Application Notes
ZHCAA09 – 5.2020

C2000 Fast Serial Interface (FSI) 在多机通信中的应用

Aki Li

ABSTRACT

针对工业控制不断更新的需求，C2000 产品已经升级到第三代，在实时控制方面，引入了三角函数运算加速单元进一步提高运算处理能力，提供了多个 ADC 模块及窗口比较器增强采样及保护的快速性和灵活性。同时，在实时通信方面，引入了新一代快速串行通信接口 Fast Serial Interface (FSI)，能支持在隔离的情况下最高 200Mbps 的数据传输速率。本文主要介绍了 FSI 的基本用法，以及如何将 FSI 应用在多机通信场合，具体包括菊花链和星型两种多机连接拓扑。本文也提供了相关的代码示例以及测试结果，以便加快用户开发过程。

Contents

Figures

1 FSI 基本功能介绍

FSI 接口是 C2000 最新推出的高可靠快速串行通信接口，具有可编程的数据长度、硬件 CRC 校验、ECC 等功能特性。针对隔离情况下的通信应用场景，FSI 的接收端口提供了 delay line control 功能，可在接收到的数据和时钟信号中插入自定义的延时，用于补偿由于信号隔离、缓冲电路、线路长度不同等引入的信号延时，从而保证接收端信号的同步性，实现高速而可靠的通信。一般来说，作为点对点的通信接口，FSI 可用在以下两种场景：

1) 两个 MCU 直接连接进行通信；

2) 芯片之间或者板级之间，通过隔离芯片进行通信；

FSI 具有独立的发送端口 FSITX 和接收端口 FSIRX，可实现完全独立的收发通信，图 1 给出了 FSI 的收发端口框图。每个端口包含 3 个信号线：时钟线 CLK 和数据线 D0 以及 D1，其中数据线 D1 可用于将数据传输速率加倍，也可选用作 GPIO 功能。因此，实现点对点的 FSI 通信至少需要 4 个信号线。由于 FSI 最大支持的通信时钟频率为 50MHz，而时钟的上升沿和下降沿皆作为数据的有效位判定时刻，因此，理论上在使能两个数据线的条件下理论上可以实现高达 200Mbps 的传输速率。

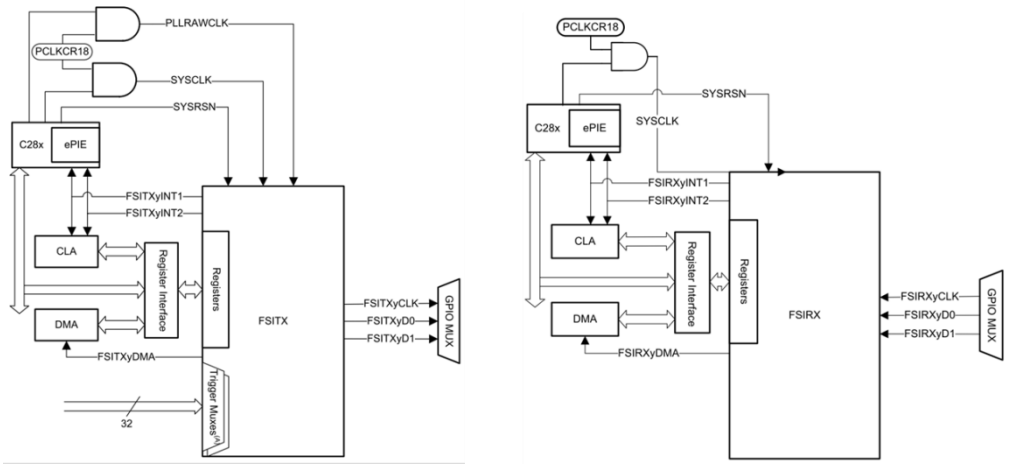

图 1 FSI 的收发端口框图

在工业电源或工控应用中，多个控制器之间的通信场景十分普遍，同时对通信实时性的要求越来越高， 相比采用传统的 CAN、SPI 和 UART 等通信接口， FSI 接口有着独有的优势，特有的 delay line control 可以确保在隔离情况下的高速通信可靠性。一般来说，由于采用 LVCMOS 接口，FSI 无需额外

添加通信收发芯片，如果实际的通信距离较长，可考虑采用 LVDS 或 RS485 收发器，增强信号传输的鲁棒性，具体可参考 FSI Adapter Board。

本文主要探究将 FSI 应用在菊花链和星型这两种常用的多机通信拓扑。目前新一代 C2000 F28004x 只提供 1 个发送端口和 1 个接口端口，只适用于菊花链拓扑；F2838x 提供 2 个发送端口和 8 个接口端口，还可以用于实现一对多的星型拓扑。

2 FSI 在菊花链多机通信拓扑中的应用

2.1 菊花链拓扑框图

图 2 给出了 FSI 用于菊花链通信拓扑的示意图，对于 N(N≥2)个 C2000 控制器互连的系统，第 i 个控制器分别与 i-1 的发送端口 FSITX 和 i+1 的接收端口 FSIRX 相连。

