**PODES\_M0O**

**Implementation User Manual**

**Ver1.2**

**Index:**

[1 概述 4](#_Toc30932502)

[2 对象和范围 6](#_Toc30932503)

[3 支持和服务 7](#_Toc30932504)

[4 PODES-M0O 功能特性 8](#_Toc30932505)

[4.1 支持的特性 8](#_Toc30932506)

[4.2 不支持的特性 10](#_Toc30932507)

[5 PODES\_M0O系统结构 11](#_Toc30932508)

[5.1 PODES-M0O功能框图 11](#_Toc30932509)

[5.2 PODES-M0O Memory Mapping 12](#_Toc30932510)

[5.3 PODES-M0O中断 13](#_Toc30932511)

[5.4 PODES-M0O代码层次结构 14](#_Toc30932512)

[6 系统控制块 15](#_Toc30932513)

[6.1 系统控制寄存器 17](#_Toc30932514)

[6.2 系统定时器 23](#_Toc30932515)

[6.3 系统中断控制器 26](#_Toc30932516)

[7 Debug 功能 29](#_Toc30932517)

[7.1 Debug 寄存器空间 29](#_Toc30932518)

[7.2 功能实现 31](#_Toc30932519)

[8 PODES-M0O寄存器 32](#_Toc30932520)

[8.1 LR 32](#_Toc30932521)

[8.2 PC 32](#_Toc30932522)

[8.3 MSP, PSP 33](#_Toc30932523)

[8.4 xPSR 33](#_Toc30932524)

[8.5 PRMASK 34](#_Toc30932525)

[8.6 CONTROL 35](#_Toc30932526)

[9 PODES\_M0O指令集 36](#_Toc30932527)

[9.1 PODES-M0O指令 36](#_Toc30932528)

[9.2 PODES-M0O指令编码列表 36](#_Toc30932529)

[10 PODES\_M0O应用指南 42](#_Toc30932530)

[10.1 时钟和复位接口 42](#_Toc30932531)

[10.2 外部中断接口 43](#_Toc30932532)

[10.3 总线接口 44](#_Toc30932533)

[10.4 PODES-M0O应用 46](#_Toc30932534)

[Change Summary 47](#_Toc30932535)

[CopyLeft 48](#_Toc30932536)



# 概述

**是什么？**

**PODES**是**P**rocessor **O**ptmization for **D**eeply **E**mbedded **S**ystem的首字母缩写。包括传统的51指令集架构，SparcV8指令集架构，ARMv6-M指令集架构以及PIC-16指令集架构等一系列MCU Core。

**PODES-M0O** 是兼容于ARMv6-M指令集架构的开源版本。**M0**指代Cortex-M0，**O**指代Open Source。

**为什么？**

做这个工作最开始的原因很简单，无聊和好玩。到后来觉得有趣，或许还有那么一点用处，所以就坚持做下来了。

**有什么用？**

**O**（Open Source）系列最基本的作用当然是学习和研究。稍微认真地搜索一下国内网络资源就会发现，很难找到有质量的IC设计开源项目。作者相信这个项目应该对开源社区会有助益。

**是谁？**

这不是一群人在战斗。这个项目是一个工程师在业余时间做的。

**愿景？**

使深嵌入式应用的MCU Core IP的License费用趋近于0。这个期望真的不过分。

**能否达成愿景？**

完全依赖于您的热心帮助！

小额赞助、购买FPGA开发板、提供开发支持、甚至是一条建议或者评论，都是鼓舞PODES前行的动力。

有意赞助者，可以扫描下方的二维码：



# 对象和范围

**PODES-M0O**是一个经过专门精简优化的开源版本，定位于学习和研究。任何想研究ARMv6-M或者Cortex-M0的人或者机构都可以从PODES-M0O获得帮助和启发。

本手册是**PODES-M0O** IP的文档描述。内容包括：系统结构、指令集、功能模块、全部寄存器定义、以及应用接口指南。阅读本文档可以方便用户完整地理解**PODES-M0O** IP的设计思路和实现的功能。

**PODES-M0O**不做修改或者稍作改动，可以直接应用于FPGA产品。用于ASIC实现则需要一些额外的设计修改工作。

PODES-M0O设计实现用户手册：（本文档）

***PODES-M0O\_Implementation\_User\_Manual\_Vxx.doc***

PODES-M0O应用用户手册：

***PODES-M0O\_Application\_User\_Manual\_Vxx.doc***

PODES-M0O评估板用户手册：

***PODES\_M0O\_Evaluation\_Board\_User\_Manual\_Vxx.doc***

Cortex-M0的相关资料，下面的文档可供参考：

***DDI0432C\_cortex\_m0\_r0p0\_trm.pdf***

***DUI0497A\_cortex\_m0\_r0p0\_generic\_ug.pdf***

***DDI0419B\_arm\_architecture\_v6m\_reference\_manual\_errata\_markup\_2\_0.pdf***

# 支持和服务

[www.mcucore.club](https://www.mcucore.club/) 是PODES开源项目的官方维护网站。

立足于保证PODES有用，作者会持续地维护这个项目。所有代码和文档资料的最新版本都可以从下面网站获得：

[www.mcucore.club](https://www.mcucore.club/)

