CMS32M5xxx 系列 IEC_60730 库

用户手册

Rev. 1.00

请注意以下有关CMS知识产权政策

＊中微半导体（深圳）股份有限公司（以下简称本公司）已申请了专利，享有绝对的合法权益。与本公司MCU或其他产品有关的专利权并未被同意授权使用，任何经由不当手段侵害本公司专利权的公司、组织或个人，本公司将采取一切可能的法律行动，遏止侵权者不当的侵权行为，并追讨本公司因侵权行为所受的损失、或侵权者所得的不法利益。

＊中微半导体（深圳）股份有限公司的名称和标识都是本公司的注册商标。

＊本公司保留对规格书中产品在可靠性、功能和设计方面的改进作进一步说明的权利。然而本公司对于规格内容的使用不负责任。文中提到的应用其目的仅仅是用来做说明，本公司不保证和不表示这些应用没有更深入的修改就能适用，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。本公司的产品不授权适用于救生、维生器件或系统中作为关键器件。本公司拥有不事先通知而修改产品的权利，对于最新的信息，请参考官方网站 www.mcu.com.cn

目录

1. IEC_60730 介绍

IEC 是 The International Electrotechnical Commission 国际电工委员会的简称，它主要对家用和类似用途的自动电气控制进行安全和基本性能方面的要求，从而保证了自动系统的嵌入式控制硬件和软件的安全性。

IEC_60730 分为 A、B、C 三类，其中 B 类用来防止被控设备的不安全操作。它对软件或硬件的干预，防止软件程序内的故障导致安全危害。

B 类的检测部件如下图所示。

需要监测的B类部件		
	要求电子控制测试的60730 B类部件(见表H.11.12.7)	故障与错误
1	CPU寄存器	阻塞
2	CPU程序计数器	阻塞
3	中断处理和执行	无中断或中断太频繁
4	时钟	错误的频率
5	固定存储器	所有的单比特错误
6	可变存储器	直流故障
7	寻址(对应于固定/可变存储器)	阻塞
8	内部数据通道	阻塞
9	寻址	地址错误
10	时序	时序/序列的错误
11	I/O外设	H.27指定的故障条件
12	模拟 A/D 和 D/A converters	H.27指定的故障条件

图 1-1：IEC_60730 B 类检测模块

中微 CMS32M5xxx 系列芯片是内核 ARM Cortex-M0，64MHz 2.1V-5.5V 的低成本高性能芯片。它拥有单周期 32 位硬件乘法器和 6~8 个周期 32 位硬件除法器，内部 FLASH 有 32KB，RAM 有 8KB（支持分区写保护功能），有内部 48MHz/64MHz（HSI）和 40KHz LSI，内置 ACMP，EPWM，ADC，可编程放大器 PGA 和运算放大器等模块。

2. IEC_60730 模块介绍

中微 IEC_60730 库包含以下几个模块的测试：

CPU 寄存器、CPU 指针、RAM 测试、FLASH 测试、I/O 测试、ADC 测试、中断测试和 CLOCK 测试。

其中对 FLASH，RAM 的操作需要对内部数据进行调用，同时也测试了内部数据通道模块。

CPU 寄存器和指针的测试中，用到了对地址的操作，对数据的直接寻址和间接寻址，也就测试了寻址模块。

程序的运行，中断和时钟等都会用到系统时序相关的处理,如果时序出现问题，测试也就不会得到正确结果，所以如果测试模块能通过，时序模块也就通过了。

这样前面列出的所有关于 IEC_60730 B 类型的模块都得到了测试。

下面章节我们对测试的各个模块分别进行介绍。

2.1 CPU 寄存器测试模块

CPU 寄存器模块测试了 Core 里的寄存器 R0~R12 和特殊寄存器（PC，SP，LR，APSR，PRIMASK）。

表 2-1：CPU 测试寄存器列表

寄存器名字	测试位	备注
R0~R12	0~31	
R13	4~31	低 4 位无效
R14	0~31	R14 测试了 LR
APSR	27~31	只有高 5 位有效
PRIMASK	0	只有位 0 有效

