

Arm@v7-M 架构的 STM32 如何启用 DWT 计数

关键字: DWT, DEMCR

1. 前言

客户在使用 STM32H7 的时候, 想要使用 DWT 计数来测量代码执行时间, 评估执行效率。客户发现在重新上电或 reset 后, 无法启用 DWT 进行计数。

2. 调研

在 ARMv7-M 架构中有个 DEMCR 寄存器, 这个寄存器可以控制 DWT 的使能。在 power-on reset 后这个寄存器所有位的值都为 0。而当 bit[24] 为 0 时, DWT 和 ITM 模块都是 disabled 的。所以为了启用 DWT 模块, 必须将 DEMCR 的 bit[24] 置为 1。如图 1 所示:

图1. DEMCR 寄存器

C1.6.5 Debug Exception and Monitor Control Register, DEMCR

The DEMCR characteristics are:

Purpose	Manages vector catch behavior and DebugMonitor handling when debugging.
Usage constraints	<ul style="list-style-type: none"> Bits[23:16] provide DebugMonitor exception control. Bits[15:0] provide Debug state, Halting debug, control.
Configurations	Always implemented.
Attributes	See Table C1-10 on page C1-699 , and also: <ul style="list-style-type: none"> A power-on reset resets all register bits to zero. A Local reset resets only the bits related to the DebugMonitor to zero. These are bits[19:16]. Reset behavior on page B1-530 defines Local reset.

The DEMCR bit assignments are:

31	25	24	23	20	19	18	17	16	15	11	10	9	8	7	6	5	4	3	1	0	
Reserved				Reserved				Reserved													
				TRCENA				MON_REQ				VC_HARDERR									
				MON_STEP				VC_INTERR													
				MON_PEND				VC_BUSERR													
				MON_EN				VC_STATERR													
								VC_CHKERR													
								VC_NOCPERR													
								VC_MMERR													
								Reserved													
								VC_CORERESET													

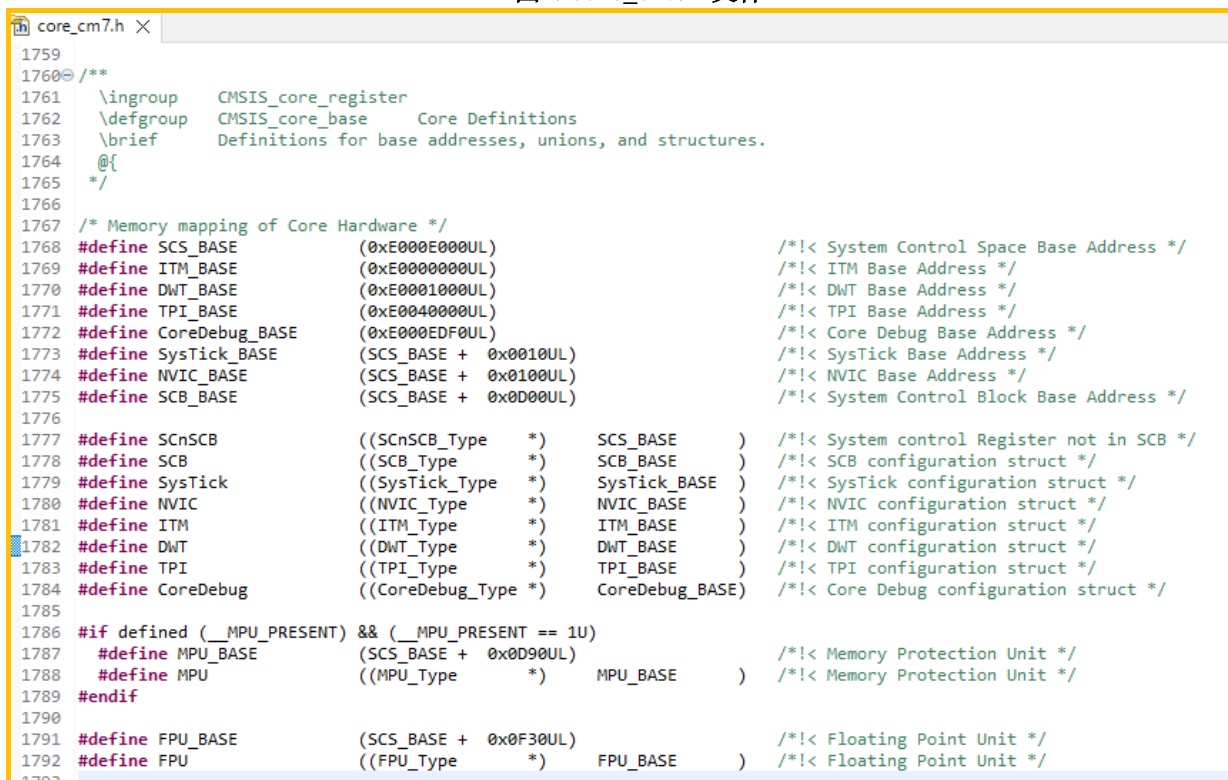
Bits[31:25]	Reserved
TRCENA, bit[24]	Global enable for all DWT and ITM features:
0	DWT and ITM units disabled.
1	DWT and ITM units enabled.
If the DWT and ITM units are not implemented, this bit is UNK/SBZP.	

