PODES_M0O

Implementation User Manual

Ver1.2

Index:

1 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:

2 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

3 支持和服务

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

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

4 PODES-M0O 功能特性

4.1 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）

4.2 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

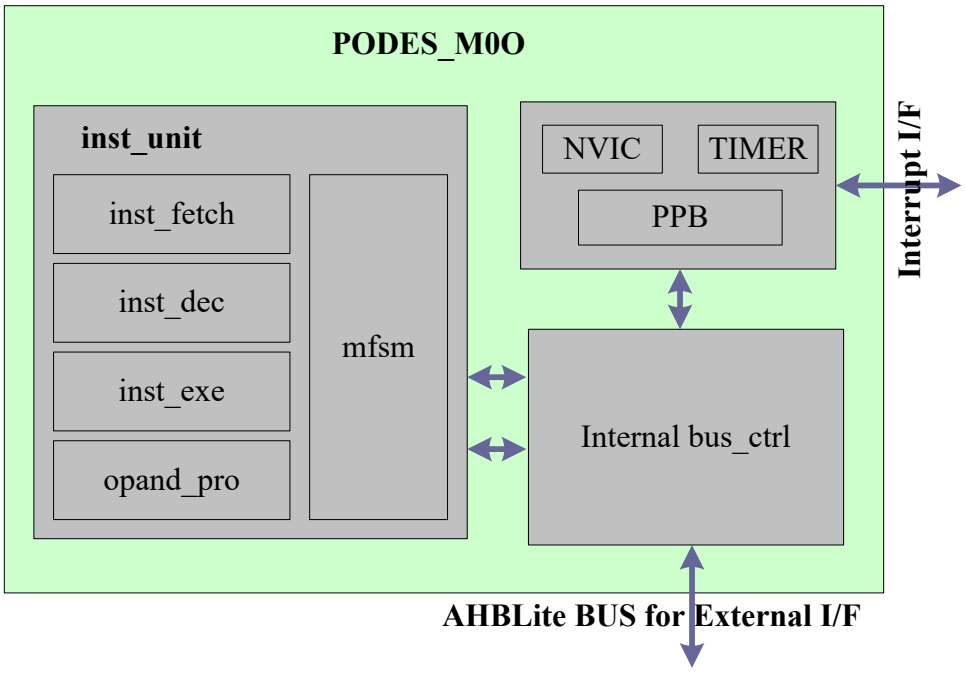
ASIC 综合或者 FPGA 综合可以直接调用工艺库提供的宏单元，或者用户自己设计结构化代码。

