基于RISC-V MCU的嵌入式开发
-- 以RV32M1为例

嵌入式系统联谊会
www.esbf.org

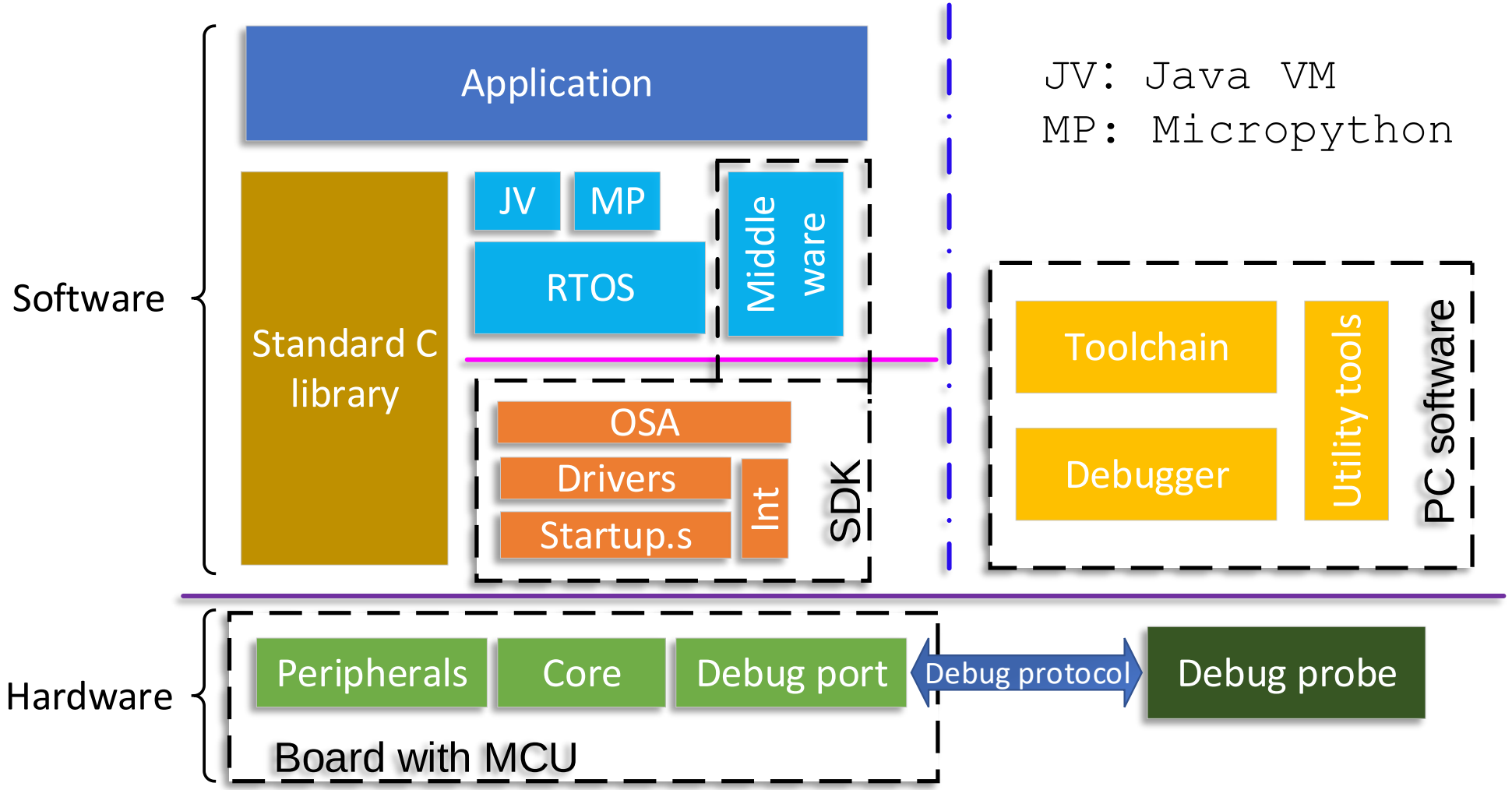
刘华东
系统应用工程师
APRIL 16, 2019

VEGA | SECURE CONNECTIONS FOR A SMARTER WORLD

主要内容

- MCU嵌入式开发软硬件构成
- RISC-V MCU嵌入式开发环境
- VEGABoard 和织女星开发板
- RV32M1 SDK 软件开发包
- 启动代码和中断处理
- RV32M1的多核应用
- RV32M1的生态链

MCU嵌入式系统软硬件构成



RISC-V MCU嵌入式开发环境

- GNU GCC RISC-V 交叉工具链
  - gcc, c/c++ compiler, as, ld, gdb 等
  - GNU二进制工具集（binutils)
    - ar, nm, objcopy, objdump, ranlib, readelf, size, strip, strings 等
  - 嵌入式C标准库
    - Newlib, Newlib-nano
  - https://github.com/pulp-platform/pulp-riscv-gnu-toolchain
- Windows Build Tools（可选）：
  - https://github.com/gnu-mcu-eclipse/windows-build-tools/releases
- Cmake
  - 代替Makefile的编译工具

