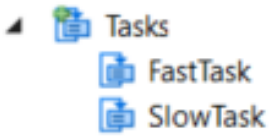
TwinCAT 中的 EtherCAT 同步 (expert)

04-Jul-2017

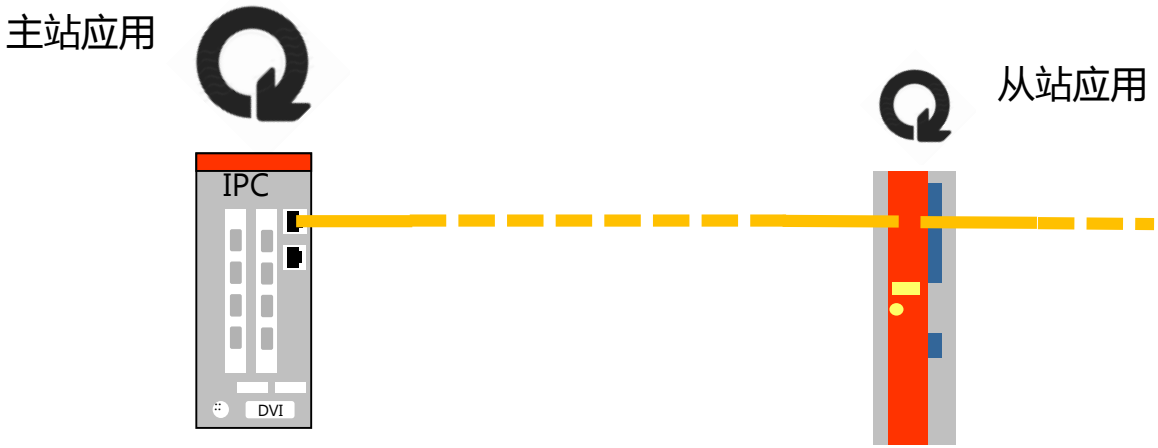
应用层的同步

主站设备和从站设备上的应用都由**周期性**执行的软件代码组成

- **主站应用**：PLC 程序, NC 任务, …



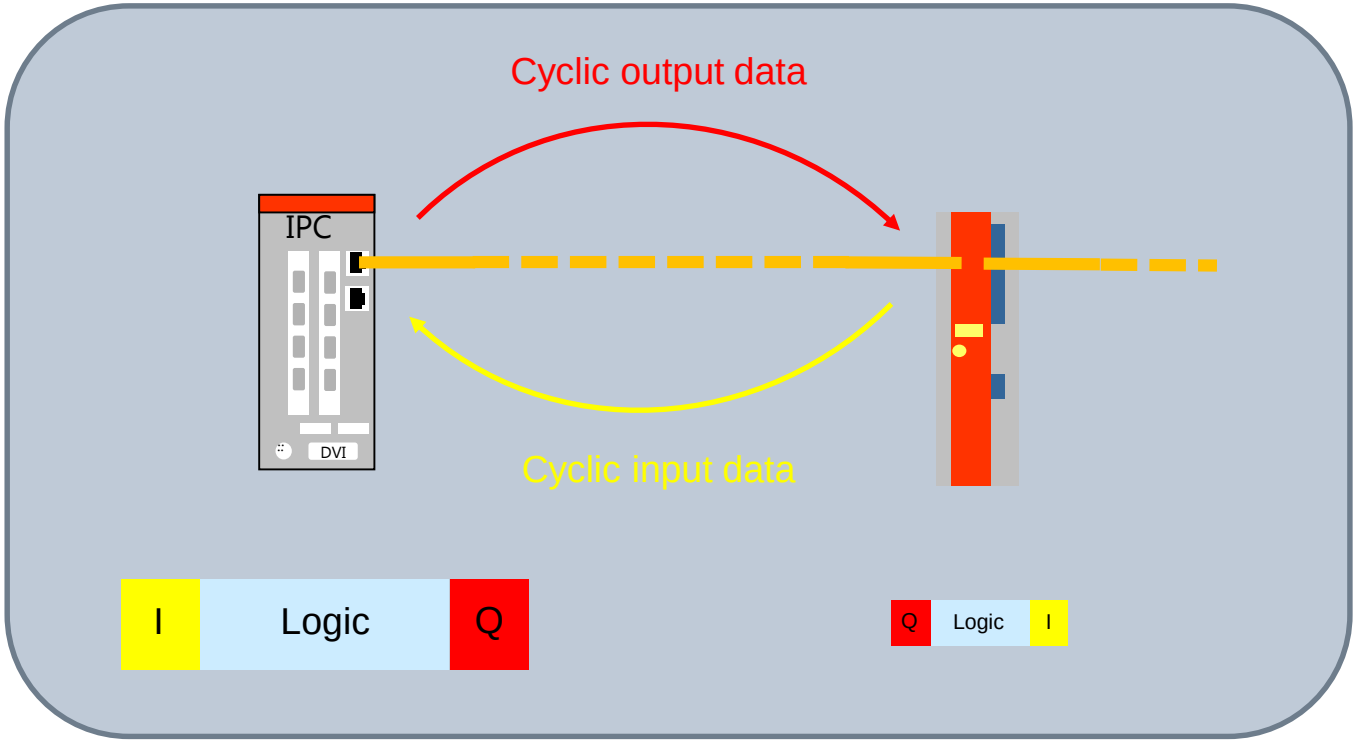
- **从站应用**：从站的固件（firmware）



应用层的同步

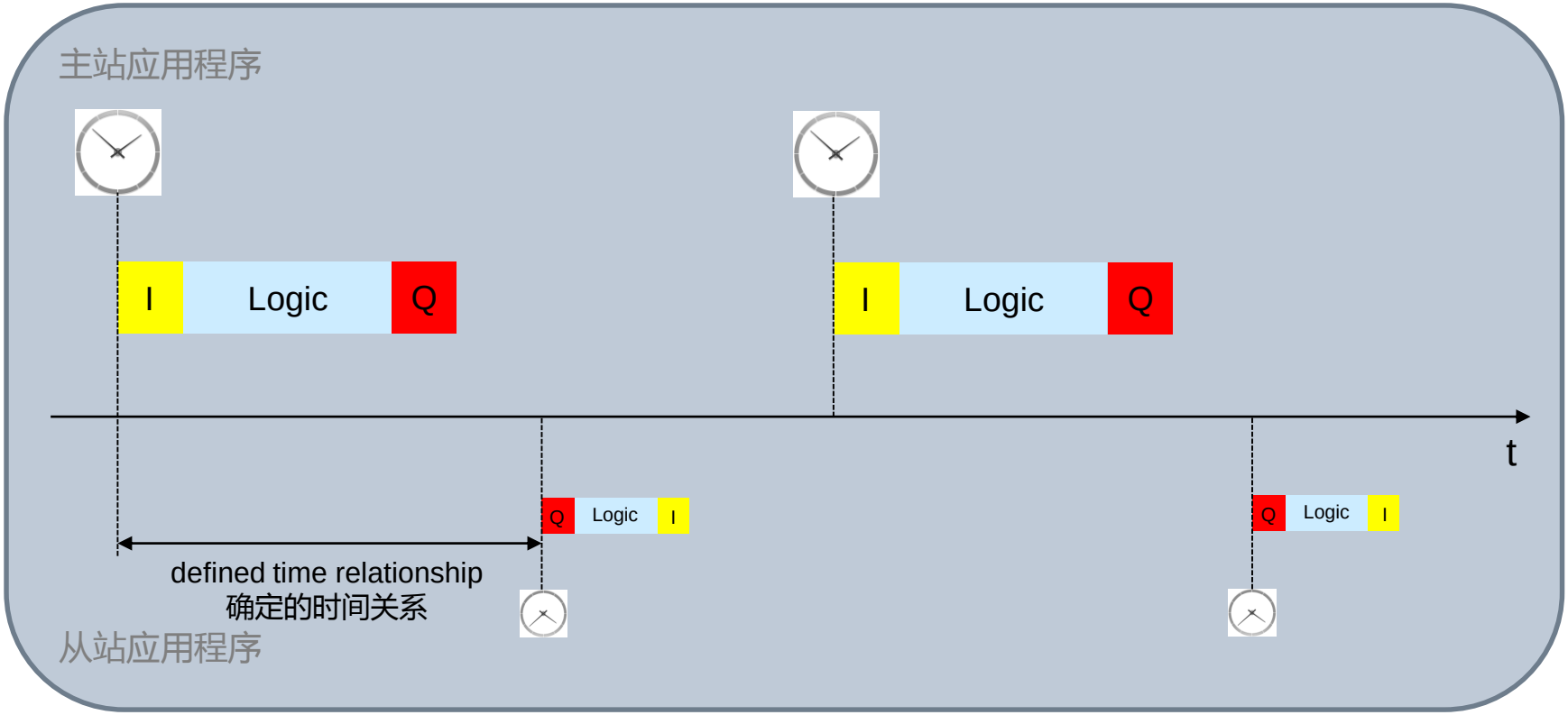
主站和从站之间的同步，从各自**应用**的角度看有何意义？

主站应用和所有从站的应用周期性地双向交换过程数据（**process data** ）：



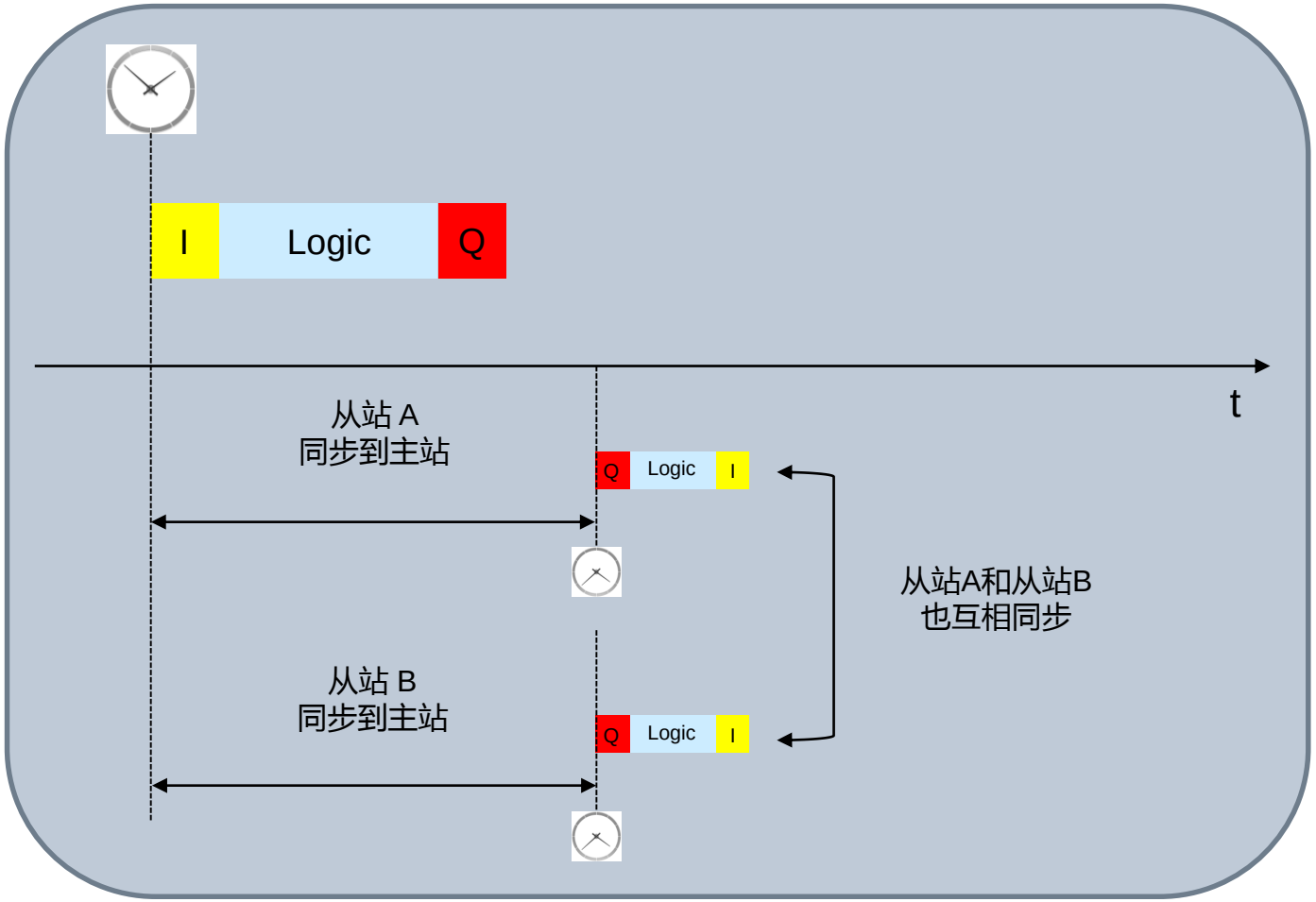
应用层的同步

总体来说，主站应用和从站应用的同步**确定**了主站循环和从站循环的起始点之间的**时间关系**。



应用层的同步

每个从站都同步到主站，意味着不同从站之间也实现了相互的同步。



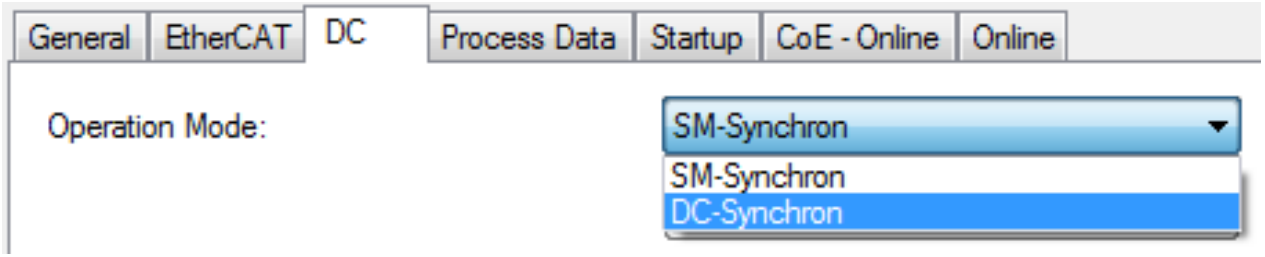
应用层的同步模式

关于从站应用相对主站循环的时间关系，EtherCAT定义了3种主要的同步模式(**synchronization modes**):

- **Free Run** (非同步): 从站的过程数据处理，由**内部事件**触发，与主站循环无关。
- **SM-Synchronous**: 从站的过程数据处理，由接收到携带过程数据的周期性数据帧时所产生的**硬件中断**触发.
- **DC-Synchronous**: 从站的过程数据处理，由基于分布时钟和系统时间的硬件中断触发。

.