所有的Issue Report或者优化建议，请投送到：[www.mcucore.club](https://www.mcucore.club/) 相关的页面，或者：[podes.mcu@qq.com](mailto:podes.mcu@qq.com) 。

# PODES-M0O 功能特性

## 支持的特性

PODES-M0O指令集设计参照ARMv6-M Architecture Reference Manual文档实现。PODES-M0O的功能模块设计参照Cortex-M0 generic user guide和Cortex-m0 technical reference manual两个文档实现。

PODES-M0O完全兼容Cortex-M0内核。为了更清楚地表明“兼容”的含义，下面逐条列出 Cortex-M0 TRM文档中的Features，对比说明。

* A low gate count processor that features:
* The ARMv6-M Thumb instruction set. （支持）
* Thumb-2 technology （支持）
* Optionally, an ARMv6-M compliant 24-bit SysTick timer（支持）
* A 32-bit hardware multiplier. This can be the standard single-cycle multiplier, or a 32-cycle multiplier that has a lower area and performance implementation. （支持单周期乘法器行为模型，结构化实现需要用户修改）
* The system interface supports either little-endian or byte invariant big-endian data accesses （支持Little-endian模式）
* The ability to have deterministic, fixed-latency, interrupt handling（支持）
* Load/store-mulitples and multicycle-multiplies that can be abandoned and restarted to facilitate rapid interrupt handling（支持）
* C Application Binary Interface compliant exception model（支持）
* Low power sleep-mode entry using Wait For Interrupt (WFI), Wait For Event (WFE) instructions, or the return from interrupt sleep-on-exit feature （不支持4个Hints指令。M0A版本支持）
* NVIC that features:
* 1,2,4,8,16,24 or 32 external interrupt inputs, each with four levels of priority

（支持32 外部中断和可配置4级优先级）

* Dedicated Non-Maskable Interupt (NMI) input（支持）
* Support for both level-sensitive and pulse-sensitive interrupt lines（支持）
* Optional Wake-up interrupt Controller (WIC), providing ultra-low power sleep mode support（不支持）
* Optional debug support: （不支持，M0A支持）
* Zero to four hardware breakpoints
* Zero to two watchpoints
* Program Counter Sampling Register (PCSR) for non-intrusive code profiling, if at least one hardware data watchpoint is implemenened
* Single step and vector catch capabilities
* Support for unlimited software breakpoints using BKPT instruction
* Non-intrusive access to core peripherals and zero-waitstate system slaves through a compact bus matrix. A debugger can access these devices, including memory, even when the processor is running.
* Full access to core registers when the processor is halted.
* Optional, lwo gate-count coresight compliant debug access through a Debug Access Port (DAP) supporting either Serial Wire or JTAG debug connections.
* Bus interfaces:
* Single 32-bit AMBA-3 AHB-Lite system interface that provides simple integration to all system peripherals and memory（支持）
* Single 32-bit slave port that supports the DAP（不支持，M0A支持）

## 不支持的特性

PODES-M0O是按照Cortex-M0内核的规范设计的。在保证可用性的前提下，有些功能在PODES-M0O中未实现。这样代码看起来更简洁，结构更条理。

* Debug功能不支持

内核评估可以不需要这个功能，ROM版本ASIC实现也不需要这个功能。

（M0A实现此功能模块）

* Hints指令不支持（SEV, WFI, WFE, YIELD）

这四条指令执行有标志信号引出，用户在低功耗或者多处理器设计中可以使用这些信号。

如果代码中有这些指令，PODES-M0O会当成NOP执行。指令流水不会停止。

（M0A实现低功耗管理功能）

* Cortex –M0的可配置特性

PODES-M0O本身源代码开源，用户可以任意修改配置。没有必要提供功能配置选项，人为把代码搞复杂。

* 乘法器和加法器使用行为模型实现

ASIC综合或者FPGA综合可以直接调用工艺库提供的宏单元，或者用户自己设计结构化代码。

# PODES\_M0O系统结构

## PODES-M0O功能框图

PODES-M0O采用三级流水结构。指令处理单元分为取指，译码和执行三个模块。流水线控制、数据相关、结构相关、分支转移、exception插入等控制都统一由主状态机完成。

PODES-M0O的系统控制包括NVIC，System-tick Timer以及PPB空间寄存器三个部分。PPB空间中与Debug功能相关的寄存器没有实现。



PODES-M0O提供AHBLite 总线接口、32个IRQ和1个NMI中断输入。外部功能模块可以使用AMBA总线连接到PODES-M0O。

## PODES-M0O Memory Mapping

Cortex-M0定义了如下的Memory空间，PODES-M0O的实现和它完全兼容。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **address** | **Name** | **Device**  **Type** | **XN** | **Cache** | **Description** |
| 0x0000\_0000 ~  0x1FFF\_FFFF | Code | Normal | - | WT | Typical ROM or flash memory. Memory required from address 0x0 to support the vector table for system boot code on reset. |
| 0x2000\_0000 ~  0x3FFF\_FFFF | SRAM | Normal | - | WBWA | SRAM region typically used for on-chip RAM. |
| 0x4000\_0000 ~  0x5FFF\_FFFF | Peripheral | Device | XN | - | On-chip peripheral address space. |
| 0x6000\_0000 ~  0x7FFF\_FFFF | RAM | Normal | - | WBWA | Memory with write-back, write allocate cache attribute for L2/L3 cache support. |
| 0x8000\_0000 ~  0x9FFF\_FFFF | RAM | Normal | - | WT | Memory with write-through cache attribute. |
| 0xA000\_0000 ~  0xBFFF\_FFFF | Device | Device  Shareable | XN | - | Shareable device space. |
| 0xC000\_0000 ~  0xDFFF\_FFFF | Device | Device | XN | - | Non-shareable device space. |
| 0xE000\_0000 ~  0xFFFF\_FFFF | System | - | - | - | System segment including the PPB. |