5 PODES_M0O 系统结构

5.1 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。

5.2 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。

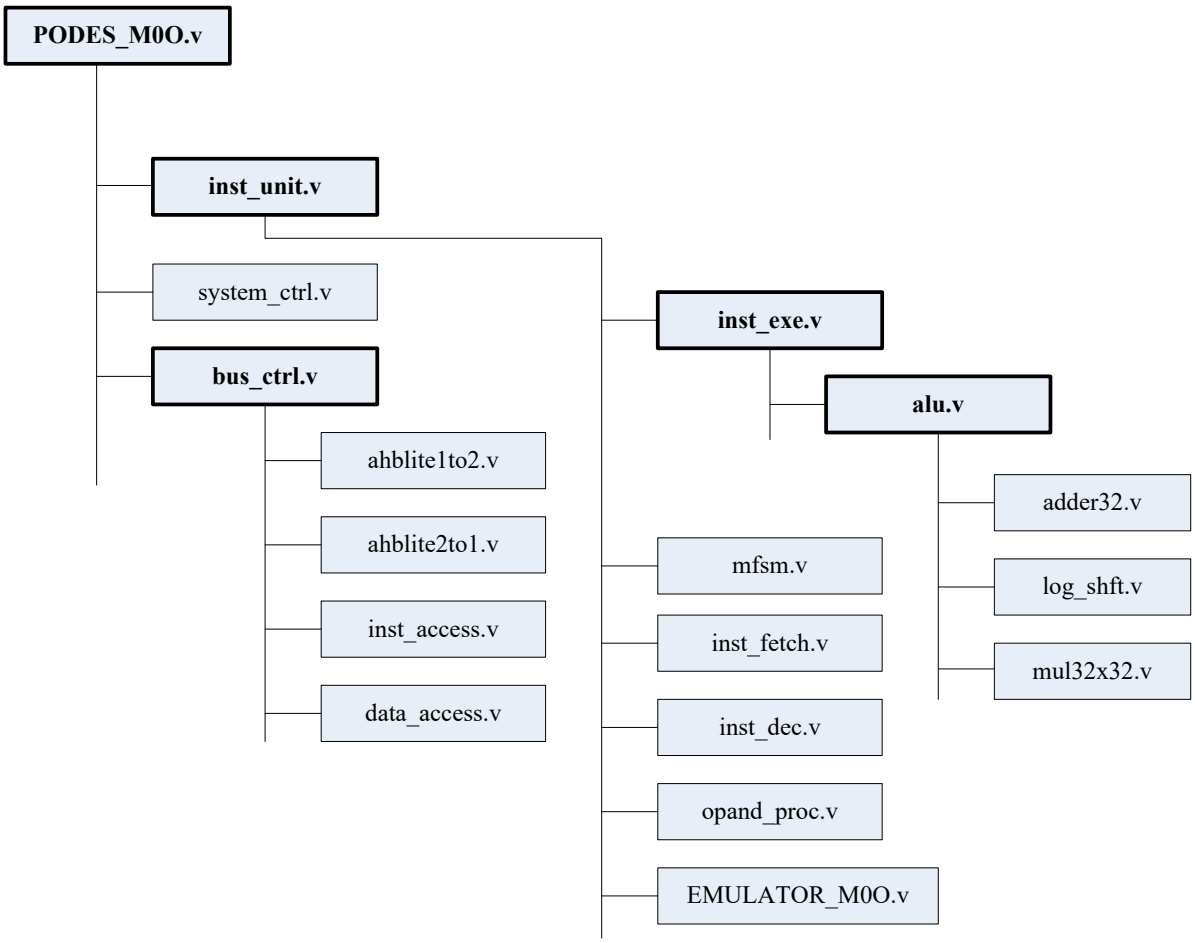
5.3 PODES-M0O 中断

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。

5.4 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

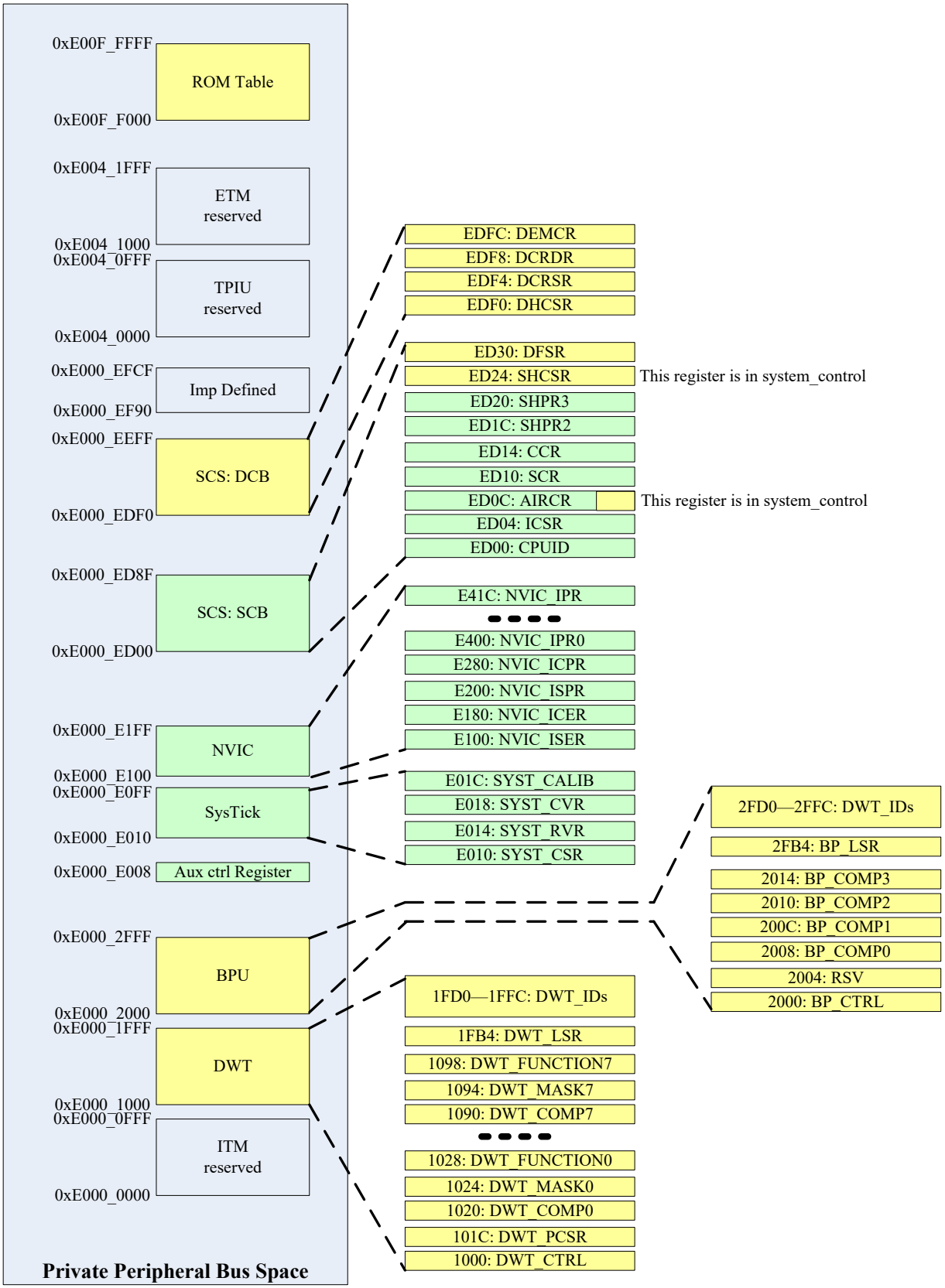
指令模拟器，代码仿真时可以输出反汇编代码。用户可以使用它辅助自己的程序仿真。

6 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。


0xE00F_FFFF
ROM Table
0xE00F_F000
0xE004_1FFF
ETM
reserved
0xE004_1000
0xE004_0FFF
TPIU
reserved
0xE004_0000
0xE000_EFCF
Imp Defined
0xE000_EF90
0xE000_EEFF
SCS: DCB
0xE000_EDF0
0xE000_ED8F
SCS: SCB
0xE000_ED00
0xE000_E1FF
NVIC
0xE000_E100
0xE000_E0FF
SysTick
0xE000_E010
0xE000_E008
Aux ctrl Register
0xE000_2FFF
BPU
0xE000_2000
0xE000_1FFF
DWT
0xE000_1000
0xE000_0FFF
ITM
reserved
0xE000_0000