支持DC功能的从站，其同步模式在 TwinCAT 中的“DC” 页面进行配置 (每个从站的同步模式互相独立):



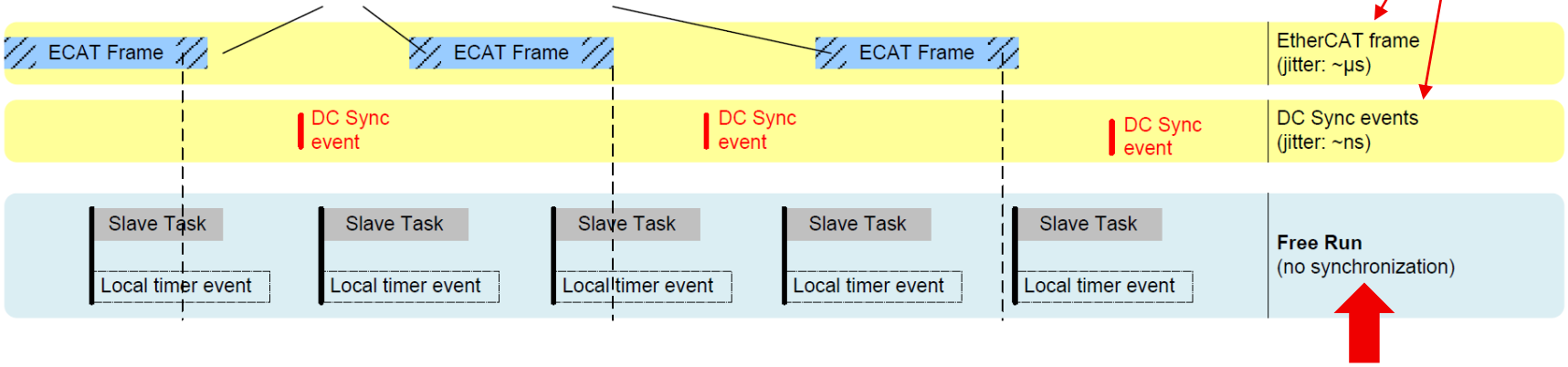
Free Run（自由运行）

Free Run. 从站的过程数据处理，由**内部事件**触发:

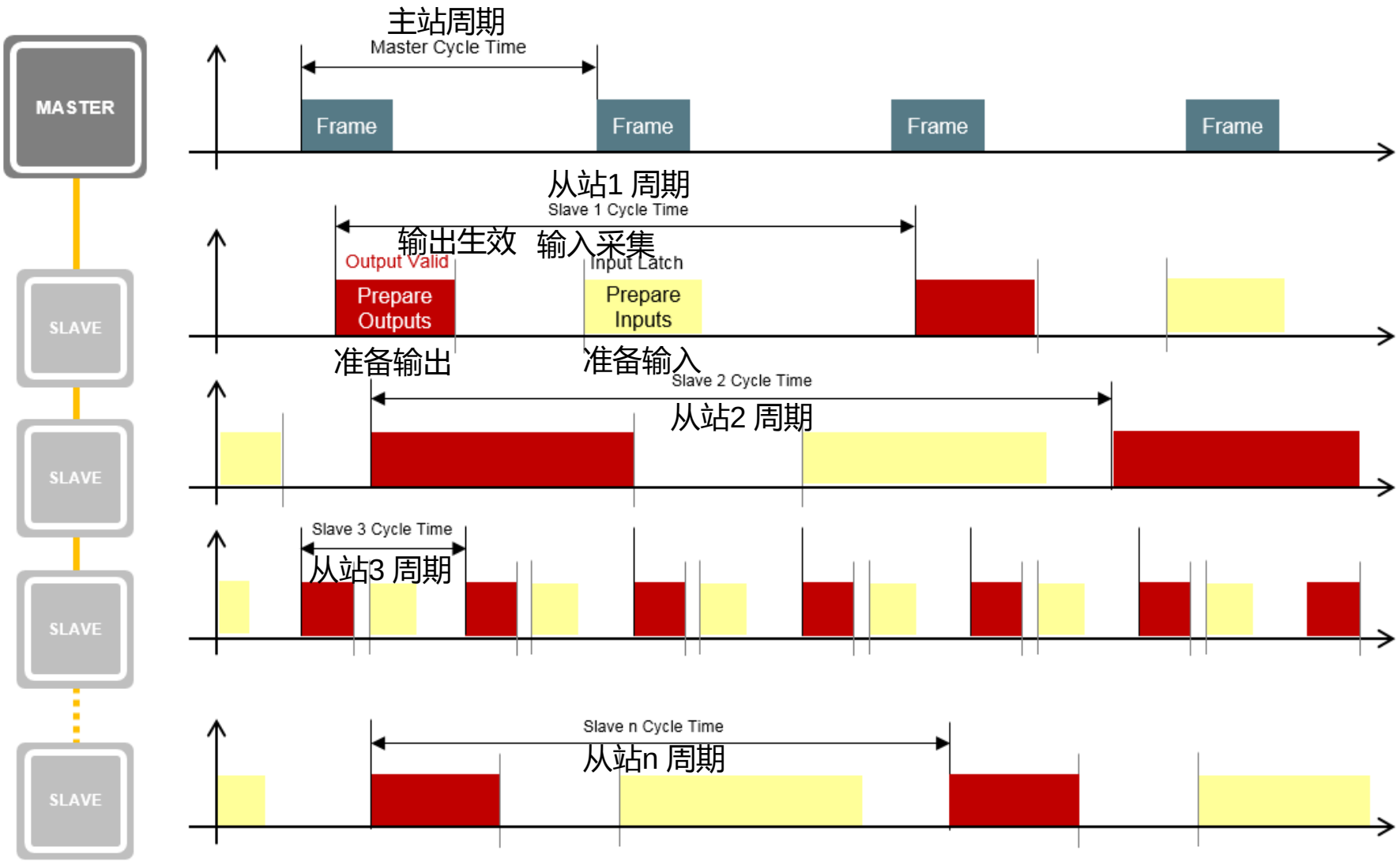
- 不用定义周期性数据帧与从站本地应用程序之间的时间关系
- 各个“Free Run” 模式的从站之间，其时间偏移量不固定
- 适用于处理信号变化缓慢的 I/O 设备 (比如温度信号 …)

EtherCAT帧的波动：us级
DC同步事件的波动：ns级

受主站应用的影响，数据帧到达时间会有所波动（~us）



Free Run（自由运行）

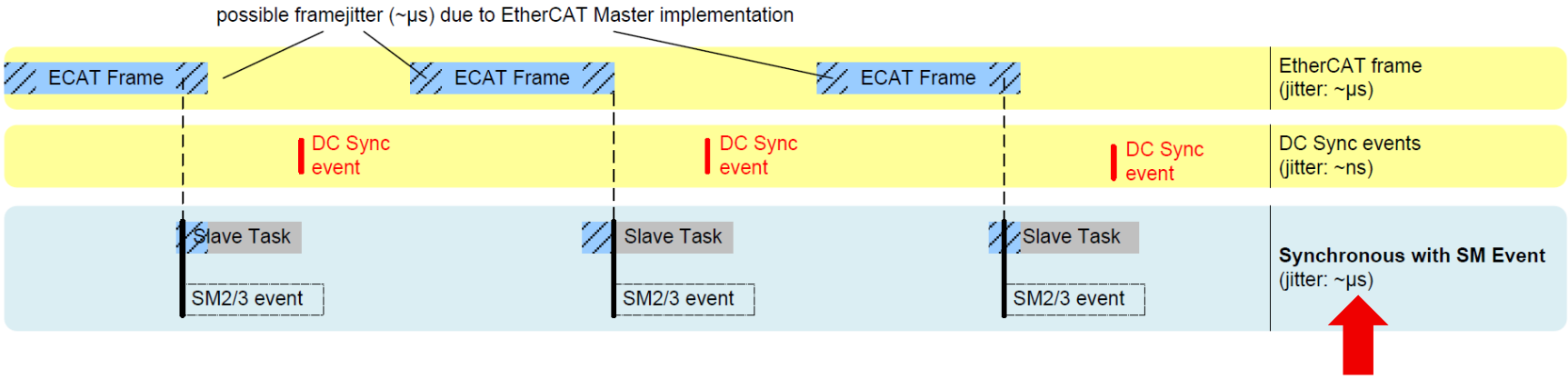


SM-同步

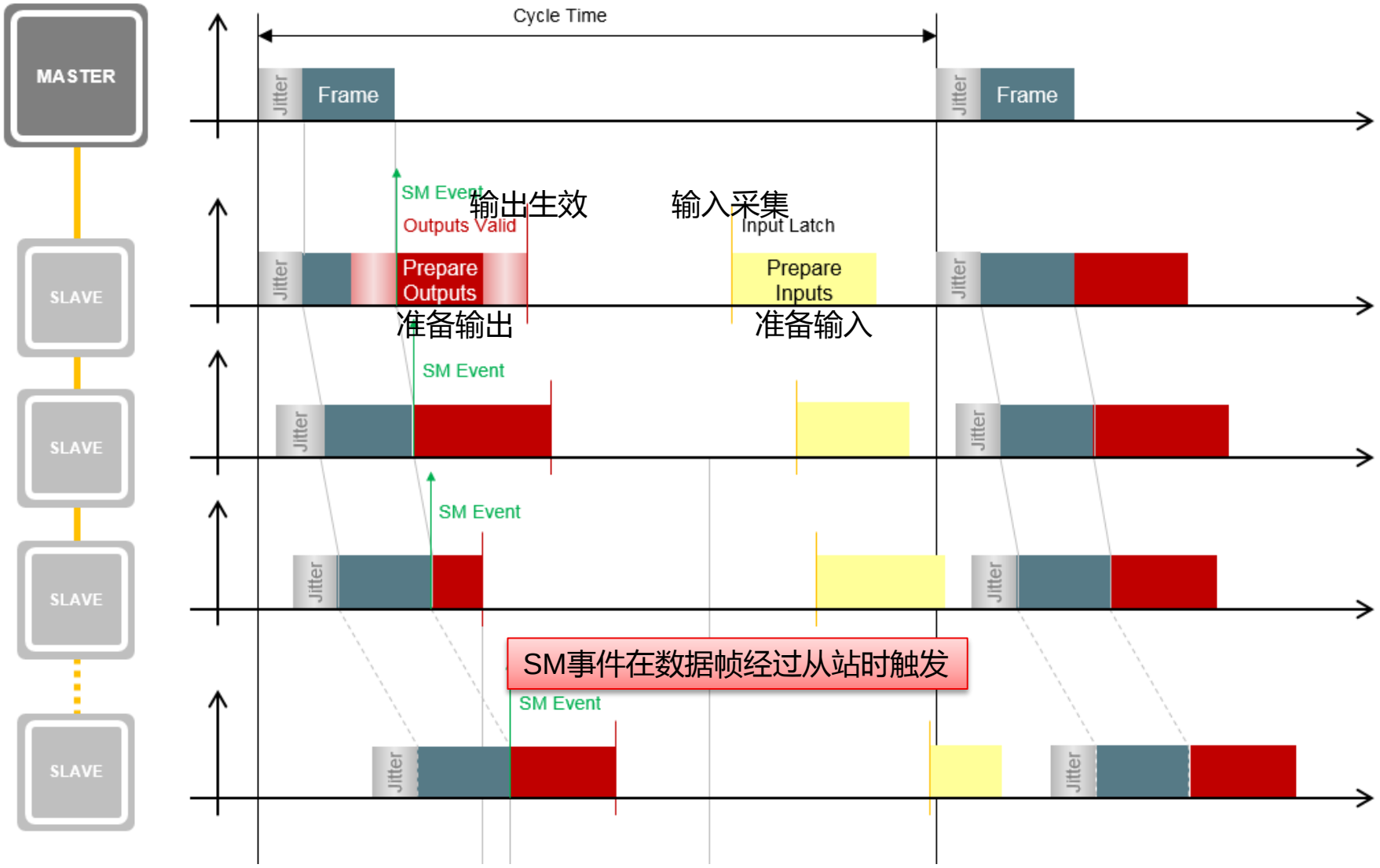
SM-Synchronous. 从站的过程数据处理，在接收到周期性数据帧时触发.