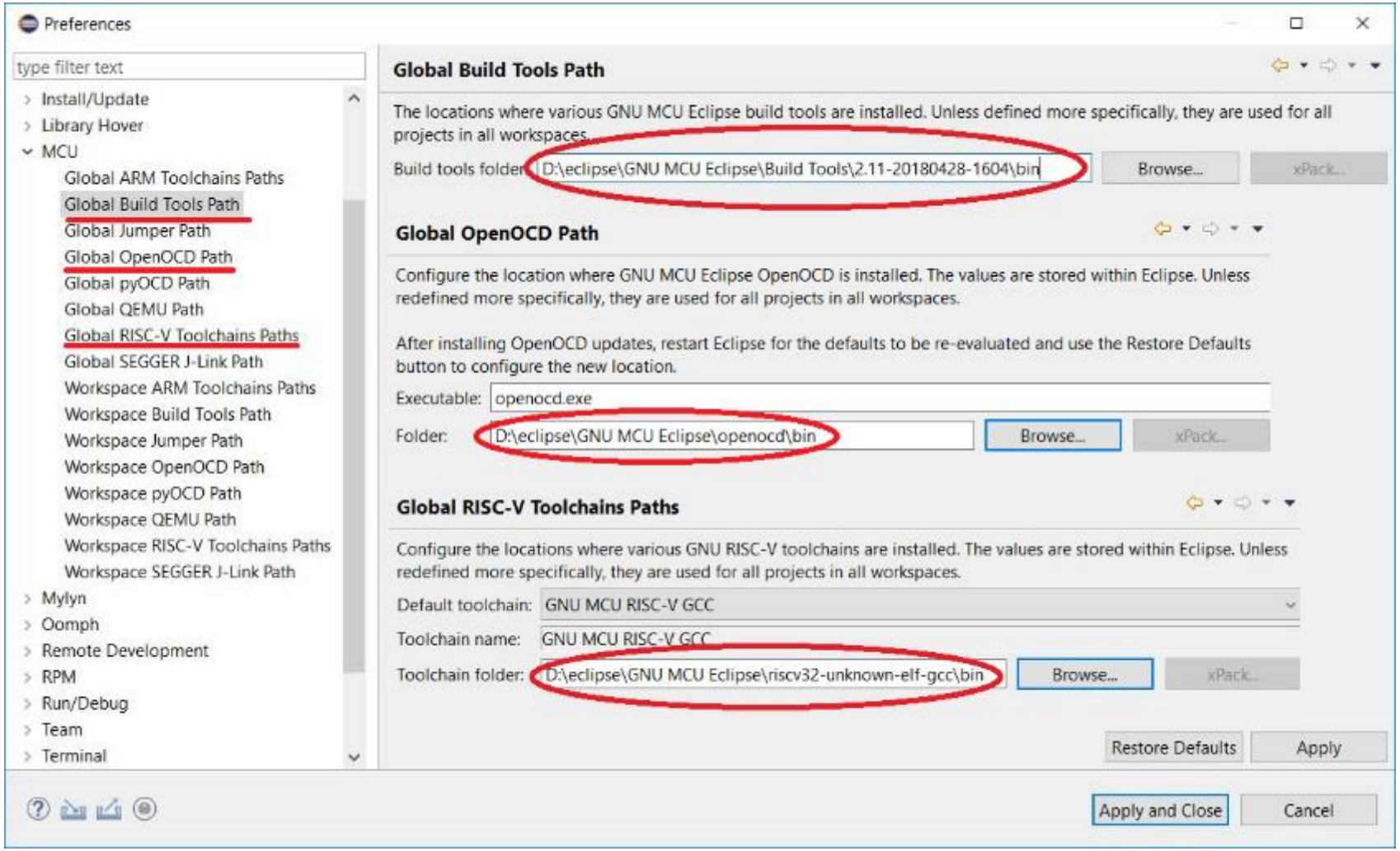
VEGA

RISC-V MCU嵌入式开发环境

- OpenOCD调试软件
  - 支持RV32M1片上Flash下载算法
  - 基于标准JTAG之上的调试协议
  - 支持实现了标准JTAG协议的调试器，J-Link及其兼容的调试器(LPC-Link2)
  - 支持GDB或者telnet调试方法，实现源代码级调试
- MingW
  - 在Windows下提供符合GNU的最小开发环境
  - 习惯于Linux下的用户，使用命令行方式开发环境
- Eclipse集成开发环境
  - 提供类似IAR，Keil和MCUXpresso的集成开发调试环境
  - https://github.com/gnu-mcu-eclipse/org.eclipse.epp.packages/releases