其中 SP 包含 MSP（main SP）和 PSP(process SP)由 R13 的测试实现。

CPU 寄存器测试函数放在 IEC60730_B_CPUTest_ARMCC.s 里，具体包括以下函数：

表 2-2：CPU 模块函数列表

函数名字	函数介绍
iec60730_reg_test	测试所有寄存器 R0~R14，SP，LR 和特殊寄存器
iec60730_reg_r0_bist_test	测试 R0
iec60730_reg_r1_r4_bist_test	测试 R1~R4
iec60730_reg_r8_r12_bist_test	测试 R8~R12
iec60730_reg_sp_bist_test	测试 SP
iec60730_reg_spec_bist_test	测试特殊寄存器
iec60730_reg_lr_bist_test	测试 LR

以上函数均无参数和返回值，如果测试失败会停留在出错处，如果测试通过会继续执行接下来的操作。

2.2 CPU 指针测试

CPU 指针测试，通过对不同函数的调用和同一函数在不同区域的调用来判断 PC 值是否能正确的获取。

CPU 指针的测试程序使用汇编来实现，放在了 IEC60730_B_CPUTest_ARMCC.s 里。

具体的函数如下表所示：

表 2-3：CPU 指针函数列表

函数名字	函数介绍
iec60730_pc_test_start_up	通过测试多个地址测试 PC 参数：无 返回：无
iec60730_pc_test	在不同的地方调用同一地址测试 PC 参数：无 返回：无

2.3 RAM 测试模块

RAM 测试主要是对 RAM 区间进行读写测试，对 RAM 区间写入不同的数据后,读出写入地址的数据，来判断读出的数据是否就是之前写入的数据，从而判断写操作。执行的方式有写入不同的数据和写入反转的数据。

RAM 测试的函数放在 IEC60730_B_RAMTest_ARMCC.s 里。有以下几个函数：

表 2-4：RAM 测试函数列表

函数名字	函数介绍
iec60730_ram_test	对设定的 RAM 区间进行测试 参数 1：起始地址 参数 2：结束地址 返回：无
iec60730_ram_march_bist_test	对 RAM 中指定地址进行读写判断 参数：要判断的 RAM 地址 返回：测试状态

2.4 FLASH 测试模块

FLASH 测试模块通过 hardware CRC 和 software CRC 对 FLASH 区间进行测试。

FLASH 测试的程序放在 IEC60730_B_FlashTest.c 里，主要有以下几个函数：

表 2-5：FLASH 测试函数列表

函数名字	函数介绍
IEC60730_HardwareCRC16Gen	通过内部硬件 CRC 实现 CRC16/32 参数 1：数据起始地址 参数 2：输入数据的大小 返回：CRC 值
IEC60730_HardwareCRC16Test	硬件 CRC16 检测 参数 1：数据的起始地址 参数 2：输入数据的大小 参数 3：期望的 CRC 值 返回：测试状态
IEC60730_SoftwareCRC16Gen	通过内部软件 CRC 实现 CRC16/32 参数 1：数据起始地址 参数 2：输入数据的大小 返回：CRC 值
IEC60730_SoftwareCRC16Test	软件 CRC16 检测 参数 1：数据的起始地址 参数 2：输入数据的大小 参数 3：期望的 CRC 值 返回：测试状态

2.5 I/O 测试模块

GPIO 是芯片对外设控制主要的通道，对它的测试是十分需要的。

GPIO 测试的程序放在 IEC60730_B_GPIOTest.c 里，在 mcu_test.c 调用。GPIO 测试函数如下：

表 2-6：GPIO 测试函数列表

函数名字	函数介绍
IEC60730_GPIOOutputTest	GPIO 输出测试 参数 1：要测试的 IO table,如 gpio_para_table 参数 2：要测试的 IO 数量 返回：测试状态
IEC60730_GPIOInputTest	GPIO 输入测试 参数 1：要测试的 IO table,如 gpio_para_table 参数 2：要测试的 IO 数量 返回：测试状态

