Application Report
ZHCA522 –April 2013

采用非对称双核 MCU 提高系统性能 –基于 C2000 Concerto 系列

董超（Knight Dong）程鹏（Patrick Cheng） MCU FAE

摘要

本文系统地介绍了 C2000 Concerto 系列非对称双核 MCU 的基础知识和重要特点。通过对比基于两个分立 MCU 和一个双核 MCU 的方案之间的差异，强调了异构双核 MCU 方案的众多优点。以 TMS320F28M35H52C 为例介绍了 C2000 Concerto 系列的 C28x 和 Cortex-M3 两个子系统的性能、外设和软件平台，重点阐述了双核通讯 IPC 的多种高效的通讯机制和 controlSUITE 软件平台。最后，通过两个设计案例来讨论如何合理地为 C28x 和 Cortex-M3 两个内核进行任务分工，从而达到提高系统性能的目的。

目录

图表

1 背景介绍

随着各个行业朝着智能化方向的发展，嵌入式产品对能耗和效率的要求越来越苛刻。特别是在智能电网、工业和医疗等领域，一个产品的核心 MCU 处理器面临多重挑战。比如，一个自动化的马达系统或者分布式工业系统，一方面需要更多的数字信号处理能力来更精确地控制马达，另一方面也需要更多和更高级的网络接口（CAN，Ethernet 或者 Wireless 等）来实现实时的分布式监控或控制功能。再比如图 1，一个太阳能逆变系统，一方面需要 DSP 引擎来实现 DC/AC 或者 DC/DC 的算法，另一方面也需要将多个逆变器通过 Wireless 或者以太网 Ethernet 组成网络，从而实现智能诊断和监控。

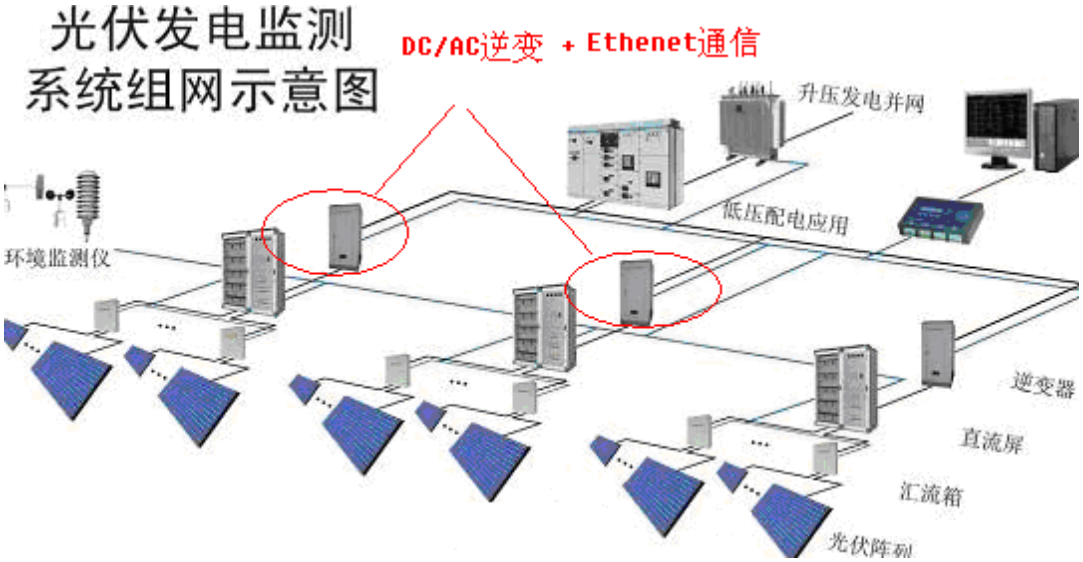

图 1 光伏发电检测 组网系统

面对这些需求，有两种传统的方案可以解决。一种方案是采用两颗单独的 MCU/DSP，其中一颗 MCU 或者 DSP 用于实现数字信号处理或者控制算法，另外一颗 MCU 实现网络协议栈或者图形显示界面等。这类方案的存在诸多缺点，首先两颗 MCU 增加了 PCB 的面积，而且双 MCU 之间的通讯的可靠性和数据吞吐率受到限制，另外，功耗也将显著增加，程序开发者甚至需要维护多个软硬件开发环境。另外一种方案是采用更高主频和更多片内资源的单核 MCU/DSP，分时地完成数据处理和辅助通信或显示功能，这种方案显著增加了系统成本和功耗，最致命的是，当客户的产品需要增加新的功能的时候，工程师需要重新计算 MCU 内核的资源和不同任务所需要的运行时间，需要更多的测试时间，因此不利于扩展和产品维护。

面对种种不足，异构双核架构应运而生，可以很好解决上述问题。事实上，非对称双核架构 MCU 可以将不同的系统任务分配于不同的 MCU 内核，分工精细，并且可以最佳地平衡性能、功耗和成本。两个 MCU 内核间的通信可以通过不同的方式来实现，比如分享内存区和消息区，非常简单和易于实现。在下面的章节，本文将以 TI 最新的 Concerto 系列产品 TMS320F28M35H52C 为例，详细阐述非对称异构双核 MCU 的优势，及其为系统带来的性能提升。

2 C2000 Concerto 双核 MCU 的特点

C2000 Concerto 系列 MCU 是 TI 推出的创新性的异构双核产品。Concerto 混合架构通过将业界最好的实时控制功能和通讯功能集成在一个芯片内，提供高性能、高效率和可靠性，从而实现实时控制环路和低延时的快速通讯响应[1]。以下从内核、存储器架构、通讯外设等方面阐述其特点。Concerto 系列 TMS320F28M35H52C 功能框图如下图 2 所示。