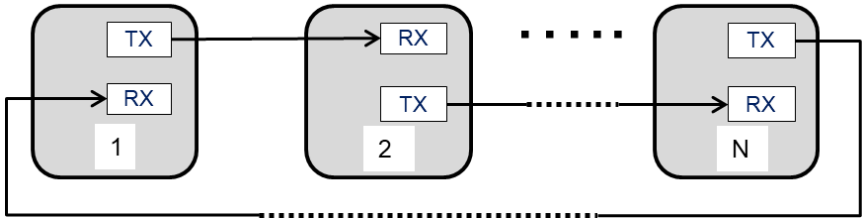

图 2 FSI 用于菊花链通信拓扑

2.2 “握手”机制

在开启菊花链式数据传输之前，需进行“握手”以确保每个芯片都做好了发送和接收数据的准备。为此，需指定其中一个芯片，如芯片 1，作为启动握手流程的主机，其余的 N-1 个芯片作为从机。握手过程主要包含以下步骤，具体的过程参考示意图 3。

1) 菊花链中的每个芯片设置 FSITX/FSIRX 的数据帧类型为 Ping，并使能 Ping Frame Received 接收中断；
2) 开始第一轮的 ping 回路：
   a. 主机发送 Flush 信号唤醒芯片 2，然后再发送 Tag0(0000)，进入等待状态；如果主机接收到 Tag0，则进入第二轮 ping 回路；否则，重复此步骤；
   b. 其他的芯片在起始时进入等待状态，若接收到有效的 ping 数据帧为 Tag0，则发送 Flush 信号和 Tag0 给下一个芯片，进入第二轮 ping 回路；否则，重复此步骤；
3) 开始第二轮的 ping 回路：
   a. 主机发送 Tag1 (0001)，进入等待状态；如果主机接收到 Tag1，则继续下一步骤（握手完成）；否则，重复此步骤；
   b. 其他的芯片进入等待状态，若收到 Tag1 则继续发送 Tag1 给下一个芯片；否则，重复此步骤；
4) 握手成功。

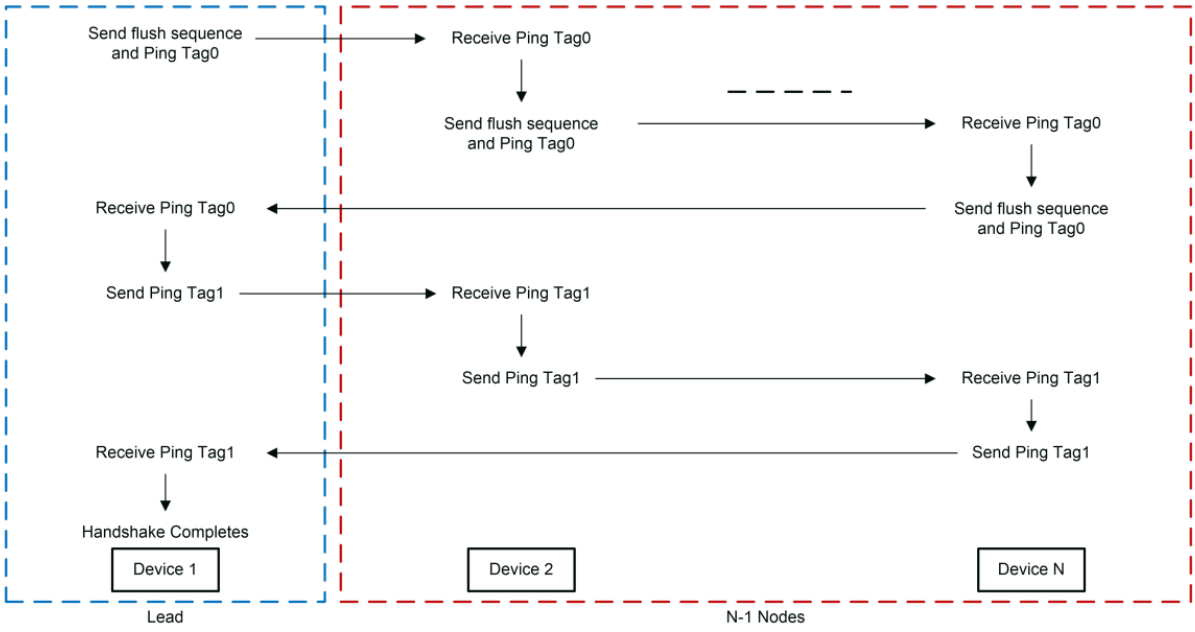

图 3 菊花链式通信握手过程

对于菊花链的多机通信拓扑，主机和从机的指定只在握手阶段有实际意义，在握手结束之后，所有芯片的地位是一样的。

2.3 数据传输配置

本文的配置示例基于 FSI 的参考例程，例程获取方式：安装最新的 C2000ware，文件路径为

C:\ti\c2000\C2000Ware_<version_number>\driverlib\f28004x\examples\fsi

- fsi_ex16_daisy_handshake_lead (用于握手过程中的主机)
- fsi_ex16_daisy_handshake_node (用于握手过程中的从机)