库中列举了一部分要测试的 IO 口，客户测试的时候要根据实际的芯片引脚进行修改，修改的位置在 mcu_test.c 里的 GPIO_Test（）函数里，如下图所示：

```
// bref: 测试输入输出
// para:
// note

static void GPIO_Test(void)
{
	uint8_t ret = IEC60730_TEST_NORMAL;
	//测试输出
	//需要测试的 GPIO 避免把 SWD 口给初始化了
	const GPIO_PARA_TYPE gpio_para_table[] =
	{
		// GPIO0
		GPIO0,1,
		GPIO0,4,
		GPIO0,5,
		GPIO0,6,
		GPIO0,7,
//		// GPIO4
//		GPIO4,3,
//		GPIO4,4,
//		GPIO4,6,
//		GPIO4,7,
//		// GPIO2
//		GPIO2,1,
//		GPIO2,2,
//		GPIO2,3,
//		GPIO2,4,
//		GPIO2,5,
//		// GPIO1
//		GPIO1,2,
//		GPIO1,3,
//		GPIO1,4,
//		GPIO1,5,
//		GPIO1,6,
//		GPIO1,7,
//		// GPIO3
//		GPIO3,0,
//		GPIO3,1,
//		GPIO3,4,
//		GPIO3,6,
	};
```

图 2-1：GPIO 口选择

如果需要测试的 IO 在列表里，打开被屏蔽的 IO 定义就好；如果例子里没有对应测试 IO 的定义，按照列出的例子添加就好。

2.6 ADC 测试模块

ADC 测试可以测试芯片 ADC 模块工作情况。

测试的函数在 IEC60730_B_ADTest.c 里，如下：

表 2-7：ADC 测试函数列表

函数名字	函数介绍
IEC60730_AD0Test	ADC0 测试 参数 1：要测试的 ADC table,如 ad0_self_test_ch_table 参数 2：要测试的通道 参数 3：测试的次数 返回：测试状态
IEC60730_AD1Test	ADC1 测试 参数 1：要测试的 ADC table 参数 2：要测试的通道 参数 3：测试的次数 返回：测试状态

ADC0 和 ADC1 使用相同的通道定义 table。

2.7 中断测试模块

程序里中断的事件可以优先被处理，中断使用频繁，它的可靠性非常重要。库中中断测试通过 timer0 中断作为计时单元，系统记录一定时间里时间变量的值是否在要求的范围内，如果变量在要求的范围内，说明中断按照计划进行了操作。如果没有在要求的范围内说明中断执行出现了错误。

中断程序在 IEC60730_B_InterruptTest.c 里，主要的函数如下：

表 2-8：中断测试函数列表

函数名字	函数介绍
IEC60730_IntCntPro	对时间变量进行操作 参数 1：计数变量选择
IEC60730_IntTestInit	初始化中断用到的各个变量 参数 1：时间变量 参数 2：低比较值 参数 3：高比较值 参数 4：初始值储存变量 参数 5 ：计数指针个数设定
IEC60730_IntTest	中断测试判断 参数：无 返回：中断模块工作状态

2.8 CLOCK 测试模块

CLOCK 测试是为了避免系统错误的时钟出现。时钟测试需要一个其他的时钟作为参考基准，来判断 CLOCK 计数是否正常。这里使用内部低时钟的 WDT 来产生计时单元，后通过计算一定时间后的 CLOCK 计数值是否在计数范围来判断 CLOCK 的准确情况。

CLOCK 测试的程序在 IEC60730_B_ClockTest.c 里，包含的函数如下：

表 2-9：CLOCK 测试函数列表

函数名字	函数介绍
IEC60730_ClkInit	初始化 clock 用到的各变量 参数 1：判断偏差的低值 参数 2：判断偏差的高值 参数 3：1s 内调用 mainloop 的次数 返回：无
IEC60730_ClkTestReset	复位各个参数 参数：无 返回：无
IEC60730_ClkCnt	每 100ms 后操作计数变量 参数：无 返回：无
IEC60730_ClkTest	CLOCK 检测 （1s 后计数值的判断） 参数：无 返回：检测状态
IEC60730_ClkMonInMainloop	CLOCK 检测 （通过进入主循环的次数来判断） 参数：无 返回：测试结果