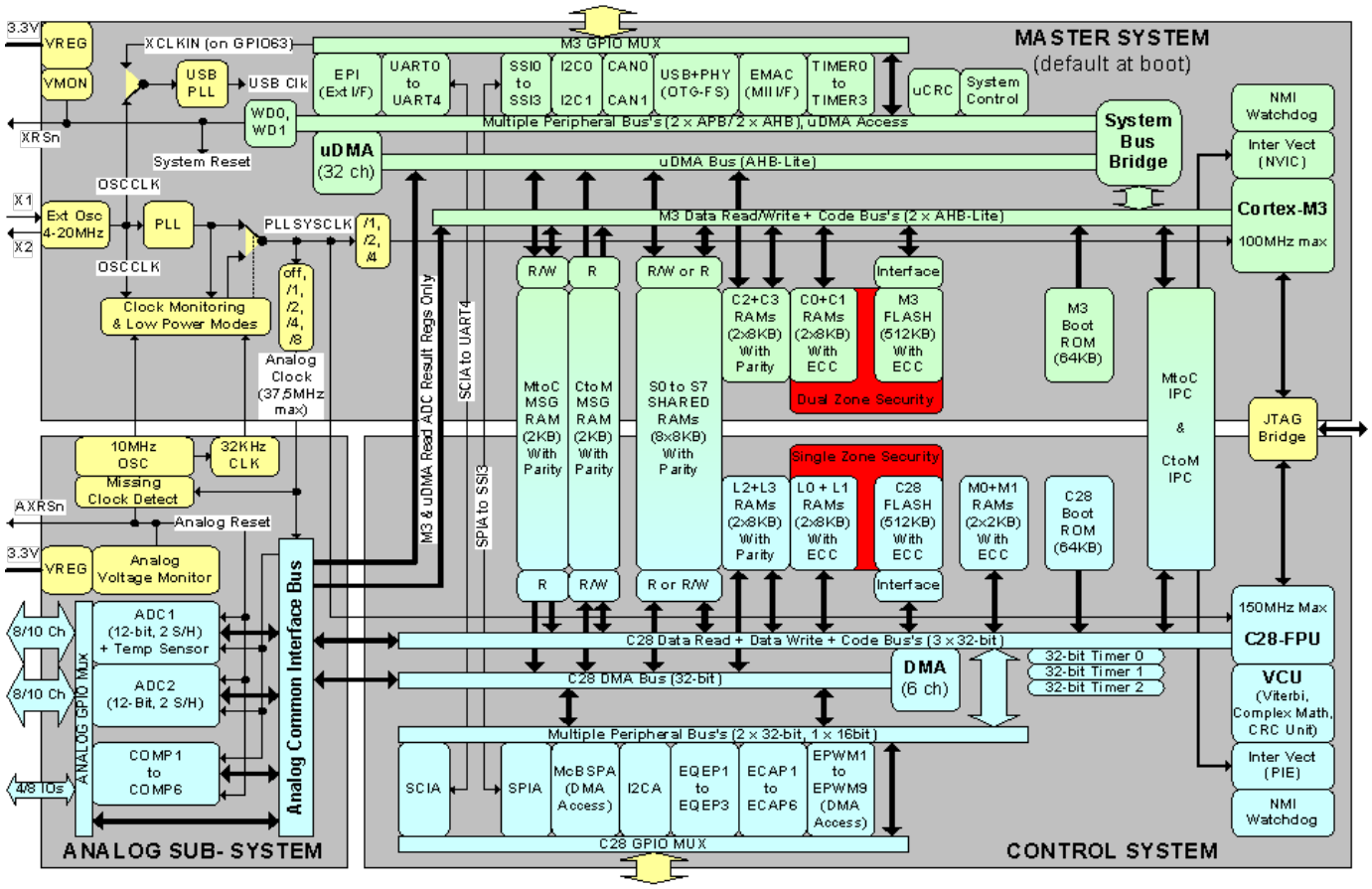

图 2 TMS320F28M35H52C 功能框图

首先是高性能的内核。Concerto 系列 MCU 包含 Cortex-M3 和 C28x 两个内核。Cortex-M3 内核是 Concerto 的主系统 Master 子系统内核，主频最高可运行于 125 MHz。Cortex-M3 内核是 32 位的 ARM 核，超高的性价比，已经被业界广泛使用，其性能和稳定性也已被用户所广泛接受，非常适用于通讯和事件控制。C28x 是新一代的 32 位 DSP 内核，是 TI 大多数现有的 C2000 产品的内核，最高可运行于 150 MHz，Concerto 中的 C28x 带浮点运算单元（Floating-Point Unit），VCU 协处理器等，性能超强，非常适用于大吞吐量的数据处理。C28x 作为 Control 子系统，宏观上受控于 Cortex-M3 Master 子系统。

其次是优化的存储器架构。如图 2 所示，TMS320F28M35H52C 的 C28x 可支配 512KB 带 ECC 校验的 Flash 存储器，64KB ROM，36KB 带 ECC 校验的 RAM；Cortex-M3 可支配 512KB 带 ECC 校验的 Flash 存储器，64KB ROM，32KB 带 ECC 校验的 RAM [3]。在两个内核之间，是共享的外设和存储区。总共 64K 字节的共享 RAM，4K 的消息 RAM。

再次是外设。如图 2 所示，TMS320F28M35H52C 的 C28x 内核可支配 DMA、高速 ADC（3MSPS）、多路高精度的 PWM(24 路 PWM 和 16 路高精度 HRPWM)、eCAP、eQEP 等为闭环控制所优化的控制外设；Cortex-M3 内核可支配多个串行接口、以太网、CAN 等工业通讯外设。同时，两个内核还可共享 ADC 等外设，增强整个系统的灵活性。

最后是软件架构。如图 3 所示，controlSUITE 是一个集成所有 C2000 MCU 的开发资源和软件包和开发平台，它为 TMS320F28M35H52C 的开发者提供了外设例程、DSP 库、文档、开发板资料。ControlSUITE 还提供免费的全功能实时操作系统 TI-RTOS 平台，如图 4 所示，TI-RTOS 是基于 SYS/BIOS 实时内核，集成了稳定的中间件，例如 TCP/IP 协议栈、USB 协议栈、FAT 文件系统、IPC 多核通讯组件等。