系统空间已经在PODES-M0O内部实现。

PODES-M0O顶层模块提供了AHBLite 总线Master接口。这个总线可以访问除0xE000\_0000 ~ 0xFFFF\_FFFF之外的全部地址范围。

## PODES-M0O中断

中断列表：

|  |  |
| --- | --- |
| **IRQ#** | **Description** |
| nmi | 不可屏蔽中断。外部端口。 |
| Irq[31:0] | 外部中断。外部端口。 |

PODES-M0O支持32个外部中断源和一个不可屏蔽中断。支持电平和边沿（脉冲）触发中断。功能与Cortex-M0完全兼容。

## PODES-M0O代码层次结构



Adder32.v

加法器，PODES-M0O使用行为模型。用户可以根据需要更改成自己专用的结构实现。

Mul32x32.v

乘法器，PODES-M0O使用行为模型。用户可以根据需要更改成自己专用的结构实现。

EMULATOR\_M0O.v

指令模拟器，代码仿真时可以输出反汇编代码。用户可以使用它辅助自己的程序仿真。

# 系统控制块

System Control 功能被分成三个部分：

1. 系统控制寄存器
2. 系统定时器
3. 系统中断控制器

寄存器空间分配如下图兰色部分所示。这部分功能属于软件正常运行所需的功能，**全部寄存器与Cortex-M0的完全一致**（除非有设计错误-:）。红颜色标注的功能是PODES-M0O提供的扩展能力。



## 系统控制寄存器

全部系统控制寄存器只能按照Word (32bit)方式读写。HalfWord，Byte读写不支持。

所有未实现的寄存器，读返回0，写没有影响。

**Auxiliary Control Register (0xE000\_E008)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:0] | RW | 0x0 | Implementation Defined |

这个留作内部实现专用功能。目前只是实现32bit寄存器读写功能，系统软件可以写入“-M0O”四个字符，用于标识PODES MCUCore家族的不同版本。

**CPUID Base Register (0xE000\_ED00)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:24] | RO | 0x41 | Implementer code assigned by ARM. ARM = 0X41. |
| [23:20] | RO | 0 | Variant. Implementation Defined. |
| [19:16] | RO | 0xC | Read as 0xC for ARMv6-M parts |
| [15:4] | RO | 0xc20 | Part No. implementation Defined （cortex-m0: 0xc20） |
| [3:0] | RO | 0 | Revision. Implementation Defined. （0） |

这个自己定义Variant （CortexM0 Clone）， part No和Revision自己规定。这个寄存器实现了读写功能。用户软件可以重写寄存器的值，建议保留当前值。

**Interrupt Control and State Register (0xE000\_ED04)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31] | RW | 0 | Nmipendset.  Setting this bit will activate an NMI.  Since NMI is the highest priority exception, it will activate as soon as it is registered. Reads back with current state (1 if pending, 0 if not). |
| [30:29] | RO | 0 | Resvered |
| [28] | RW | 0 | Pendsvset.  Set a pending PendSV interrupt. This is normally used to request a context switch.  Reads back with current state (1 if pending, 0 if not) |
| [27] | WO | 0 | Pendsvclr.  Clear a pending PendSV interrupt. |
| [26] | RW | 0 | Pendstset.  Set a pending SysTick.  Reads back with current state (1 if pending, 0 if not) |
| [25] | WO | 0 | Pendstclr.  Clear a pending SysTick (Wether set here or by the timer hardware). |
| [24] | RO | 0 | Reserved. |
| [23] | RO | 0 | Isrpreempt.  If set, a pending exception will be serviced on exit from the debug halt state. The bit applies to the Debug Extension only, otherwise it is reserved. |
| [22] | RO | 0 | Isrpending.  Indicates if an external configurable (NVIC generated) interrupt is pending. This bit applies to the Debug Extension only, otherwise it is reserved. |
| [21] | RO | 0 | Reserved. |
| [20:12] | RO | 0 | Vectpending.  Indicates the exception number for the highest priority pending exception. The pending state includes the effect of memory-mapped enable and mask registers. It does not include the PRIMASK special-purpose register qualifier. A value of zero indicates no pending exceptions. |
| [8:0] | RO | 0 | Vectactive.  0: Thread mode.  Value > 1: the exception number for the current executing exception. Debug Extension only, otherwise reserved. |

**Vector Table Offset Register (0xE000\_ED08)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:0] | RO | 0 | Fixed value. For ARMv6-M, Vector table base Address is 0x0000\_0000. |

