Application Note

F28P65+DP83826 EtherCAT 帧错误排查方法及应用建议

Yunjing Wang, Zane Wei *North West China FAE team*

摘要

F28P65 是 TI 新一代集成倍福 EtherCAT IP，针对工业以太网应用的低成本高性能 C2000 芯片，搭配 TI 新一代低延迟百兆以太网 PHY DP83826，为 TI 实现最新一代高性价比 EtherCAT 的明星方案，广泛应用于工业实时以太网通信场景。在实际的工业以太网通信过程中，通常会遇到各种各样的通信问题，比如 EtherCAT 主站扫描不到从设备、通信过程中出现帧错误，甚至出现严重的物理层 Link down 问题。对于初次接触 EtherCAT 通信的新手工程师，无法从系统层面逐一排查和定位问题，因此，本应用手册从 EtherCAT 通信的 OSI 模型切入，介绍 EtherCAT 官方提供的一些故障排查硬件机制和排查手段，并且针对 EtherCAT 应用常见的帧错误问题提供排查定位和应对方案。

内容

修改记录 1
1 EtherCAT 概述及通信模型介绍 2
2 EtherCAT 从站帧传输及诊断功能 4
2.1 EtherCAT 从站帧处理过程 4
2.2 ESC 诊断功能 4
3 EtherCAT 帧错误故障案例及排查方法 7
3.1 帧错误故障案例 7
3.2 帧错误故障排查案例——故障隔离分析 9
3.3 帧错误故障排查案例—— F28P65 ESC 问题排查 11
4 总结 16
5 参考文献 16

修改记录

Version	Date	Author	Notes
1.0	Dec 14th 2025	Yunjing Wang,Zane Wei	First version

1 EtherCAT 概述及通信模型介绍

EtherCAT 是 Ethernet control and Automation Technology 的缩写，全称为基于以太网的工业自动化技术。EtherCAT 实时工业以太网，最早由德国倍福 Beckhoff 公司于 2003 研发推出，2007 年成为国际标准，2014 年成为中国国家标准，因其超高速、高实时性、开放性的特点，成为当前广为应用的工业以太网技术之一。

图 1-1 为 EtherCAT 通信模型，分为物理层、数据链路层和应用层。物理层通常采用 RJ45 接口以及物理层 PHY 芯片，也有用 LVDS 形式提供电路级的协议传输，EtherCAT 物理层是 EtherCAT 通信模型中的最底层，它为网络信号的传输提供了物理接口，同时接收来自数据链路层的数据单元，并对它们进行相应的解码。数据链路层为物理层把数据帧转化为原始位，并负责将来自高层的数据打包成帧以及流量控制、纠错和帧的重发。数据链路层最基本的功能就是通过该层的协议使相邻两个节点之间进行可靠的数据传输。数据链路层由 ESC 实现，主要是计算、比较、生成帧校验序列，并通过从以太网帧中提取或插入数据来实现通信。EtherCAT 应用层协议支持多种协议如 VoE、FoE、EoE 和 CoE 等。

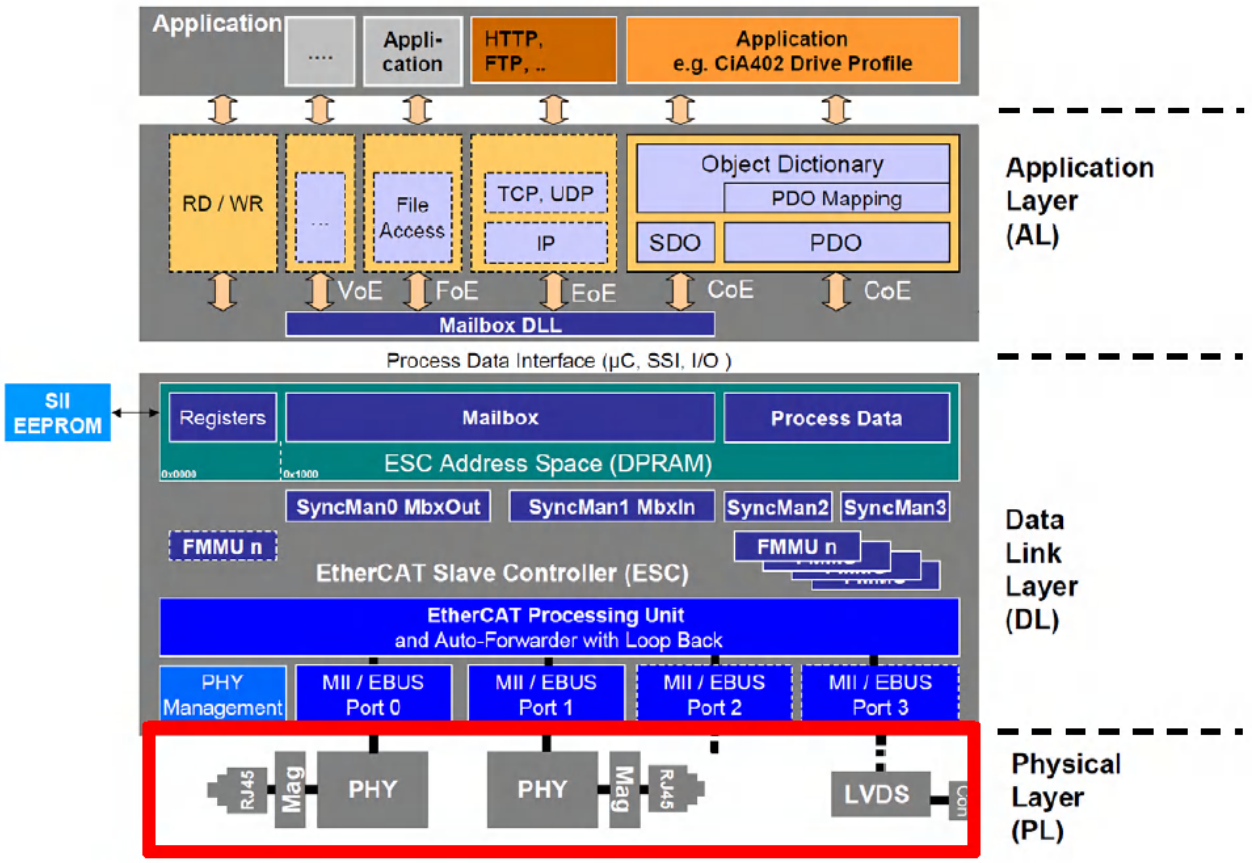

图 1-1. EtherCAT 通信模型

传统的以太网在通信的过程当中每个节点是以接收、处理、转发依次进行，而 EtherCAT 则是同时传输和处理 EtherCAT 数据。每个从站的节点都有 FMMU（现场总线内存管理单元），FMMU 的功能是对经过从站节点的数据包进行地址分析，将逻辑地址转化为物理地址，如果数据包中有当前从站所用的数据则读出，并同时转发报文至下一个从站节点，同样的，在报文经过时也可以插入数据。如 图 1-2 所示。

