IoT OS on RISC-V with RT-Thread
Outline

- RTOS Porting on RISC-V
- What is RT-Thread
- Highlights of RT-Thread
- Future of RT-Thread and RISC-V
RT-Thread porting on RISC-V
E310-based porting

- Base HiFive1 Board
- Base QEMU(branch: riscv-all)
  - qemu-system-riscv32 -M sifive_e ...
E310-based porting

- RTOS Porting
  - Interrupt Enable/Disable
  - Context Switching
  - Interrupt Handling
  - OS Tick
  - [Cache]
E310–based porting

RT-Thread’s libcpu porting interface

- Functions:
  - rt_hw_interrupt_enable
  - rt_hw_interrupt_disable
  - rt_hw_stack_init
  - rt_hw_context_switch_to
  - rt_hw_context_switch
  - rt_hw_context_switch_interrupt

- Variable
  - rt_thread_switch_interrupt_flag
  - rt_interrupt_from_thread
  - rt_interrupt_to_thread
E310–based porting

Interrupt enable/disable

- rt_base_t rt_hw_interrupt_disable(void)
  - Saves the global interrupt status, then disable it and returns the saved state

- void rt_hw_interrupt_enable(rt_base_t level)
  - Restores global interrupt status from variable ‘level’
E310–based porting

- **Context switching**
  - Switch to the first thread:
    - `rt_uint8_t *rt_hw_stack_init(`
      - `void *tentry,`
      - `void *parameter,`
      - `rt_uint8_t *stack_addr,`
      - `void *texit);`
    - `void rt_hw_context_switch_to(rt_uint32 to);`
**E310-based porting**

- **Context switching**
  - Thread to Thread: `rt_hw_context_switch(rt_uint32 from, rt_uint32 to)`

```
thread_from
   EPC
   RA
   MSTATUS
   X3
   X4
   X5
   ...
   X30
   X31

thread_from
   stack

thread_to
   EPC
   RA
   MSTATUS
   X3
   X4
   X5
   ...
   X30
   X31

stack
```
E310-based porting

- **Interrupt**
  - Save thread from context after entering trap

```
EPC
RA
MSTATUS
X3
X4
X5
...
X30
X31
```

- `trap`
- `switch to interrupt stack`
- `thread from stack`
- `handle_trap`
E310-based porting

- **Interrupt**
  - handle_trap() calls the ISR (Interrupt Service Routine)
  - Code in ISR may trigger Thread Scheduling

```plaintext
handle_trap  →  ISR  →  rt_sem_release
              →  rt_mutex_release
              →  rt_event_send
              →  ......  
              →  rt_schedule

rt_thread_switch_interrupt_flag = 1  →  rt_hw_context_switch_interrupt

save from thread sp to rt_interrupt_from_thread

save to thread sp to rt_interrupt_from_thread

rt_hw_context_switch_interrupt
```
E310–based porting

- **Interrupt and Context switching**
  - Restore context when the ISR is exited

```plaintext
after ISR

rt_thread_switch_interrupt_flag = 1

rt_thread_switch_interrupt_flag = 0

thread to stack

thread from stack

EPC
RA
MSTATUS
X3
X4
X5
...
X30
X31
```
E310-based porting

- System Tick
  - RTOS requires a timer for OS tick,
  - In the porting of HiFive1 E310, using LFROSC as the clock source triggers the mtime interrupt;
  - In the mtime interrupt handler, every tick calls rt_tick_increase() once;
  - The current tick configuration is 10ms.
RT-Thread Introduction
What is RT–Thread?

1. RT-Thread is a company. Real-Thread Technology is the service company behind RT-Thread and promotes the development, maintenance, update, and operation of RT-Thread.

2. RT-Thread is a RTOS kernel that was born in 2006. It was developed by Bernard Xiong. It is open source and has small footprint; It has been adopted by mainstream companies in many fields and has become the most mature and stable RTOS with the largest installed capacity in China.

3. RT-Thread is an IoT OS/middleware platform, includes many software components such as file systems, device frameworks, graphics libraries/GUIs, application frameworks, and more for IoT fields.

4. RT-Thread is a developer community, the biggest open source community for embedded software system in China, with tens of thousands of developers and rapid growth.
Software ecology

- Mainstream toolchain support for rapid development deployment;
- Rich application frameworks and third-party software tools support;

Industry/Customer

- Used in many high-reliability fields such as energy, medical, automotive, etc.
- Used by hundreds of well-known companies in various fields

Hardware support

- Multiple major MCU architectures
- Almost all mainstream chip on the market
## RT-Thread Application

<table>
<thead>
<tr>
<th></th>
<th>Computing Device</th>
<th>High-end Smart Devices</th>
<th>Smart Device</th>
<th>Simple Device Node</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High-performance computing</td>
<td>complex but cost sensitive</td>
<td>low power and cost sensitive</td>
<td>Sensor</td>
</tr>
<tr>
<td></td>
<td>Higher resolution graphics</td>
<td>Harsh real-time</td>
<td>High real-time</td>
<td>Very low power and cost</td>
</tr>
<tr>
<td></td>
<td>High-end image processing</td>
<td>High resolution graphics</td>
<td>Button or touch interaction</td>
<td>Simple application</td>
</tr>
<tr>
<td></td>
<td>Complex touch interaction</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RT–Thread Software Architecture
IoT Connection Components

IoT Cloud Access
- MQTT
- HTTP
- CoAP
- mDNS
- LwM2M
- uPnP

DTLS
- TLS
- Tiny Crypt

SAL (Socket Abstraction Layer)
- lwIP(IPv4/IPv6)
- AT Command
- RPL 6LoWPAN
- Socket CAN