通常情况下，在数据传输前，需对 FSITX 和 FSIRX 配置发送/接收的数据的帧类型、数据的长度、user data 和 tag 的内容，若是发送数据帧，还需将发送的数据写入 TX 的数据缓冲器。其中 user data 或 tag 的内容皆可自定义配置，可用于判断收到的数据来自哪个发送芯片，或作为其他功能的标记。参考采用 Driverlib 形式的配置示例：

```
// TX setting part
FSI_setTxFrameType(FSITXA_BASE, FSI_FRAME_TYPE_NWORD_DATA);
FSI_setTxSoftwareFrameSize(FSITXA_BASE, nWords);
FSI_setTxDataWidth(FSITXA_BASE, nLanes);
FSI_setTxUserDefinedData(FSITXA_BASE, txUserData);
FSI_setTxFrameTag(FSITXA_BASE, txDataFrameTag);
```

```
// RX setting part
FSI_setRxSoftwareFrameSize(FSIRXA_BASE, nWords);
FSI_setRxDataWidth(FSIRXA_BASE, nLanes);
```

触发 FSITX 发送数据有 3 种方式，分别是软件触发、外部触发源(EPWMx-SOCA/B)触发和利用 DMA 进行触发控制。本文给出的代码示例基于 DMA 的触发方式，基于以下两点考虑：

1) 利用 DMA 将数据写入或移出 FSI 的数据缓冲器，无需 CPU 干预，将节省大量带宽；

2) FSITX 发送完成数据或者 FSIRX 接收完成数据，可触发 DMA 开始数据搬运工作，实现两者的无缝配合；

利用 DMA 进行触发 FSITX 进行发送数据，FSIRX 收到数据后用 DMA 将数据移出缓冲器，需注意在例程的宏定义中采用#define FSI_DMA_trigger 1 和 #define TX_DMA_enable 0 的配置方式，其他的配置方式可参考应用文档 Using the Fast Serial Interface (FSI) With Multiple Devices in an Application。

对于 FSITX，需将寄存器 TX_OPER_CTRL_LO.START_配置为 0x2，或采用 Driverlib 方式配置如下，即意味着一旦 DMA 写入数据帧中的 user data 和 tag 部分即触发 FSITX 进行发送数据。

```
FSI_setTxStartMode(FSITXA_BASE, FSI_TX_START_FRAME_CTRL_OR_UDATA_TAG);
```

在此基础上，需配置两个 DMA 通道分别对传输的数据和帧中的 tag/user data 部分进行处理，而后者的对应的通道应仅次于前者，这样才能保证数据部分写入完成后才会进行发送。例如，采用 DMA 通道 1 将数据写入 TX 的数据缓冲寄存器，采用 DMA 通道 2 将 tag 和 user data 内容写入数据帧的对应位置。值得注意的是，由于 FSI 的数据缓冲寄存器是一个 16 个字的环形缓冲寄存器，因此当传输的数据量超过 16 个字，DMA 在写入缓冲寄存器时需要配置 wrap 机制，如下

```
DMA_configWrap(DMA_CH1_BASE, DMA_TRANSFER_SIZE_IN_BURSTS, 0, dest_WrapSize, 0);
```

此处的 dest_WrapSize 对应 16/nWords（nWords 为 DMA 单次 burst 发送的字数）。

FSIRX 模块的配置方法类似，采用 DMA 通道 3 和通道 4 分别将接收到的数据和帧中的 tag/user data 移动到目标存储区域，而区别于 FSITX，此处对 DMA 的通道选择次序没有要求。以下的配置使能 FSIRX 完成数据接收后产生触发 DMA 工作的信号。

```
FSI_enableRxDMAEvent(FSIRXA_BASE);
```

2.4 测试结果

测试结果基于 3 个芯片构成的菊花链多机通信系统，采用 3 个评估板 F280049C ControlCARD Evaluation Modules 构成，测试平台见图 4。

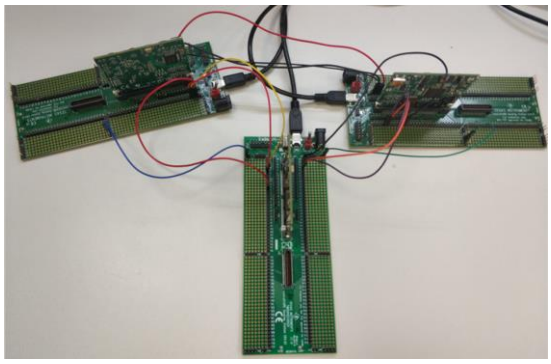

图 4 3 芯片通信菊花链拓扑测试平台

测试条件如下：

1) 数据长度为 4 个字（DMA: 4 words/burst, 1 burst/transfer），采用单根数据线，TXCLK = 50 MHz；
2) 芯片 1 将数据发送给芯片 2， 芯片 2 接收到数据并读取数据，接着将数据写入 FSITX 数据缓冲器，并发送至芯片 3，芯片 3 再经过同样的过程将数据发送给芯片 1；

