
jocktos-docs

Release v0.1

Nick Schneider, Joshua Goard

Mar 07, 2024

CONTENTS:

1	Source Code!	3
1.1	Embedded LaTeX Equations	3
1.2	Data Structure Reference	3
1.3	Enum Reference	4
2	Indices and tables	7
	Index	9

A journey of learning how to create an RTOS kernel from the ground up, by Nick Schneider and Joshua Goard

SOURCE CODE!

1.1 Embedded LaTeX Equations

Here's some LaTeX inside the reStructured-Text:

$$\frac{\sum_{t=0}^N f(t, k)}{N}$$

1.2 Data Structure Reference

Here's an auto-gen blurb on a coreRegisters data structure:

```
struct coreRegistersDef
    STM32 Cortex-M4 Core Registers.
```

Public Members

```
uint32_t registers[13]
```

R0-R12 are 32-bit general-purpose registers for data operations.

```
uint32_t *stack_pointer
```

The Stack Pointer (SP) is register R13.

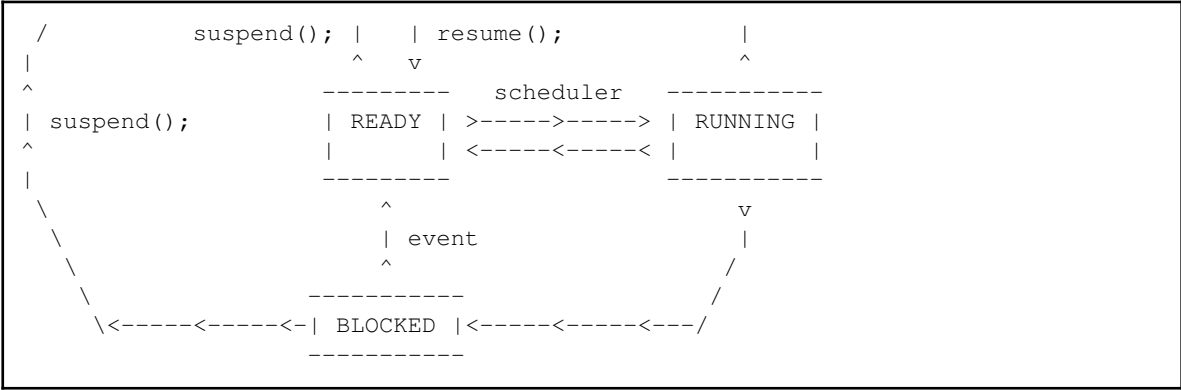
In Thread mode, bit[1] of the CONTROL register indicates the stack pointer to use:

- 0: Main Stack Pointer (MSP). This is the reset value.
- 1: Process Stack Pointer (PSP). On reset, the processor loads the MSP with the value from address 0x00000000.

```
uint32_t link_register
```

The Link Register (LR) is register R14.

It stores the return information for subroutines, function calls, and exceptions. On reset, the processor loads the LR value 0xFFFFFFFF.



Values:

enumerator **RUNNING**

Currently active task.

enumerator **READY**

In the queue and ready to run.

enumerator **BLOCKED**

Awaiting a resource.

enumerator **SUSPENDED**

Delayed or intentionally released.

This was all in the header file!

The figure and LaTeX can both be found in jocktos/main.h

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

C

[coreRegistersDef \(C++ struct\), 3](#)
[coreRegistersDef::control_register \(C++ member\), 4](#)
[coreRegistersDef::exception_markers \(C++ member\), 4](#)
[coreRegistersDef::link_register \(C++ member\), 3](#)
[coreRegistersDef::program_counter \(C++ member\), 3](#)
[coreRegistersDef::program_status_register \(C++ member\), 4](#)
[coreRegistersDef::registers \(C++ member\), 3](#)
[coreRegistersDef::stack_pointer \(C++ member\), 3](#)

J

[jocktos_TaskState \(C++ enum\), 4](#)
[jocktos_TaskState::BLOCKED \(C++ enumerator\), 5](#)
[jocktos_TaskState::READY \(C++ enumerator\), 5](#)
[jocktos_TaskState::RUNNING \(C++ enumerator\), 5](#)
[jocktos_TaskState::SUSPENDED \(C++ enumerator\), 5](#)