Private Peripheral Bus Space
EDFC: DEMCR
EDF8: DCRDR
EDF4: DCRSR
EDF0: DHCSR
ED30: DFSR
ED24: SHCSR
This register is in system_control
ED20: SHPR3
ED1C: SHPR2
ED14: CCR
ED10: SCR
ED0C: AIRCR
This register is in system_control
ED04: ICSR
ED00: CPUID
E41C: NVIC_IPR
E400: NVIC_IPR0
E280: NVIC_ICPR
E200: NVIC_ISPR
E180: NVIC_ICER
E100: NVIC_ISER
E01C: SYST_CALIB
E018: SYST_CVR
E014: SYST_RVR
E010: SYST_CSR
2FD0—2FFC: DWT_IDs
2FB4: BP_LSR
2014: BP_COMP3
2010: BP_COMP2
200C: BP_COMP1
2008: BP_COMP0
2004: RSV
2000: BP_CTRL
1FD0—1FFC: DWT_IDs
1FB4: DWT_LSR
1098: DWT_FUNCTION7
1094: DWT_MASK7
1090: DWT_COMP7
1028: DWT_FUNCTION0
1024: DWT_MASK0
1020: DWT_COMP0
101C: DWT_PCSR
1000: DWT_CTRL

6.1 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.

6.2 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。

6.3 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.