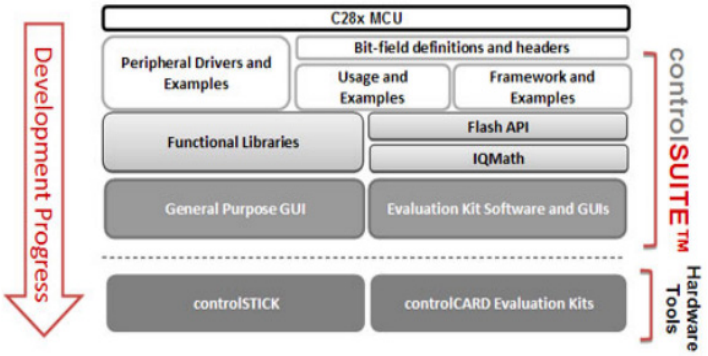

图 3 controlSUITE 软件架构

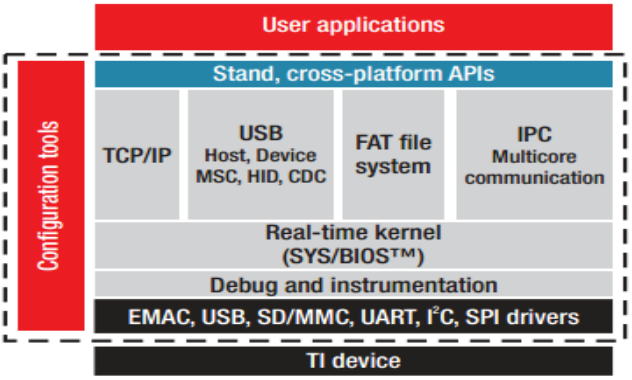

图 4 TI-RTOS 组件架构

3 IPC 内核间通信

Cortex-M3 和 C28x 内核之间的通信主要完成两大功能，一是数据通信，二是传递状态和控制信息。IPC（内核间通讯）的数据通信需要较大的 RAM 来支持，而传递状态和控制等信息只需要一系列状态标志位即可。此外，Cortex-M3 侧的 UART4 与 C28x 侧的 SCIA；以及 Cortex-M3 侧的 SSI3 与 C28x 侧的 SPIA 在 Concerto 内部实现互联，不需要在芯片外部硬件连接，而是否使能这类功能则有 Cortex-M3 系统配置。

3.1 Message RAM 内存区

TMS320F28M35H52C 使用 Message RAM 实现 IPC 的数据通信。如图 5 所示，2K 字节的 MTOC Message RAM 用于从 Master （Cortex-M3）子系统向 Control（C28x）子系统传递消息；2K 字节的 CTOM Message RAM 用于从 Control 子系统向 Master 子系统传递消息。由于两个子系统都配有 DMA 外设，因此，DMA 也可以读写 Message RAM，从而提高系统效率。Message RAM 区通过 RAM 内存的读写权限保证了 Message 的互斥访问，例如，C28x CPU 与 DMA 可以读写访问 CTOM Message RAM 区，而 Cortex-M3 CPU 和 uDMA 只能读访问 CTOM Message RAM。同样，两个内核对于 MTOC Message RAM 区的读写访问权限则正好相反。

	C28x	M3	28x DMA	M3 µDMA
C28x to M3 Message RAM (2KB)	R/W	R	R/W	R
M3 to C28x Message RAM (2KB)	R	R/W	R	R/W

图 5 IPC Message RAM

Message RAM 仅仅作为 IPC 的数据缓存，IPC 还需借助于特定的控制逻辑电路来完成。如图 6 所示，Master 子系统和 Control 子系统都是通过 5 个寄存器来实现 IPC 的逻辑流程控制：IPCACK、IPCSTS、IPCFLG、IPCCLR、IPCSET。这 5 个寄存器都是 32 位，每一个 bit 对应于 IPC 的一个通道，因此最多可实现 32 个通道的握手通信。Bit0 到 Bit3 总共 4 个通道可以触发消息接收方的 IPC 中断，Bit4 到 Bit31 共 28 个通道则需要消息接收方的软件查询来获取 Message RAM 中是否收到数据。如果两个内核之间仅仅传递状态和控制信息（例如 RTOS 中的 Semaphore），仅通过以上寄存器便可以实现，而无需 Message RAM 的参与。

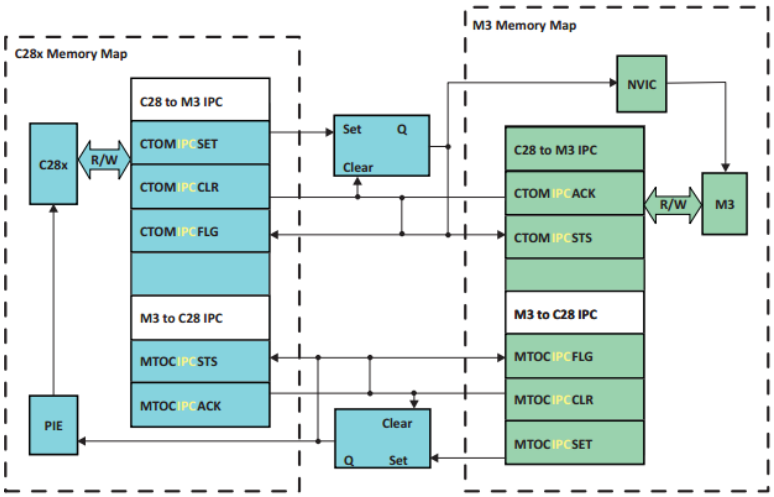

图 6 Concerto IPC 控制模块

以下通过举例 Master 子系统往 Control 子系统发送一帧数据，来简单介绍 IPC 模块的操作流程。

1. Cortex-M3 先在 MTOC Message RAM 中写入一帧数据；
2. Cortex-M3 置位 MTOCIPCSET（CM3 映射存储器区）的 Bit9，如图 6 所示，此时 MTOCIPCSTS（C28x 映射存储器区）的 Bit9 也将置位；
3. C28x 轮询 MTOCIPCSTS 的 Bit9，查询到 Bit9 已置位；（如果之前的操作是 Bit0 到 Bit3 其中之一， 则将触发 C28x 产生一个 IPC 中断）