每个 EtherCAT 从站在接收到的报文中提取或者插入用户数据，然后将处理好的报文发送到下一个 EtherCAT 从站，报文在依次被所有从站处理之后由最后一个 EtherCAT 从站回传，并作为响应报文返回给 EtherCAT 主站控制单元。整个过程基于以太网的全双工模式，通过 TX 线发送出去的报文会从 RX 线返回，因此在 EtherCAT 通信系统中，任何物理拓扑结构在逻辑上永远是环形。

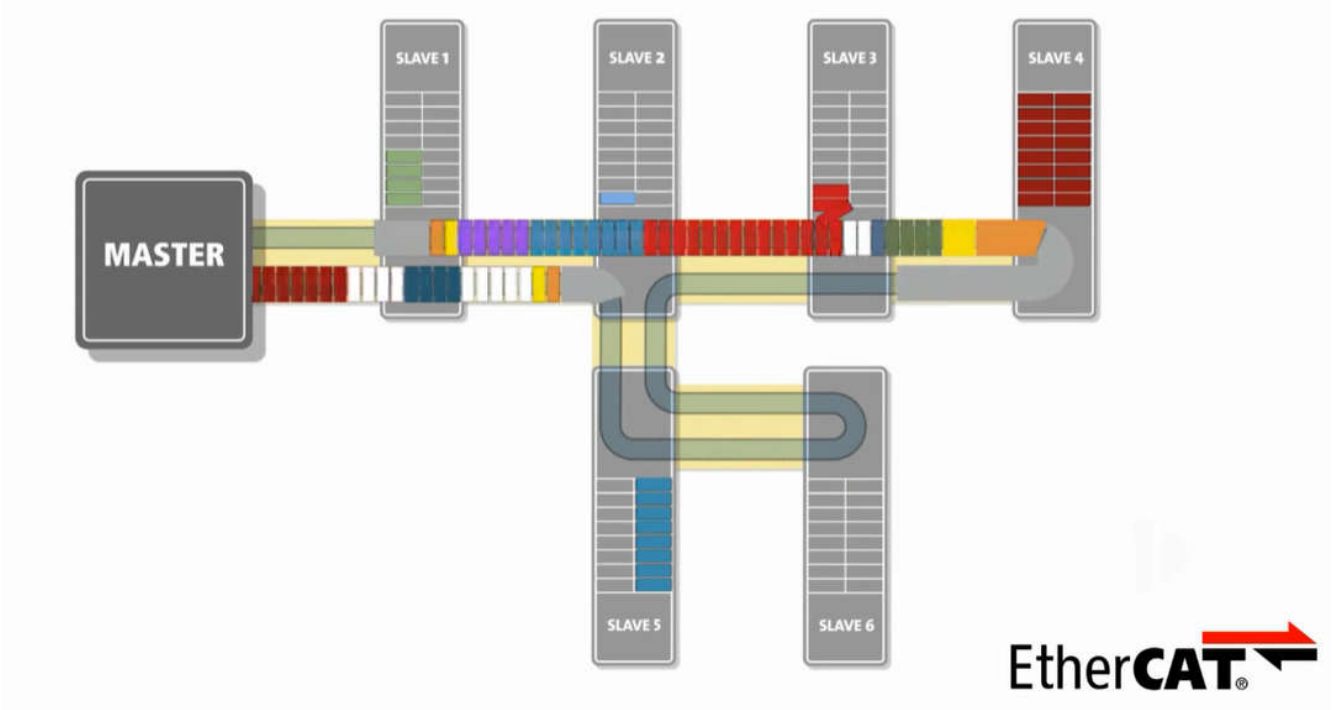

图 1-2. EtherCAT 通信过程图

2 EtherCAT 从站帧传输及诊断功能

2.1 EtherCAT 从站帧处理过程

如 图 2-1 所示，为 EtherCAT 官方提供的帧处理流程，对于 F28P65 来讲，只支持两个 Port，所以 Port2 和 Port3 为关闭状态。帧会从 Port0 进入经 Auto-Forwarder 后进入 EtherCAT 处理单元，再到 Port1，直到最后一个 ESC 设备，然后从 Port1 回环到主站。