**Application Interrupt and Reset Control Register (0xE000\_ED0C)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:16] | WO | 0 | Vector Key.  Should be written with the value 0x05FA, otherwise the register write is unpredictable. |
| [31:16] | RO | 0 | Vector key state. Unpredictable. |
| [15] | RO | 0 | Endianess.  0: little endian. 1: big endian. |
| [14:3] | RO | 0 | Reserved. |
| [2] | WO | 0 | Systemresetreq.  Writing this bit 1 will cause a signal to be asserted to the external system to indicate a reset is requested.  The bit self-clears as part of the reset sequence. |
| [1] | WO | 0 | Vectclractive.  Clears all active state information for fixed and configurable exception.  This bit self-clears and is classified as a debug resource. This bit can only be written when the core is halted.  It is the debugger’s responsibility to re-initialize the stack.  It is implementation defined whether VECTCLRACTIVE also clears pending status associated with any non-cofigurable exceptions, specifically HardFault and NMI. |
| [0] | RO | 0 | Reserved. |

Bit15的值来自系统配置参数ENDIANESS输入。PODES-M0O固定为little-endian模式。

**System Control Register (0xE000\_ED10)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:5] | RO | 0 | reserved |
| [4] | RW | 0 | Sevonpend.  When enable, interrupt transitions from inactive to pending are included the list of wakeup events for the WFE instruction. |
| [2] | RW | 0 | Sleepdeep.  A qualifiying hint that indicates waking from sleep might take longer. Implementations can take advantage of the feature to identify a lower power sleep state. |
| [1] | RW | 0 | Sleeponexit.  When set, the implementation can enter a sleep state on an exception return that loads the IPSR with Exception Number == 0. |
| [0] | RO | 0 | Reserved. |

PODES-M0O实现了这些寄存器位，但是没有提供对应的功能。PODES-M0O本身不支持Sleep和Wakeup事件。

**Configuration and Control Register (0xE000\_ED14)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:10] | RO | 0 | reserved |
| [9] | RO | 1 | Stkalign.  RAO: on exception entry, the SP used prior to the exception is adjusted to be 8-byte aligned and the context to restore it is saved. The SP is restored on the associated exception return. |
| [3] | RO | 1 | Unalign\_trp.  RAO: unaligned word and halfword accesses generate a Hardfault exception. |
| [2:0] | RO | 0 | Reserved. |

**System Handler Priority Register2 (0xE000\_ED1C)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:30] | RW | 0 | Pri\_11.  Priority of system handler 11 – SVCall. |
| [29:0] | RO | 0 | Reserved. |

**System Handler Priority Register3 (0xE000\_ED20)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:30] | RW | 0 | Pri\_15.  Priority of system handler 15 – SysTick. |
| [29:24] | RW | 0 | Reserved. |
| [23:22] | RW | 0 | Pri\_14.  Priority of system handler 14 – PendSV. |
| [21:0] | RW | 0 | Reserved. |

**System Handler Control and State Register (0xE000\_ED24)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:16] | RO | 0 | Reserved. |
| [15] | RW | 0 | Svcallpended.  Reads as 1 if SVCall is pending.  A state bit that is set when the exception started to invoke but was replaced by a higher priority exception. |
| [14:0] | RO | 0 | Reserved. |

## 系统定时器

与Cortex-M0 完全兼容。Calibration register 与实现相关。不支持外部参考时钟，默认的Cali值为10ms/50Mhz。

**SysTick Control and Status Register (0xE000\_E010)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:17] | RO | 0 | Reserved. |
| [16] | RO | 0 | Countflag.  Returns 1 if timer counted to 0 since last time this register was read. COUNTFLAG is set by a count transition from 1 to 0. COUNTFLAG is cleared on read or by a write to current value register. |
| [2] | RW | 0 | Clksource.  0: clock source is (optional) external reference clock.  1: core clock used for SysTick.  If no external clock provided, this bit will read as 1 and ignore writes. |
| [1] | RW | 0 | Tickint.  1: counting down to 0 will cause the SysTick exception to be pended. Clearing the SysTick current value register by a register write in software will not cause SysTick to be pended.  0: counting down to 0 does not cause the SysTick exception to be pended. Software can use COUNTFLAG to determine if a count to zero has occurred. |
| [0] | RW | 0 | Enable.  0: the counter is disabled.  1: the counter will operate in a multi-shot manner. |

**SysTick Reload Value Register (0xE000\_E014)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:24] | RO | 0 | Reserved. |
| [23:0] | RW | 0 | Reload.  Value to load into the current value register when the counter reaches 0. |

**SysTick Current Value Register (0xE000\_E018)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:24] | RO | 0 | Reserved. |
| [23:0] | RW | 0 | Current.  Current counter value. This is the value of the counter at the time it is sampled. The counter does not provide read-modify-write protection. The register is write-clear. A software write of any value will clear the register to 0 and clears the SYST\_CSR.COUNTFLAG bit to 0. |