- 同步不准的第1个原因: 所有从站接收的周期性数据帧具有同样的抖动（Jitter），该抖动值受发送此帧的主站影响。
- 同步不准的第2个原因: 即使没有抖动（Jitter），受实际硬件传播延时的影响，最末端的从站接受数据帧的时间必然晚于第一个从站的接收到该数据帧时间。

- 适用于PLC控制的 定位动作。



SM-同步


MASTER
SLAVE
SLAVE
SLAVE
SLAVE
Cycle Time
Jitter
Frame
Jitter
Frame
SM Event
输出生效
Outputs Valid
Prepare Outputs
准备输出
输入采集
Input Latch
Prepare Inputs
准备输入
SM Event
SM Event
SM事件在数据帧经过从站时触发
SM Event

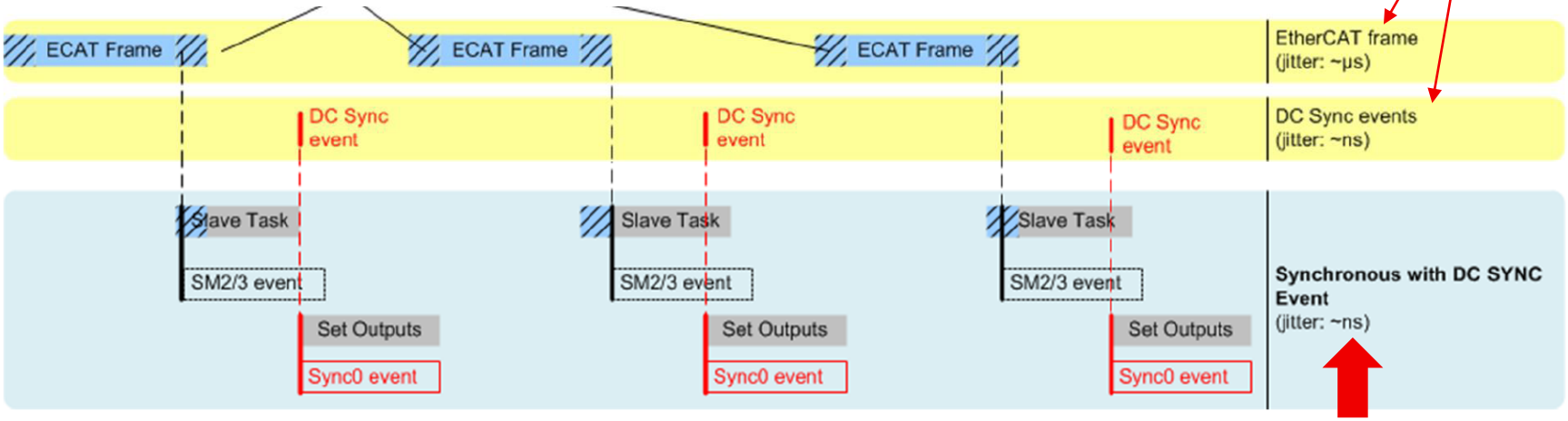
DC-同步

DC-Synchronous. 从站过程数据的处理，由从站中基于DC系统时间产生的硬件同步事件（SYNC Event）触发。

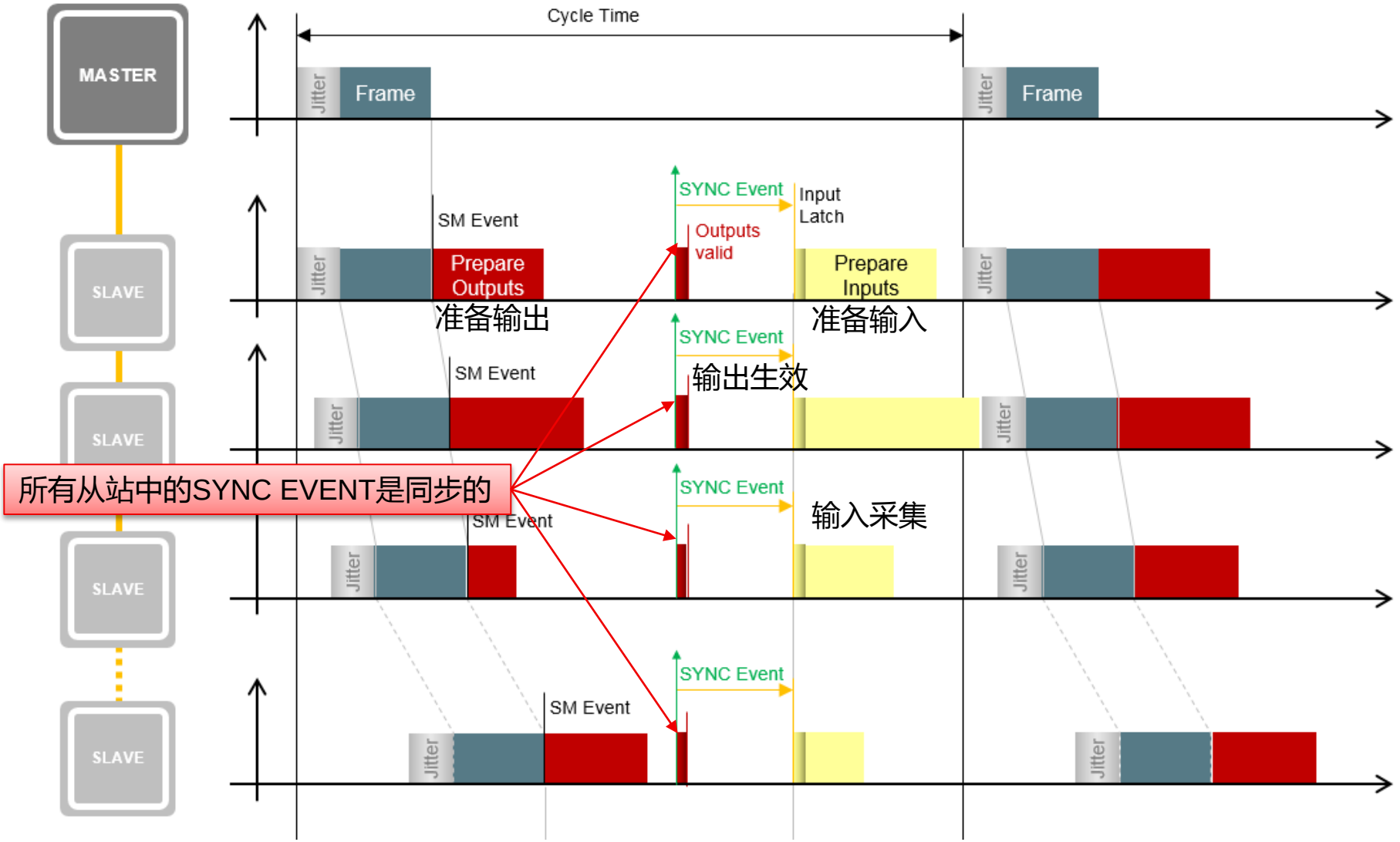
- 基于分布时钟的系统时间（DC System Time），在每个DC从站的内部产生硬件同步事件。
- 每个从站中的触发事件不受主站抖动和传播延时的影响。
- 适用于 NC任务控制的伺服驱动，以及超采样的IO模块

EtherCAT帧的波动：us级
DC同步事件的波动：ns级

受主站应用的影响，数据帧到达时间会有所波动（~us）

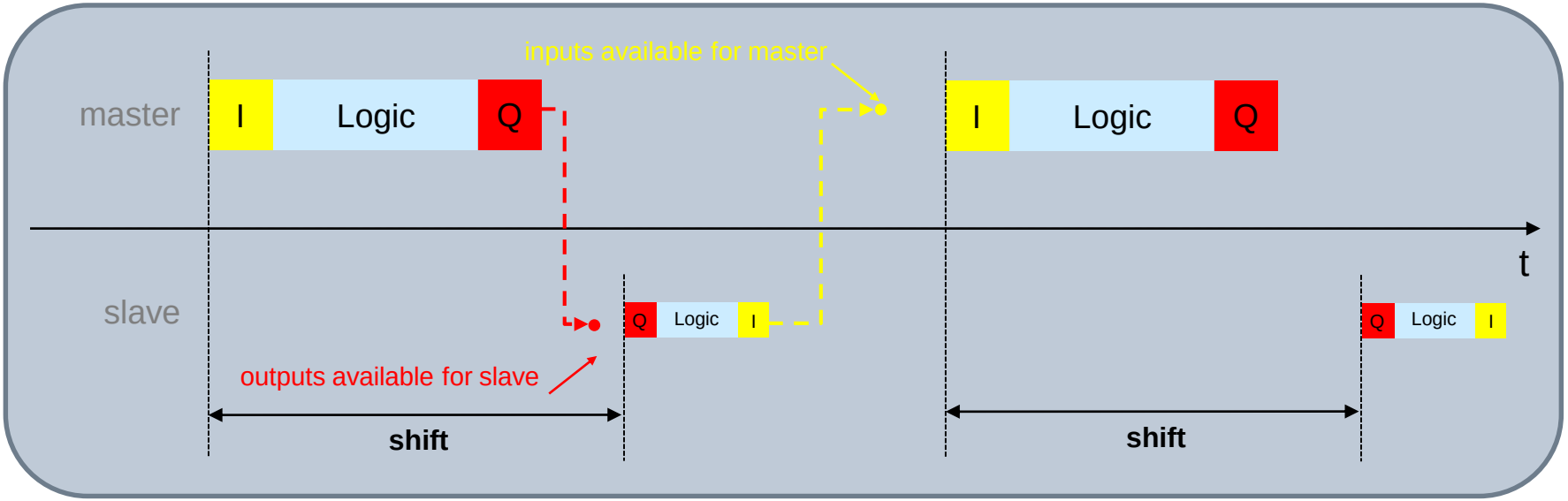


DC- 同步


MASTER
SLAVE
SLAVE
SLAVE
SLAVE
Cycle Time
Jitter
Frame
Jitter
Frame
SM Event
SYNC Event
Input
Latch
Outputs
valid
Prepare
Outputs
准备输出
Prepare
Inputs
准备输入
SM Event
SYNC Event
输出生效
所有从站中的SYNC EVENT是同步的
SM Event
SYNC Event
输入采集
SM Event
SYNC Event

工作在同步模式: Time Shift

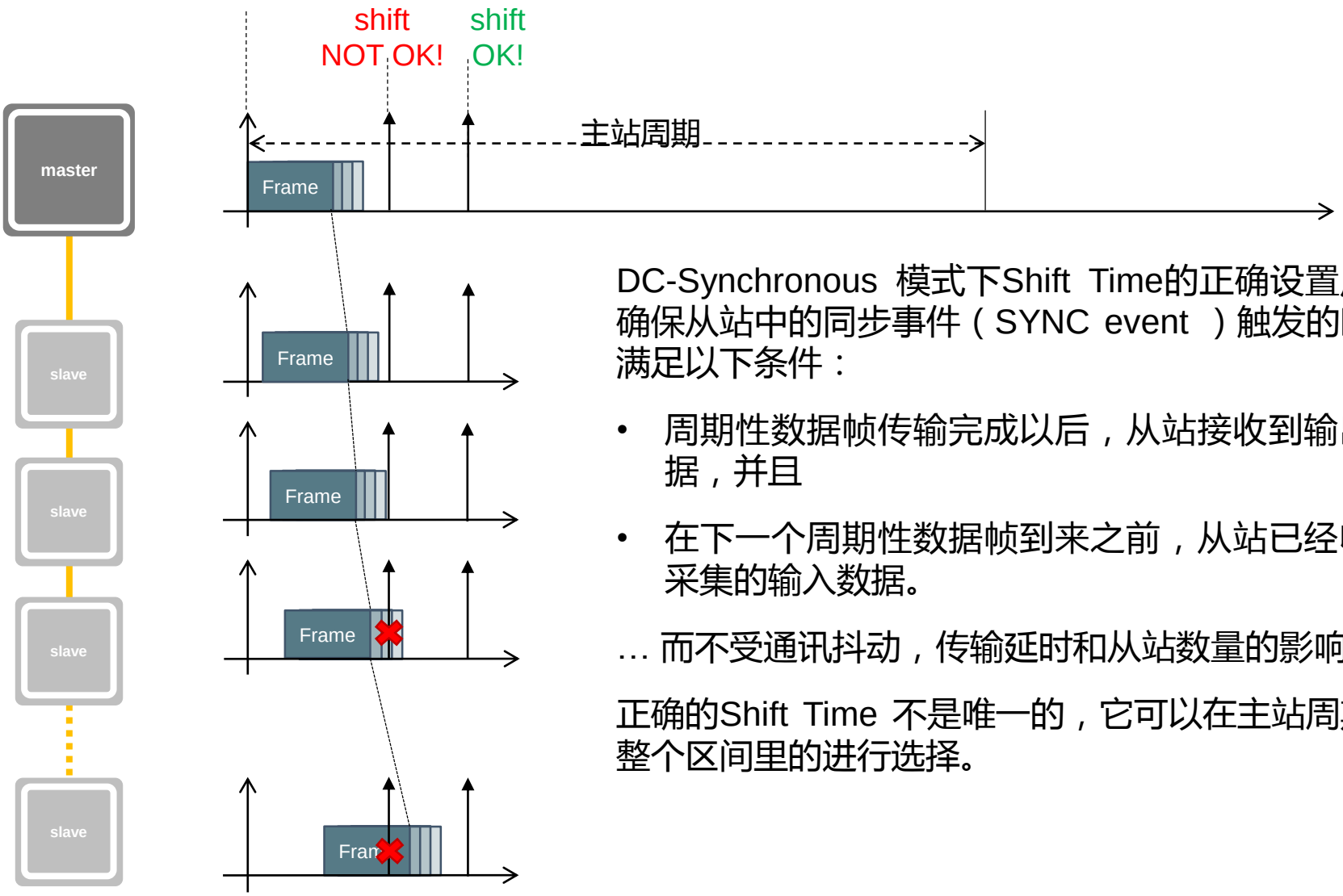
在 **SM-** 和 **DC-synchronous**模式，在主站和从站应用程序的循环起始点之间总是需要若干偏移时间 **shift**, 以使所有从站都能在它的循环程序开始之前接收到数据：