测试代码中用 GPIO 的高低电平信号标记每个芯片的收发数据时刻，图 5 给出了利用 DMA 触发 FSI 在 3 芯片菊花链通信中的测试结果。结果表明，从芯片 1 发送数据经菊花链回送数据总时长为 5.42 µs。图 6 测得在菊花链回路中，每增加一个芯片，所需增加的时间为 2.10 µs，该时长包括了数据传输时间、读取数据时间以及写入发送数据时间等。

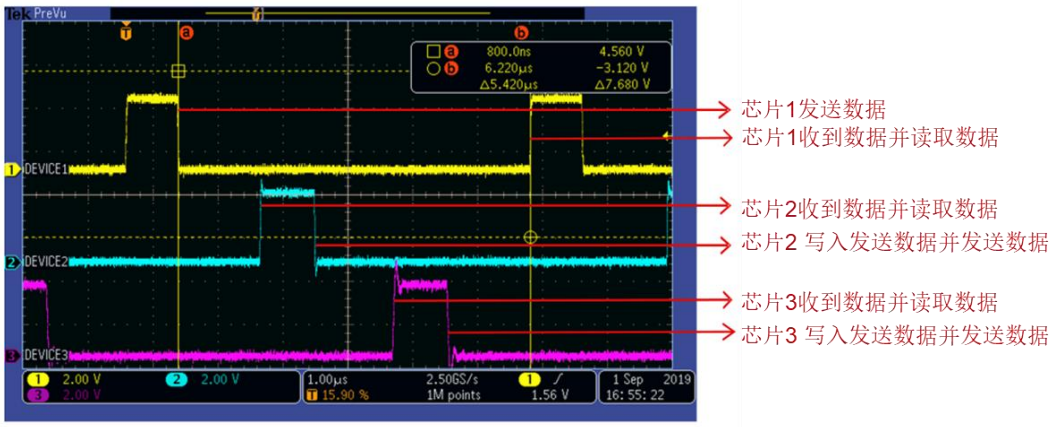

图 5 利用 DMA 触发 FSI 在 3 个芯片菊花链通信中的测试结果

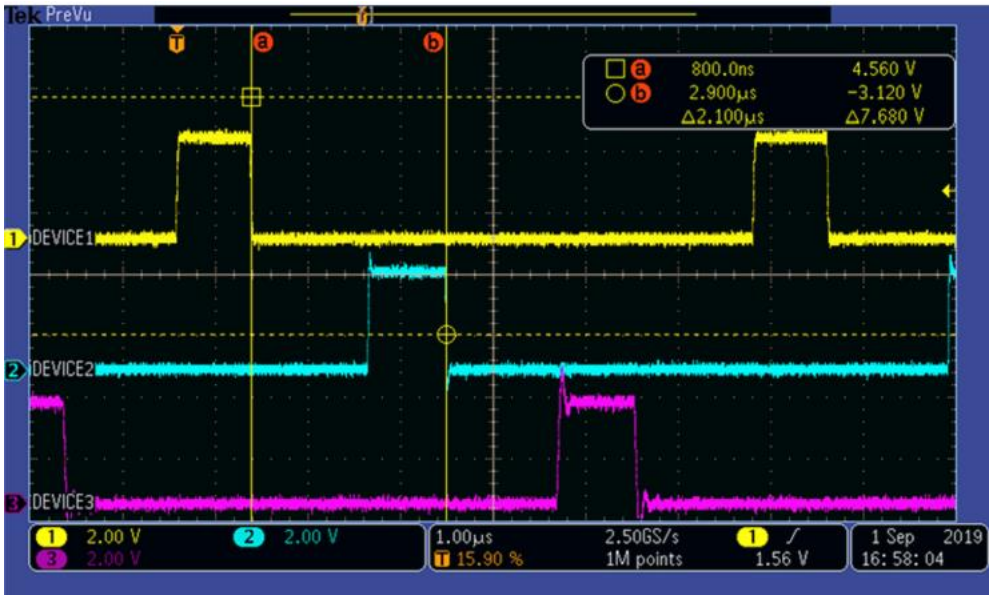

图 6 在单个芯片中的处理时间

采用菊花链多机通信拓扑的 FSI 应用实例也可以参考 TIDM-02006，该参考设计展示了利用 FSI 实现 1 主机/4 从机的多轴伺服控制应用场景。

3 FSI 在星型多机通信拓扑中的应用

3.1 星型拓扑框图

针对如 F2838x 等具有多个 FSIRX 端口的芯片，除了可应用于上述的菊花链拓扑，也可用于实现一对多的星型多机通信拓扑，如图 7 所示，作为 Master 主机的 FSITX 和每个 slave 从机的 FSIRX 直接相连，而每个从机的 FSITX 分别与主机不同的 FSIRX 连接。若以 F2838x 为主机，则最多可实现 1 对 8 的星型通信拓扑，而从机可以选用 F2838x 或 F28004x。