**SysTick Calibration Value Register (0xE000\_E01C)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31] | RO | ??  1 | Noref.  If read as 1, the reference clock is not provided – the CLKSOURCE bit of the SysTick control and Status register will be forced to 1 and cannot be clear to 0. |
| [30] | RO | ??  1 | Skew.  If read as 1, the calibration value for 10ms is inexact (due to clock frequency) |
| [29:24] | RO | 0 | Reserved. |
| [23:0] | RO | ??  7a120 | Tenms.  An optional reload value to be used for 10ms (100HZ) timing, subject to system clock skew error. If the value reads as 0, the calibration value is not known. |

Calibration register 与实现相关。不支持外部参考时钟，默认的Cali值为10ms/50Mhz。

## 系统中断控制器

支持32个外部中断源。支持电平和边沿（脉冲）触发中断。寄存器定义与Cortex-M0 完全一致。

**Interrupt Set-Enable Register (0xE000\_E100)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:0] | RW | 0 | Setena  Enable one or more interrupts within a group of 32. Each bit represents an interrupt number from N to N+31 (starting at interrupt 0, 32, 64, etc.).  Writing a 1 will enable the associated interrupt.  Writing a 0 has no effect.  The register reads back with the current enable state. |

**Interrupt Clear-Enable Register (0xE000\_E180)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:0] | RW | 0 | clrena  Disable one or more interrupts within a group of 32. Each bit represents an interrupt number from N to N+31 (starting at interrupt 0, 32.64, etc.).  Writing a 1 will disable the associated interrupt.  Writing a 0 has no effect.  The register reads back with the current enable state. |

**Interrupt Set-Pending Register (0xE000\_E200)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:0] | RW | 0 | Setpend.  Writing a 1 to a bit pends the associated interrupt under software control. Each bit represents an interrupt number from 0 to 31.  Writing a 0 to a bit have no effect on the associated interrupt. The register reads back with the current pending state. |

**Interrupt Clear-Pending Register (0xE000\_E280)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:0] | RW | 0 | clrpend.  Writing a 1 to a bit un-pends the associated interrupt under software control. Each bit represents an interrupt number from 0 to 31.  Writing a 0 to a bit have no effect on the associated interrupt. The register reads back with the current pending state. |

**Interrupt Priority Register (0xE000\_E400 ~ 0xE000\_E41C)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bits** | **R/W** | **Reset** | **Description** |
| [31:30] | RW | 0 | Pri\_n3.  Priority of interrupt number N+3 (3, 7, 11, etc). |
| [29:24] | RO | 0 | Reserved. |
| [23:22] | RW | 0 | Pri\_n2.  Priority of interrupt number N+2 (2, 6, 10, etc). |
| [21:16] | RO | 0 | Reserved. |
| [15:14] | RW | 0 | Pri\_n1.  Priority of interrupt number N+1 (1, 5, 9, etc). |
| [13:8] | RO | 0 | Reserved. |
| [7:6] | RW | 0 | Pri\_n.  Priority of interrupt number N (0, 4, 8, etc). |
| [5:0] | RO | 0 | Reserved. |

# Debug 功能

ARMv6-M Debug支持如下功能：

提供local reset

处理器挂起

单步/执行

读写处理器寄存器

访问Exception有关的信息

软件断点

硬件断点

观察点（watchpoint）

通过DAP访问系统memory

## Debug 寄存器空间

Debug\_Control在PPB 空间的地址分配如下图示。图中黄色的寄存器只能被DAP访问，在硬件设计上已经把他们放到了DAP的空间（CPU无法访问）。



## 功能实现

在深嵌入式应用中，大多数情况下程序代码是固定好的，比如直接使用ROM掩模或者FLASH下载。不像通用MCU那样需要终端用户编程实现不确定的功能。并且Debug 和DAP功能实现会占用相当比例的Die Size。因此，深嵌入式设计往往不提供Debug访问功能。

为了使代码更简洁精炼，PODES\_M0O设计去掉了Debug和DAP功能模块。这样也不影响PODES-M0O的实用性。

如果需要Debug功能，用户可以使用PODES-M0A 内核。

# PODES-M0O寄存器

PODES-M0O内部寄存器定义与CortexM0手册完全兼容。下面描述具体实现相关的特性。

|  |  |  |
| --- | --- | --- |
| **Item** | **Register Name** |  |
| 0-12 | R0-R12 | General purpose registers |
| 13 | MSP | Main SP |
| 13 | PSP | Process SP |
| 14 | LR | Link Register |
| 15 | PC | Program Counter |
|  | APSR | Application Program Status Register |
|  | IPSR | Interrupt Program Status Register |
|  | EPSR | Execution Program Status Register |
|  | PRIMASK | Priority mask register |
|  | CONTROL | SP control register |

## LR

用来存储subroutines, function calls, exceptions的返回信息。复位时的值不关心。

## PC

在复位时，硬件自动取出0x0000\_0004位置的reset\_vector[31:1]装载到PC中。Bit[0]装载到ESPR-T中。这个bit[0]必须为1，否则HardFault。

PC的值为当前执行指令的地址+4。一个指令读寄存器号R15(4’b1111)时，返回的结果是当前执行的指令地址值加4。

三级流水设计中，PC实际上是在取指时生成，然后在译码时可能被使用，最后在执行阶段可能会产生新的分支地址。如果规定“当前指令地址”为IE处理时对应的指令地址，那么与当前IE对应的IF已经前进了2步。译码处理使用的PC值即为当前指令地址+4。汇编器计算地址偏移值时必须知道确切的当前已经执行的指令地址，否则可能漏掉或者重复执行了指令。PODES-M0O设计实现基于上面的考虑。