4. C28x 读 MTOC Message RAM 中的数据，此时，Cortex-M3 成功将一帧数据发送至 C28x。

3.2 Shared RAM 内存区

大部分情况下，2K 字节的 IPC Message RAM 区能够满足 C28x 和 M3 子系统之间的数据通信，配合 DMA，通信效率也可以进一步提高。如果用户希望一次性在两个子系统传递更大块的数据，另一种方法是通过 Shared RAM 内存。

TMS320F28M35H52C 有一个 64K 字节大小的 Shared RAM 区，总共 8 块 S0-S7，每块 8K 字节大小，如图 7 所示。Cortex-M3 可以设置让任何一块 Shared RAM 区由 C28x 或 M3 主控，比如，映射 S0 至 C28x 侧以后，C28x CPU 和 DMA 可以读写 S0，而 M3 和 uDMA 将只能读 S0，不能写入和预取。

假如 Cortex-M3 需要一次性发送 6K 字节的数据到 C28x 侧，它可以先将 Shared RAM 区 S0 映射到本地存储器空间，接着通过 IPC 发送一个标志位给 C28x 来通知其可以将数据取走。

µDMA Access	M Address (Byte-Aligned)(1)	Master Subsystem RAMs	Size (Bytes)	C Address (x16 Aligned)(2)	C DMA Access(2)
no	2000 0000 – 2000 1FFF	C0 RAM (ECC, Secure)	8K		
no	2000 2000 – 2000 3FFF	C1 RAM (ECC, Secure)	8K		
yes	2000 4000 – 2000 5FFF	C2 RAM (Parity)	8K		
yes	2000 6000 – 2000 7FFF	C3 RAM (Parity)	8K		
yes	2000 8000 – 2000 9FFF	S0 RAM (Parity, Shared)	8K	0000 C000 – 0000 CFFF	yes
yes	2000 A000 – 2000 BFFF	S1 RAM (Parity, Shared)	8K	0000 D000 – 0000 DFFF	yes
yes	2000 C000 – 2000 DFFF	S2 RAM (Parity, Shared)	8K	0000 E000 – 0000 EFFF	yes
yes	2000 E000 – 2000 FFFF	S3 RAM (Parity, Shared)	8K	0000 F000 – 0000 FFFF	yes
yes	2001 0000 – 2001 1FFF	S4 RAM (Parity, Shared)	8K	0001 0000 – 0001 0FFF	yes
yes	2001 2000 – 2001 3FFF	S5 RAM (Parity, Shared)	8K	0001 1000 – 0001 1FFF	yes
yes	2001 4000 – 2001 5FFF	S6 RAM (Parity, Shared)	8K	0001 2000 – 0001 2FFF	yes
yes	2001 6000 – 2001 7FFF	S7 RAM (Parity, Shared)	8K	0001 3000 – 0001 3FFF	yes

图 7 Shared RAM 区

3.3 IPC 的软件驱动

controlSUITE 软件开发包中提供 2 种 IPC 的软件驱动库，IPC Driver 和 IPC_Lite Driver。IPC_Lite Driver 仅使用 IPC 寄存器来实现通信，不需要额外的 RAM，但是用户只能支持一个 IPC 中断服务 ISR，且不支持以队列形式来处理 IPC 请求。IPC_Lite Driver 使用方式如下：

Sending Processor:

1. **Calls command function:**
   - Example:

```
while (IPCLiteMtoCDataRead
(IPC_FLAG2, addr, IPC_LENGTH_16_BITS, IPC_FLAG17) != STATUS_PASS) {}
```

   - Returns:
     - STATUS_PASS (0) if status/interrupt flags are clear and command is sent.
     - STATUS_FAIL(1) if status or interrupt flags are still set .
2. **Calls GetResult function when data is needed:**
   - Example:

```
while(IPCLiteMtoCGetResult
(&Data,IPC_LENGTH_16_BITS, IPC_FLAG17) != STATUS_PASS) {}
```

   - Returns:
     - STATUS_PASS (0) if data is ready in *pvData parameter.
     - STATUS_FAIL(1) if status flag is still set (command failed, so no result available).

Receiving Processor:

1. **In STORIPC ISR, decodes command in STORIPCCOM register. (S=Sending, R=Receiving Processor)**
2. **Based on Command, calls function with same name as command function*:**
   - Example:

```
IPCLiteMtoCDataRead(IPC_FLAG2, IPC_FLAG17);
```

3. **Receiving function:**
   - If command is VALID:
     Processes command and status and interrupt flags are ack'd.
   - If command is INVALID:
     Only interrupt flag is ack'd. Status flag remains set. Function returns without processing.

*Exception: IPCLiteCtoMReqMemAccess() function (M3 responds with IPCLiteCtoMSetBitsProtected()/IPCLiteCtoMClearBitsProtected())

1，主动发起数据请求的内核会首先调用 IPC_Lite Driver 提供的名函数。在这个例子汇总，M3 是发送数据的内核并执行"IPCLiteMtoCDataRead" 函数。

- IPC_FLAG2 是 C28 中断标志，指示 C28 内核一个消息到来。
- IPC_FLAG17 是响应标志，C28 用其指示 M3 核一个命令已经被处理。
- 需要读取数据的 C28 的地址也被作为一个参数传递给 C28 内核。
- 这个函数在 while 循环中被调用的原因是，它可能返回 STATUS_FAIL 并且不会发送信息给 C28 直至 MtoC IPC 中断 2 和标志 17 可用, 之后，该函数返回 STATUS_PASS.