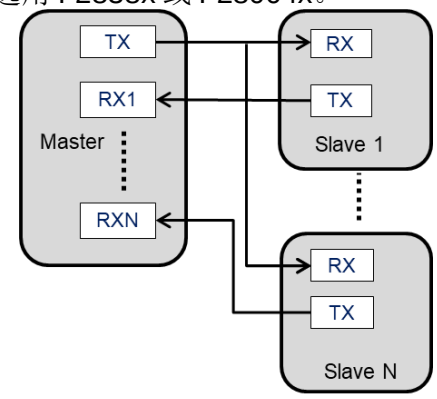

图 7 FSI 用于星型通信拓扑

3.2 “握手”机制

星型拓扑的通信握手过程相对菊花链式较为简化，主要包含以下步骤：

1) 所有芯片设置 FSITX/FSIRX 的数据帧类型为 Ping，并使能 Ping Frame Received 接收中断；
2) 开始第一轮的 ping 回路：
   a. 主机发送 Flush 信号唤醒所有从机，然后再发送 Tag0(0000)，进入等待状态；如果主机接收到所有来自从机的 Tag0，则进入第二轮 ping 回路；否则，重复此步骤；
   b. 所有从机在起始时进入等待状态，若接收到有效的 ping 数据帧为 Tag0，则发送 Flush 信号和 Tag0 给主机，进入第二轮 ping 回路；否则，重复此步骤；
3) 开始第二轮的 ping 回路：
   a. 主机发送 Tag1 (0001)，进入等待状态；如果主机接收到所有来自从机的 Tag1，则继续下一步骤（握手完成）；否则，重复此步骤；
   b. 从机进入等待状态，若收到 Tag1 则发送 Tag1 给主机；否则，重复此步骤；
4) 握手成功。

实际上，通过第一轮的 ping 回路已经能确保主机和从机握手成功，但主机还需通过第二轮的 ping 回路通知从机握手已成功，双方退出握手等待状态，进入数据传输状态。

3.3 数据传输配置

针对星型多机通信的例程，可参考 C2000ware 中的 “fsi_ex9_star_broadcast”，文件路径：

C:\ti\c2000\C2000Ware_<version_number>\driverlib\f2838x\examples\c28x\fsi

该例程提供了 1 对 3 的多机通信示例，采用 CPU 进行软件干预数据的收发控制，而本文主要说明针对星型拓扑采用 DMA 触发数据传输的配置方法。基础配置可参考 2.3 节，主要区别在于主机对于不同的从机在发送数据和接收数据时可进行区分处理，也更符合实际的应用场景。由于所有的从机会同时接收到主机发送的数据，因此主机可以指定不同的 user data 或 tag 内容，对应不同的从机，而从机根据该内容选择是否回应主机。

假设在主机中仍然利用 DMA 进行触发 FSITX 进行发送数据，FSIRX 收到数据后用 DMA 将数据移出缓冲器，以 1 对 2 的星型系统为例，参考配置示例如下。

对于 FSITXA，可选择 0x101 作为对应 slave1 的 user data 和 tag 内容，选择 0x202 作为对应 slave2 的 user data 和 tag 内容，在每次利用 DMA 通道 2 写入该内容前，需先对 DMA 通道 2 的数据源进行赋值，参考代码如下，即每次 DMA 完成接收数据的搬运后会产生中断，在中断里进行该处理。

```
interrupt void fsirxadma_isr(void)
{
   GPIO_writePin(RXA_DATA_FRAME_IO,1);
   // after receiving data from slave 1, rewrite the user data/tag for slave 2
   *((uint16_t *)Slave_userdata_tag_ADDR) = 0x202;

   DMA_forceTrigger(DMA_CH1_BASE);
```

```
    DMA_forceTrigger(DMA_CH2_BASE);

    Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP7);
}

interrupt void fsirxbdma_isr(void)
{
    GPIO_writePin(RXB_DATA_FRAME_IO,1);
// after receiving data from slave 2, rewrite the user data/tag for slave 1
    *((uint16_t *)Slave_userdata_tag_ADDR) = 0x101;

    DMA_forceTrigger(DMA_CH1_BASE);
    DMA_forceTrigger(DMA_CH2_BASE);

    Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP7);
}
```