使用ADD或者MOV更新PC时，bit[0]被忽略掉。但是使用BX, BLX, POP更新PC时，如果bit[0]为0，则产生HardFault。其实就是所有的指令更新PC都可以忽略bit[0]，只不过BX, BLX, POP时可能产生HardFault。

## MSP, PSP

ARMv6-M支持两个堆栈指针。按照下面的规则选择使用哪一个SP。

如果Control[1] 为1并且当前模式为Thread Mode，SP使用ProcessSP，如果为Handler mode，结果不可预期（设计实现仍然使用ProcessSP）。

如果Control[1]为0，则使用MainSP。

SP的bit[1:0]总是保留为00，写[1:0]被忽略，读出为00。

上电复位后MSP的值来自memory地址0x0000\_0000。PSP的值未知（设计实现直接复位为全0）。

## xPSR

xPSR通指APSR, IPSR和EPSR。

APSR[31:28]四个bit分别表示指令运行的N，Z，C和V标志。指令运行过程中这些标志被更新，同时MSR, MRS指令可以访问他们。这些标志在复位时的值未知（设计实现为全0）。

IPSR[5:0]指示当前执行的Exception 的序号。在Exception进入和退出时被更新。可以使用MRS读出。MSR写入被忽略。在Thread Mode， IPSR的值为0，在Handler Mode，IPSR反映当前执行的Exception 的Number。

IPSR[8:6]在ARMv6-M中是保留bits，值为全0。在PODES-M0O设计中，这三个bit有扩展功能，记录当前exception的优先级标志。分别为

|  |  |
| --- | --- |
| **IPSR[8:6]** | **Exception level** |
| 3’B111 | Fixed Level -3 |
| 3’B001 | Fixed Level -2 |
| 3’B010 | Fixed Level -1 |
| 3’B011 | Configurable Level 0 |
| 3’B100 | Configurable Level 1 |
| 3’B101 | Configurable Level 2 |
| 3’B110 | Configurable Level 3 |
| 3’B000 | Reserved |

EPSR[24]为T-bit，EPSR不能被软件（MSR, MRS）读写，可以在debug状态读写。T-bit用于支持ARM 架构的互操作模型，T-bit为1表示执行Thrumb指令，否则执行ARM指令。ARMv6-M只支持Thrumb指令，所以T-bit应该保持为1。更新PC的指令会同时更新T-bit，如果T-bit为0，则产生HardFault。T-bit在reset时为1。

## PRMASK

用于优先级提升。PRMASK[0]复位时被清零，可以使用MSR/MRS指令访问。PRMASK[0]被置位会将执行优先级提升到0，这会阻止所有可配优先级的Exception被执行。也就是只有NMI，Hardfault和Reset可以被执行，其他Exception都被屏蔽掉。

## CONTROL

CONTROL[1]定义堆栈的用法。0：MSP用做当前堆栈；1：Thread mode下PSP用作当前堆栈。这个寄存器在Exception进入和退出时被更新。复位时被清0。MRS/MSR可以读写这个寄存器。只能在Thread mode下读写，Handler mode下的访问被忽略。

# PODES\_M0O指令集

## PODES-M0O指令

PODES-M0O支持的指令完全兼容ARMv6-M指令集。下面四条指令（NOP-Compatible Hints指令）的硬件行为与MCU外部功能实现相关（比如多处理器设计或者低功耗处理），在PODES-M0O中有保留控制信号，用户可以做扩展设计。

WFE

WFI

SEV

YIELD

默认情况下，PODES-M0O执行到这些指令会等效为NOP指令，指令流水不会停止。其行为与这些指令的期望功能有出入（用户扩展设计需要处理的事情）。建议用户在简单的PODES-M0O评估中不要使用这些指令。