WiFi/ETH/Cellar
- GPRS/WiFi/NBIOT
- BLE/Lora/Sub-1G
- 485/232/CAN
Highlights of RT-Thread

- Stable/Reliable
- Rich Components
- Easy to Use
- Highly Scalable
- Cross Platform
Building the Foundation for IoT

Security Features

- **Cloud Connection**: All communications can support encrypted transmission. For example: https, mqtt(tls), CoAP(dtls).
- **Thread Protection**: Isolated thread stack without affecting each other. Automatic detection for stack overflow.
- **Secure Startup**: Secure bootloader. Integrated highly reliable OTA components.
- **Data Storage**: Cryptographic Library. Supports AES, base64, SHA, MD5, etc.
RT–Thread Script Engine Packages

**Python Script**
- MicroPython Engine for MCU

```plaintext
//
// RT - Thread Operating System
// 3.0.0 build Nov 10 2017
// 2006 - 2017 Copyright by rt-thread team

[Flash](./components/flash/src/ef_env.c:144) ENV start address is 0x08000000, size is 262144 bytes.
[Flash](./components/flash/src/ef_env.c:760) Calculate ENV CRC32 number is 0x1AB8F50D.
[Flash](./components/flash/src/ef_env.c:772) Verify ENV CRC32 result is OK.
[Flash](./components/flash/src/ef_env.c:760) Calculate ENV CRC32 number is 0x079120BD.
[Flash](./components/flash/src/ef_env.c:772) Verify ENV CRC32 result is OK.

msh >/dev/tty  // EasyFlash V3.0.3 is initialize success.
```

**JavaScript Script**
- Lightweight JerryScript Engine

```plaintext
\|
\|
RT - Thread Operating System
// 3.0.0 build Nov 10 2017
// 2006 - 2017 Copyright by rt-thread team

wIP-2.0.2 initialized!
ERROR: Can't support gpio #63
[SFUD] Find a GigaDevice flash chip. Size is 16777216 bytes.
[SFUD] flash flash device is initialize success.
root file system initialized!
ERROR: Can't support gpio #63
ERROR: gpio_request, GPIO 63 is already in use
fh_mmc_request.get response returns -2, cmd: 5
SD card capacity 3915776 KB
probe mmcblk block device!
found part[0], begin: 1048576, size: 3.751GB
sdcard file system initialized!
media process init success!!
PAE init success!
vpu init success!!
JPEG init success!
YOU init success!
cis_clk_out: parent='pll0'
```
Highly scalable

 ENV

 Package Manager

 Configurator

 Toolchain

 Simulator

 Highly scalable

 Simple, reliable, low resources

 Audio and video streaming

 Network Protocol

 Basic Components

 RTOS Kernel

 Building the Foundation for IoT
Configuration Tool

- **Configuration System**: Kconfig
  - Easy-to-use configuration tools that can be used to adjust the configuration of the kernel and components
  - Set up the system as easy as building blocks
  - UI-based configuration with good interactivity
  - A text help explains the configuration
  - Automatic processing dependencies
  - Efficient configuration checking

- **Build System**: SCons
  - Supports multiple tool chains;
  - Supports direct compilation using scons;
  - Supports generation of IDE project files including MDK/IAR, etc.
Package Management Tools

- Package Management System
  - Packages for different application areas, such as system, language, network protocol and application, security etc.
  - A package is composited with description information, package source code, or library files.
  - Based on RT-Thread ENV tool, packages can be added or removed easily.

- Why use the package
  - Splits kernel and packages for low coupling and high maintainability;
  - Supports continuous integration, high reliability;
  - It’s an open platform, everyone (RT-Thread development team or others) can share the package in the open source or libraries.
  - With the more software packages, the user can develop applications quickly.
  - There are currently 50+ software packages
Future of RT-Thread/RISC-V
Why RT–Thread is suitable for RISC–V

- Why RT-Thread is suitable for RISC-V?
  - Open source ISA + Open source OS
  - Rich components
  - Highly scalable, from MCU, IoT Soc, to MPU
  - The basic kernel support in HiFive1 E310 demand for resources is:
    - FLASH: 8.8KB
    - RAM: 5.4KB
Why RT-Thread is suitable for RISC-V

- Why RT-Thread is suitable for RISC-V?
  - IoT Chip + IoT OS
  - Tiny resource requirements with full features TCP/IP stack: lwIP
    - Typical footprint with 64KB ROM, 24KB RAM
  - Nano resource requirements with AT framework:
    - AT Client: 4.3KB ROM, 2.0KB RAM
    - AT Client + AT Socket + SAL: 14KB ROM, <4KB(with 5 sockets)
Why RT–Thread is suitable for RISC–V

- Why RT-Thread is suitable for RISC-V?
  - POSIX-compliant IoT OS:
    - File System
    - Dynamic Linking
    - PThread/Semaphore
    - Memory Management
    - Network
    - Device File
    - ...

---

Building the Foundation for IoT

RT-Thread
The Experience/Issues

- The Experience/Issues in RT-Thread porting, QEMU etc
  - Toolchain
    - There are several different versions of the tool chain, how to choose?
  - QEMU
    - RT-Thread/HiFive1 currently only runs on QEMU's RISC-ALL branch
    - "set riscv use_compressed_breakpoint no"
    - is required before using breakpoints
  - Debugger
    - Compared to the ARM platform, the debugger is hard to use.
RT-Thread for future planning of RISC-V:

- Support E200 soft core CPU in Lichee Tang
- Support GAP8 IoT application processor
- Helping the promotion of RISC-V
Thank You