FreeRTOS on RISC-V

Richard Barry
Founder, FreeRTOS Project
Principal Engineer, AWS IoT
Agenda

The FreeRTOS Kernel

Running FreeRTOS on RISC-V
FreeRTOS—Open source real time kernel is everywhere...
FreeRTOS downloads per month over 15 years

Official RISC-V port released February 2019
Use cases
Recent trend - IoT

- Secure device connectivity and messaging
- Fleet onboarding, management and SW updates
- Fleet audit and protection
- IoT data analytics and intelligence

Intelligence
Insights & Logic → Action
Bare metal, assembly or C

- User (application) code
- Vendor supplied libraries
- Hardware
Application view with individual libraries

User (application) code

TCP/IP
MQTT
Wi-Fi

Provisioning
TLS
Key management
Secure storage

Vendor supplied libraries

Hardware
Introducing a library that implements multithreading

Application Design Goals:

- Meet real time requirements!
- Maximize responsiveness
- Use as little CPU/Power as possible
- Maximize maintainability
- Maximize portability (hardware change)
- Simplicity!
- Fast to market
- Meet requirements with minimum expenditure
Application view with individual libraries

User (application) code

- TCP/IP
- MQTT
- Wi-Fi

Vendor supplied libraries

Hardware

Secure storage
Key management
OTA
Provisioning
TLS
Application view with security and connectivity libraries

- User (application) code
  - FreeRTOS kernel
  - Middleware libraries
    - MQTT agent
    - Greengrass discovery
    - OTA agent
  - Vendor supplied libraries
  - Hardware
FreeRTOS on RISC-V: Source Files
Common source files and port specific source files
Example for Arm

```
c:\
  event_groups.c
  list.c
  queue.c
  stream_buffer.c
  tasks.c
  timers.c

  include

  portable

  GCC

  ARM_CM0
    port.c
    portmacro.h

  ARM_CM3
    port.c
    portmacro.h

  ARM_CM3_MPU
    port.c
    portmacro.h
```
Additional header file for RISC-V

event_groups.c
list.c
queue.c
stream_buffer.c
tasks.c
timers.c

include
portable
  GCC
    RISC-V:
      port.c
      portASM.S
      portmacro.h
    chip_specific_extensions
      Pulpino_Vega_RV32M1RM
      freertos_risc_v_chip_specific_extensions.h
      RV32I_CLINT_no_extensions
      freertos_risc_v_chip_specific_extensions.h
FreeRTOS on RISC-V: Interrupt stack
Defining the interrupt stack

Task 1
Max Task Stack
Max IRQ Stack

Task 2
Max Task Stack
Max IRQ Stack

Task 3
Max Task Stack
Max IRQ Stack

Max Task Stack
Max IRQ Stack
Max Task Stack
Max IRQ Stack
FreeRTOS on RISC-V: Setting CLINT base address
FreeRTOS on RISC-V: Installing the FreeRTOS trap handler
FreeRTOS on RISC-V: Calling an external interrupt handler
Category:
- General Options
- Static Analysis
- C/C++ Compiler
- Assembler
- Output Converter
- Custom Build
- Build Actions
- Linker
- Debugger
- I-jet
- Simulator

Factory Settings

Language | Output | List | Preprocessor | Diagnostics | Extra Options
--- | --- | --- | --- | --- | ---

- **Ignore standard include directories**

Additional include directories: (one per line)
```
$PROJ_DIR$\..\..\Source\portable\IAR\RISC-V\chip_specific_extensions
```

Preinclude file:

Defined symbols: (one per line)
```
\RUPT=vApplicationHandleTrap
```
Thank You!

Download, share and support

https://www.freertos.org

https://github.com/aws/amazon-freertos

@real_FreeRTOS  |  @AWSOpen