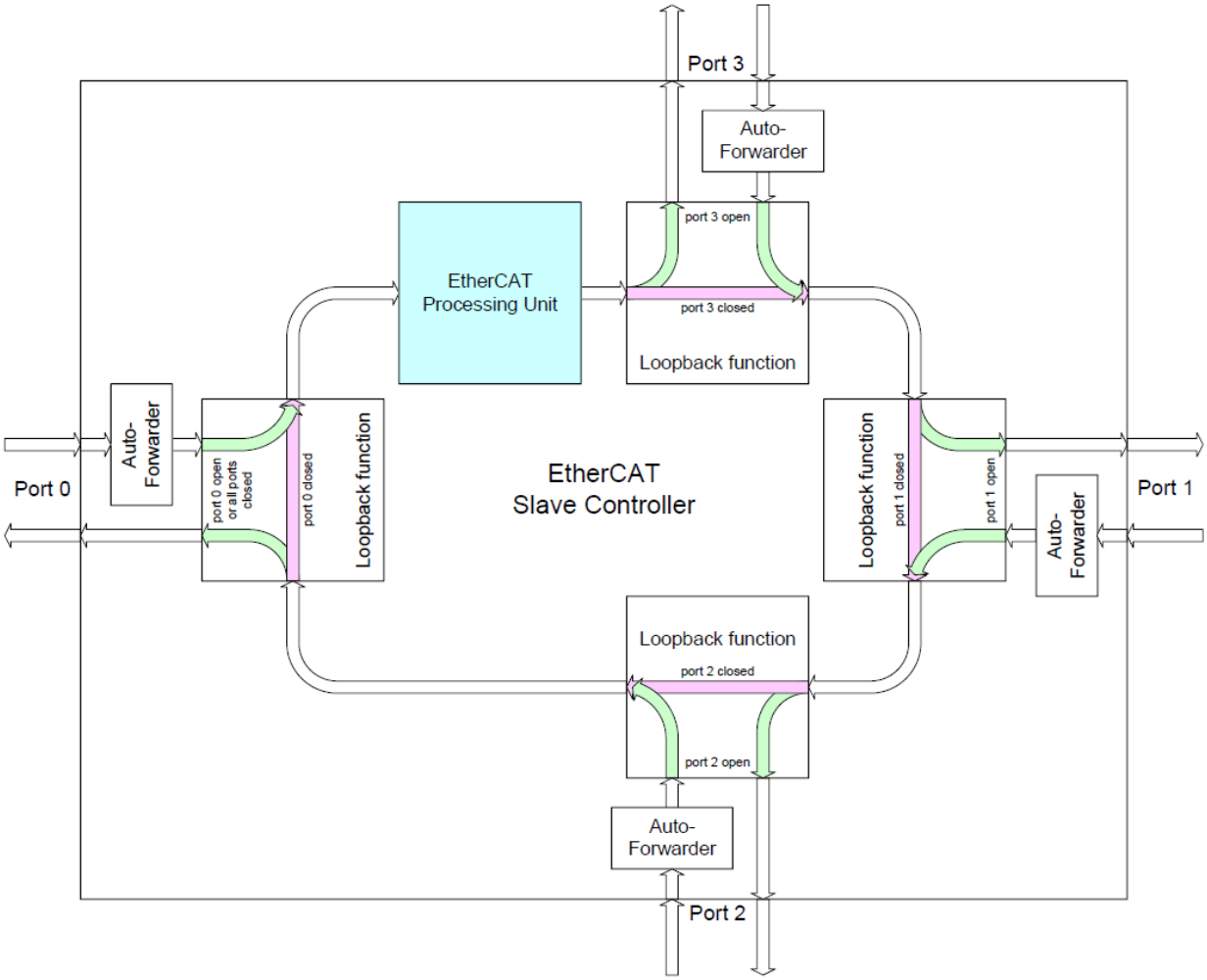

图 2-1. EtherCAT 帧传输流程

2.2 ESC 诊断功能

每个 EtherCAT 报文的结尾都会有一个 16bit 的工作计数器(WKC)，表示相应命令所请求的内存访问成功的次数。如 图 2-2 所示，当报文从 EtherCAT 主站发出时，WKC 计数为 0，之后每个从站(ESC)按照相应规则访问指令请求的内存后，WKC 计数会相应增加。主设备通过对比收到的 WKC 数值与预期值之间的差异，可以判断 EtherCAT 数据包的有效处理情况。如果主站收到的 WKC 数值与预期值不对应，则表示该报文寻址的一个或者多个 ESC 存在访问失败的情况，ESC 物理层连接故障、ESC OP 状态异常、帧错误等情况都可能导致 WKC 与预期值不匹配的情况，用户可以根据 ESC 支持的多种错误计数器来进一步定位故障原因和故障位置。

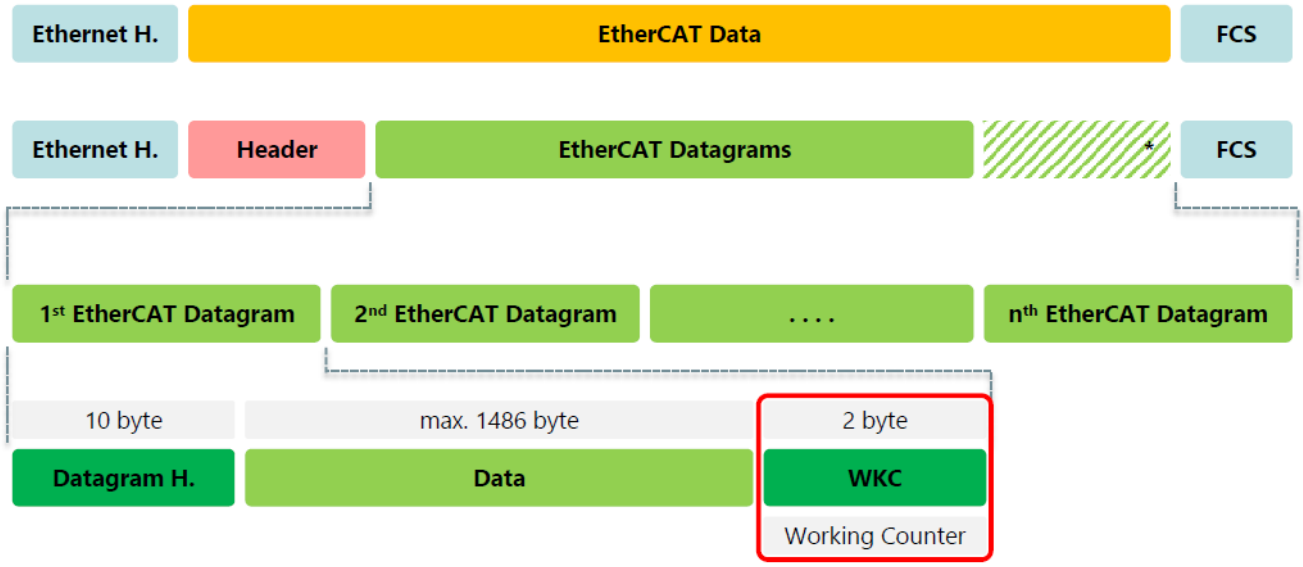

图 2-2. EtherCAT 通信帧结构及 WKC 计数器

EtherCAT 环路的通信问题首先要确认物理层连接状态是否异常，用户可以通过 ESC 0x0110 寄存器来确认每个 Port 的 link 状态，因篇幅因素，本文档主要针对 Frame error 故障的分析定位，EtherCAT 物理层 link 状态异常的故障分析方法本系列后续会有相关文档提供案例分析。

即使 EtherCAT 设备之间的 Link 没有任何问题，EtherCAT 帧也可能在传输过程中发生损坏，例如 EMC 干扰，ESC 设备故障等。ESC 提供了多种帧错误相关的错误计数器，方便用户定位帧错误原因和故障位置。具体的错误计数器及其功能请参考 图 2-3。

Error Counter	Register	Description
Port Error Counters	0x0300:0x0307	Errors counted at the Auto-Forwarder (per port):
Invalid Frame Counter	0x0300/2/4/6	Invalid frame initially detected (includes RX Errors)
RX Error Counter	0x0301/3/5/7	Physical layer RX Errors (inside/outside frame): MII: RX_ER EBUS: Manchester violations
Forwarded RX Error Counter	0x0308:0x030B	Invalid frame with marking from previous ESC detected (per port)
ECAT Processing Unit Error Counter	0x030C	Invalid frame passing the EtherCAT Processing Unit (additional checks by processing unit)
PDI Error Counter	0x030D	Physical Errors detected by the PDI. Refer to PDI description in Section III for details.
Lost Link Counter	0x0310:0x0313	Link lost events (per port), counts only if port is in Auto or Auto close mode
Watchdog Counter Process Data	0x0442	Watchdog timeout events
Watchdog Counter PDI	0x0443	Watchdog timeout events

图 2-3. ESC 错误计数器

如下几个错误寄存器仅在 Port Open 时才会进行计数，因此故障分析时需要注意 ESC 的 Port 状态：

RX Error 计数器：RX_ERR 是指物理层芯片能够检测到任何的错误，如 4B/5B 编码中的符号错误，或者硬件 MLT-3 信号的电压级别错误。这些错误会通过 PHY 在 MII 接口上的 RX_ER 引脚报告给 ESC，进而计入 ESC RX Error 寄存器。

Invalid Frame 计数器：每个端口在接收到帧时都会自动执行循环冗余校验操作，不论是物理层错误还是帧内错误也被视为帧错误。

Forward Error 计数器：当 ESC 首次检测到帧错误时，该 ESC 将此数据帧发送给后续 ESC 时将会以错误的 CRC 序列以及帧末尾的一个额外终止字节形式来标记。这一情况会被后续 ESC 检测到，并且计入 Forward Error 错误计数器。