上述指令在PODES-M0A中有专门的处理逻辑。

## PODES-M0O指令编码列表

下面表格列举出PODES-M0O实现的全部指令的二进制编码。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LSL(immediate)** |  | 0 | 0 | 0 | 0 | 0 | imm5 | | | | | Rm | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LSR(immediate)** |  | 0 | 0 | 0 | 0 | 1 | imm5 | | | | | Rm | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ASR(immediate)** |  | 0 | 0 | 0 | 1 | 0 | imm5 | | | | | Rm | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ADD(Register)** |  | 0 | 0 | 0 | 1 | 1 | 0 | 0 | Rm | | | Rn | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **SUB(Register)** |  | 0 | 0 | 0 | 1 | 1 | 0 | 1 | Rm | | | Rn | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ADD(immediate)** |  | 0 | 0 | 0 | 1 | 1 | 1 | 0 | imm3 | | | Rn | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **SUB(immediate)** |  | 0 | 0 | 0 | 1 | 1 | 1 | 1 | imm3 | | | Rn | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **MOV(immediate)** |  | 0 | 0 | 1 | 0 | 0 | Rd | | | imm8 | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **CMP(immediate)** |  | 0 | 0 | 1 | 0 | 1 | Rn | | | imm8 | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ADD(immediate)** |  | 0 | 0 | 1 | 1 | 0 | Rdn | | | imm8 | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ADR** |  | 1 | 0 | 1 | 0 | 0 | Rd | | | imm8 | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **SUB(immediate)** |  | 0 | 0 | 1 | 1 | 1 | Rdn | | | imm8 | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **AND(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Rm | | | Rdn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **EOR(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Rm | | | Rdn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LSL(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | Rm | | | Rdn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LSR(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Rm | | | Rdn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ASR(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | Rm | | | Rdn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ADC(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | Rm | | | Rdn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **SBC(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | Rm | | | Rdn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  | | |  | | |
| **ROR(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | Rm | | | Rdn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **TST(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | Rm | | | Rn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **RSB(immediate)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | Rn | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **CMP(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | Rm | | | Rn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **CMN(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | Rm | | | Rn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ORR(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | Rm | | | Rdn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **MUL** |  | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | Rn | | | Rdm | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **BIC(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | Rm | | | Rdn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **MVN(register)** |  | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | Rm | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ADD(register)** |  | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | DN | Rm | | | | Rdn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **CMP(register)** |  | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | N | Rm | | | | Rn | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **MOV(register)** |  | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | D | Rm | | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **BX** |  | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | Rm | | | | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **BLX(register)** |  | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | Rm | | | | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **STR(register)** |  | 0 | 1 | 0 | 1 | 0 | 0 | 0 | Rm | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **STRH(register)** |  | 0 | 1 | 0 | 1 | 0 | 0 | 1 | Rm | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **STRB(register)** |  | 0 | 1 | 0 | 1 | 0 | 1 | 0 | Rm | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LDRSB(register)** |  | 0 | 1 | 0 | 1 | 0 | 1 | 1 | Rm | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LDR(register)** |  | 0 | 1 | 0 | 1 | 1 | 0 | 0 | Rm | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LDRH(register)** |  | 0 | 1 | 0 | 1 | 1 | 0 | 1 | Rm | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LDRB(register)** |  | 0 | 1 | 0 | 1 | 1 | 1 | 0 | Rm | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LDRSH(register)** |  | 0 | 1 | 0 | 1 | 1 | 1 | 1 | Rm | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **STM(/IA/EA)** |  | 1 | 1 | 0 | 0 | 0 | Rn | | | register\_list | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **STR(immediate)** |  | 0 | 1 | 1 | 0 | 0 | imm5 | | | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LDM(/IA/FD)** |  | 1 | 1 | 0 | 0 | 1 | Rn | | | register\_list | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LDR(immediate)** |  | 0 | 1 | 1 | 0 | 1 | imm5 | | | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LDR(literal)** |  | 0 | 1 | 0 | 0 | 1 | Rt | | | imm8 | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **STRB(immediate)** |  | 0 | 1 | 1 | 1 | 0 | imm5 | | | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LDRB(immediate)** |  | 0 | 1 | 1 | 1 | 1 | imm5 | | | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **STRH(immediate)** |  | 1 | 0 | 0 | 0 | 0 | imm5 | | | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LDRH(immediate)** |  | 1 | 0 | 0 | 0 | 1 | imm5 | | | | | Rn | | | Rt | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **STR(immediate)** |  | 1 | 0 | 0 | 1 | 0 | Rt | | | imm8 | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **LDR(immediate)** |  | 1 | 0 | 0 | 1 | 1 | Rt | | | imm8 | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **CPS** |  | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | im | 0 | 0 | I | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ADD(SP immediate)** |  | 1 | 0 | 1 | 0 | 1 | Rd | | | imm8 | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | imm7 | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **SUB(SP immediate)** |  | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | imm7 | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **SXTH** |  | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | Rm | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **SXTB** |  | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | Rm | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **UXTH** |  | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | Rm | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **UXTB** |  | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | Rm | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **PUSH** |  | 1 | 0 | 1 | 1 | 0 | 1 | 0 | M | register\_list | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **REV** |  | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | Rm | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **REV16** |  | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | Rm | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **REVSH** |  | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | Rm | | | Rd | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **POP** |  | 1 | 0 | 1 | 1 | 1 | 1 | 0 | P | register\_list | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **BKPT** |  | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | imm8 | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **NOP** |  | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **YIELD** |  | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **WFE** |  | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **WFI** |  | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **SEV** |  | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **B** | T1 | 1 | 1 | 0 | 1 | cond | | | | imm8 | | | | | | | |
| T2 | 1 | 1 | 1 | 0 | 0 | imm11 | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **SVC(formerly SWI)** |  | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | imm8 | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **MSR(register)** | | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | Rn | | | |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | SYSm | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **DSB** | | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | option | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **DMB** | | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | option | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ISB** | | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | option | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **MRS** | | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | Rd | | | | SYSm | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **BL** | | 1 | 1 | 1 | 1 | 0 | S | imm10 | | | | | | | | | |
| 1 | 1 | J1 | 1 | J2 | imm11 | | | | | | | | | | |