对于 FSIRX，主机对应 slave1 和 slave2 的接收端口分别为 FSIRXA 和 FSIRXB, 分别采用 DMA 通道 3/4 和 DMA5/6 搬运接收缓冲器中的数据。2 个从机在接收到数据后，将对数据中的 Tag 进行判断，例如收到 Tag1，则 slave1 回送数据到主机；若收到 Tag2，则 slave2 回送数据到主机，参考配置如下：

```
//Slave1
interrupt void fsirxdma_isr(void)
{
    if(FSI_getRxFrameTag(FSIRXA_BASE)==FSI_FRAME_TAG1)
    {
        GPIO_writePin(RX_DATA_FRAME_IO,1);
        DMA_forceTrigger(DMA_CH1_BASE);
        DMA_forceTrigger(DMA_CH2_BASE);
    }
    Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP7);
}

//Slave2
interrupt void fsirxdma_isr(void)
{
    if(FSI_getRxFrameTag(FSIRXA_BASE)==FSI_FRAME_TAG2)
    {
        GPIO_writePin(RX_DATA_FRAME_IO,1);
        DMA_forceTrigger(DMA_CH1_BASE);
        DMA_forceTrigger(DMA_CH2_BASE);
    }
    Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP7);
}
```

3.4 测试结果

本测试采用 1 对 2 的星型通信拓扑，基于 1 个 F28388D ControlCARD Evaluation Module 为主机和 2 个 F280049C ControlCARD Evaluation Modules 为从机进行通信平台搭建，测试条件如下：

1) 数据长度为 4 个字（DMA: 4 words/burst, 1 burst/transfer），采用单根数据线，TXCLK = 50 MHz;
2) F28388 轮流发送不同数据给 F280049-1 和 F280049-2， 若 F280049-1 识别到该数据带有 Tag1，则回送数据给 F28388;若 F280049-2 识别到该数据带有 Tag2，则回送数据给 F28388;

测试代码中用 GPIO 的高低电平信号标记每个芯片的收发数据时刻，IO 口置高为数据接收完毕的时刻，IO 口置低为数据发送完毕的时刻，采用 Kingst Logic Analyzer 测得的结果如图 8 所示。

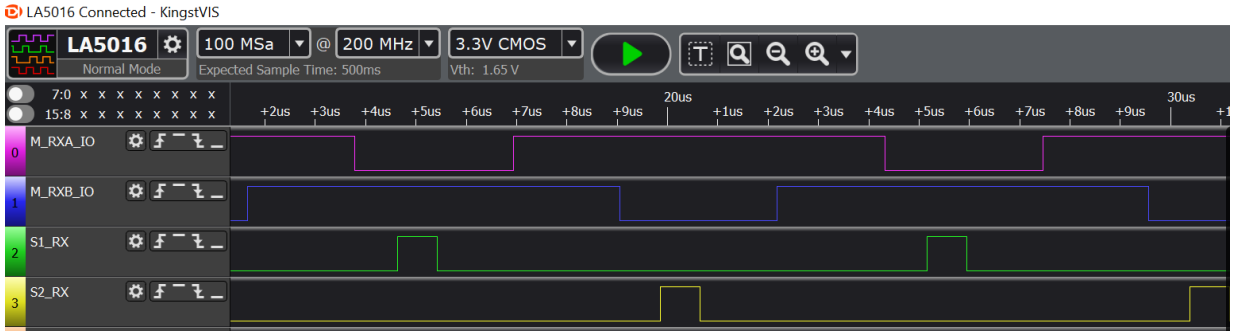

图 8 1 对 2 星型通信测试结果

3.5 其他

3.5.1 星型多机通信拓扑

本节讨论的基于星型拓扑的多机通信应用目前仅限于采用 F2838x 作为主机的场景，且可支持的最多从机数为 8 个，在 TRM 手册中的 Multi-Slave Configurations 部分，也提出了图 9 所示的类星型的多机通信拓扑结构，主机的发送端口与所有从机的接收端口连接，所有从机再经过链式相连。该拓扑需利用 FSI 的 tag matching 特性和 CLB 可编程逻辑模块，每个从机可根据接收到的数据帧的 tag 内容，选择发送自身的数据，或者进入旁路模式，将前一个从机的数据直接发送给下一个从机，具体实现细节有待进一步更新。

