc. |

The Zilog header files are found under …/ZDSII\_eZ80Acclaim!\_5.3.5/include/ (where … means your installation path).

The FreeRTOS header files are found under …/FreeRTOS-for-Agon/FreeRTOSv202212.01-LTS/FreeRTOS/Source/include/.

Each application will maintain and include its own FreeRTOSConfig.h header file, to bring in exactly the FreeRTOS functions it requires and no more. See for example the Demos under …/FreeRTOS-for-Agon/FreeRTOSv202212.01-LTS/FreeRTOS/Demo/Alpha/.

### Beta Capability

The Beta capability adds support for most of the MOS API defined in [MOS API] (including the FFS API), plus an additional DEV API.

#### MOS API

|  |  |
| --- | --- |
| Header Files | Functions |
| mosapi.h | mos\_getkey, mos\_load, mos\_save, mos\_cd, mos\_del, mos\_ren, mos\_mkdir, mos\_getsysvars, mos\_copy, |
|  | mos\_fopen, mos\_fclose, mos\_fgetc, mos\_fputc, mos\_feof |
|  | mos\_getfil, mos\_fread, mos\_fwrite, mos\_flseek |
|  | mos\_geterror (plus mos\_printerr) |
|  | mos\_getrtc, mos\_setrtc, mos\_setintvector, mos\_uopen, mos\_uclose, mos\_ugetc, mos\_uputc, mos\_setkbvector, mos\_getkbmap, mos\_i2c\_open, mos\_i2c\_close, mos\_i2c\_write, mos\_i2c\_read |
| ffsapi.h | ffs\_fopen, ffs\_fclose, ffs\_fread, ffs\_fwrite, ffs\_flseek, ffs\_feof, ffs\_fstat |
| mosConfig.inc | Application Beta MOS API configuration:  configUSE\_MOS\_FILE\_OPS  configUSE\_MOS\_DIR\_OPS  configUSE\_MOS\_BUFFERED\_FILE\_OPS  configUSE\_MOS\_SYSVARS  configUSE\_FFS\_FILE\_OPS  configUSE\_FFS\_DIR\_OPS  configUSE\_MOS\_KEYBOARD\_OPS  configUSE\_MOS\_DEVICE\_RTC  configUSE\_MOS\_DEVICE\_UART  configUSE\_MOS\_DEVICE\_I2C |

The MOS header files mosapi.h and ffsapi.h are found under …/FreeRTOS-for-Agon/FreeRTOSv202212.01-LTS/FreeRTOS/Source/mos/ (where … means your installation path).

Each application will maintain and include its own mosConfig.inc header file, to bring in exactly the MOS functions it requires and no more. See for example the Demos under …/FreeRTOS-for-Agon/FreeRTOSv202212.01-LTS/FreeRTOS/Demo/Beta/. Note this is an assembly header file as the MOS API is implemented mainly in assembler.

#### DEV API

The DEV API extends the Beta capability with access to the Agon Extensions Interface directly. It provides a safeguarded API for SPI, UART, I2C and GPIO; and through them to connected devices. The DEV API enhances the Uart and I2C capability of MOS, and provides APIs for GPIO and SPI that are absent in the MOS API.

|  |  |
| --- | --- |
| Header Files | Functions |
| devapi.h | uart\_open, uart\_close, uart\_read, uart\_write, uart\_poll |
|  | i2c\_open, i2c\_close, i2c\_read, i2c\_write |
|  | spi\_open, spi\_close, spi\_read, spi\_write |
|  | gpio\_open, gpio\_close, gpio\_read, gpio\_write |
| devConfig.h | Application Beta DEV API configuration:  configUSE\_DRV\_UART  configUSE\_DRV\_I2C  configUSE\_DRV\_SPI  configUSE\_DRV\_GPIO  configUSE\_DEV\_SAFEGUARDS  configUSE\_FAST\_INTERRUPTS |

The DEV header file devapi.h is found under …/FreeRTOS-for-Agon/FreeRTOSv202212.01-LTS/FreeRTOS/Source/mos/ (where … means your installation path). (Private header files for each of the major device types are also found there, which are not intended for direct inclusion by application programs.)

Each application will maintain and include its own devConfig.inc header file, to bring in exactly the DEV functions it requires and no more. See for example the Demos under …/FreeRTOS-for-Agon/FreeRTOSv202212.01-LTS/FreeRTOS/Demo/Beta/.

## References

[Agon] <https://github.com/TheByteAttic/AgonLight> Agon Light hardware

[FreeRTOS API]

[FreeRTOS Kernel] <https://www.freertos.org/RTOS.html> About the FreeRTOS kernel

[MOS] <https://agonconsole8.github.io/agon-docs/MOS/> Machine Operating System for Agon Light and compatibles.

[MOS API] <https://agonconsole8.github.io/agon-docs/MOS-API/#the-mos-api>