在**SM-Synchronous** 模式，偏移时间由同步模式本身设定（不需要参数配置），因为从站的应用程序是由循环发送的数据帧直接触发的。

在 **DC-synchronous** 模式，同步信号与主站循环超始点的偏移时间由主站在网络启动时设置，必要时用户可以修改。

怎样才是正确的偏移时间（Shift Time）



DC-Synchronous 模式下Shift Time的正确设置应该确保从站中的同步事件（SYNC event ）触发的时间满足以下条件：

- 周期性数据帧传输完成以后，从站接收到输出数据，并且
- 在下一个周期性数据帧到来之前，从站已经收到采集的输入数据。

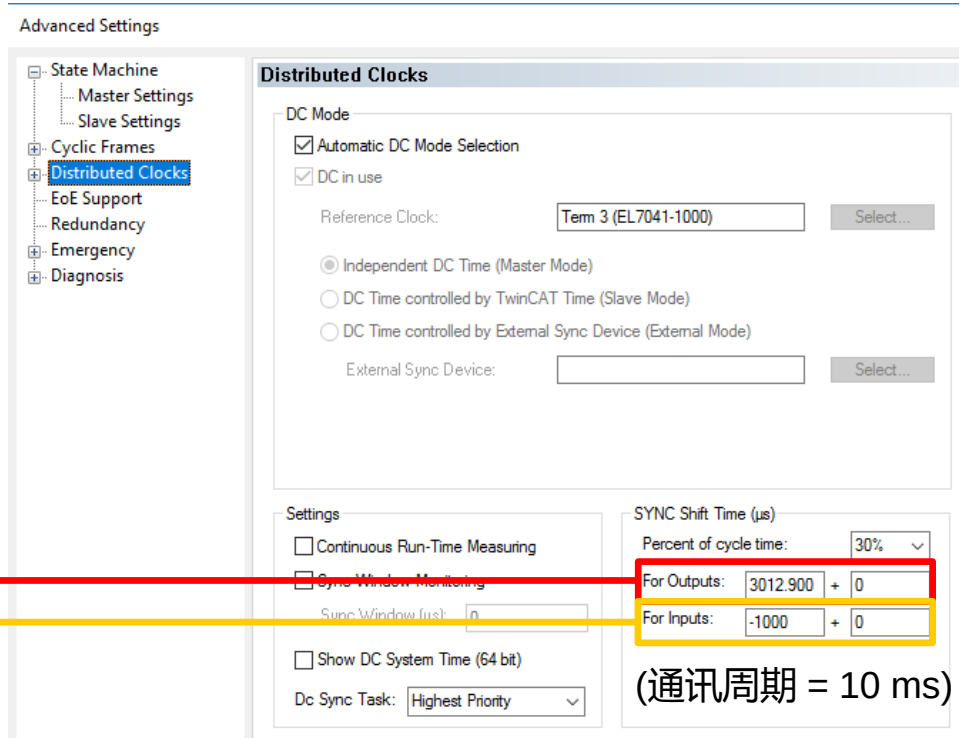
… 而不受通讯抖动，传输延时和从站数量的影响。

正确的Shift Time 不是唯一的，它可以在主站周期的整个区间里的进行选择。

主站的 SYNC Shift Times（同步偏移时间）

通过主站的 **SYNC Shift Time** 设置所有DC同步模式的从站的时间偏移. 可以分别设置两个不同的 SYNC Shift Time 参数:

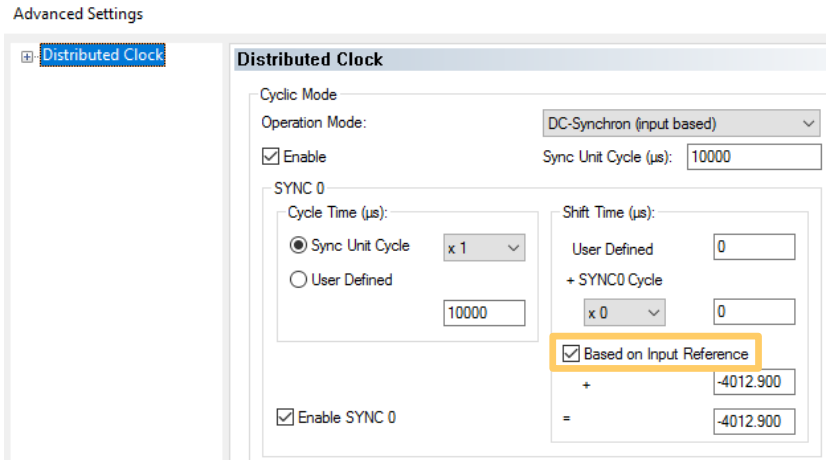
- **For Outputs**: 通常用于过程映像区中支持周期性输出的从站
- **For Inputs**:通常用于只有周期性输入的从站（过程映像区中没有周期性输出）

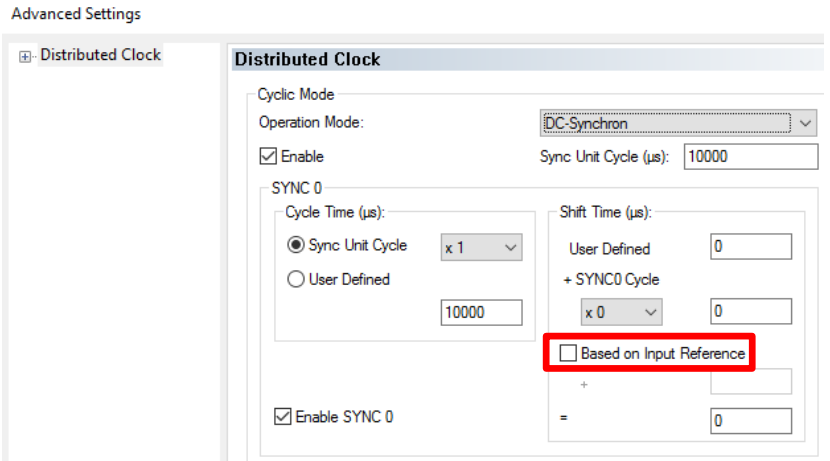


Outputs 及 Inputs 的同步偏移时间（ SYNC Shift Time ）

每个DC-Synchronous模式的从站都配置为使用主站的SYNC Shift Time 作为输出偏移或者输入偏移, 从来不会同时用作输出偏移和输入偏移。在ESI文件中有标明 SYNC Shift Time 是用作输入还是输出偏移，并显示在TwinCAT的从站设置页面。

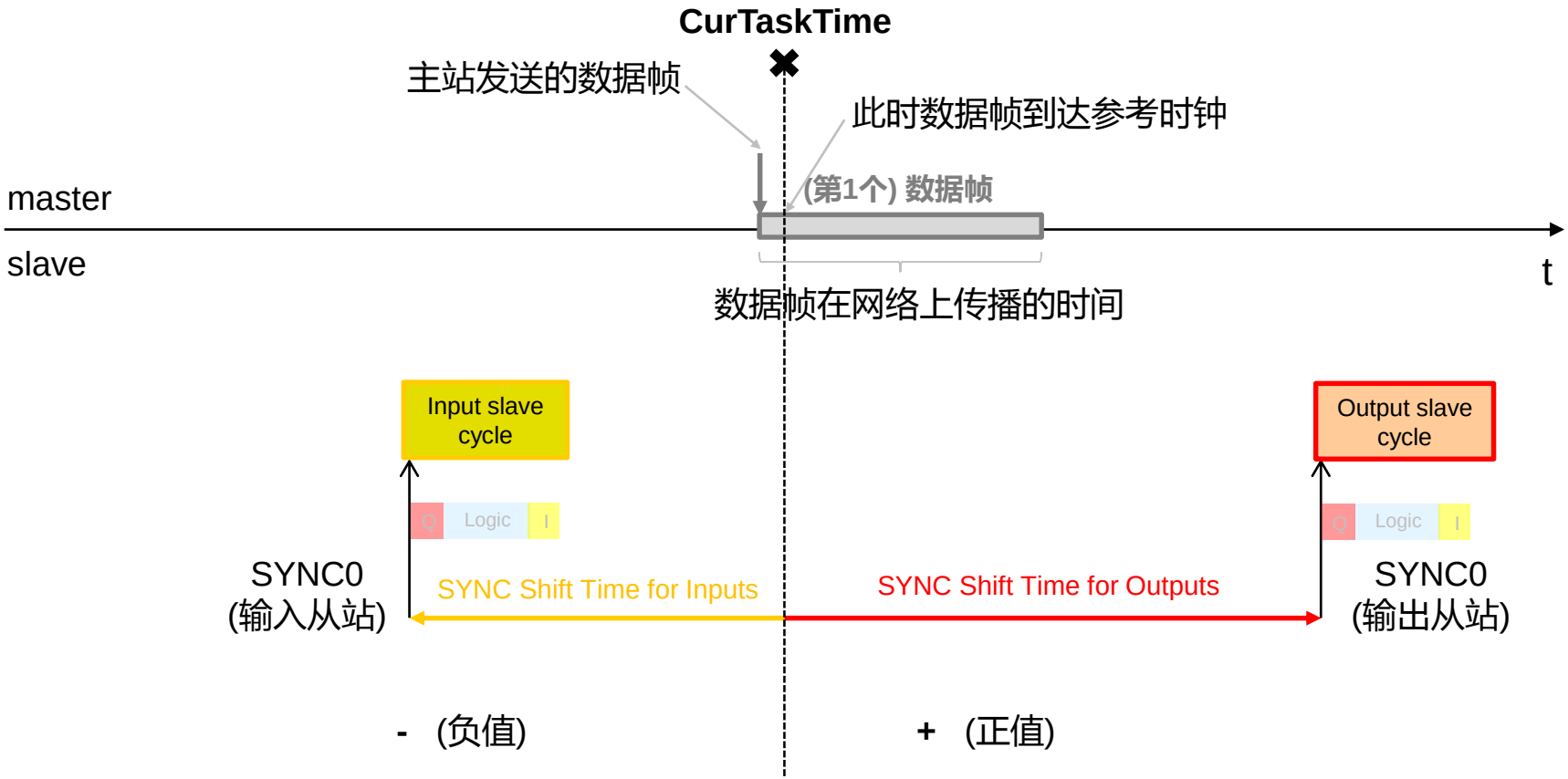
- 如果在slave Advanced Settings 中使用默认设置的 “**Based on Input Reference**”标记，主站在对从站进行初始化时就会使用 SYNC Shift Time 作为输入的同步偏移时间 …
- 否则就使用SYNC Shift Time 作为输出的同步偏移时间