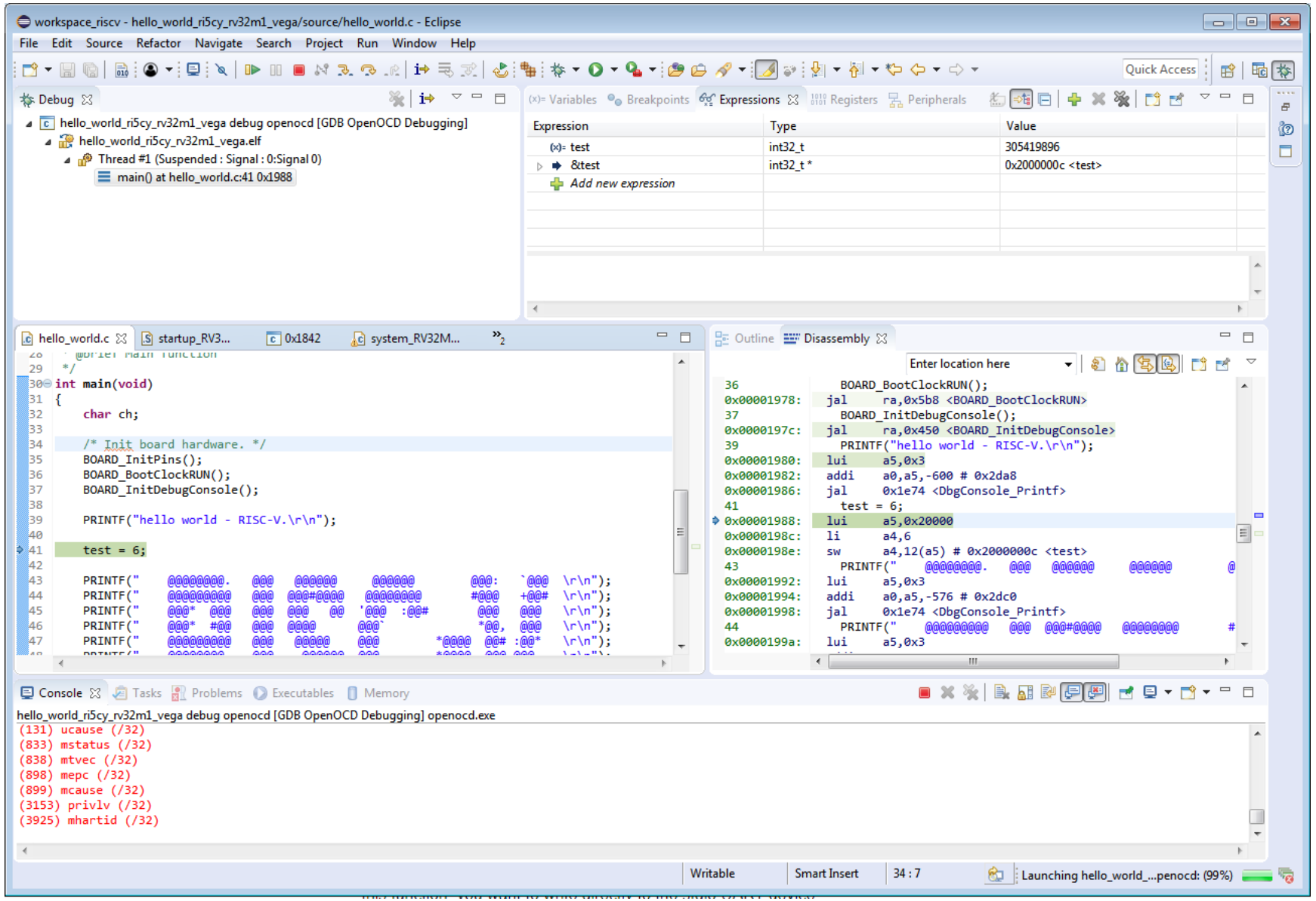
VEGA

RISC-V MCU嵌入式开发环境

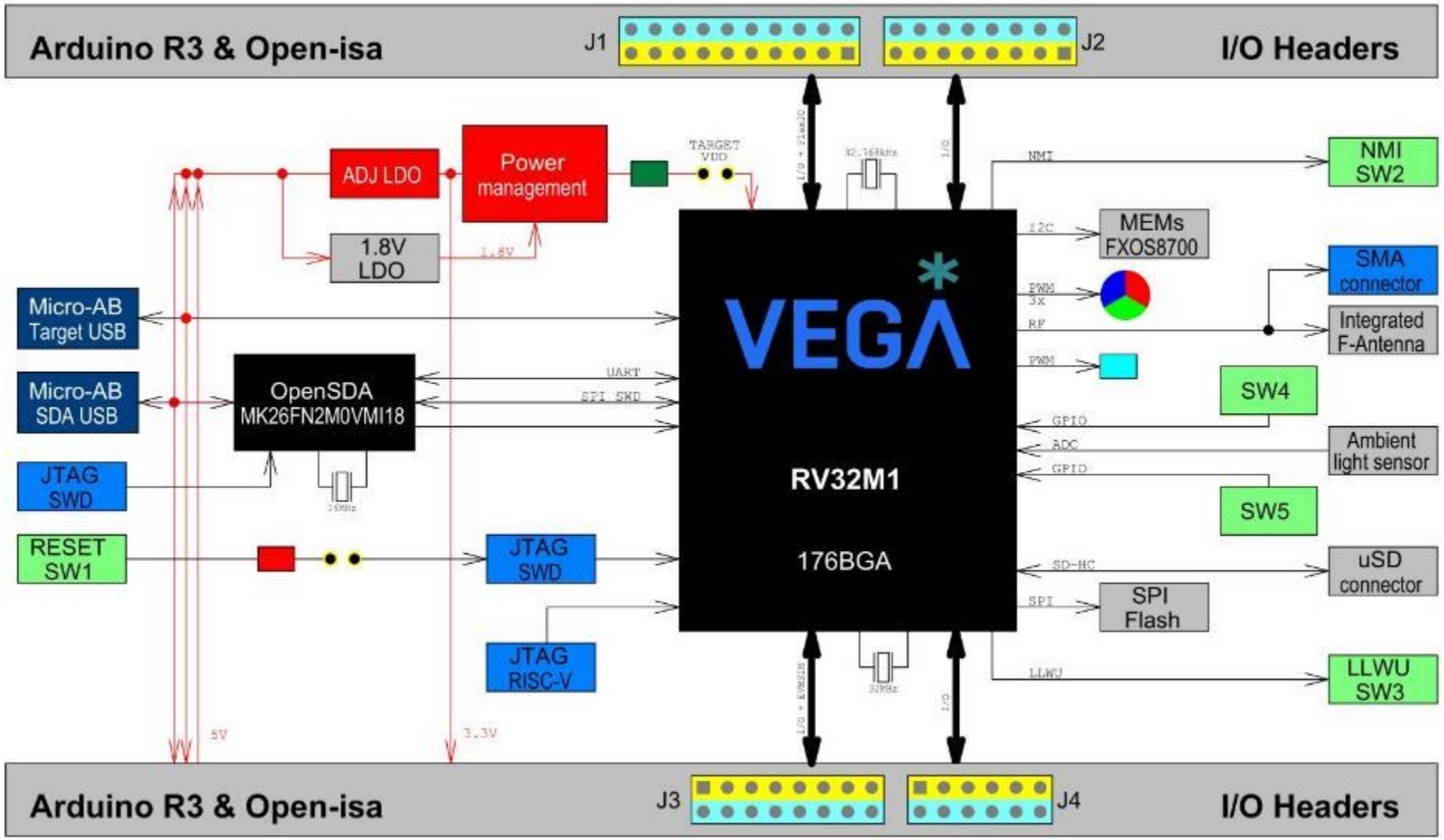


VEGA

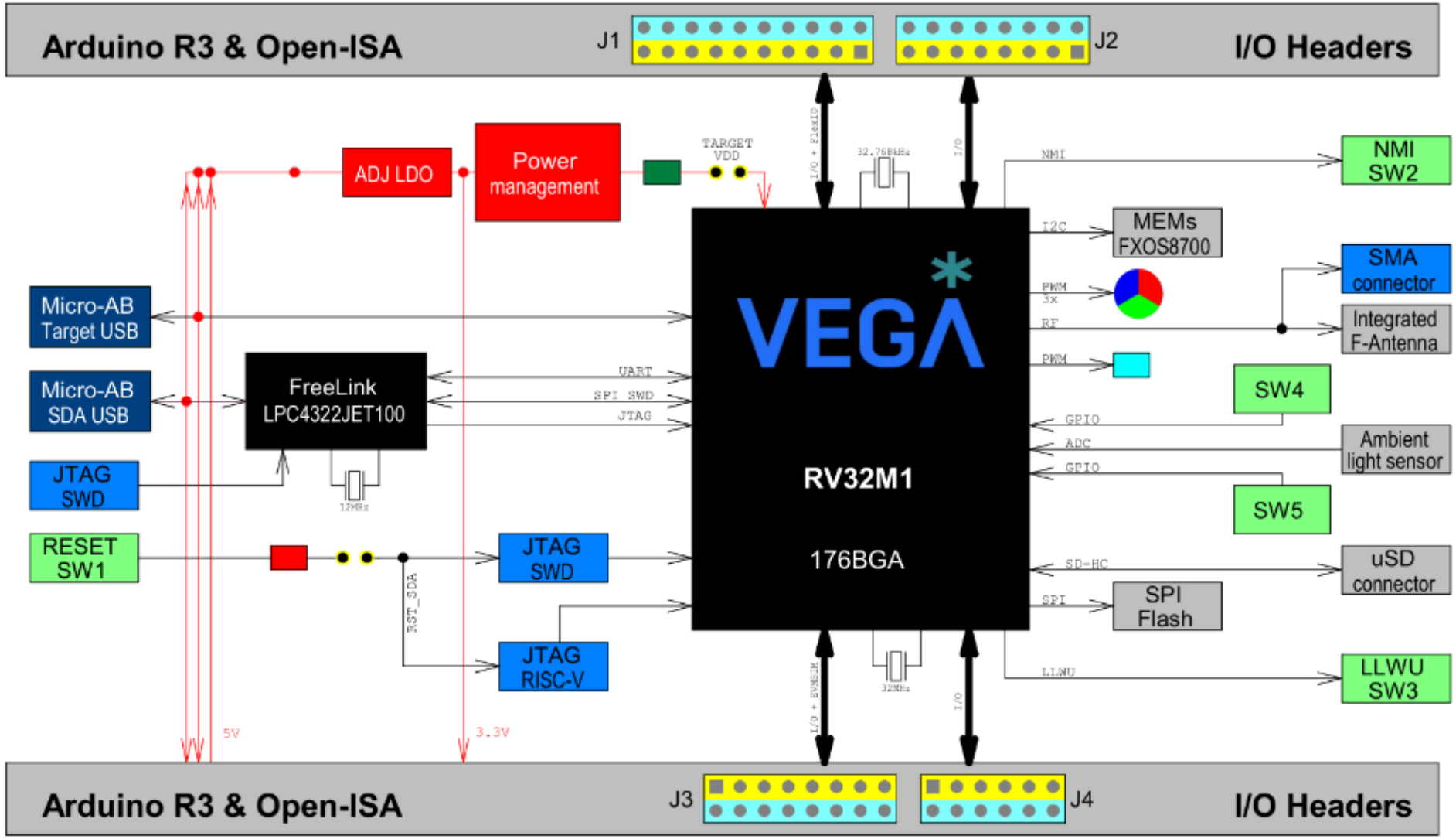
RISC-V MCU嵌入式开发环境



VEGABoard


Arduino R3 & Open-isa
J1
J2
I/O Headers
ADJ LDO
Power management
TARGET VDD
1.8V LDO
1.8V
Micro-AB Target USB
Micro-AB SDA USB
JTAG SWD
RESET SW1
OpenSDA MK26FN2M0VMI18
UART
SPI SWD
JTAG SWD
JTAG RISC-V
5V
3.3V
VEGA
RV32M1
176BGA
NMI
NMI SW2
I2C
MEMs FXOS8700
PWM 3x
RF
SMA connector
Integrated F-Antenna
PWM
SW4
GPIO
ADC
GPIO
Ambient light sensor
SW5
SD-HC
uSD connector
SPI
SPI Flash
LLWU
LLWU SW3
Arduino R3 & Open-isa
J3
J4
I/O Headers