# PODES\_M0O应用指南

## 时钟和复位接口

|  |  |
| --- | --- |
| **IO** | **Signals** |
| input | clk |
| input | rst\_n |
| output | sysresetreq |

PODES-M0O使用单时钟设计，全部模块代码属于单一clock domain。对工作频率没有特殊限制，最高时钟频率依赖综合结果。

PODES-M0O使用单复位设计，全部模块代码属于单一reset domain。Rst\_n是低电平有效。外部接口电路必须先用clk同步复位信号，rst\_n有效宽度建议至少保持一个时钟周期。

Sysresetreq源自AIRCR （0XE000BD0C[2]）寄存器。此信号为高，表示请求外部逻辑产生一个复位信号，将CPU Core复位。

Sysresetreq信号是clk同步信号。外部reset\_gen模块可以采样sysresetreq信号，只要发现它为高电平，就输出rst\_n的低电平（复位）。等到Sysresetreq 变低后，再释放rst\_n信号。Sysresetreq 和rst\_n之间没有特定的相位要求。

## 外部中断接口

|  |  |
| --- | --- |
| **IO** | **Signals** |
| input [31:0] | irq |
| input | nmi |

支持32个外部中断源以及一个不可屏蔽中断。支持电平和边沿（脉冲）触发中断。

电平中断的含义是指，直到外部设备的中断源被清掉（比如中断服务函数的操作），这个电平请求才会消除。如果处理器从中断服务返回后中断请求还在，那么这个中断又会进入Pending，导致处理器可能再一次进入中断处理。

脉冲中断指外部设备的中断请求脉冲至少保持一个系统时钟周期的宽度。中断控制器连续检查中断信号的上升沿并且更新Pending位。在Active Period，如果连续到达多个脉冲，只会被看作一个中断请求事件。

处理器在响应一个中断请求后(进入中断服务的时刻)会自动做一次Pending清除动作。因此对于电平中断来说，这个清除没有意义（pending还在）；对脉冲中断来说，这个清除有意义（Pending被清掉直到新的中断到来）。

脉冲类型的中断可以接任何外部设备中断（电平或者脉冲）不会有问题。电平类型的中断只能连接输出电平类型的外部设备中断。设计应该告诉用户哪些中断是电平中断，哪些是脉冲中断。在PODES-M0O设计实现中对外部中断不加区分，统一采样信号的高电平并更新pending 。外部中断信号可以直接/任意连接到中断控制器，不用区分哪些信号是电平那些是脉冲。

接入的中断信号必须先使用clk同步。

## 总线接口

|  |  |
| --- | --- |
| **IO** | **Signals** |
| output | ext\_mhready |
| output | ext\_mhsel |
| output [31:0] | ext\_mhaddr |
| output [1:0] | ext\_mhtrans |
| output | ext\_mhwrite |
| output [31:0] | ext\_mhwdata |
| output [2:0] | ext\_mhsize |
| output [2:0] | ext\_mhburst |
| output [3:0] | ext\_mhprot |
| input [31:0] | ext\_mhrdata |
| input | ext\_mhready\_out |
| input | ext\_mhresp |

PODES-M0O总线接口兼容AHBLite协议。

PODES-M0O内部设计已经确保了ext-mhready一直为高电平。AMBA总线兼容性考虑，用户模块设计必须在ext\_mhready信号为高时，才可以确认PODES-M0O AHB master访问的address phase。

接口AHB Slave模块如果使用全地址译码，可以不用关心ext\_mhsel信号的状态。否则ext\_hsel必须引入译码逻辑。

PODES-M0O设计使用简化的总线访问模式。ext\_mhburst恒定为3’b000；ext\_mhprot恒定为4’b0001；没有masterlock信号输出。

简单的总线接口模式如下面的图示例子。



## PODES-M0O应用

PODES-M0O应用包括代码集成、功能仿真、代码综合、软件开发、软件测试等工作。

PODES-M0O代码集成可以参照上述接口规范。

PODES-M0O使用工艺无关HDL风格设计。功能仿真不需要调用工艺库model。

PODES-M0O实现的指令集完全兼容ARMv6-M指令集。可以使用支持Cortex-M0的各种开发编译工具完成软件开发工作。

AMY-M0O是一个以PODES-M0O为内核的MCU实例。包括源代码、两个User Manual文档和一个FPGA评估板，提供完整详细的代码集成、功能仿真、代码综合、软件开发、软件测试参考。

***PODES\_M0O\_Application\_User\_Manual\_Vxx.doc***

***PODES\_M0O\_Evaluation\_Board\_User\_Manual\_Vxx.doc***

上述资源可以从[www.mcucore.club](http://www.mcucore.org) 获得。

# Change Summary

**REVISION HISTORY**

|  |  |  |
| --- | --- | --- |
| **Revision No.** | **Description of change** | **Release Date** |
| Ver1.0 | First Release | 20200101 |
|  |  |  |
|  |  |  |
|  |  |  |

# CopyLeft



除非明确声明，PODES项目的软件代码都以LGPL方式发行。所有文档则以CC-BY-SA-4.0方式发行。PODES项目中涉及到第三方软件和工具遵守第三方版权规定。

分发开源软件代码时请保留原始file header注释。分发开源文档时请完整保留本文档第一节至第三节信息。