3. 库函数调用

IEC_60730 各个模块的调用大致可以用下图来概述：

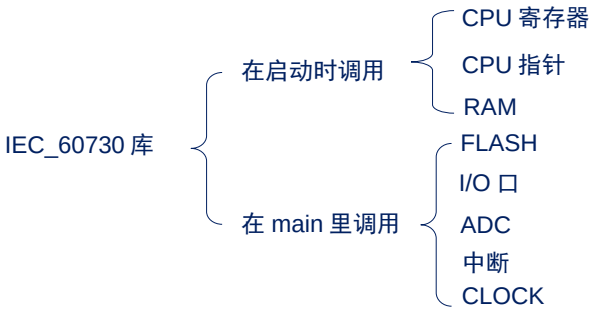

图 3-1：IEC_60730 模块调用图

详细的各个函数的调用位置主要有几个地方，分别是启动 startup_cms32m5xxx.s 的 Reset_Handler、SysTick_Handler 的 Self_Test_Periodic、main.c 的 IEC_Init 和 iec_test_loop 里。（中断和 CLOCK 模块的测试在 TMR0_IRQHandler、WDT_IRQHandler 和 SysTick_Handler 中调用）

下表详细的列出了样例工程各个模块库函数的调用情况：

表 3-1：库函数调用情况

程序调用区	中间函数	低层模块函数	所在模块
startup_cms32m5xxx.s 的 Reset_Handler	iec60730_reg_test		CPU 寄存器模块，在 main 之前调用
	iec60730_pc_test_start_up		CPU 指针模块，在 main 之前调用
	iec60730_ram_test		RAM 模块，在 main 之前调用
SysTick_Handler 的 Self_Test_Periodic	Register_Test	iec60730_reg_r0_bist_test	CPU 寄存器模块
		iec60730_reg_r1_r4_bist_test	
		iec60730_reg_r8_r12_bist_test	
		iec60730_reg_sp_bist_test	
		iec60730_reg_spec_bist_test	
		iec60730_reg_lr_bist_test	
	iec60730_pc_test		CPU 指针模块，
	MarchX_Test	iec60730_ram_march_bist_test	RAM 模块
	CRC_Test_Para	IEC60730_HardwareCRC16Gen	FLASH 模块， 在需要做 CRC16 操作的地方都可以调用，如果使用 IEC_60730 库不需要再调用此函数
main.c 的 IEC_Init	ROM_Test	IEC60730_HardwareCRC16Gen	FLASH 模块。在需要做 CRC16 操作的地方都可以调用，如果使用 IEC_60730 库不需要再调用此函数。
		IEC60730_HardwareCRC16Test	FLASH 模块
		IEC60730_SoftwareCRC16Gen	FLASH 模块，在需要做 CRC16 操作的地方都可以调用，如果使用 IEC_60730 库不需要再调用此函数。
		IEC60730_SoftwareCRC16Test	FLASH 模块
	GPIO_Test	IEC60730_GPIOOutputTest	GPIO 模块

程序调用区	中间函数	低层模块函数	所在模块
		IEC60730_GPIOInputTest	GPIO 模块
	ADC_Test	IEC60730_AD0Test	ADC 模块
		IEC60730_AD1Test	ADC 模块
	Interrupt_Test	IEC60730_IntTestInit	中断模块
	Clock_Test	IEC60730_ClkInit	CLOCK 模块
	Hal_TimerInit		调用 Timer 驱动，实现 timerc 初始化，不调用测试模块函数
	Mem_Test_init		只初始化栈自检变量，不调用任何函数
main.c 的 iec_test_loop	iec_test_loop	IEC60730_ClkMonInMainloop	CLOCK 模块
TMR0_IRQHandler	IEC60730_IntCntPro		中断模块
SysTick_Handler	Self_Test_Periodic		每次进入中断都调用
	iec_test_int	IEC60730_IntTest	中断模块，每 100ms 调用一次
	IEC60730_ClkCnt		CLOCK 模块，每 10ms 调用一次
WDT_IRQHandler	IEC60730_ClkTest		CLOCK 模块，判断 CLOCK 正常与否