3 EtherCAT 帧错误故障案例及排查方法

帧错误是 EtherCAT 应用中的一种常见错误，它会导致通信中断、设备掉线、设备不稳定等严重后果。通常来说，EtherCAT 帧错误是指主站检测到从站返回的报文不符合预期或者通信过程中出现了数据损坏或者丢失等问题,甚至可能会因为某一台设备故障导致整个通信链路通信中断，此时主站将不能扫描到任何一台从站设备。帧错误故障可能是物理层数据错误、MDI 时钟偏移、ESC 故障等等，故障定位需要通过物理层和 ESC 故障隔离、ECAT 主站状态机、错误计数器等诊断信息来排查。

本章节以一个帧错误故障实例为分析案例，介绍帧错误的判断方式、故障位置隔离方法、以及 F28P65 ESC 对应的应用建议。

3.1 帧错误故障案例

稳定复现的通信故障通常能够快速定位，但是对于随机复现的故障，因其复现难、故障 ESC 位置随机，因此更难定位。本故障案例为多次连续上下电(POR)测试时，上电后初始化后会偶发性出现主站扫描不到从站的情况，导致 EtherCAT 环路无法正常通信，且前期已确认物理层 link 状态良好。7 台 ESC 从站多次连续上下电(POR)测试，问题发生时，主站扫描不到任何一个从站设备。

只要将 ESC5 从系统中拿除，主站能扫描到 ESC1~4，ESC6，ESC7，初步定位故障 ESC 设备为 ESC5。对 ESC5 执行 F28P65 芯片复位，ESC5 能从故障中恢复，但仅执行内部 ESC 复位，并不能让 ESC5 从故障中恢复。抓取问题复现时的每一台 ESC 错误寄存器如 故障状态下 ESC 错误寄存器相关信息 所示，通过对每台设备的 ESC 错误寄存器进行分析，可以确定 ESC5 在传输过程中首先出现了 Port 故障计数。

表 3-1. 故障状态下 ESC 错误寄存器相关信息

Register	ESC1	ESC2	ESC3	ESC4	ESC5	ESC6	ESC7
0x300 PORT0 bit[15:8]-RX ERR bit[7:0]-Invalid frame	0	0	0	0	00FF	00FF	00FF
0x302 PORT1 bit[15:8]-RX ERR bit[7:0]-Invalid frame	00FF	00FF	00FF	00FF	00FF	00FF	0
0x308 Forward Error Port 0-bit[7:0], Port1-bit[15:8]	FF00	FF00	FF00	FF00	FF00	FFFF	00FF
0x30C EPU Err	0	0	0	0	00FF	00FF	00FF
0x30D PDI ERR	0	0	0	0	00FF	00FF	00FF

根据 F28P65 ESC 的内部结构和环路数据传输过程，分析整个环路的错误累加和传播过程如下。

如 图 3-1 所示，ESC5 设备为故障源。ESC5 Port0 首先收到无效帧，0x300 Invalid Frame, 0x308[7:0] Forward Error 和 0x30C EPU 错误累加并传播到 ESC6 和 ESC7，当环路末端在 ESC7 Port1 闭合并且数据帧开始回传时，除了 ESC7 Port1 之外(ESC7 Port1 处于环路末端为关闭状态，因此其 Port1 0x302 Invalid frame 不累计错误计数)，其后所有 ESC 包括 ESC6~ESC1 的所有 Port1 端口 Invalid frame(0x302), Forward Error(0x308[15:8])都开始累加计数并传播。

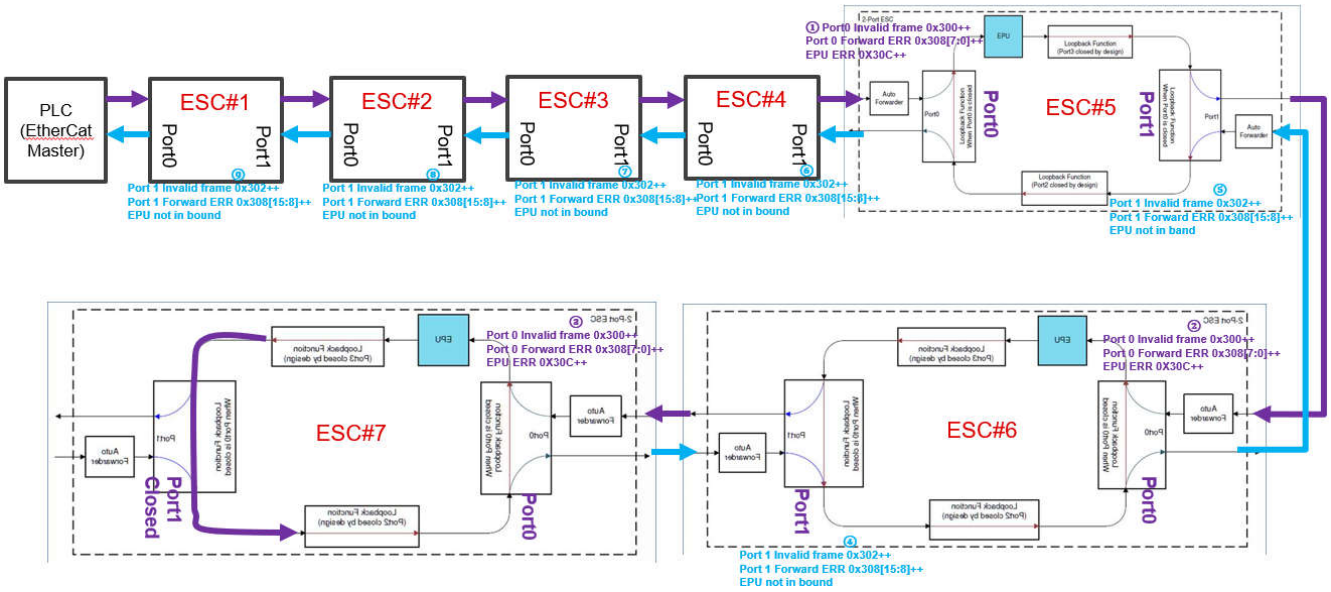

图 3-1. 帧错误故障案例-错误累加和传播过程分析

ESC5 的故障计数包括主站 Lost frame, 从站 CRC Error，Forward Error, EPU Error, 参考 图 3-2 所示的基于 ESC 错误寄存器的故障定位指南，本案例对应的故障寄存器信息指向的故障原因存在多种可能，包括物理层 PHY 故障、MII 接口时序故障、F28P65 内部时序故障等。为了快速定位故障位置，3.2 节将介绍如何利用 PHY 的 loopback 模式进行故障隔离，锁定故障位置。