SYNC Shifts Times 的参考点

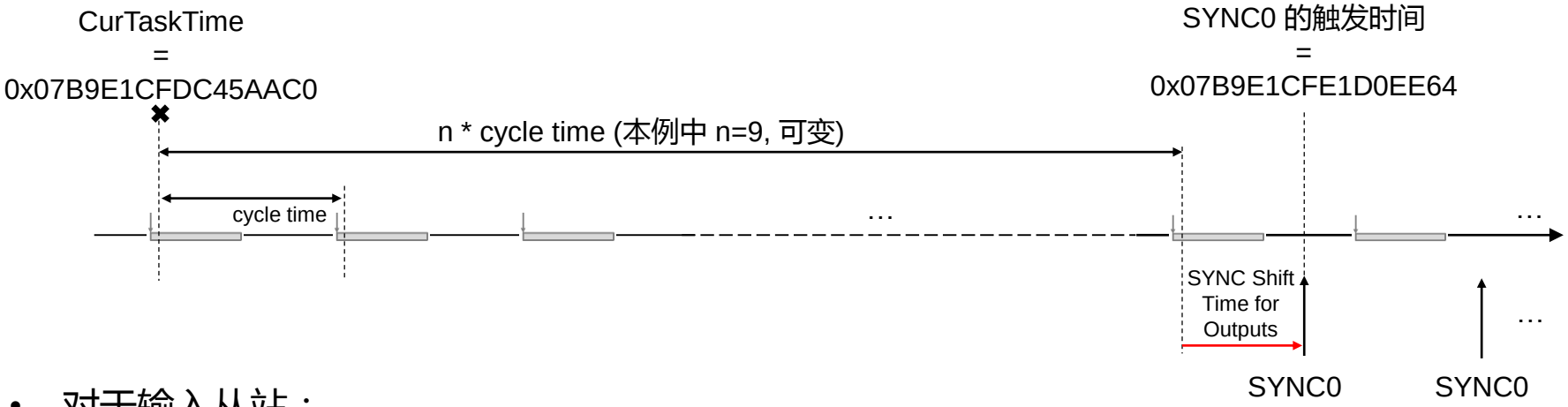
在TwinCAT中，SYNC Shift Time 的参考点称为 **CurTaskTime**, 对应于周期性数据帧第1次到达参考时钟设备 (网络中第1个 DC从站)的时间。



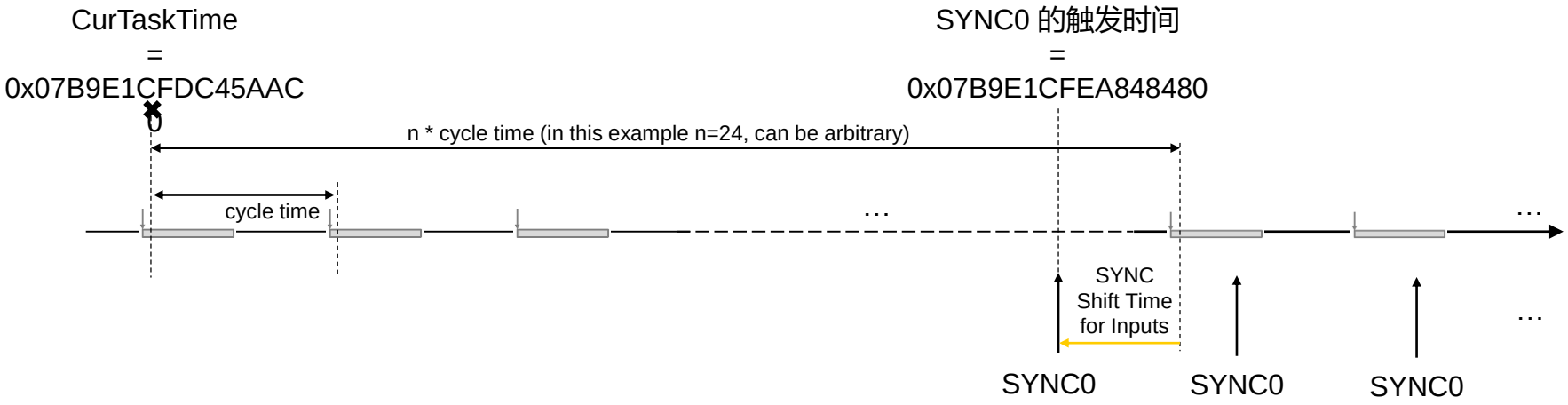
深入了解 – 初始化时 SYNC Shift Times 的使用

初始化一个DC同步的从站的时候，主站并不向从站发送 SYNC Shift Time, 而是发送产生第1个SYNC信号的绝对时间值……

- 对于输出从站：

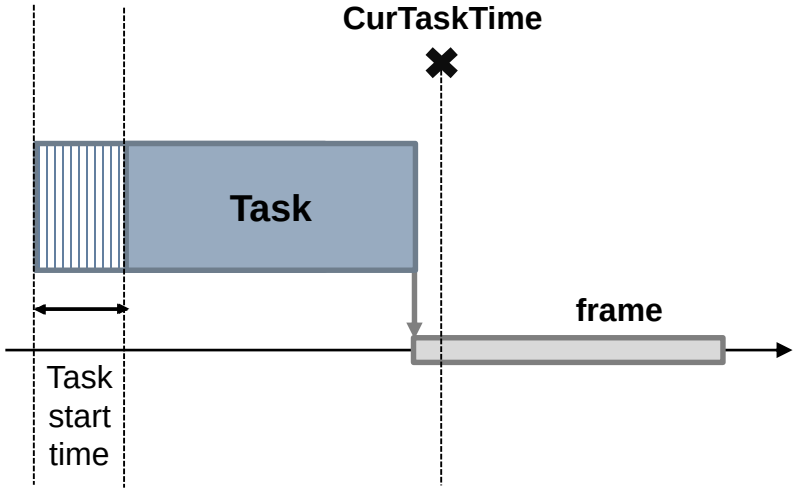


- 对于输入从站：



深入了解 – TwinCAT 中的 CurTaskTime 和 任务开始

CurTaskTime 通常是所有DC时间的参考点。TwinCAT 调整软件任务 (PLC, NC, …) 的开始时间，以保持 CurTaskTime 为恒定的时间间隔 (该间隔对应于软件配置的任务周期)：

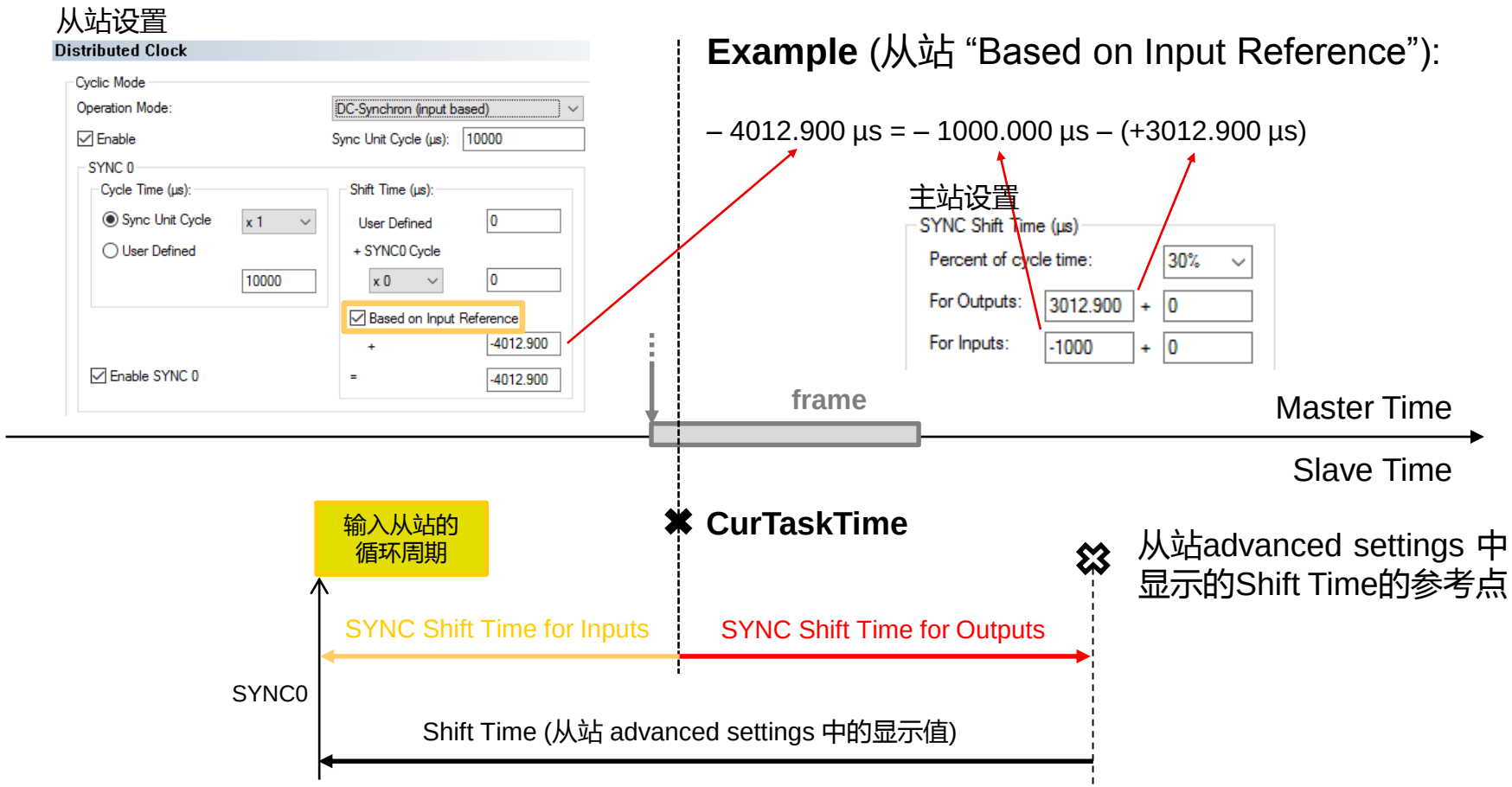


对于执行时间波动较大的任务，连接到“I/O at Task End” 操作， 要维持 CurTaskTime 稳定可能会非常困难甚至是不可能的，可能会导致EtherCAT网络报告同步错误。可能的解决方案是：

1. 增加 SYNC Shift Time 的值 (见下页 next slides)
2. 把 “I/O at Task End” 改成 “I/O at Task Begin” (见下页 next slides)

从站 Advanced Settings 中的 Shift Time 值

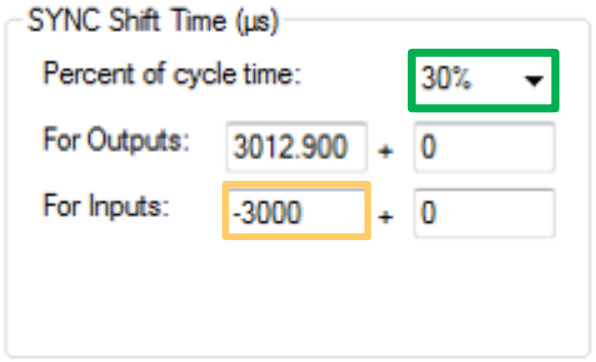
从站“Advanced Settings”中 “Shift Time ”的**显示值**是从站同步脉冲相对于主站“Advanced Settings”中的SYNC Shift Time For Output 的偏移量。 如果把参考点的差异考虑在内，绝对偏移量与主站中的 Advanced Settings设置是一致 (但是，如果从站置了自定义的偏移量):



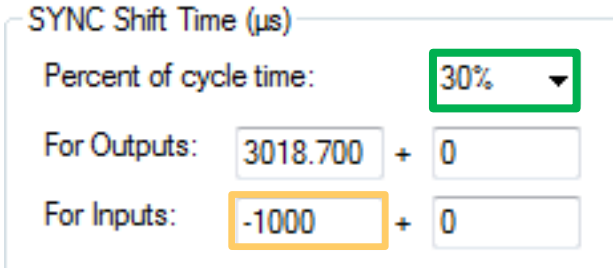
SYNC Shift Times 的默认值

TwinCAT 自动把 SYNC Shift Time for Outputs 的默认值设置为通讯周期（**communication cycle time**）的**30%，**而 SYNC Shift Time for Inputs 的值与 TwinCAT 版本有关）：

TwinCAT 2



TwinCAT 3



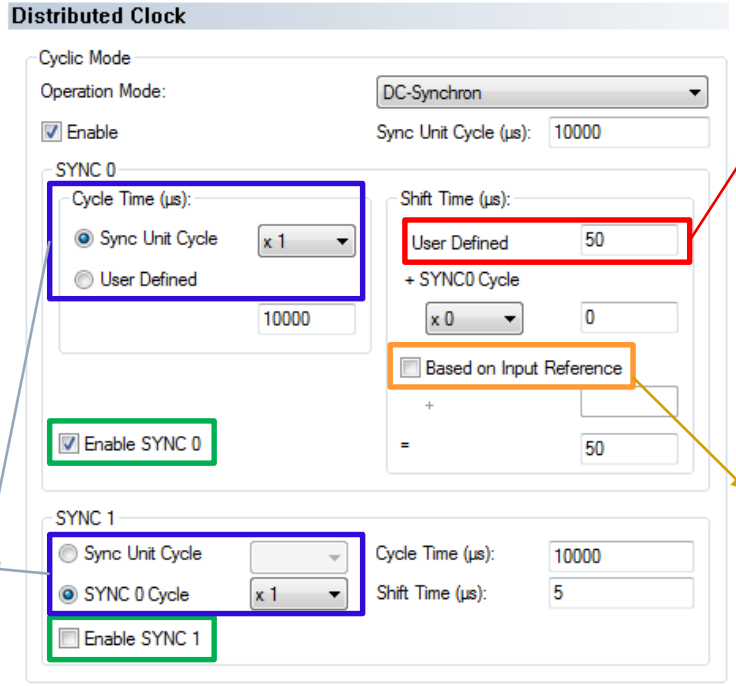
这个默认值是一个很好的平衡，在绝大部分实际的应用项目中可以确保实现正确的时间同步。

只有非常有挑战性的应用项目（任务周期极短，从站数量极多），或者EtherCAT网络启动后发生了同步错误，才需要更精确地评估并调整 SYNC Shift Time 的默认值。示例见下页 following slides.