中间函数调用低层模块函数后再被调用区函数调用，实现各个模块在工程里的测试。

用户调用 IEC_60730 库可以参考以上调用来实现。

4. 库函数时间设置

IEC_60730 库的各个 timer 以及 Self_Test_Periodic 中各个测试块的时间，根据测试需要必须设置成固定的值或满足一定的要求。具体的要求请参考下面的列表：

1) Timer0 设置为 2.5ms 的定时器，每次进中断的时候时间变量变化 1
2) Systick 设置 1ms 的中断间隔
3) 看门狗设置产生溢出中断时间为 1.0s
4) 每进入 1 次 SysTick_Handler 中断调用一次 Self_Test_Periodic，其中 Self_Test_Periodic 包含了 T_RAM，T_REG，T_ROM，T_PC，T_ADC（根据硬件情况添加，硬件支持就添加）。每一个 periodic case 的执行时间不超过 10us。

5. 工程设置要求

在调用 IEC_60730 库的时候需要做一些系统或变量的设置。具体如下：

a) CMS32M5XXX.Sct 文件需要根据使用芯片具体情况来进行设置，具体的设置参考下图(主要是设置堆栈，RAM 区间)

```
LR_IROM1 0x00000000 0x00008000  {    ; load region size_region
  ER_IROM1 0x00000000 0x00008000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
   .ANY (+XO)
  }
  RW_IRAM1 0x20000000 0x200017E0  {  ; RW data
   .ANY (+RW +ZI)
  }
  ;2K stack
  STACK_NO_HEAP 0x200017E4 UNINIT 0x00002000 { ;  Stack  and  magic  pattern  for
stack overflow detection
        mcu_test.o (STACK_BOTTOM)
        startup_cms32m5xxx.o (STACK,+Last)
      }
}
```

图 5-1：Stack 和 RAM 设置

b) startup_cms32m5xxx.s 文件里，reset_handler 的 RAM 测试，需要设置具体要测试的 RAM 的起始地址和结束地址（R6 是起始地址，R7 是结束地址）。

```
Reset_Handler        PROC
                     EXPORT  Reset_Handler            [WEAK]
                     IMPORT  SystemInit
                     IMPORT  iec60730_pc_test_start_up
                     IMPORT  iec60730_reg_test
                     IMPORT  iec60730_ram_test
                     IMPORT  __main

                     BL      iec60730_pc_test_start_up; test all D-RAM area
                     BL      iec60730_reg_test        ; test register
                     LDR     R6, =0x20000000          ; set RAM start address
                     LDR     R7, =0x20001FFF          ; set RAM end address
                     BL      iec60730_ram_test        ; test all D-RAM area

                     ;LDR     R0, =SystemInit
                     ;BLX     R0
                     LDR     R0, =__main
                     BX      R0
                     ENDP
```

图 5-2：RAM 测试地址设置

R6 和 R7 定义的 RAM 地址根据使用芯片的实际地址设置。

c) 工程 OPTION 里 IRAM 的设置，需要根据芯片的实际大小设置

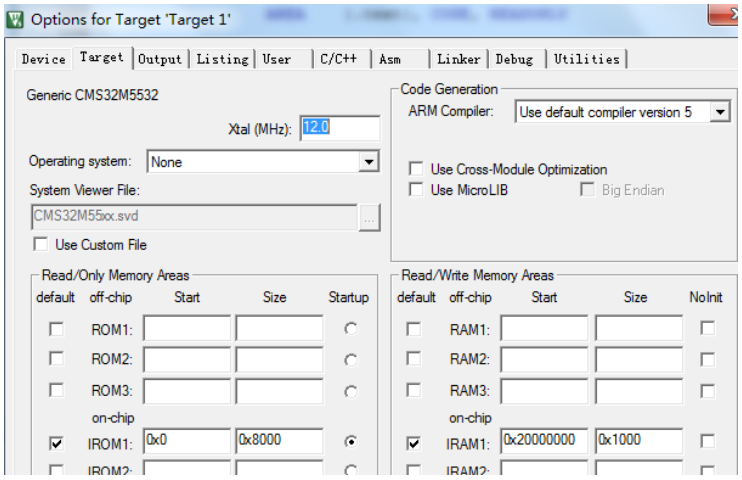

图 5-3：Option IRAM 设置

d) 在 option 的 Flash Download 界面里， 需要根据芯片的 RAM 实际大小设置 RAM for Algorithm 的 RAM Size。如下图：