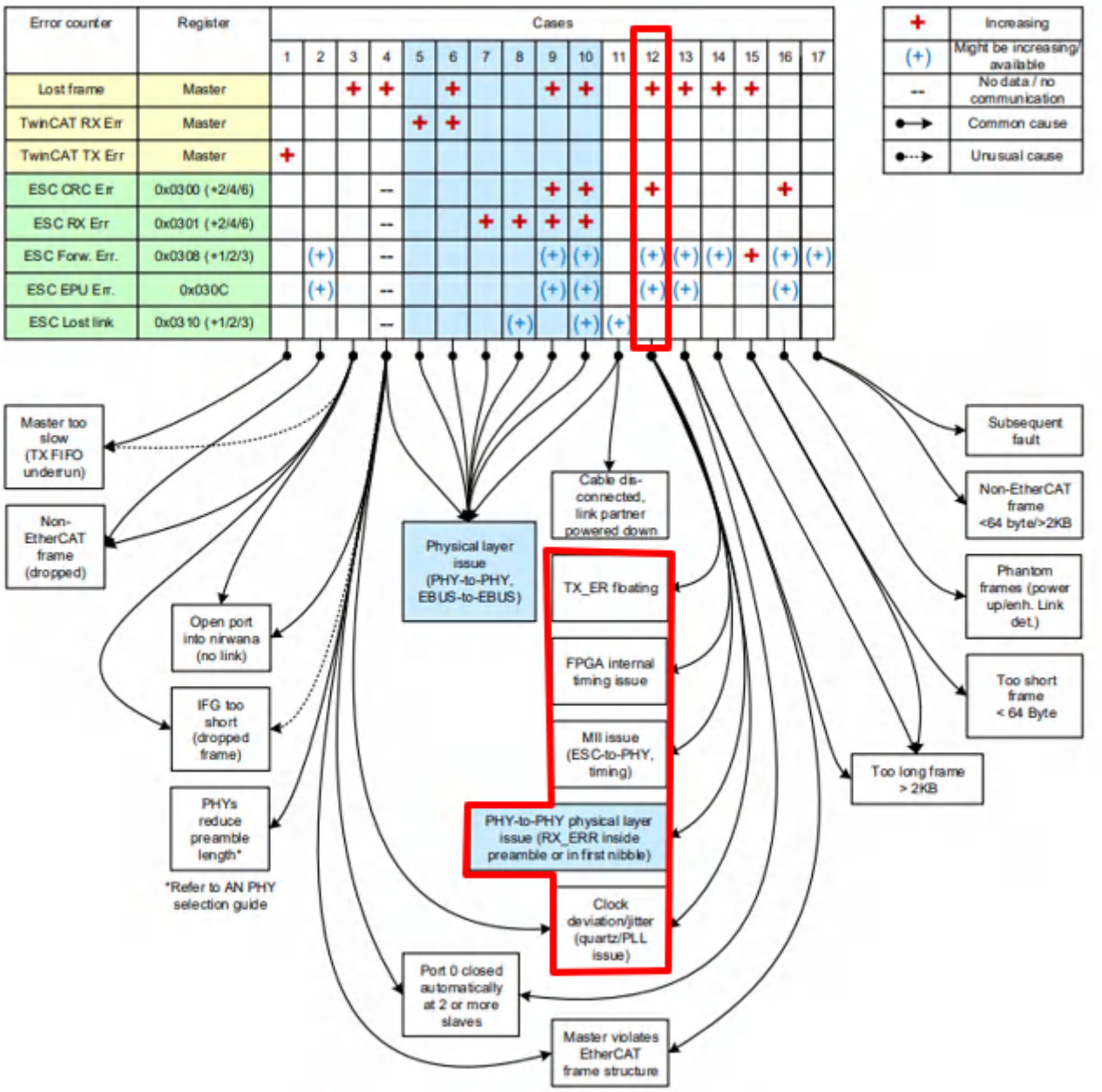

Figure 2: Error counters interpretation guide

图 3-2. 基于 ESC 错误寄存器的故障定位指南

3.2 帧错误故障排查案例——故障隔离分析

利用 PHY 芯片的 Loopback 模式可以有效进行 EtherCAT 故障位置的隔离，在 PHY、MII、F28P65 之间锁定故障位置，帮助下一步分析。复现故障时，可以根据以下 3.2.1~3.2.2 的方式配置环路，以分别验证故障设备 ESC5 上的 PHY0/PHY1, MII 和 F28P65 是否为故障源。

3.2.1 ESC PHY0/PHY1 故障隔离分析

如 图 3-3 所示，复现故障时，使用 PHY Reverse loopback 可以分别验证 ESC5 PHY0, PHY1 的工作状态是否正常。

将主站接在最左侧 ESC1 ，使用如下 DP83826 寄存器配置，将故障设备 ESC5 的 PHY0 配置为 Reverse loopback 模式，数据将在 ESC5 PHY0 的 MII 端口处环回，可以验证 ESC5 PHY0 是否正常工作。

DP83826 PHY Reverse loopback 配置方法如下：

```
0000 2100 //Disables Auto-Neg, Selects 100 Mbps
0016 0010 //Select Reverse Loopback
001F 4000 //Soft Reset
```

故障案例针对 ESC5 PHY0 做 Reverse loopback 实测结果表明，该配置下主站能够正常扫描到 ESC1~ESC4，并且 ESC4 0x0110 寄存器显示 Port1 为 Open 状态，说明 ESC4 PHY1 与 ESC5 PHY0 之间 link 状态良好，数据经过 ESC5 PHY0 之后由 MII 接口成功回环，ESC5 PHY0 并无故障。

将主站连接位置改为最右侧 ESC7，环路流向改为从右向左，将故障设备 ESC5 的 PHY1 做同样的 Reverse loopback 配置，使用同样的方法和判据可以验证 ESC5 PHY1 也无故障。

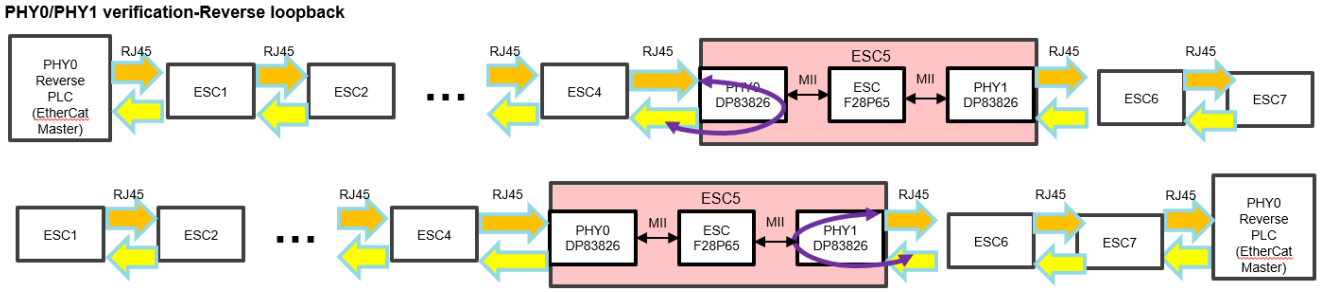

图 3-3. 故障设备 ESC5 PHY0/PHY1 故障隔离

3.2.2 ESC F28P65 以及 MII 故障隔离分析

在 3.2.1 排除 PHY 故障后，如 图 3-4 所示，使用 PHY MII Loopback 和 F28P65 Port Force close 对比测试，可以实验 ESC5 F28P65 内部 Port 和外部 MII 接口的故障隔离分析。

测试 1(紫色-PHYx MII loopback 路径)：若主站连接在最左侧的 ESC1，将故障设备 ESC5 的 PHY1 配置为 MII loopback 模式，数据从主站到达 ESC5 之后，将在 ESC5 PHY1 内部经 MII 接口环回，不经过 PHY1 内部除 MII 之外的其他功能块。