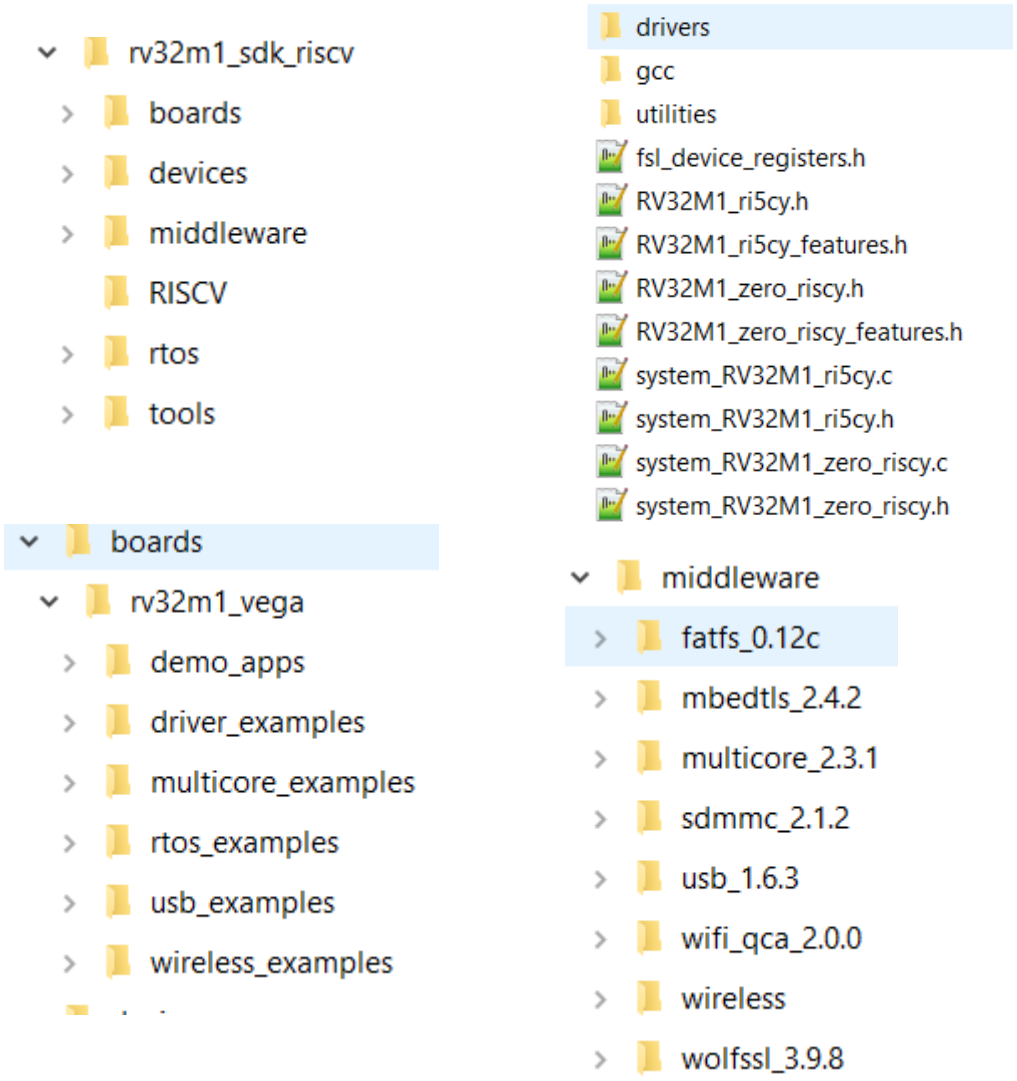
织女星开发板


Arduino R3 & Open-ISA
J1
J2
I/O Headers
ADJ LDO
Power management
TARGET VDD
NMI SW2
MEMs FXOS8700
SMA connector
Integrated F-Antenna
Micro-AB Target USB
Micro-AB SDA USB
FreeLink LPC4322JET100
UART
SPI SWD
JTAG
VEGA
RV32M1
176BGA
SW4
Ambient light sensor
SW5
JTAG SWD
RESET SW1
JTAG SWD
JTAG RISC-V
RST_SDA
uSD connector
SPI Flash
LLWU SW3
5V
3.3V
Arduino R3 & Open-ISA
J3
J4
I/O Headers

RV32M1的介绍

- RV32M1是异构四核MCU：
  ① 一个RISC-V RI5CY 核、一个RISC-V ZERO_RISCY 核
  ② 一个ARM Cortex-M4F核、一个Cortex-M0+核
- RV32M1是一个超低功耗的高集成度的芯片，
  - 单芯片就可以支持低功耗蓝牙(BLE)、Generic FSK（支持250/500/1000/2000 kbps）
  - IEEE 802.15.4 标准，可运行BLE Mesh/Thread/Zigbee协议特别适用超低功耗的移动设备
  - 低功耗的密码加速单元，支持AES128/196/256, DES/3DES, SHA-256, RSA and ECC PK-256/Curve25519
  - USB2.0 FS、SAI支持I2S和AC‘97、SDHC、 EMVSIM
  - 1x32ch FlexIO、4xUART、4xI2C、 4x16-bit LPSPI、1x12bit ADC
  - 2x6ch的PWM，1x2ch的PWM，RTC， LPTimer
  - 1.25 MB Flash 、384 KB SRAM

RV32M1 SDK 软件开发包



- Boards
  - 大量的例子工程及其代码、板级相关代码（管脚，系统时钟配置）
- Devices
  - RV32M1芯片的头文件定义、外设的驱动源代码、每个核的启动代码和调试串口打印实现代码
- Middleware
  - 多核管理库，USB设备和OS抽象层、BLE协议栈等源代码

启动代码和中断处理

- 启动代码
  - 初始化堆栈、全局变量的指针、设置MTVEC和UTVEC来设置中断入口地址
  - 建立C运行时环境 （LMA和VMA）
    - 初始化全局变量和静态变量，没有初始化的值为零
    - Flash中的代码拷贝到RAM中运行。

```
copy_section(&__etext, &__data_start__, &__data_end__);
zero_section(&__bss_start__, &__bss_end__);
```

  - 建立C/C++运行时环境
    - 初始化C/C++对象

```
/* Reset Handler */
Reset_Handler:

    # Disable global interrupt. */
    csrci mstatus, 8

    # initialize stack pointer
    la sp, __StackTop

    # initialize global pointer
    la gp, __global_pointer

#ifndef __NO_SYSTEM_INIT
    jal SystemInit
#endif

    call __libc_init_array

    # Enable global interrupt. */
    csrsi mstatus, 8

    jal main
    ebreak
```


VEGA

启动代码和中断处理 -初始化C运行时环境

- Linker Scripts

```
.bss :
{
  /* This is used by t
  . = ALIGN(4);
  __START_BSS = .;
  __bss_start__ = .;
  *(.bss)
  *(.bss*)
  *(.sbss)
  *(.sbss*)
  *(COMMON)
  . = ALIGN(4);
  __bss_end__ = .;
  __END_BSS = .;
} > m_data
```

```
__etext = .;    /* define a global symbol at end of code */
__global_pointer = .;    /* define a global symbol at end of code */
__DATA_ROM = .; /* Symbol is used by startup for data initialization */

.data : AT(__DATA_ROM)
{
  . = ALIGN(4);
  __DATA_RAM = .;
  __data_start__ = .;      /* create a global symbol at data start */
  *(.data)                 /* .data sections */
  *(.data*)                /* .data* sections */
  *(.sdata .sdata.*)
  *(.heapsram*)            /* This is only for the pulpino official test code. */
  __noncachedata_start__ = .;   /* create a global symbol at ncache data start */
  *(NonCacheable)
  __noncachedata_end__ = .;     /* define a global symbol at ncache data end */
  KEEP(*(.jcr*))
  . = ALIGN(4);
  __data_end__ = .;        /* define a global symbol at data end */
} > m_data

__DATA_END = __DATA_ROM + (__data_end__ - __data_start__);
text_end = ORIGIN(m_text) + LENGTH(m_text);
ASSERT(__DATA_END <= text_end, "region m_text overflowed with text and data")

_edata = .;
```