深入了解 – 从站的 DC Advanced Settings

与主站设置中的全局性的 SYNC Shift Time 参数不同，大部分从站 Advanced Settings 中的DC分布时钟参数是从它们的ESI文件提取的：这些参数由从站的生产厂家决定，保证实现正确的操作，用户**切勿修改**。

周期性触发的同步信号(SYNC signal)应当具备以下条件：



从站初始化时，主站会把从站的 User Defined 的 Shift Time 迭加到主站定义的全局（global）偏移量SYNC Shift Time

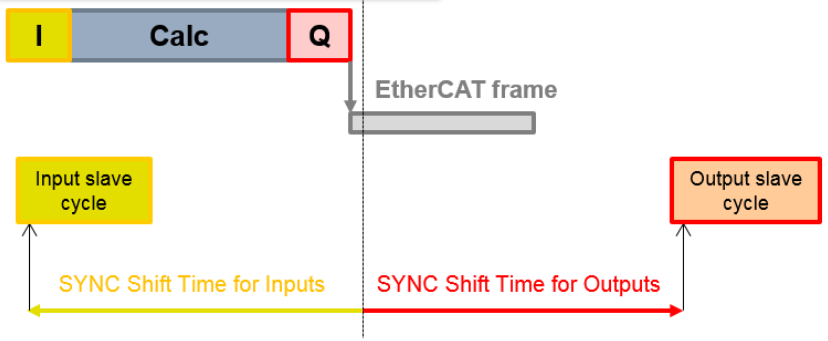
从站初始化时，主站根据此项设置确定从站的同步脉冲基于读取输入的参考点还是基于刷新输出的参考点。

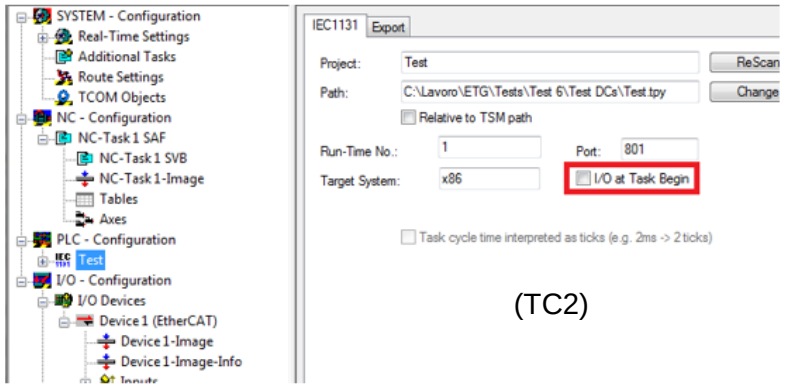
哪个同步信号启用，而哪个同步信号禁用

TwinCAT 设置 - I/O at Task End 及 I/O at Task Begin

TwinCAT 发送 EtherCAT 数据帧的时间与任务循环的时间可以有两种关系：

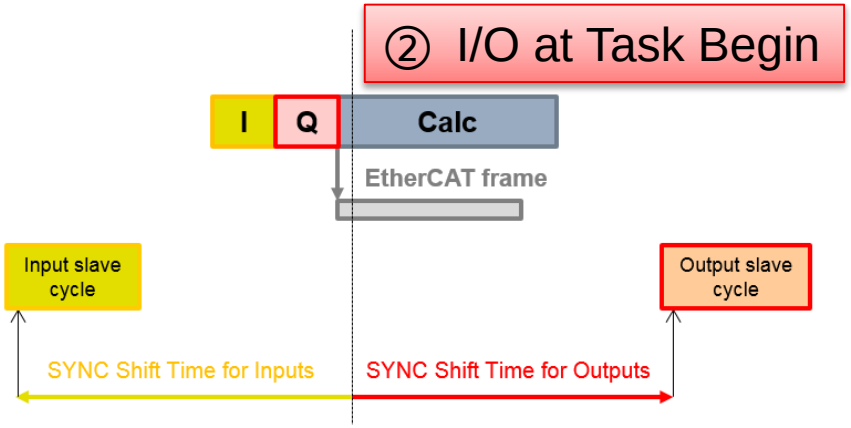
① I/O at Task End

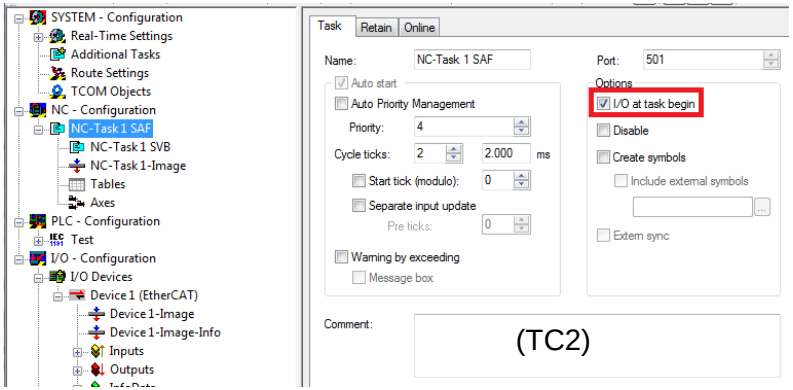




(TC2)

② I/O at Task Begin





(TC2)

- “I/O at Task Begin” 是NC 任务的默认设置(不可修改)
- “I/O at Task End” 是PLC 任务和 IO 任务的默认设置

深入了解 – TwinCAT 3 中的 I/O at Task Begin

在 TwinCAT 3 中， PLC 任务和 C++ Modules 都不提供 “I/O at Task Begin”的标记。此时可以使用属性 **TcCallAfterOutputUpdate** 实现同样的功能。

PLC

Attribute TcCallAfterOutputUpdate

The pragma **{attribute 'TcCallAfterOutputUpdate'}** defines whether a program is to be executed after an output update. This attribute replaces the TwinCAT 2 functionality of the option "IO at Task begin".

Syntax:

{attribute 'TcCallAfterOutputUpdate' }

This attribute must be added to all program POUs, which are to be called after the output update.

Example:

```
{attribute 'TcCallAfterOutputUpdate'}
PROGRAM MAIN
VAR
END_VAR
```

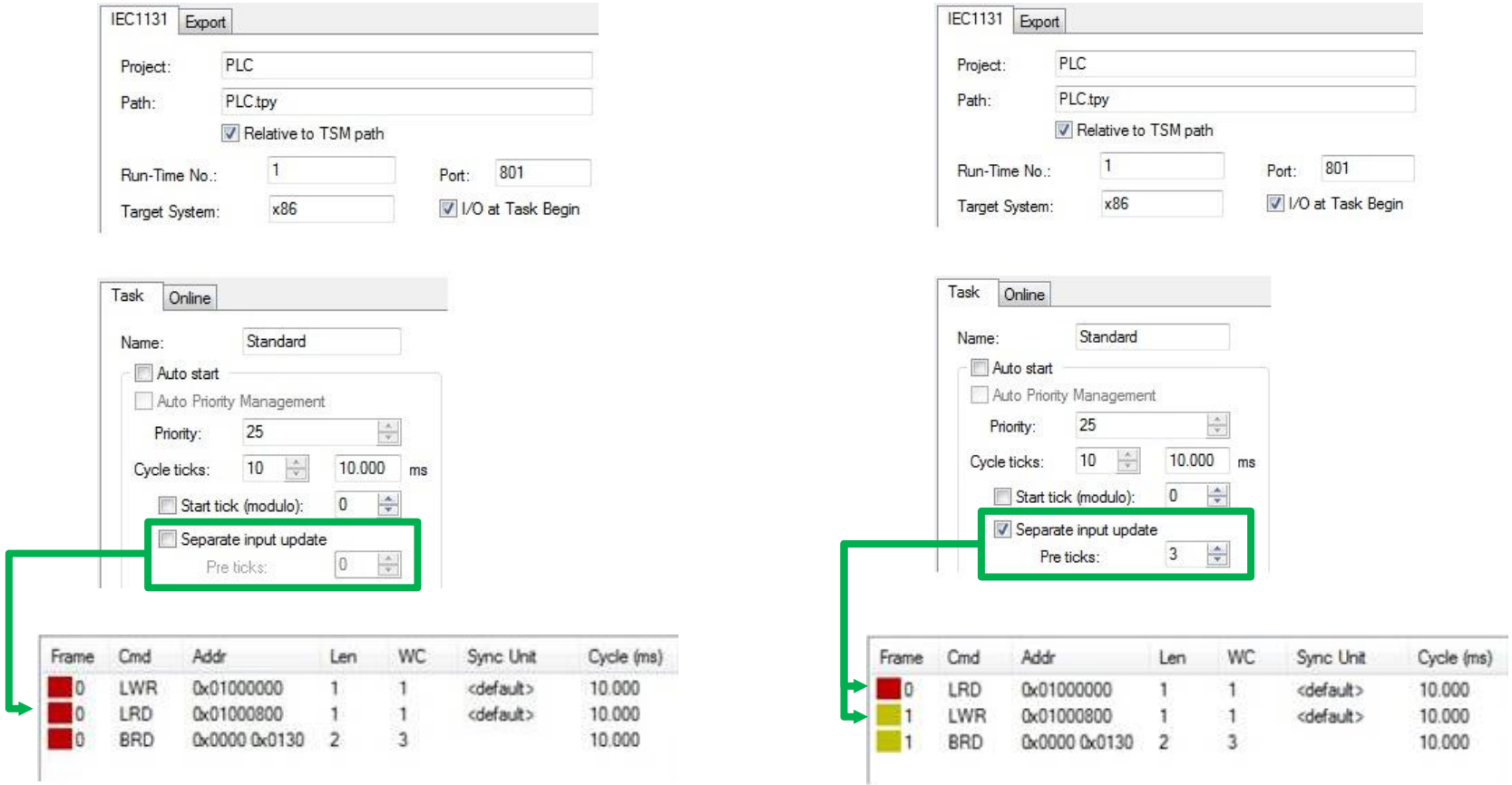
C/C++

TcCallAfterOutputUpdate for C++ modules

Comparable to PLC Attribute TcCallAfterOutputUpdate C++ modules could be called after the output update.
Equivalent to the ITcCyclic Interface: Please make use of the ITcPostCyclic Interface

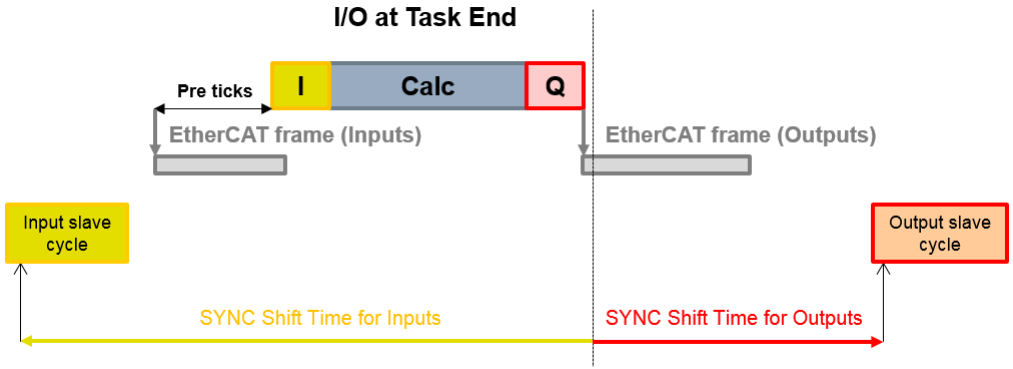
TwinCAT 设置 – 独立的输入刷新

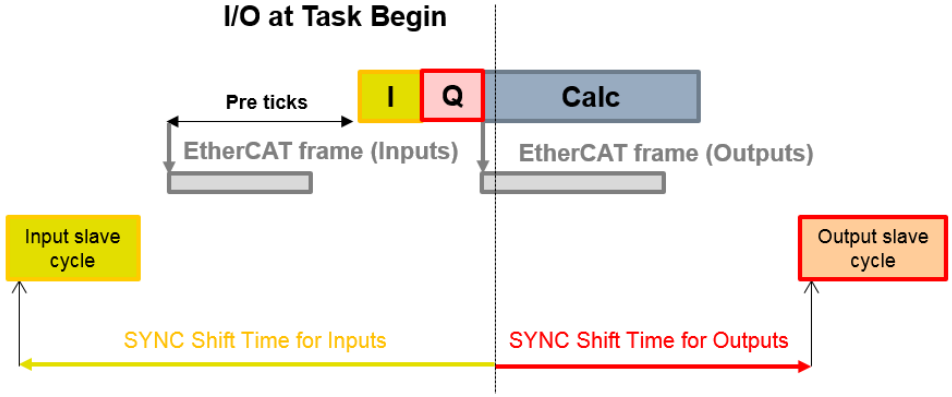
TwinCAT 提供一特别的选项，为输入信号发送一个独立的数据帧来收集输入信号。如果选中了“**Separate input update**”选项，EtherCAT主站就会配置一个独立的数据帧来读取输入过程数据 (这个选项可用于减少 input-output 响应时间)：