PHY MII loopback 配置方法如下：

```
0000 6100 //Disables Auto-Neg, Selects 100 Mbps, MII Loopback enabled.
0016 0104 //Select MII Loopback in BISCR
001F 4000 //Soft Reset
```

具体代码配置可以参考：

```
/* MII Management and PHY Addressing defines */
#define ESC_MII_CTRL_STATUS_1_OFFSET         0x0510
#define ESC_MII_CTRL_STATUS_2_OFFSET         0x0511
#define ESC_PHY_ADDRESS_OFFSET               0x0512
#define ESC_PHY_REG_ADDRESS_OFFSET           0x0513
#define ESC_PHY_DATA_OFFSET                  0x0514
#define ESC_MII_ECAT_ACCESS_OFFSET           0x0516
#define ESC_MII_PDI_ACCESS_OFFSET            0x0517

// Adding PHY configuration code here
HW_EscWriteByte(0x01,ESC_MII_PDI_ACCESS_OFFSET); // Give PDI access to MII management
HW_EscWriteByte(0x00,ESC_PHY_REG_ADDRESS_OFFSET); // Set PHY register to read/write
HW_EscWriteByte(0x01,ESC_MII_CTRL_STATUS_2_OFFSET); // Read PHY reg
HW_EscWriteWord(0x6100,ESC_PHY_DATA_OFFSET); // Set the value to write to register
HW_EscWriteByte(0x02,ESC_MII_CTRL_STATUS_2_OFFSET); // Write PHY reg

HW_EscWriteByte(0x16,ESC_PHY_REG_ADDRESS_OFFSET); // Set PHY register to read/write
HW_EscWriteByte(0x01,ESC_MII_CTRL_STATUS_2_OFFSET); // Read PHY reg
HW_EscWriteWord(0x0104,ESC_PHY_DATA_OFFSET); // Set the value to write to register
HW_EscWriteByte(0x02,ESC_MII_CTRL_STATUS_2_OFFSET); // Write PHY reg

HW_EscWriteByte(0x1F,ESC_PHY_REG_ADDRESS_OFFSET); // Set PHY register to read/write
HW_EscWriteByte(0x01,ESC_MII_CTRL_STATUS_2_OFFSET); // Read PHY reg
```

```
HW_EscWriteWord(0x4000,ESC_PHY_DATA_OFFSET); // Set the value to write to register
HW_EscWriteByte(0x02,ESC_MII_CTRL_STATUS_2_OFFSET); // Write PHY reg
```

在此配置下，如果主站能够正常扫到从站 ESC1~ESC5，则表示 ESC5 F28P65 Port 0/1 和内外 MII 接口均无异常，ESC5 设备无异常，无需实行测试 2。反之，若主站无法正常扫到 ESC5，表示 ESC5 F28P65 内部 Port0/1 或者外部 MII 接口存在异常，下一步使用测试 2 对 MII 接口进行验证。

测试 2(蓝色-ESC 控制器 PortX 内部回环路径)：若主站连接在最左侧的 ESC1，将故障设备 ESC5 F28P65 的 Port1 force close(如果主站扫不到 ESC5，可以通过翻转 Port1 的 link 状态实现 Port1 强制 close)，这时数据由主站顺序经过 ESC1~ESC4，然后经由 ESC5 Port0 内部转发至 ESC5 Port1，最终在 ESC5 Port1 内部回环，不会由 ESC5 Port1 送出。

若测试 1 里主站无法扫到 ESC5，测试 2 配置下主站依然无法扫到 ESC5，则说明 ESC5 设备确为异常，并且故障源头为 ESC5 的 ESC 控制器 F28P65；若测试 1 里主站无法扫到 ESC5，但是测试 2 配置下主站可以扫到 ESC5，则说明 ESC5 Port1 对外的 MII 接口存在异常，比如外部 MII 走线不匹配导致 MII 接口经过 PCB 线路之后出现时序异常。

以上为主站连接在最左侧 ESC1 的情况，可以判断故障设备 ESC5 的 Port1 以及 Port1 对外 MII 接口是否异常。针对 Port0 以及 Port0 对外的 MII 接口是否异常，将主站连接在最右侧 ESC7 进行相应分析即可。

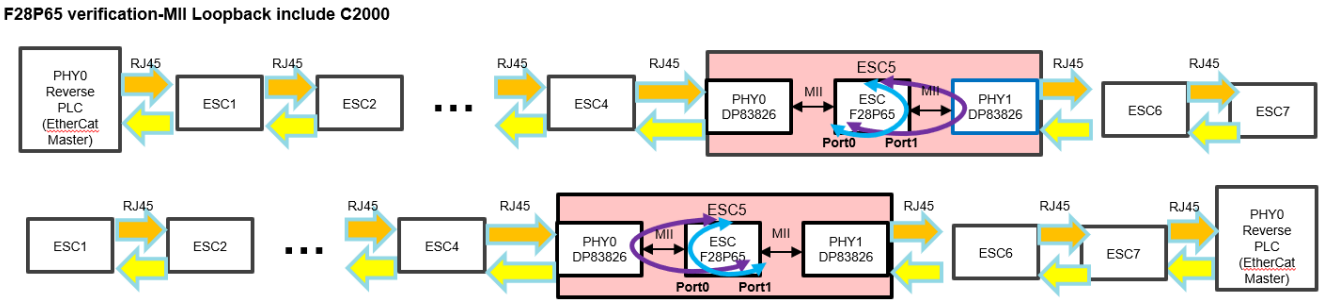

图 3-4. 故障设备 ESC5 MII 接口，F28P65 故障隔离

故障案例实测结果：将主站连接在最左侧 ESC1，在 ESC5 PHY1 做 MII loopback 测试(测试 1)，主站能够正常扫描到 ESC1~ESC4 却无法扫到 ESC5；在 ESC5 Port1 做 Port1 force close 测试(测试 2)，主站依然无法扫到 ESC5。将主站连接在最右侧 ESC7，在 ESC5 PHY0 做 MII loopback 测试(测试 1)，主站能够正常扫描到 ESC1~ESC4 却无法扫到 ESC5；在 ESC5 Port0 做 Port1 force close 测试(测试 2)，主站依然无法扫到 ESC5。测试结果表明，故障设备 ESC5 的故障源在 ESC 控制器 F28P65 内部。

3.2 节以实际帧错误故障案例的故障隔离分析为例，说明了 EtherCAT 帧错误故障在错误寄存器信息指向多种故障可能的情况下，如何进行故障位置隔离分析。下文 3.3 将根据本节案例分析定位的故障位置，进行 F28P65 内部排查分析和应用建议说明。

3.3 帧错误故障排查案例—— F28P65 ESC 问题排查

基于 3.2 的故障隔离分析结果，物理层 PHY、MII 接口故障已经排除，因此本节将对 ESC 内部时序进行排查。

值得注意的是，除了 3.1 节的错误累计和传播分析所涉及的 Invalid Frame, Forward Error 和 EPU Error 之外, 从 ESC5 开始到 ESC7 都显示有 PDI Error 计数，而 PDI 是 ESC 与 F28P65 CPU 交互的重要接口，ESC 官方手册中也提到，应在上电后确保 PDI 错误计数为 0。因此基于 3.2 故障隔离分析的结果，结合 PDI Error 信息，更加可以断定 ESC5 设备的 F28P65 芯片内部 SC IP 与 F28P65 CPU 交互出现了很严重的问题，本节将对此问题进行详细分析并给出解决方法。