VEGA

启动代码和中断处理

- TVEC寄存器

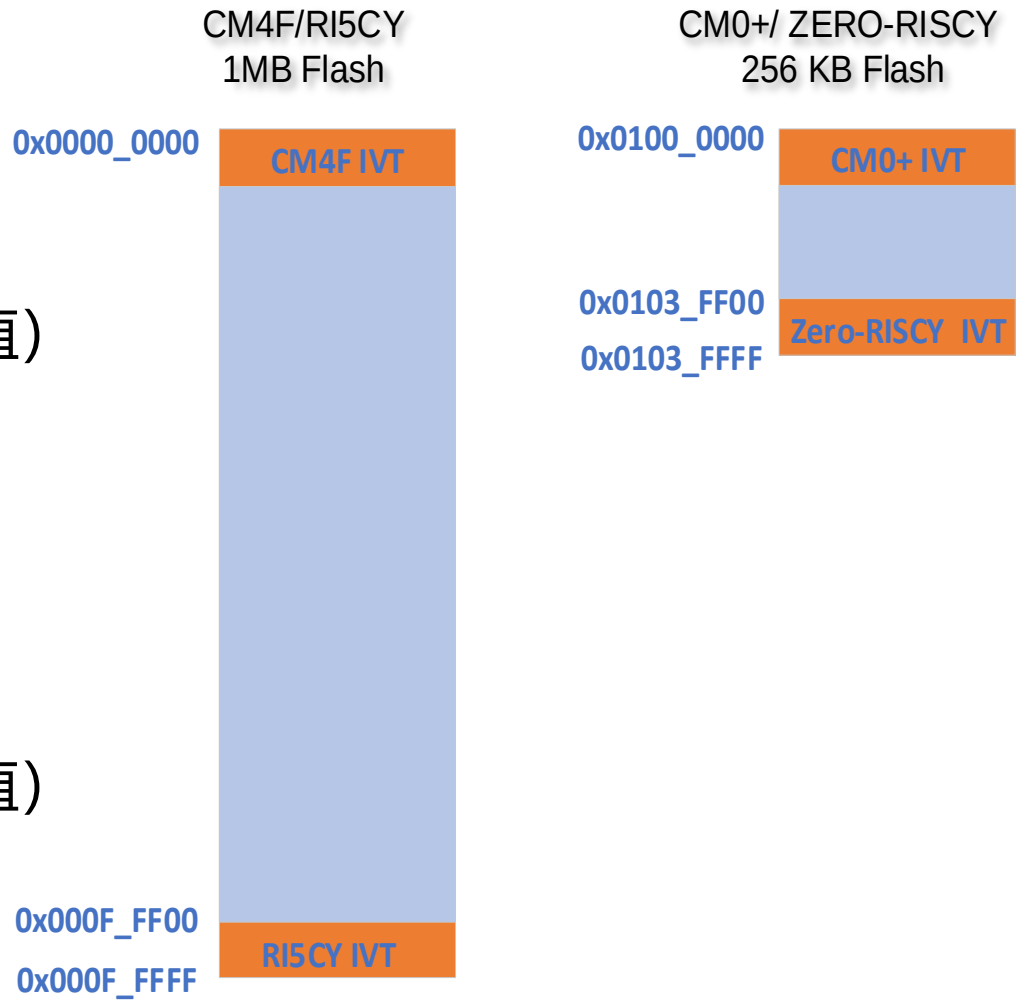
  - BASE[XLEN-1:2]，中断处理相关的地址，意义由MODE的值决定
  - MODE=0时，所有中断发生时pc=BASE；
  - MODE=1时，中断发生时pc=BASE+4*cause
- RISC-V标准化的PLIC中断控制器
- RV32M1非标准的向量化中断控制器

VEGA

启动代码和中断处理

- RV32M RI5CY核的中断向量表（IVT）
  - 中断向量表：0x000F_FF00~0x000F_FF7F
  - 系统异常表：0x000F_FF80~0x000F_FFFF
  - Reset异常执行位置：0x000F_FF80(复位后pc的值)

- RV32M Zero_riscy核的中断向量表（IVT）
  - 中断向量表：0x0103_FF00~0x0103_FF7F
  - 系统异常表：0x010F_FF80~0x010F_FFFF
  - Reset异常执行位置：0x010F_FF80(复位后pc的值)