深入了解 – 独立的输入刷新和 I/O 周期

Pre ticks 参数是 TwinCAT Base Time 的整数倍，它代表 发送输入数据帧的时间 相比于软件任务开始时间 的提前量。





为了更精细地设置 Pre ticks，可以减小TwinCAT Base Time。

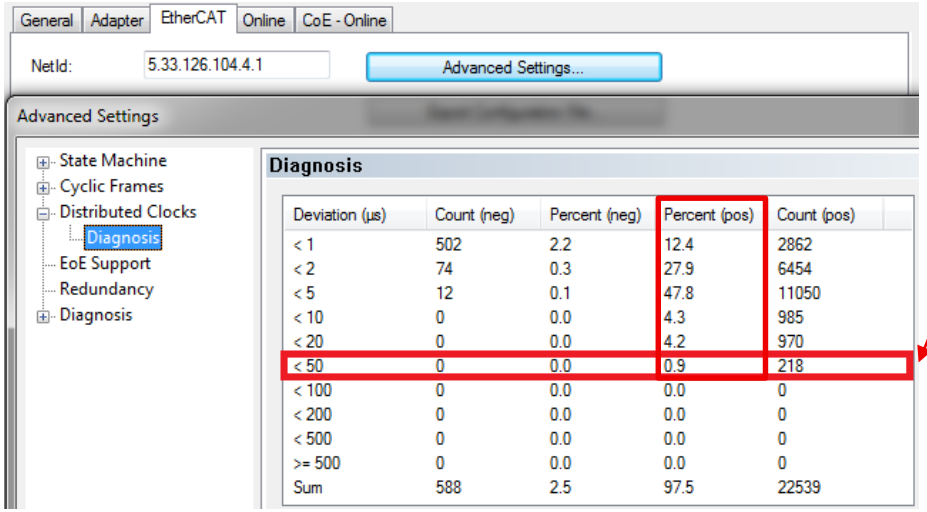
“SYNC Shift for Outputs” 的估算实例

当需要对 “SYNC Shift for Outputs” 进行优化时，可以通过以下分量的**代数累加**估算其最小时间

+ 从站内部电路引起的硬件延时 (从参考时钟 **Reference Clock** 开始):
  - 1 µs * MII 端口的从站数量 + 0.3 µs * EBUS 接口的从站数量

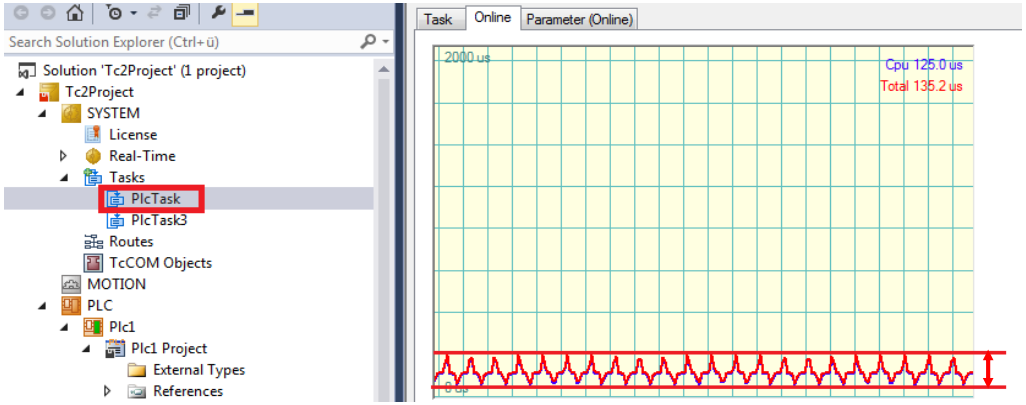
+ 电缆引起的硬件延时(从参考时钟 **Reference Clock** 开始):
  - 5.3 ns/m * 网络中的铜缆长度 [单位 m]

+ 主站的Jitter 最大值 (正偏差):

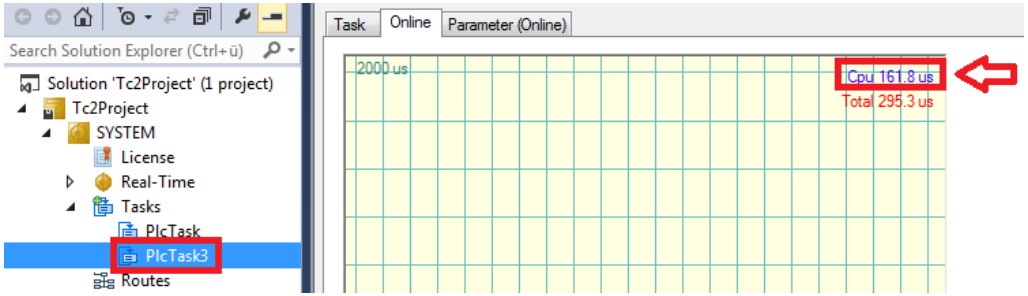


“SYNC Shift for Outputs” 的估算实例

+ 最高优先级的软件任务的执行时间的波动幅值 jitter (仅当设置为 “I/O at Task End”).



+ 多任务配置时， Σ 各个任务的执行时间求和，连接到DC同步从站的第2个到最后一个任务的执行时间求和。 (第1个任务的循环周期也应当包含在总和中，除非第1个任务配置为 “I/O at Task Begin”)

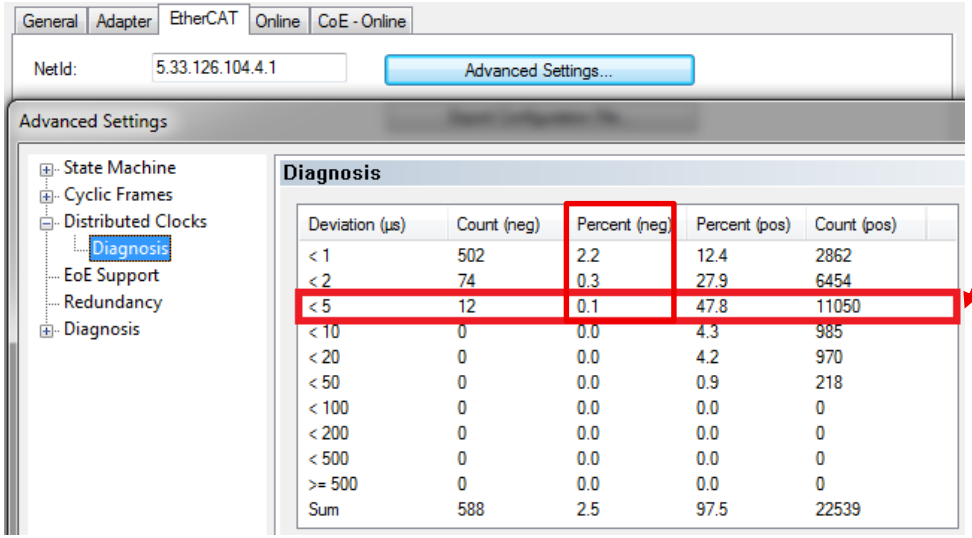


+ 一定的公差余量…

“SYNC Shift for Inputs” 的估算实例

必要时，可以通过以下分量的**代数累加**估算“SYNC Shift for Inputs” 的最小时间：

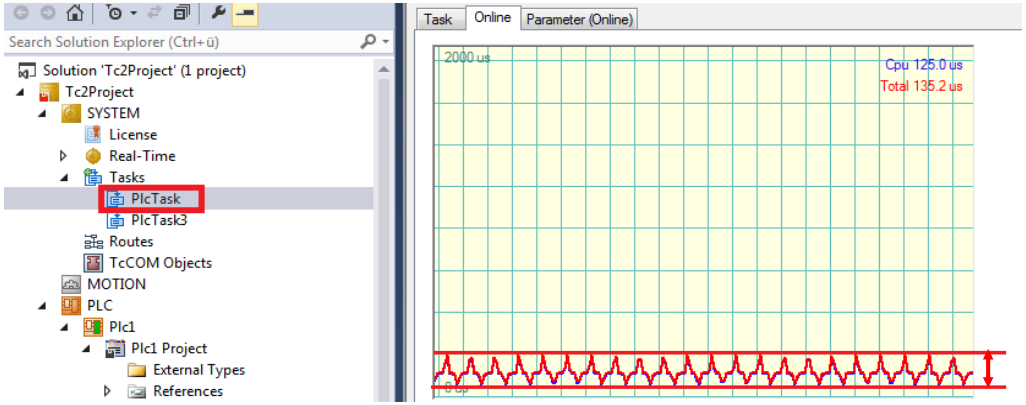
- 主站通讯的最大 jitter (负偏差):



- 对于工作在 “Based on Input Reference” 模式下的那些从站，它们的 minimum cycle times (0x1C33:05) 的最大值
- Pre ticks 对应的时间 (仅当激活了 “Separate input update” 模式).

“SYNC Shift for Inputs” 的估算实例

- 最高优先级的软件任务的执行时间的波动幅值 jitter (仅当设置为 “I/O at Task End”).