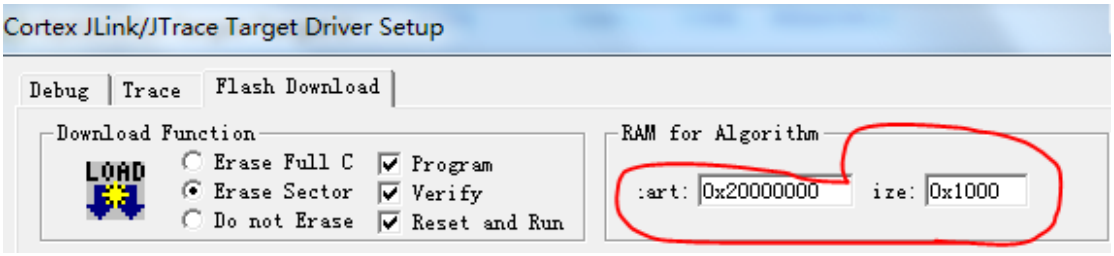

图 5-4：RAM 测试地址设置

e) 在 mcu_test.c 里，需要根据芯片 RAM 的实际大小设置下面的变量

```
#define  RAM_BIST_SIZE          4 /*  4 bytes every time */
#define  MCU_RAM_START_ADDR     0x20000000
#define  MCU_RAM_SIZE           (8*1024)
#define  MCU_RAM_END_ADDR   (MCU_RAM_START_ADDR + MCU_RAM_SIZE - RAM_BIST_SIZE)
static volatile stl uint32 t RamBistStartAddr = MCU RAM START ADDR;
```

图 5-5：芯片 RAM 地址变量设置

f) 在 mcu_test.c 里，需要根据芯片 ROM 的实际大小设置下面的变量

```
#define ROM_BIST_SIZE           8 /*  8 bytes every time */
#define ROM_START (0x0)
#define FLASH_ROM_START_ADDR (0x0)
#define FLASH_ROM_SIZE        (32*1024)
#define ROM_END    ((uint32_t)(FLASH_ROM_SIZE - 1))
#define ROM SIZE   ((uint32 t)ROM END - (uint32 t)ROM START + 1)
```

图 5-6：芯片 ROM 地址变量设置

6. IEC_60730 库使用的 driver

IEC_60730 库用到了芯片的驱动文件，在调用库的时候，请把下面的 driver 加上，如果没有添加可能在编译的时候会出现编译错误。

具体的调用情况请参考下面的列表：

```
Cms32m5xxx.h(根据实际使用的芯片头文件来)
ADCtest: ADC0.c, ADC0.h, ADC1.c, ADC1.h
Clocktest: wdt.c, core_cm0.h
FLASHtest: crc.c
GPIOtest: gpio.h
中断: timer.c
Mcu_test.c: timer.c, crc.c, system.c, system.h, wdt.c, core_cm0.h
```

7. 测试反馈

如果在测试过程中遇到了某个模块有异常，系统会根据函数的返回值给出错误的常量，如果像 CPU 寄存器或 PC 等没有返回的模块，系统会一直停留在错误的地方，直到 WDT 给出错误指示。

错误的返回有：IEC60730_TEST_FUNC_ERROR，IEC60730_TEST_PARA_ERROR。

如果每个模块都测试通过，函数会返回成功或正确的值，如果没有返回的模块，程序在模块测试成功后会继续执行下一个模块。

正确的返回有：IEC60730_TEST_NORMAL。

8. 版本修订说明

版本号	时间	修改内容
V1.00	2020 年 3 月	初始版本