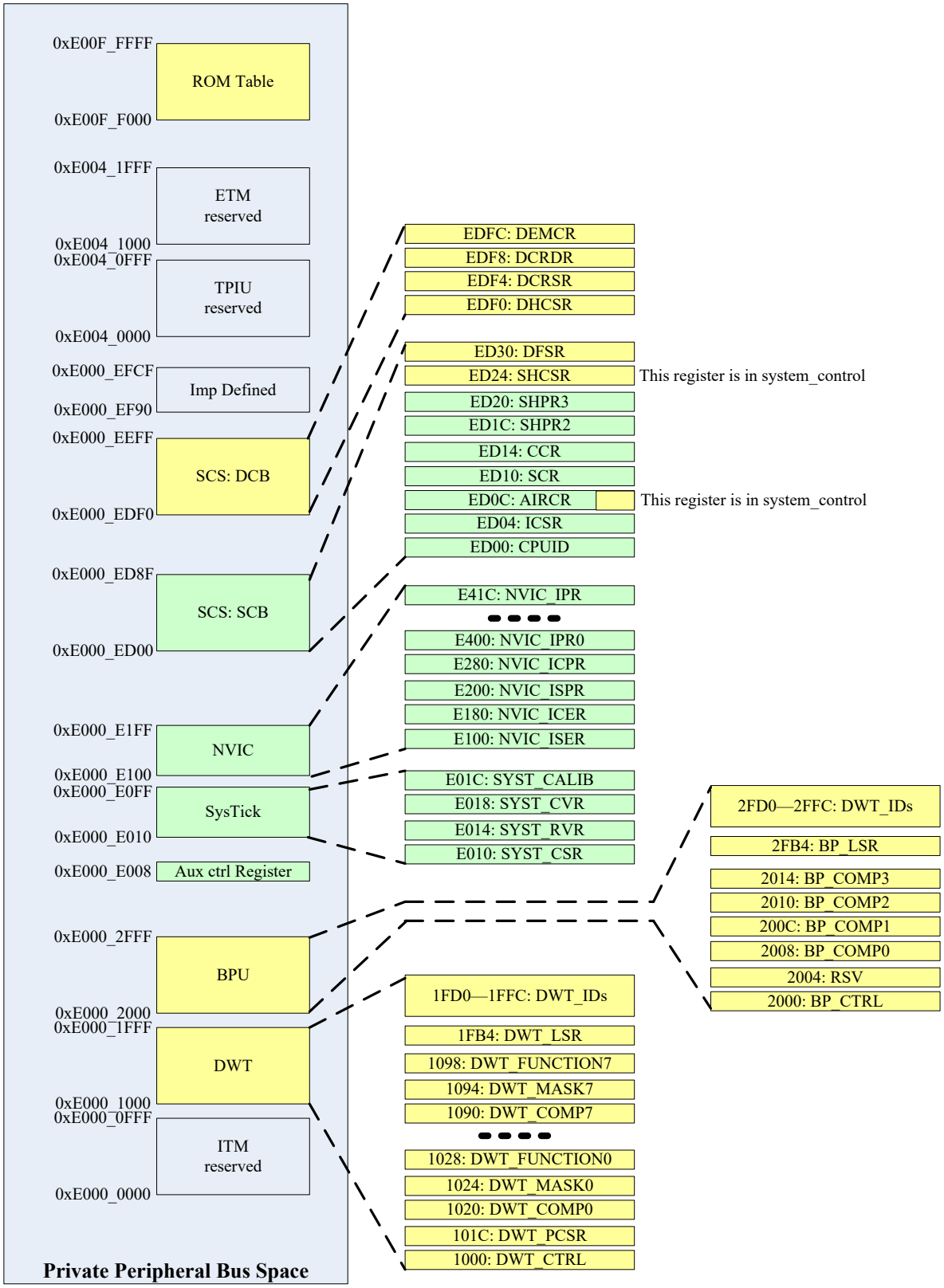
7 Debug 功能

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

7.1 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）。


0xE00F_FFFF
ROM Table
0xE00F_F000
0xE004_1FFF
ETM
reserved
0xE004_1000
0xE004_0FFF
TPIU
reserved
0xE004_0000
0xE000_EFCF
Imp Defined
0xE000_EF90
0xE000_EEFF
SCS: DCB
0xE000_EDF0
0xE000_ED8F
SCS: SCB
0xE000_ED00
0xE000_E1FF
NVIC
0xE000_E100
0xE000_E0FF
SysTick
0xE000_E010
0xE000_E008
Aux ctrl Register
0xE000_2FFF
BPU
0xE000_2000
0xE000_1FFF
DWT
0xE000_1000
0xE000_0FFF
ITM
reserved
0xE000_0000
Private Peripheral Bus Space
EDFC: DEMCR
EDF8: DCRDR
EDF4: DCRSR
EDF0: DHCSR
ED30: DFSR
ED24: SHCSR
This register is in system_control
ED20: SHPR3
ED1C: SHPR2
ED14: CCR
ED10: SCR
ED0C: AIRCR
This register is in system_control
ED04: ICSR
ED00: CPUID
E41C: NVIC_IPR
E400: NVIC_IPR0
E280: NVIC_ICPR
E200: NVIC_ISPR
E180: NVIC_ICER
E100: NVIC_ISER
E01C: SYST_CALIB
E018: SYST_CVR
E014: SYST_RVR
E010: SYST_CSR
2FD0—2FFC: DWT_IDs
2FB4: BP_LSR
2014: BP_COMP3
2010: BP_COMP2
200C: BP_COMP1
2008: BP_COMP0
2004: RSV
2000: BP_CTRL
1FD0—1FFC: DWT_IDs
1FB4: DWT_LSR
1098: DWT_FUNCTION7
1094: DWT_MASK7
1090: DWT_COMP7
1028: DWT_FUNCTION0
1024: DWT_MASK0
1020: DWT_COMP0
101C: DWT_PCSR
1000: DWT_CTRL

7.2 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。

8 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