2，被动接收数据请求的内核会在 ISR 中解析其 IPCCOM 寄存器的命令。这个例子中，C28 MtoCIPCINT2 ISR 知道标志置位，解析 MTOCIPCCOM 寄存器的命令，识别出是读数据命令。

3，被动接收数据请求的内核会调用与主动发起数据请求的内核相同的函数名。这个例子中，C28 执行 IPCLiteMtoCDataRead， IPC_FLAG2 作为中断标志参数， IPC_FLAG17 作为状态标志参数。

4，如果接收到命令有效，IPC_Lite 的驱动函数会处理读命令并确认（acknowledges）状态和中断标志。如果接收到的命令无效，则只有中断标志被确认（acknowledged）用来释放中断给后续的命令，而状态标志仍然置位。

IPC Driver 通过在 Message RAM 中建立环形缓冲区，使得多个 IPC 通信命令可以以队列的形式被缓冲，然后逐个处理，并且可以同时支持多个 IPC 中断服务程序 ISR，当然，IPC Driver 需要更多的 RAM 来支持。和 IPC-Lite 不同，为了使用 IPC 驱动，需要在 M3 和 C28 的项目中增加一些设置。

第一步是在 M3 和 C28 的链接定位文件（.cmd）中添加 IPC 循环缓冲区和指针段到 CTOM 和 MTOC message RAM。如下所示：

C28 .CMD File:

```
MEMORY
{
    PAGE 0 :
       CTOMRAM : origin = 0x03F800,
                   length = 0x000400
       MTOCRAM : origin = 0x03FC00,

                 length = 0x000400
}
SECTIONS
{
   GROUP : > CTOMRAM, PAGE = 1
   {
        PUTBUFFER
        PUTWRITEIDX
        GETREADIDX
   }
   GROUP : > MTOCRAM, PAGE = 1
   {
        GETBUFFER :    TYPE = DSECT
        GETWRITEIDX :  TYPE = DSECT
        PUTREADIDX :   TYPE = DSECT
   }
}
```

M3 .CMD File:

```
MEMORY
{
    CTOMRAM (RX) : origin = 0x2007F000,
                   length = 0x00000800
    MTOCRAM (RWX): origin = 0x2007F800,
                   length = 0x00000800
}
SECTIONS
{
   GROUP : > MTOCRAM
   {
        PUTBUFFER
        PUTWRITEIDX
        GETREADIDX
   }

   GROUP : > CTOMRAM
   {
        GETBUFFER :   TYPE = DSECT
        GETWRITEIDX : TYPE = DSECT
        PUTREADIDX :  TYPE = DSECT
   }
}
```

第二步，应用程序源码中必须定义并且初始化至少一个 volatile global tIpcController 变量 (为 C28 – M3 IPC 中断使用)，如下所示：

M3 tIpcController Definition and Intialization example:

```
//*************************************************************
//
// At least 1 volatile global tIpcController instance is required
// when using IPC API Drivers.
//
//*************************************************************
volatile tIpcController g_sIpcController1;
volatile tIpcController g_sIpcController2;

void
main(void)
{
    //
    // Initialize IPC Controllers
    //
    IPCMInitialize (&g_sIpcController1, IPC_INT1, IPC_INT1);
    IPCMInitialize (&g_sIpcController2, IPC_INT2, IPC_INT2);
```

Sending Processor:

1. **Calls command function:**
   - Example:

```
IPCMtoCSetBits(&g_sIpcController1, addr,(unsigned long)SETMASK_16BIT,
    IPC_LENGTH_16_BITS, ENABLE_BLOCKING);
```

2. **Returns:**
   - **STATUS_PASS (0):** There was room in "Put" buffer – command sent successfully.
   - **STATUS_FAIL (1):** "Put" buffer was full of commands – command not sent.
3. **Call the next command function…**

Receiving Processor:

1. **In STORIPC ISR, call IpcGet function to grab messages as long as there are still messages in the "Get" buffer.**
   - **Example:**

```
while (IpcGet (&g_sIpcController1, &sMessage, DISABLE_BLOCKING)!=
STATUS_FAIL)
{…}
```

2. **Inside IpcGet() loop, application code decodes sMessage.uICommand andcalls function with same name as command function*:**
   - Example:

```
IPCMtoCSetBits(&sMessage);
```

3. **Outside of IpcGet() loop, application code acknowledges IPC interrupt flag.**

*Exception: IPCCtoMReqMemAccess() function (M3 responds with IPCCtoMSetBitsProtected()/IPCCtoMClearBitsProtected())

1. 主动发起数据请求的内核会首先调用 IPC Driver 提供的一个命令函数。这个例子中，M3 是发起数据请求的内核，执行"IPCMtoCSetBits"函数。

- g_sIpcController1 是 tIpcController 类型的变量，控制 M3 和 C28 IPC 中断通道之间的通信。
- SETMASK_16BIT 是 16-bit 掩码，指示应该被置位的位域。IPC_LENGTH_16_BITS 指示命令操作的数据对象是 16-bits。
- 函数被配置成允许阻塞 "ENABLE BLOCKING", 意味着函数会一直等待直到 M3 PutBuffer 有空的缓冲区。如果函数被配置成不许阻塞 "DISABLE BLOCKING", 一旦"Put"缓冲区满，它会立即返回 STATUS_FAIL 并且不会发送消息到 C28。如果"Put"缓冲区有空余，函数会返回 STATUS_PASS，消息被成功发送到 C28.