ESC 为 F28P65 的外设功能模块，图 3-5 展示了 ESC 与 F28P65 的连接方式。16 位异步接口（PDI）和中断请求是 F28P65 CPU 和 ESC 交互的主要通道。中断请求可用于根据 ESC 内部事件、异常情况或者时间同步事件触发操作。同时，ESC 的初始化由 F28P65 CPU 控制，因此需要对初始化过程中的每一个环节进行排查，并分析是否可能造成 ESC 功能异常。

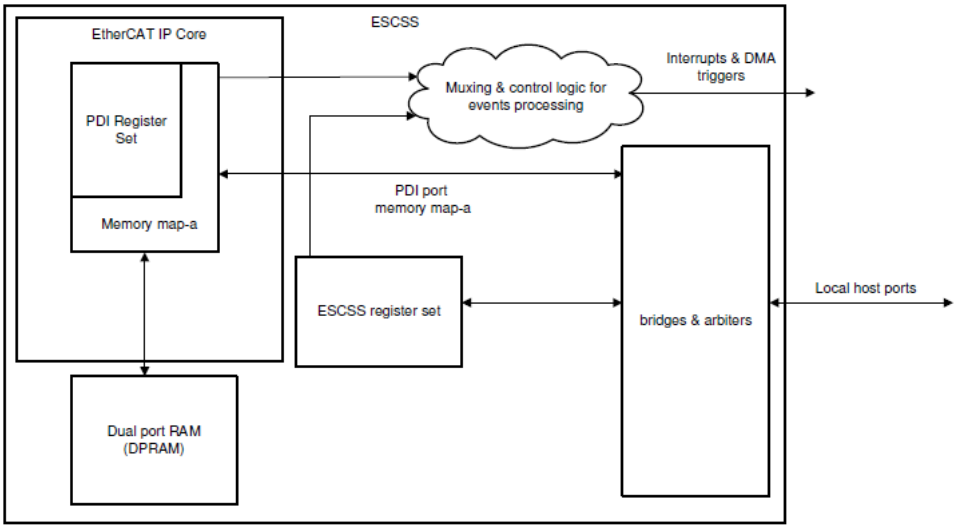

图 3-5. EtherCAT IP 与 F28P65 通信模型

图 3-6 为 F28P65 CPU1 初始化 ESC IP 的流程，通过对之前的案例进行分析，可以确定 ESC4 的 CPU1 正常工作，并且 ESC4 EtherCAT IP 也在工作，否则不会记录到错误计数。按照 ESC 初始化流程逐一进行分析，步骤 5 是一个需要重点排查的点，它会设置 EtherCAT 时钟源和分频系数，如果系统时钟配置出现问题，将会导致严重的通信故障，这个也和我们观察到的现象匹配。结合 F28P65 TRM 上系统控制寄存器配置的限制，系统控制寄存器内存运行在 10MHz 的 INTOSC1 时钟，而 CPU 运行在 200MHz，因此对相关寄存器的写入应加入一定的延时，否则可能会出现第二次写入失败，而步骤 5 的设置时钟源和分频系数的寄存器 ETHERCATCLKCTL 刚好就在这个低频 10MHz 时钟上。通过检查 TI 提供的库函数，确实是对 ETHERCATCLKCTL 执行了两次连续的写入，而且第二次写入的为需要配置的时钟源和分频系数。但根据之前的描述，第二次的正确配置很可能不能正确写入，将会导致严重的通信错误。在发现这个问题后，针对不同的延时进行了测试，发现不同的延时时间，故障复现的概率也不同。

Step	Action
1	General device initialization (configure clock, enable PLL, enable peripheral clocks except EtherCAT)
2	Configure Aux Clock for EtherCAT (if using Aux clock as source)
3	Configure GPIOs for EtherCAT (set pin configurations, set GPIO qualification mode, set pad configuration)
4	Initialize interrupts and register ISR handlers
5	Set EtherCAT clock source and divider. Then configure if EtherCAT PHY is clocked from device or external PHY clock.
6	Configure the EEPROM size
7	Bring ESC out of reset using system control register
8	Perform EtherCAT memory initialization and wait until memory initialization is complete
9	(Optional) Enable debug access to the EtherCAT registers
10	(Optional) Check that EEPROM loaded successfully
11	EtherCAT subsystem configurations for interrupt masking, SYNCx connections, and so on(1)

(1) Applications must make sure that ESC outputs are in a safe state until the EEPROM is loaded and that SYNC and LATCH are configured only after the EEPROM is loaded.

图 3-6. F28P65 CPU1 初始化 ESC 流程

```
static inline void
SysCtl_setECatClk(SysCtl_ECatClkDivider divider, SysCtl_PLLClockSource source,
                  uint16_t enable)
{
    //
    // Clears the divider & the source, then configures it.
    //
    EALLOW;
    HWREGH(CLKCFG_BASE + SYSCTL_O_ETHERCATCLKCTL) =
                   (HWREGH(CLKCFG_BASE + SYSCTL_O_ETHERCATCLKCTL) &
                    ~(SYSCTL_ETHERCATCLKCTL_PHYCLKEN   |
                      SYSCTL_ETHERCATCLKCTL_ECATDIV_M |
                      SYSCTL_ETHERCATCLKCTL_DIVSRCSEL));


    HWREGH(CLKCFG_BASE + SYSCTL_O_ETHERCATCLKCTL) |=
                ((uint16_t)divider << SYSCTL_ETHERCATCLKCTL_ECATDIV_S) |
                ((uint16_t)source << SYSCTL_ETHERCATCLKCTL_DIVSRCSEL_S) |
                (enable << SYSCTL_ETHERCATCLKCTL_PHYCLKEN_S);

    EDIS;
}
```

图 3-7. F28P65 设置 Ethercat 时钟库函数

3.3.1 ECAT 时钟配置失效分析

为了分析 ECAT 时钟配置失败的根本原因，设计团队创建了测试环境并进行了仿真，如 图 3-8 所示设计团队在大约 40 个周期内对 ETHERCATCLKCTL 寄存器执行两次写操作。如 图 3-8 红色标记了第一次和第二次写入（VBUS32_REQ: 黄色波形）。这次写操作必须配置 ETHERCAT 时钟分频器的值，它处于较慢的时钟域（INTOSC1）。因此，需要从 200 MHz 到 10 MHz 的脉冲传输。在脉冲传输信号"Pulse sync"下，可以看到信号 i_src_clk（源时钟）、i_src_pulse（源脉冲）、i_des_clk（目的时钟）、o_des_pulse（目的脉冲）和 des_ack（来自目标域的确认信号）。只要 ack 信号为高电平，对该寄存器的任何其他写入操作都会被遗漏。在此场景中，当 des_ack 信号（品红/粉色信号）为高电平时，对寄存器的第二次写入发生。但该值从未到达分频器，分频器只配置了旧值。