8.1 LR

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

8.2 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。

8.3 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）。

8.4 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。

8.5 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。

8.6 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。

9 PODES_M0O 指令集

9.1 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。

9.2 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							
B 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										

10 PODES_M0O 应用指南

10.1 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。

10.2 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。

10.3 总线接口

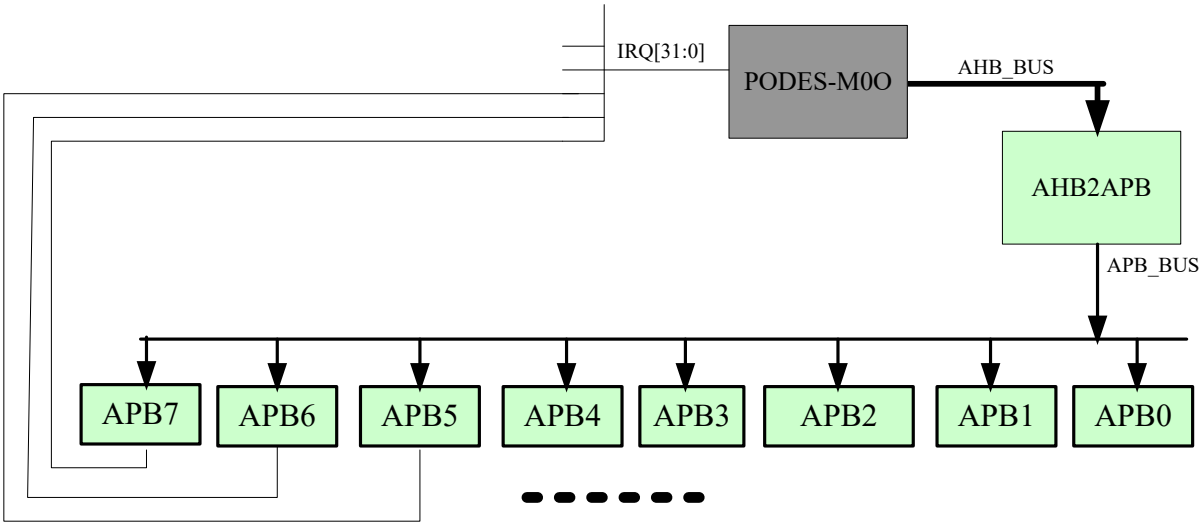
IO	Signals
output	ext_mhready
output	ext_mhsel
output [31:0]	ext_mhaddr
output [1:0]	ext_mhtrans
output	ext_mhwrite
output [31:0]	ext_mhwdata
output [2:0]	ext_mhsize
output [2:0]	ext_mhburst
output [3:0]	ext_mhprot
input [31:0]	ext_mhrdata
input	ext_mhready_out
input	ext_mhresp

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。


IRQ[31:0]
PODES-M0O
AHB_BUS
AHB2APB
APB_BUS
APB7
APB6
APB5
APB4
APB3
APB2
APB1
APB0

10.4 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 获得。

Change Summary

REVISION HISTORY

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

CopyLeft🄯