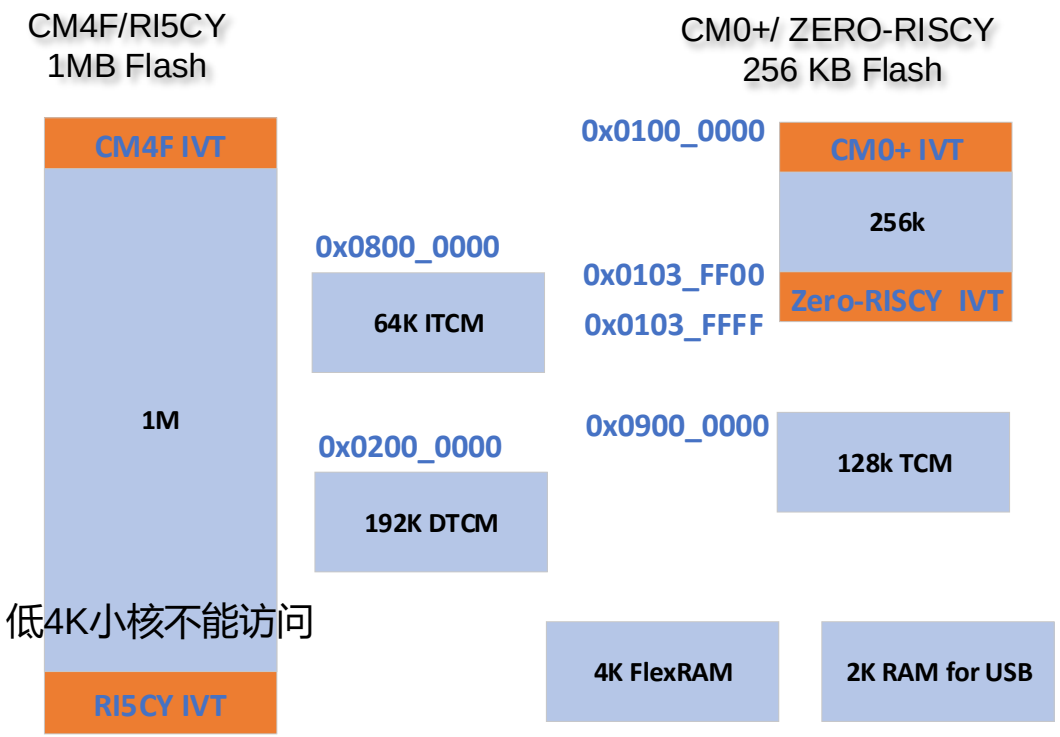
启动代码和中断处理

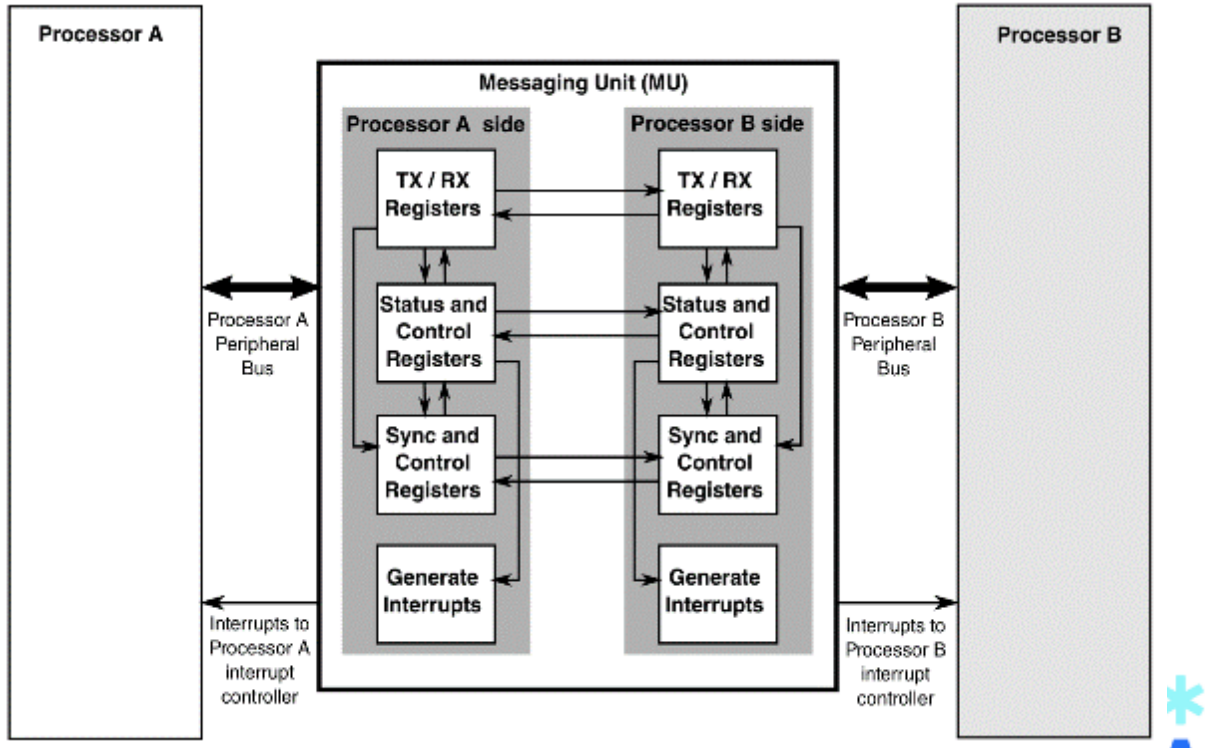
- RV32M1支持的RISC-V私有扩展
  - 乘加，比特计数等DSP相关的指令
  - 硬件循环指令（两个硬件循环模块）执行循环操作零跳转开销
  - 中断入口保存和中断返回恢复系统寄存器时有6个额外的寄存要保存

VEGA

RV32M1的多核应用

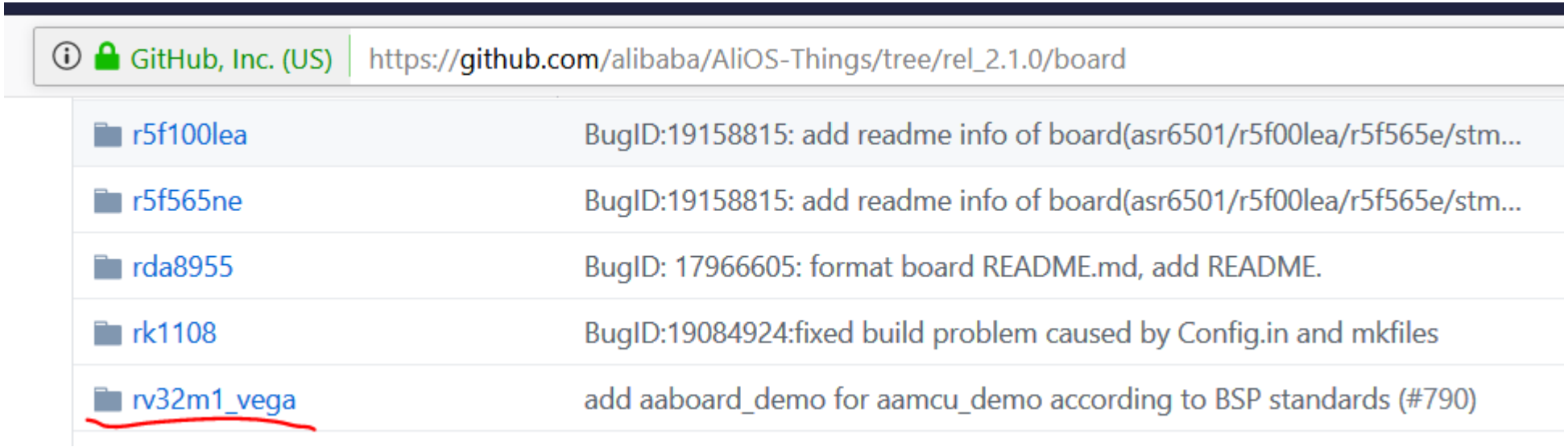
- 支持多核的功能模块：
  - 消息传递单元(MU)、Semaphore2(SEMA42)、扩展的资源域控制器(XRDC)
  - SDK提供抽象的中间件，包括多核管理器和轻量级的RPMsg，提供简单易用的API