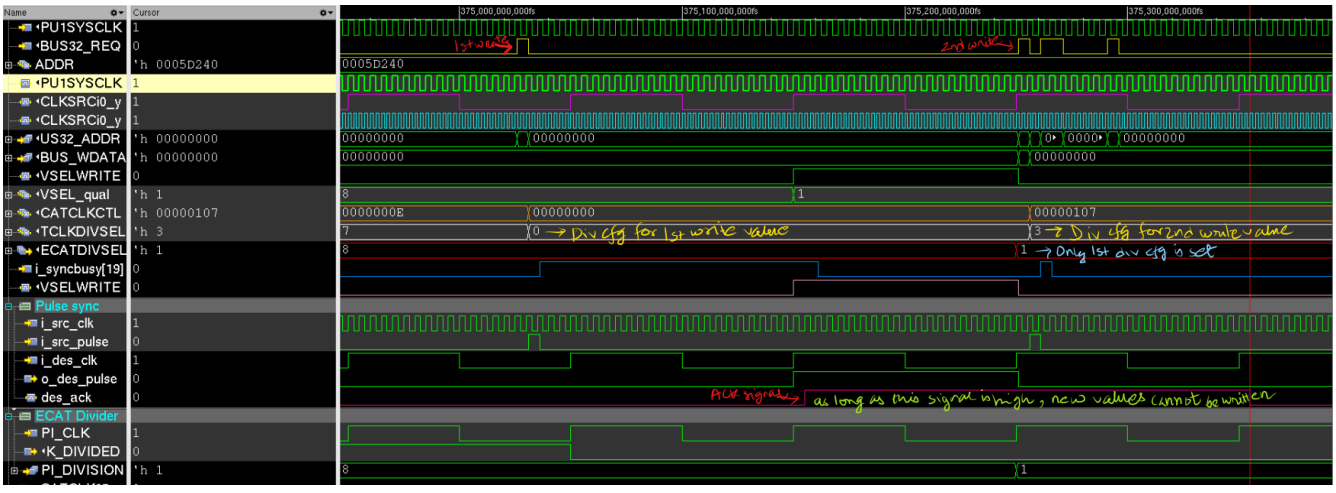

图 3-8. ESC 时钟配置 RTL 仿真

3.3.2 ECAT 时钟配置建议

上述失效问题，本质上是由跨时钟域的亚稳态导致的，即当一个信号从一个时钟域传输到另一个时钟域，且两个时钟相位关系不确定或频率不同时，就会发生跨时钟域传输。而亚稳态是数字电路中触发器的一种不确定状态，

他发生在当时钟采样信号到来时，数据输入信号整处于变化。为了保证第二次成功，并且确保跨时钟域的亚稳态问题的发生建议使用如下公式计算两次寄存器写入的延时：

Delay = 6 x (F_{SYSCLK} / $F_{INTOSC1}$) + 9

因此，为了让 EtherCAT 时钟的分频值正确写入，应至少延时 129 cycle。如 图 3-9 所示：

#define SYSCTL_REGWRITE_DELAY **asm**(" RPT #129 || NOP")

```
static inline void
SysCtl_setECatClk(SysCtl_ECatClkDivider divider, SysCtl_PLLClockSource source,
                        uint16_t enable)
{
    //
    // Clears the divider & the source, then configures it.
    //
    EALLOW;
    HWREGH(CLKCFG_BASE + SYSCTL_O_ETHERCATCLKCTL) =
                    (HWREGH(CLKCFG_BASE + SYSCTL_O_ETHERCATCLKCTL) &
                     ~(SYSCTL_ETHERCATCLKCTL_PHYCLKEN  |
                       SYSCTL_ETHERCATCLKCTL_ECATDIV_M |
                       SYSCTL_ETHERCATCLKCTL_DIVSRCSEL));
    SYSCTL_REGWRITE_DELAY;

    HWREGH(CLKCFG_BASE + SYSCTL_O_ETHERCATCLKCTL) |=
                ((uint16_t)divider << SYSCTL_ETHERCATCLKCTL_ECATDIV_S) |
                ((uint16_t)source << SYSCTL_ETHERCATCLKCTL_DIVSRCSEL_S) |
                (enable << SYSCTL_ETHERCATCLKCTL_PHYCLKEN_S);
    SYSCTL_REGWRITE_DELAY;
    EDIS;
}
```

图 3-9. 加入延时的设置 Ethercat 时钟库函数

F28P65 库函数通常在对一个寄存器写入前会执行寄存器清零，从而可能导致第二次写入因延时不够而失败。另一种简单的做法是，在配置 EtherCAT 时钟时只对寄存器执行一次正常值写入，避免二次写入所造成的配置失败。

3.3.3 利用 DCC 解决跨时钟域的亚稳态问题

DCC (Daul Clock Comparator)是 F28P65 中用于监控时钟频率和检测时钟故障的安全特性模块。它的核心功能是比较两个独立的时钟源频率，确保他们在预期的范围内运行。

虽然在两次写入之间加入足够的延时能够保证第二次写入成功，但跨时钟域所引入的亚稳态问题是不可避免的，会有极低的概率发生亚稳态，仍然有可能导致 ESC 时钟配置错误，进而引发通信错误。可以参考图 3-6 ，利用 C2000 DCC 外设监控 ESC 时钟是否正确配置。

```
// Verify the frequency of ECATPHYCLK clock using the SYSCLK as reference clock
// FClk1 = eCAT PHY Clock frequency = 100MHz
// FClk0 = SYSCLK frequency = 200MHz
// Tolerance = 1%
// Allowable Frequency Tolerance = 0% (update as per the error in the ECATPHYCLK frequency)
// SysClk Freq = 200MHz
//
// Note: Update the parameters in case you are using different PLL or XTAL
// frequencies,
//
status = DCC_verifyClockFrequency(DCC0_BASE,
                                  DCC_COUNT1SRC_ECATPHYCLK, 100.0F,
                                  DCC_COUNT0SRC_SYSCLK, 200.0F,
                                  1.0F, 0.0F, 200.0F);
```

图 3-10. 利用 DCC 模块监控 EtherCAT 时钟

4 总结

本文从 EtherCAT 通信的基本概念入手，介绍了 EtherCAT 通信的特点及原理，EtherCAT 通信模型和 ESC 的诊断功能，从实际帧错误故障案例出发，提供了一套 EtherCAT 帧错误的故障分析方法，包括诊断信息排查、错误累加和传播分析、故障位置隔离分析，给出了 F28P65 在 EtherCAT 应用上的使用建议，为使用 F28P65+DP83826 实现 EtherCAT 应用的场景提供了分析和调试经验。

5 参考文献

1. EtherCAT_Diagnosis_For_Developers
2. ethercat_esc_datasheet_sec1_technology_2i3
3. ethercat_esc_datasheet_sec2_registers_3i0
4. ethercat_et1100_datasheet_v2i1
5. F28P65 TRM
6. DP83826 Troubleshooting guidance

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（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，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(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、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月