2. 被动接受数据请求的内核会连续调用 IpcGet 函数来读取 sMessage 结构体里的消息，只要有消息在"Get"缓冲区。在 ISR 中 IpcGet 函数被调用，C28 侧的 tIpcController 变量被用来绑定两个相同的 M3 和 C28 的 IPC 中断通道（和 M3 侧用来发送命令的 tIpcController 相同）。

3. 即使被动接收数据的内核没有确认（acknowledged）IPC 中断标志，主动请求数据的内核仍然可以连续发送消息，因为 tIpcController 变量会把消息排队放到"Put"缓冲区（与被动接收数据请求的内核的"Get"缓冲区相同）。被动接收数据请求的内核的 ISR 会连续获取并处理消息，直至"Get"缓冲区为空。

4 Cortex M3 和 C28x 核的任务分工

Cortex-M3 子系统的优势在于处理事务和管理通讯外设的能力，C28x 内核子系统在实时控制和数据处理方面性能优越。因此，在一个系统中，合理地分配两个子系统的所处理的事务，优化资源的配置是至关重要的。基于 Concerto 的系统，一方面应当最大化地使用 C28x 的 DSP 和实时控制优势，发挥 ADC、PWM、C28x 组成的闭环系统的优势；另一方面应将人机界面、通讯协议栈、文件系统等尽可能运行在 Cortex-M3 子系统一侧。下面通过两个应用案例来讨论如何通过合理任务分工来提高系统效率。

4.1 光伏逆变器网络节点

光伏逆变器的主要功能是把光伏面板输出的 DC 直流电逆变为 110V/220V 的 AC 交流电，最终接入电网或者离网输电至用电设备。在一个大功率的光伏发电网络拓扑中，往往有许多个光伏逆变器，这些逆变器需要被监测，控制中心需要实时观测各个光伏逆变器的工作状态。因此，光伏逆变器网络节点的功能主要包括 DC/AC 逆变器和网络连接。如图 9 所示，C28x 子系统（运行于 100MHz）完成 MPPT 和 DC/AC 逆变算法。网络连接可以有多种方式，常用的方式包括 Ethernet 以太网、RS485 或 CAN 等，TMS320F28M35H52C 的 Cortex-M3 子系统（100 MHz）带 Ethernet、RS485 和 CAN 等接口，支持多种有线和无线连接功能。

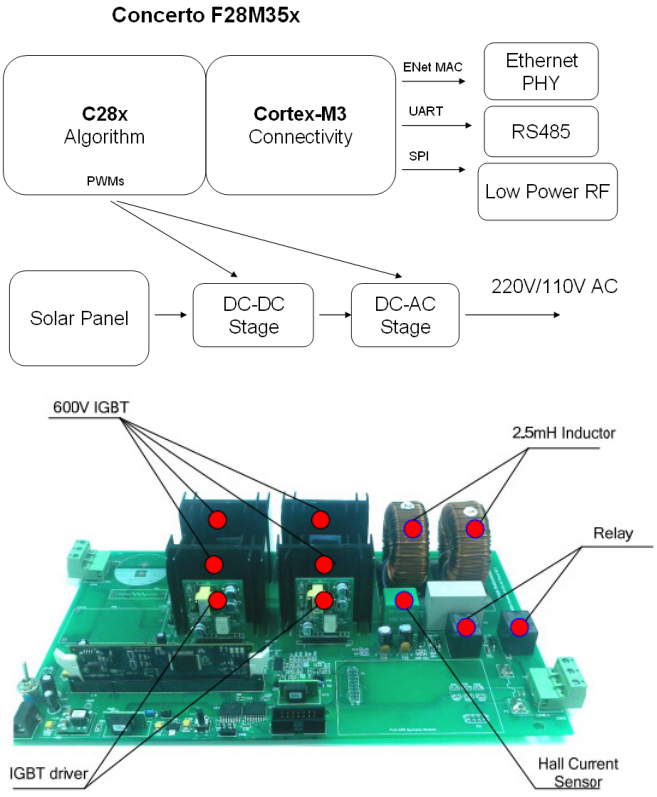

图 8 Solar HV DC-AC Kit

对于 C28x 子系统，采用状态机的设计思路来区别不同的系统状态。不同的状态代表着不同的运行模式，其它的任务能够根据特定的运行模式采取相应的行动。例如，可以采用下面 5 种不同的运行模式。

- Power On Mode: 系统上电后进入 Power On Mode，系统上电后，F28M35H52C1 中的 Cortex-M3 内核 boot 程序首先启动，此时 C28x 控制子系统和模拟子系统处于复位状态，需要 M3 主子系统将其从复位状态解除。M3 主子系统设定 M3 和 C28x 内核的时钟频率，由于 M3 和 C28x 的主频之比必须为整数比，因此 M3 和 C28x 的主频设定只能为 60/60MHz、75/150MHz、100/100MHz。在 M3 和 C28x 的主频设定完成之后，需要由 M3 主子系统对整个芯片的外设资源以及 GPIO 进行配置，来决定哪些 GPIO 可以由 C28x 控制子系统进行配置。本系统中 M3 和 C28x 主频设定为 75/150MHz。当所有的初始化操作完成后，系统自动转入到 Standby Mode。
- Standby Mode：所有的 PWM 和继电器被关闭。系统等待启动命令，也检测是否发生错误。
- Soft Start Mode: 接收到启动命令，系统进入软启动模式，PWM 和继电器开启。如果启动成功而且没有错误发生，系统自动进入正常逆变模式。
- Normal Inverter Mode: 该模式下系统输出功率，如果没有错误发生也没有收到关闭命令，系统会一直处于这个模式。
- Fault Mode: 如果发生错误，例如母线过压，系统立即进入 Fault Mode。所有 PWM 输出被封锁，输出继电器被断开。Fault 状态可以被按键或者 GUI 清除。清除后，系统会返回到 Standby Mode