3. 启用 DWT 进行计数

STM32H7 基于 Arm Cortex-M7 内核，而 Cortex-M7 是 ARMv7-M 架构，所以 H7 在配置 DWT 模块之前需要将 DEMCR 的 bit[24]置位。在基于 Cortex-M7 的芯片中，需要使用 DWT-LAR 来解锁 DWT（其他核可能不需要，应具体分析），然后对 DWT_CTRL 进行相应使能即可。

在 CMSIS 文件中已经提供了相关寄存器的宏定义（例如在“core_cm7.h”文件中包提供了 DWT 和 DEMCR 的宏定义），我们可以使用这些宏定义方便的进行配置，如图 2 所示：

图2. core_cm7.h 文件



```

1759
1760 /**
1761  \ingroup   CMSIS_core_register
1762  \defgroup  CMSIS_core_base Core Definitions
1763  \brief     Definitions for base addresses, unions, and structures.
1764  @{
1765  */
1766
1767 /* Memory mapping of Core Hardware */
1768 #define SCS_BASE      (0xE000E000UL)    /*< System Control Space Base Address */
1769 #define ITM_BASE      (0xE0000000UL)    /*< ITM Base Address */
1770 #define DWT_BASE      (0xE0001000UL)    /*< DWT Base Address */
1771 #define TPI_BASE      (0xE0040000UL)    /*< TPI Base Address */
1772 #define CoreDebug_BASE (0xE000EDF0UL)   /*< Core Debug Base Address */
1773 #define SysTick_BASE   (SCS_BASE + 0x0010UL) /*< SysTick Base Address */
1774 #define NVIC_BASE      (SCS_BASE + 0x0100UL) /*< NVIC Base Address */
1775 #define SCB_BASE       (SCS_BASE + 0x0D00UL) /*< System Control Block Base Address */
1776
1777 #define SCnSCB          ((SCnSCB_Type *)   SCS_BASE)  /*< System control Register not in SCB */
1778 #define SCB             ((SCB_Type *)      SCB_BASE)  /*< SCB configuration struct */
1779 #define SysTick         ((SysTick_Type *)   SysTick_BASE) /*< SysTick configuration struct */
1780 #define NVIC            ((NVIC_Type *)      NVIC_BASE)  /*< NVIC configuration struct */
1781 #define ITM             ((ITM_Type *)       ITM_BASE)   /*< ITM configuration struct */
1782 #define DWT             ((DWT_Type *)       DWT_BASE)   /*< DWT configuration struct */
1783 #define TPI             ((TPI_Type *)       TPI_BASE)   /*< TPI configuration struct */
1784 #define CoreDebug       ((CoreDebug_Type *) CoreDebug_BASE) /*< Core Debug configuration struct */
1785
1786 #if defined (__MPU_PRESENT) && (__MPU_PRESENT == 1U)
1787 #define MPU_BASE        (SCS_BASE + 0x0D90UL)    /*< Memory Protection Unit */
1788 #define MPU            ((MPU_Type *)      MPU_BASE) /*< Memory Protection Unit */
1789 #endif
1790
1791 #define FPU_BASE        (SCS_BASE + 0x0F30UL)    /*< Floating Point Unit */
1792 #define FPU            ((FPU_Type *)      FPU_BASE) /*< Floating Point Unit */
1793

```

示例（如下）：使用 DWT 测量代码执行所用的时钟 cycle 数。

```

// In you code you should #include "core_cm7.h"
// then

// DWT and ITM units enabled
CoreDebug->DEMCR |= CoreDebug_DEMCR_TRCENA_Msk;
// Enable write access
DWT->LAR = 0xC5ACCE55;
// Clear CYCCNT
DWT->CYCCNT = 0;
// CYCCNT counter enable
DWT->CTRL |= DWT_CTRL_CYCCNTENA_Msk;

uint32_t cycles = DWT->CYCCNT;
/* Your codes */

```

```
cycles = DWT->CYCCNT - cycles;
```

4. 小结

在使用 ARMv7-M 架构的 STM32 时，对 DWT 配置之前应确保 DEMCR 中的 bit[24] 已经被配置（使能 DWT），然后才能使用 DWT。

参考文献

文件编号	文件标题	版本号	发布日期
ARM DDI 0403E	Arm®v7-M Architecture Reference Manual	E.e	15-Feb-2021
RM0468	STM32H723/733, STM32H725/735 and STM32H730 Value line advanced Arm®-based 32-bit MCUs	3	13-Dec-2021

版本历史

日期	版本	变更
2023 年 01 月 30 日	1.0	首版发布

重要通知 - 请仔细阅读

意法半导体公司及其子公司（“ST”）保留随时对 ST 产品和 / 或本文档进行变更的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用，ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。若需 ST 商标的更多信息，请参考 www.st.com/trademarks。所有其他产品或服务名称均为其各自所有者的财产。

本文档是 ST 中国本地团队的技术性文章，旨在交流与分享，并期望借此给予客户产品应用上足够的帮助或提醒。若文中内容存有局限或与 ST 官网资料不一致，请以实际应用验证结果和 ST 官网最新发布的内容为准。您拥有完全自主权是否采纳本文档（包括代码，电路图 etc）信息，我们也不承担因使用或采纳本文档内容而导致的任何风险。

本文档中的信息取代本文档所有早期版本中提供的信息。

© 2020 STMicroelectronics - 保留所有权利