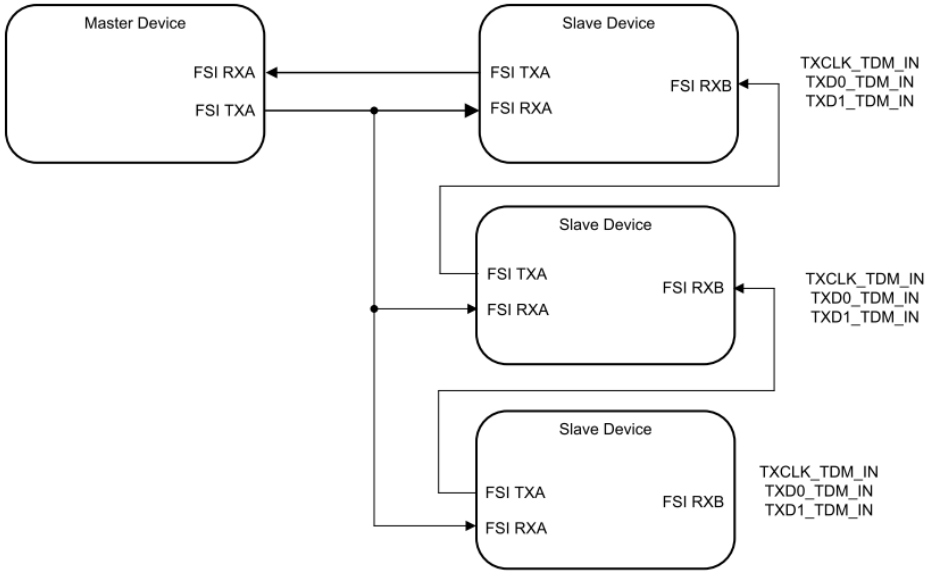

图 9 类星型多机通信拓扑

3.5.2 多机调试技巧

在多机通信系统的评估开发过程中，若实现在同一台 PC 上对多个 C2000 的通信状态同时进行在线调试，将大大提升开发效率。如果不同的 C2000 芯片搭配不同的烧录器，这是直接可行的；若不同 C2000 搭配的收录器相同，则可以参考以下的操作方法，此处以 2 个 F280049C 评估板 TMDSCNCD280049C(皆采用 XDS100V2 为仿真烧录器)为例进行说明。

1) 针对两个需进行调试的工程分别建立不同的 workspace，然后开启两个 CCS 窗口，导入工程；

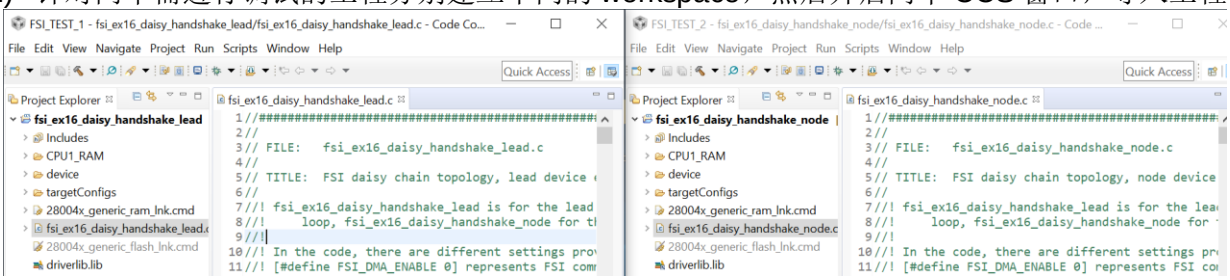

图 10 两个 CCS 窗口及工程导入

2) 由于两个 F280049C 评估板都是采用 XDS100V2 仿真器，需要进一步获取每个仿真器独立的序列号，并在 target configuration 配置文件.ccxml 中进行区分配置。
首先，对于 XDS100 系列的仿真器，在连接 PC 后，利用 CMD 窗口运行 xds100serial.exe 程序，该程序位于 CCS 安装文件路径(以 CCS8.3 为例) C:\ti\CCS8.3\ccsv8\ccs_base\common\uscif，即可获取对应该仿真器的序列号，如图 11。然后在对应的工程配置文件.ccxml 中进入 “Connection Properties”，修改 “Debug Probe Selection” 为 “Select by serial number”，即可输入仿真器对应的序列号，如图 12。