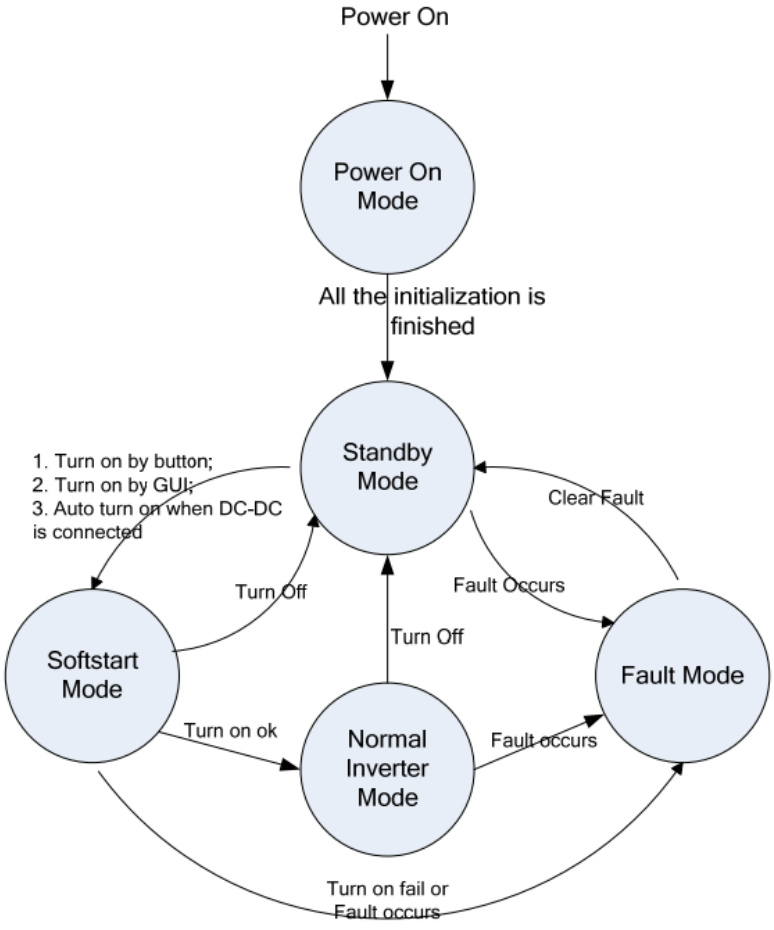

图 90 C28x 端程序系统状态机

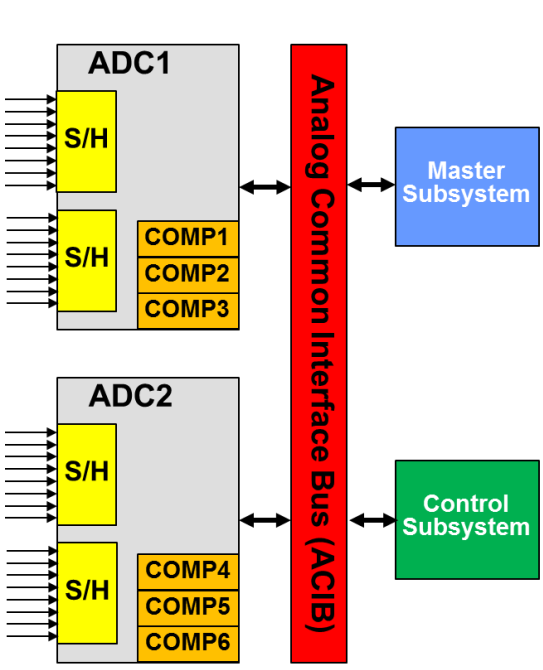

图 11 Concerto ADC 框图 1

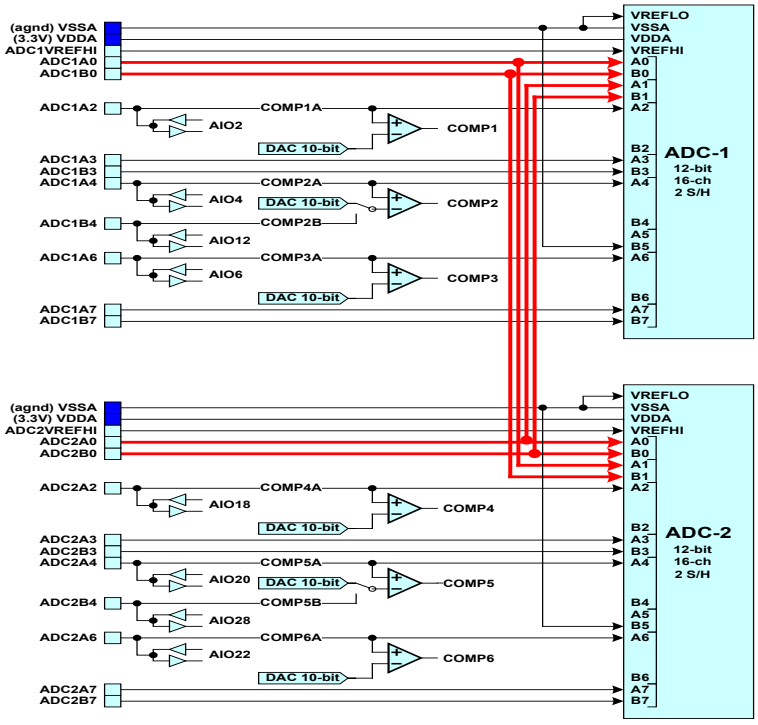

图 101 Concerto ADC 框图

Concerto 系列有两个 12-bit ADC 模块，每个 ADC 模块包含两个采样保持电路，支持同步或者顺序采样模式，3 个带 10-bitDAC 的模拟比较器，模拟信号的输入范围 0V~3.3V(内部参考)或者 VREFHI/VREFLO 比例关系（外部参考）。

图 11 给出了详细的 ADC 配置，TMS320F28M35H52C 的 Cortex-M3 和 C28x 内核都能够访问 ADC 的结果寄存器，而且 2 个 ADC 模块共享 4 个模拟输入， Concerto ADC 模块的这个特性允许对关键信号进行安全性验证，提高系统的可靠性。

4.2 电力线载波通讯 PLC 智能家居网关

智能家居网关能够将房间内的智能电器以有线或者无线的方式组成网络，集中进行管理。如图 10 所示，TMS320F28M35H52C 的 C28x（运行于 150MHz）主要完成电力线载波通信（Power Line Carrier Communication）PLC 的 OFDM 物理层算法。Cortex-M3（75MHz）的运行 TCP/IP 协议接入以太网，其次，可选地通过 UART 接口外接 GPRS 模块或者通过 EBI 外扩总线连接 TFT 彩屏用户界面。