RV32M1的生态链

- 自去年12月份VEGABoard发布以来，在RISC-V社区很受欢迎
- 开源社区许多开发者和各个合作伙伴都积极提供对VEGABoard的支持
- 国内AliOS Things， https://github.com/alibaba/AliOS-Things

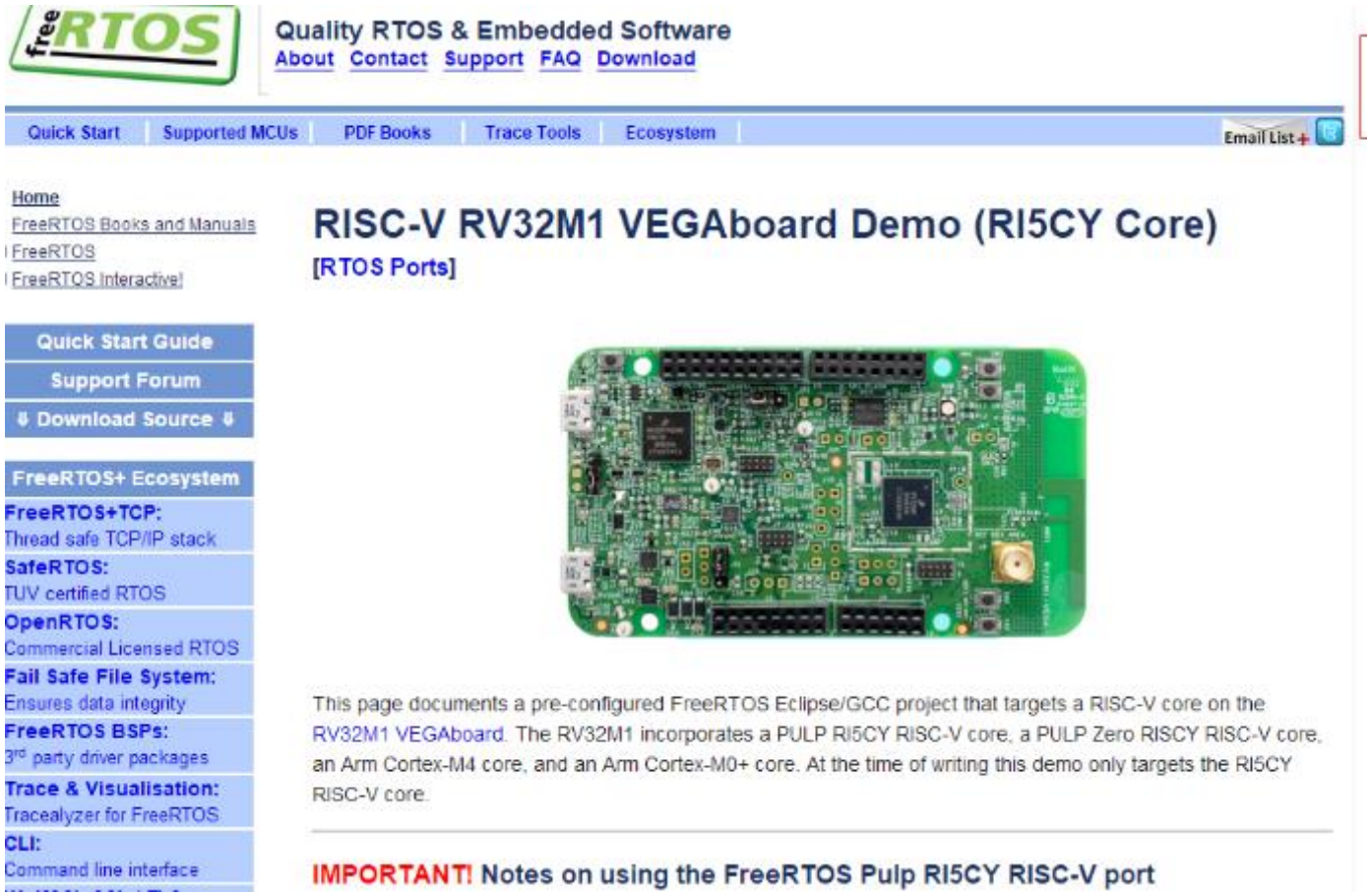


RV32M1的生态链

- RT-Thread
  - https://github.com/RT-Thread/rt-thread

RV32M1的生态链

- Amazon FreeRTOS
  - https://www.freertos.org/RTOS-RISC-V-Vegaboard_Pulp.html



RV32M1的生态链

- Zephyr
  - https://github.com/zephyrproject-rtos/zephyr
  - https://docs.zephyrproject.org/latest/boards/riscv32/rv32m1_vega/doc/rv32m1_vega.html

RV32M1的生态链

- Micropython语言
  - Zephyr，RT-Tread 社区已经将Micropython移植到各自的RTOS上
  - 来自社区的个人也有将Micropython直接移植到VEGABoar板子上
    - https://github.com/AaronKel/micropython-vega
  - 我们自己也开始Micropython移植，目前已经完成time模块，mcu模块，machine模块，pyb模块
  - 会将目前的代码整理后，上传到Github，欢迎更多开源社区的开发人员加入
- Forth语言 (Mecrisp-Quintus Forth)
  - https://sourceforge.net/projects/mecrisp/files/
  - Forth的核心部分是纯手工写的RISC-V汇编代码，想学习RISC-V汇编人，可以学习参考

VEGA

THANKS

VEGA

Q&A