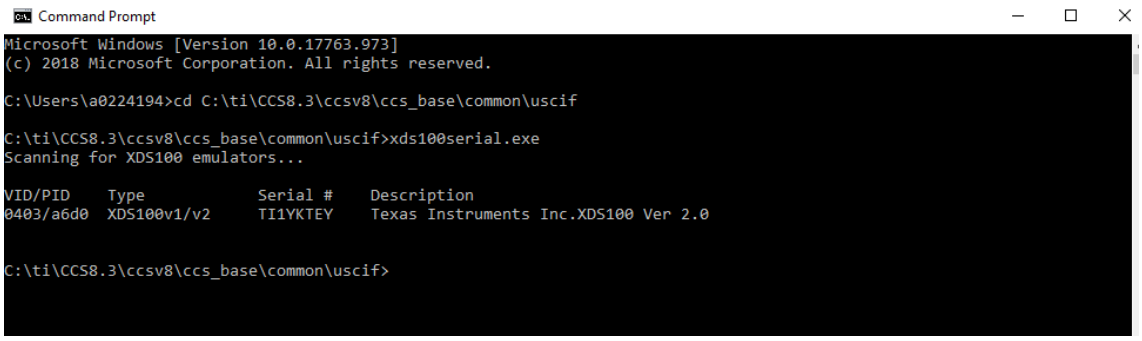

图 11 获取 XDS100 仿真器序列号

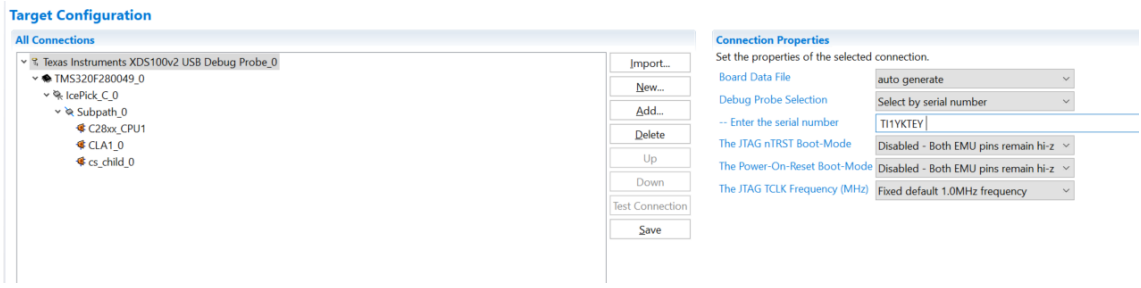

图 12 配置仿真器连接属性中的序列号

3) 在完成以上步骤后，利用两个 CCS 窗口可以同步对两个工程进行调试，如图 13。

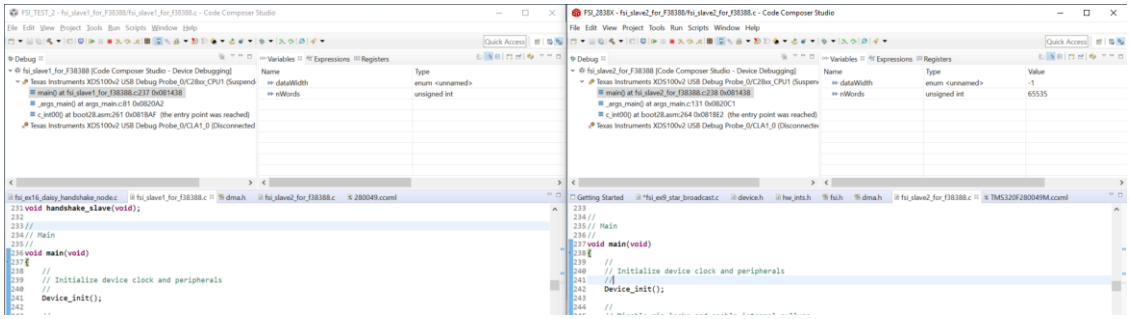

图 13 同步调试两个工程

4) 在某些情况下，仿真器在出厂设置时没有烧写对应序列号，或者连接的仿真器的序列号相同，则需要重新烧写该序列号，具体方法可参考以下链接。

https://processors.wiki.ti.com/index.php/XDS100#Q:_Can_I_change_the_serial_number_on_my_XDS100v2.3F

以上的方法基于 XDS100 系列仿真器，其他型号的仿真器对应的操作可参考

http://software-dl.ti.com/ccs/esd/documents/sdto_ccs_multi-probe-debug.html

4 参考文献

1. TMS320F28004x Piccolo™ Microcontrollers Datasheet
2. TMS320F28004x Piccolo Microcontrollers Technical Reference Manual
3. TMS320F2838x Microcontrollers With Connectivity Manager datasheet
4. TMS320F2838x Microcontrollers TRM (Rev. A)
5. Using the Fast Serial Interface (FSI) With Multiple Devices in an Application
6. TIDM-02006: Distributed multi-axis servo drive over fast serial interface (FSI) reference design

重要声明和免责声明

TI 均以“原样”提供技术性及可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证其中不含任何瑕疵，且不做任何明示或暗示的担保，包括但不限于对适销性、适合某特定用途或不侵犯任何第三方知识产权的暗示担保。

所述资源可供专业开发人员应用TI 产品进行设计使用。您将对以下行为独自承担全部责任：(1) 针对您的应用选择合适的TI 产品；(2) 设计、验证并测试您的应用；(3) 确保您的应用满足相应标准以及任何其他安全、安保或其他要求。所述资源如有变更，恕不另行通知。TI 对您使用所述资源的授权仅限于开发资源所涉及TI 产品的相关应用。除此之外不得复制或展示所述资源，也不提供其它TI或任何第三方的知识产权授权许可。如因使用所述资源而产生任何索赔、赔偿、成本、损失及债务等，TI对此概不负责，并且您须赔偿由此对TI 及其代表造成的损害。

TI 所提供产品均受TI 的销售条款 (http://www.ti.com.cn/zh-cn/legal/termsofsale.html) 以及ti.com.cn上或随附TI产品提供的其他可适用条款的约束。TI提供所述资源并不扩展或以其他方式更改TI 针对TI 产品所发布的可适用的担保范围或担保免责声明。

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