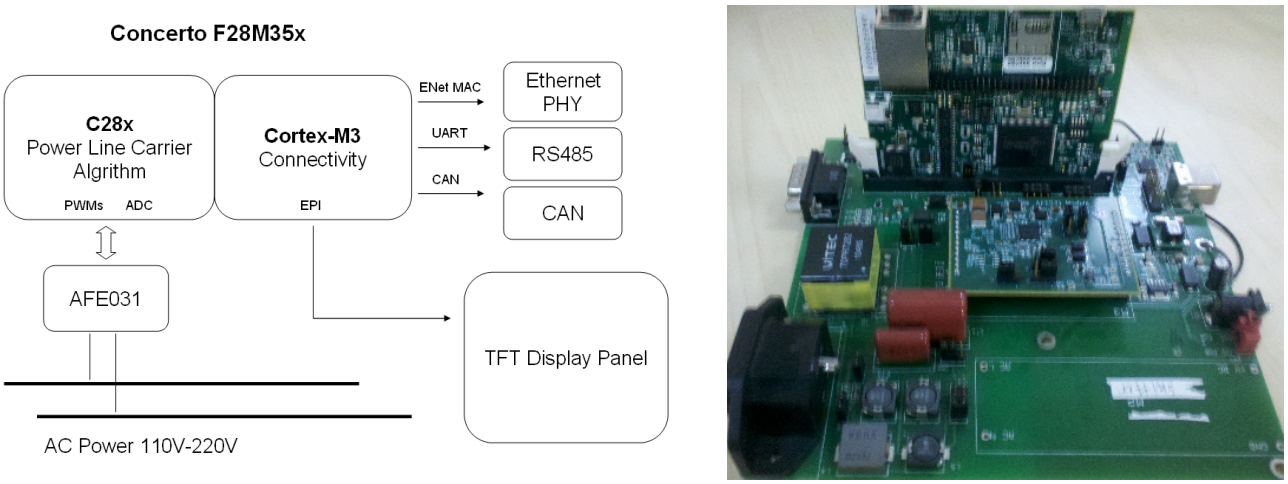

图 112 Concerto PLC 电力线载波 EVM 板

5 总结

Concerto C2000 异构双核 MCU 将 C28x DSP 内核与 ARM 公司的 Cortex-M3 内核融合在一起，展示出高效的数据处理、数据通讯和事件管理的强大性能。C28x 和 Cortex-M3 两个子系统分工明确，又通过 IPC 模块巧妙实现了实时高效地核间通讯。在软件方面，controlSUITE 开发平台提供多种组件，包括 TCP/IP 协议栈、IPC 驱动、USB 协议栈、FAT 文件系统等，可帮助用户更快地开发出创新性的产品。

6 参考

[1]. Performance Without Compromise: Implementing Real-time Control And Communications With a Dual-subsystem Microcontroller, Sangmin Chon, Brett Novak
[2]. Concerto Brochure
[3]. TMS320F28M35Hx Concerto Microcontrollers (Rev. D)
[4]. Concerto F28M35x Technical Reference Manual (Rev. B)
[5]. F28M35x Peripheral Driver Library User's Guide
[6]. Quick Start Guide for Concerto-based Solar Explorer Development Kit

重要声明

德州仪器(TI) 及其下属子公司有权根据 JESD46 最新标准, 对所提供的产品和服务进行更正、修改、增强、改进或其它更改， 并有权根据 JESD48 最新标准中止提供任何产品和服务。客户在下订单前应获取最新的相关信息, 并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的组件的性能符合产品销售时 TI 半导体产品销售条件与条款的适用规范。仅在 TI 保证的范围内，且 TI 认为 有必要时才会使用测试或其它质量控制技术。除非适用法律做出了硬性规定，否则没有必要对每种组件的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用 TI 组件的产品和应用自行负责。为尽量减小与客户产品和应 用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 组件或服务的组合设备、机器或流程相关的 TI 知识产权中授予 的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从 TI 获得使用这些产品或服 务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是 TI 的专利权或其它 知识产权方面的许可。

对于 TI 的产品手册或数据表中 TI 信息的重要部分，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况 下才允许进行复制。TI 对此类篡改过的文件不承担任何责任或义务。复制第三方的信息可能需要服从额外的限制条件。

在转售 TI 组件或服务时，如果对该组件或服务参数的陈述与 TI 标明的参数相比存在差异或虚假成分，则会失去相关 TI 组件 或服务的所有明示或暗示授权，且这是不正当的、欺诈性商业行为。TI 对任何此类虚假陈述均不承担任何责任或义务。

客户认可并同意，尽管任何应用相关信息或支持仍可能由 TI 提供，但他们将独力负责满足与其产品及在其应用中使用 TI 产品 相关的所有法律、法规和安全相关要求。客户声明并同意，他们具备制定与实施安全措施所需的全部专业技术和知识，可预见 故障的危险后果、监测故障及其后果、降低有可能造成人身伤害的故障的发生机率并采取适当的补救措施。客户将全额赔偿因 在此类安全关键应用中使用任何 TI 组件而对 TI 及其代理造成的任何损失。

在某些场合中，为了推进安全相关应用有可能对 TI 组件进行特别的促销。TI 的目标是利用此类组件帮助客户设计和创立其特 有的可满足适用的功能安全性标准和要求的终端产品解决方案。尽管如此，此类组件仍然服从这些条款。

TI 组件未获得用于 FDA Class III（或类似的生命攸关医疗设备）的授权许可，除非各方授权官员已经达成了专门管控此类使 用的特别协议。

只有那些 TI 特别注明属于军用等级或“增强型塑料”的 TI 组件才是设计或专门用于军事/航空应用或环境的。购买者认可并同 意，对并非指定面向军事或航空航天用途的 TI 组件进行军事或航空航天方面的应用，其风险由客户单独承担，并且由客户独 力负责满足与此类使用相关的所有法律和法规要求。

TI 已明确指定符合 ISO/TS16949 要求的产品，这些产品主要用于汽车。在任何情况下，因使用非指定产品而无法达到 ISO/TS16949 要求，TI不承担任何责任。

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP应用处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity	德州仪器在线技术支持社区	www.deyisupport.com

邮寄地址： 上海市浦东新区世纪大道 1568 号，中建大厦 32 楼 邮政编码： 200122