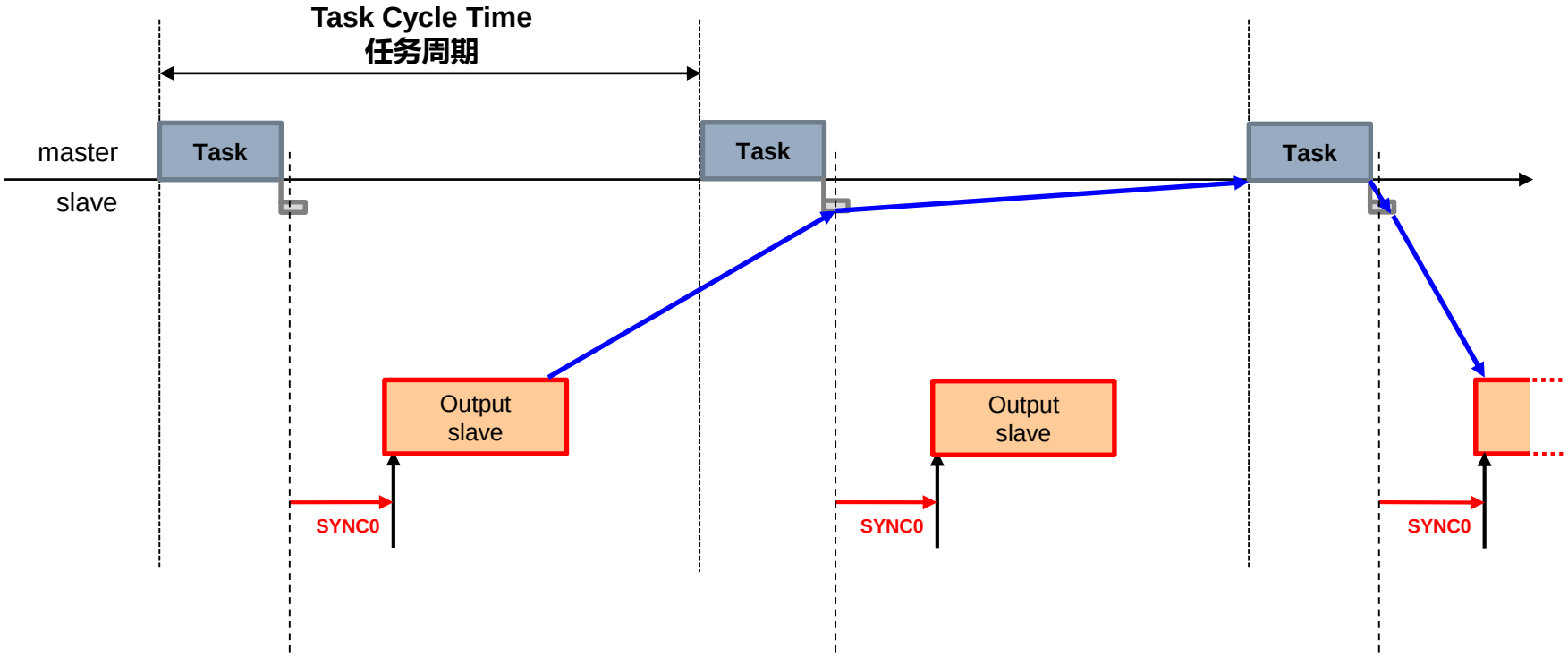


- 一定公差余量…

深入了解 – 时序示例 1

本例 基于以下设置：

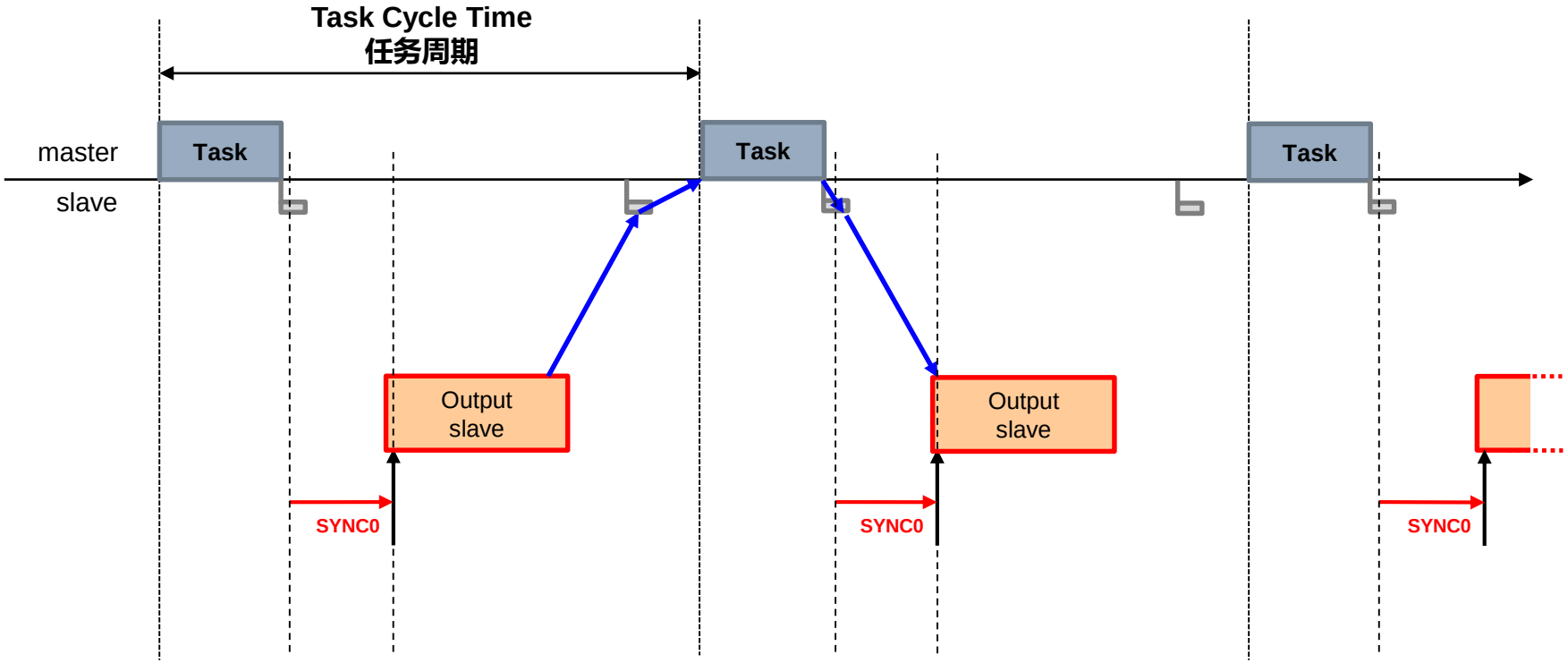
- 任务设置为 “**I/O at Task End**”
- 读取数据和写入数据到 **同一个**Input and Output 从站



深入了解 – 时序示例 2

本例 基于以下设置：

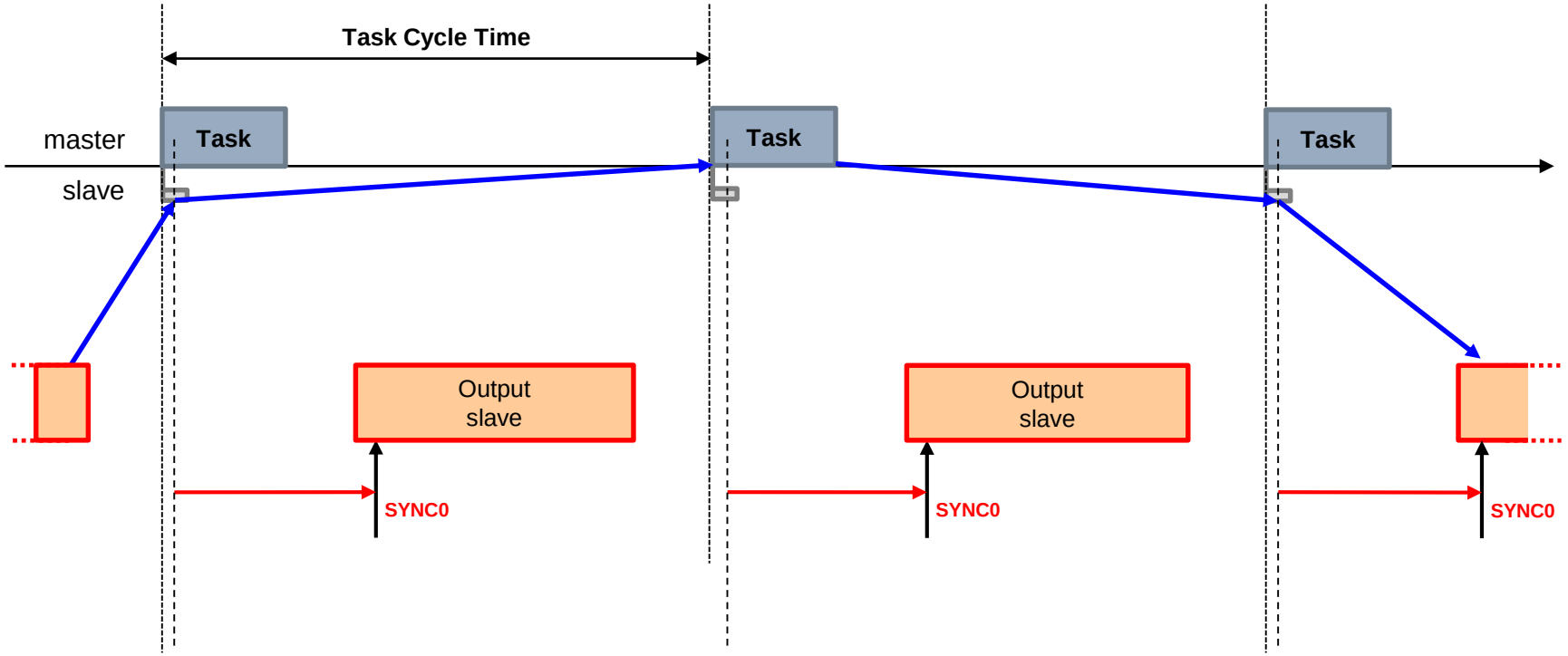
- 任务设置为 “**I/O at Task End**” 以及 “**Separate Input Update**”
- 读取数据和写入数据到 **同一个**Input and Output 从站



深入了解 – 时序示例 3

本例基于以下设置：

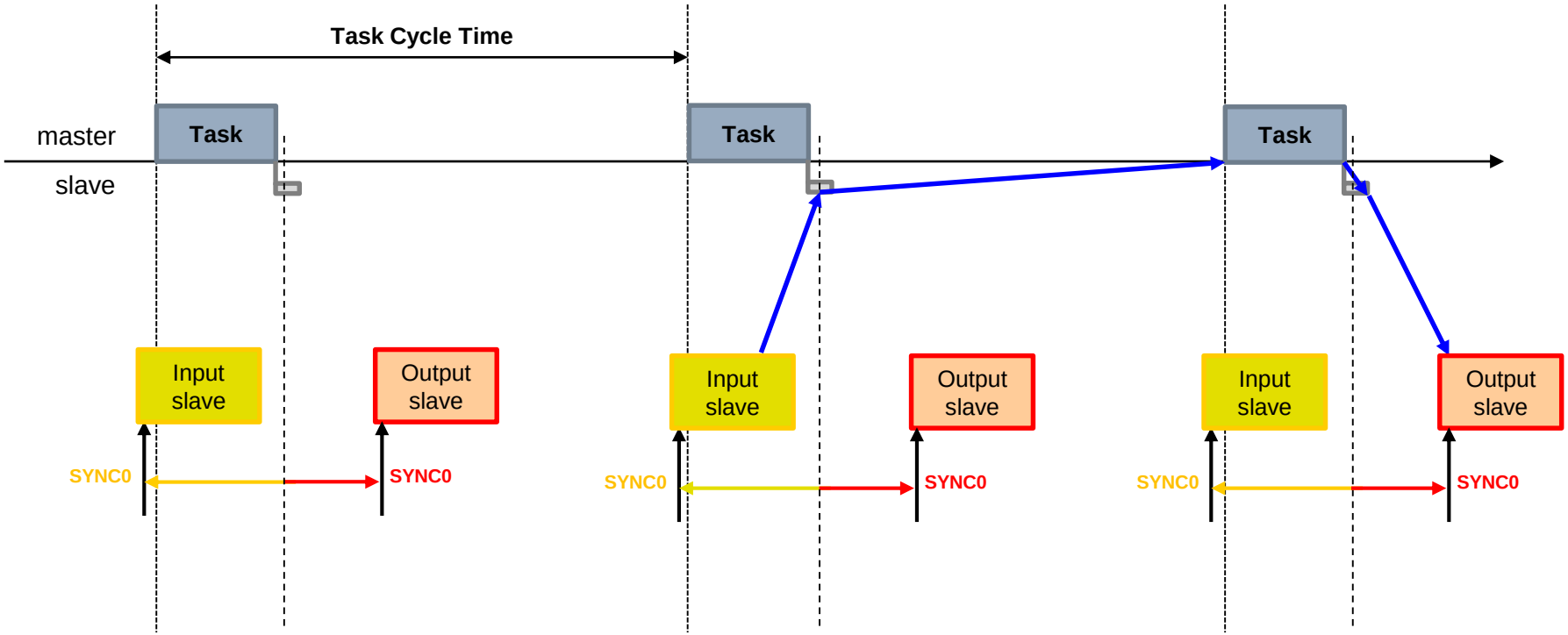
- 任务设置为 “**I/O at Task Begin**”
- 读取数据和写入数据到 **同一个**Input and Output 从站



深入了解 – 时序示例 4

本例基于以下设置：

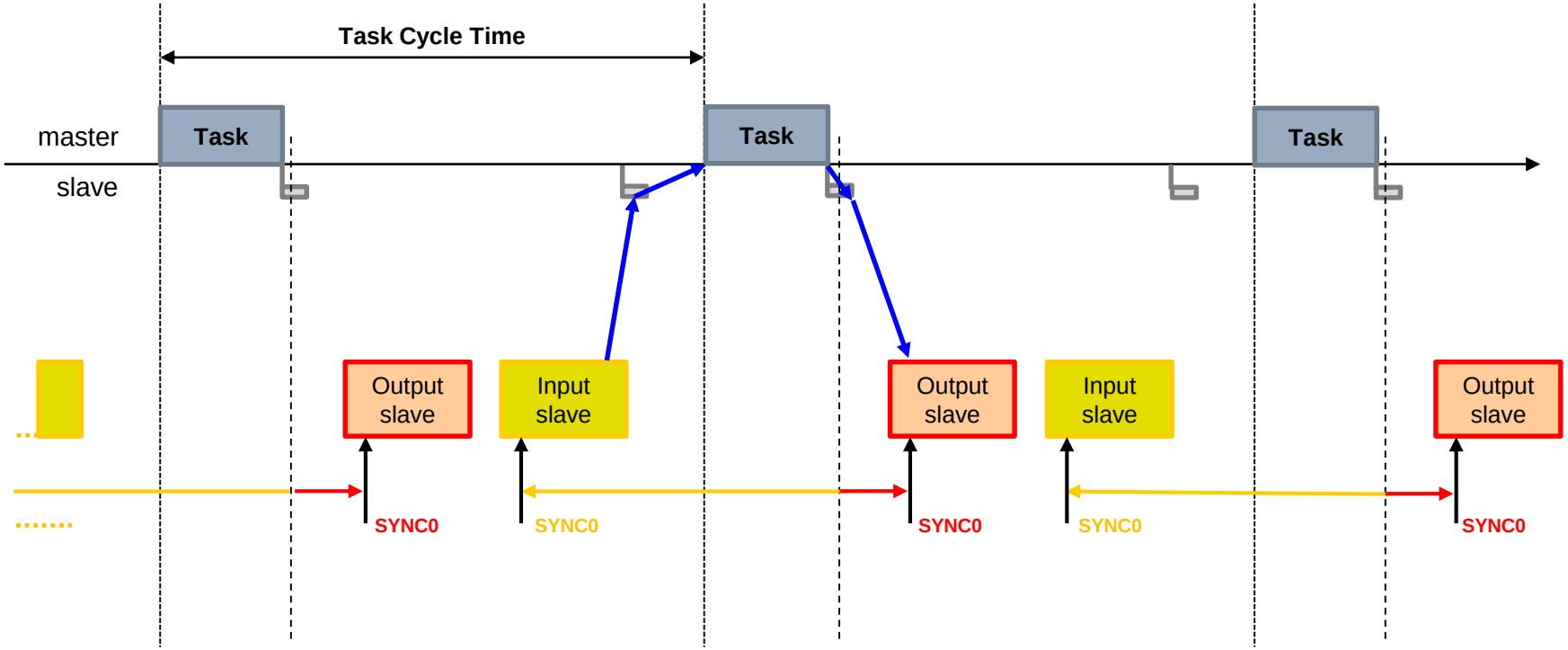
- 任务配置为 “I/O at Task End”
- 从 Input 从站读取数据，写入数据到 Output从站



深入了解 – 时序示例 5

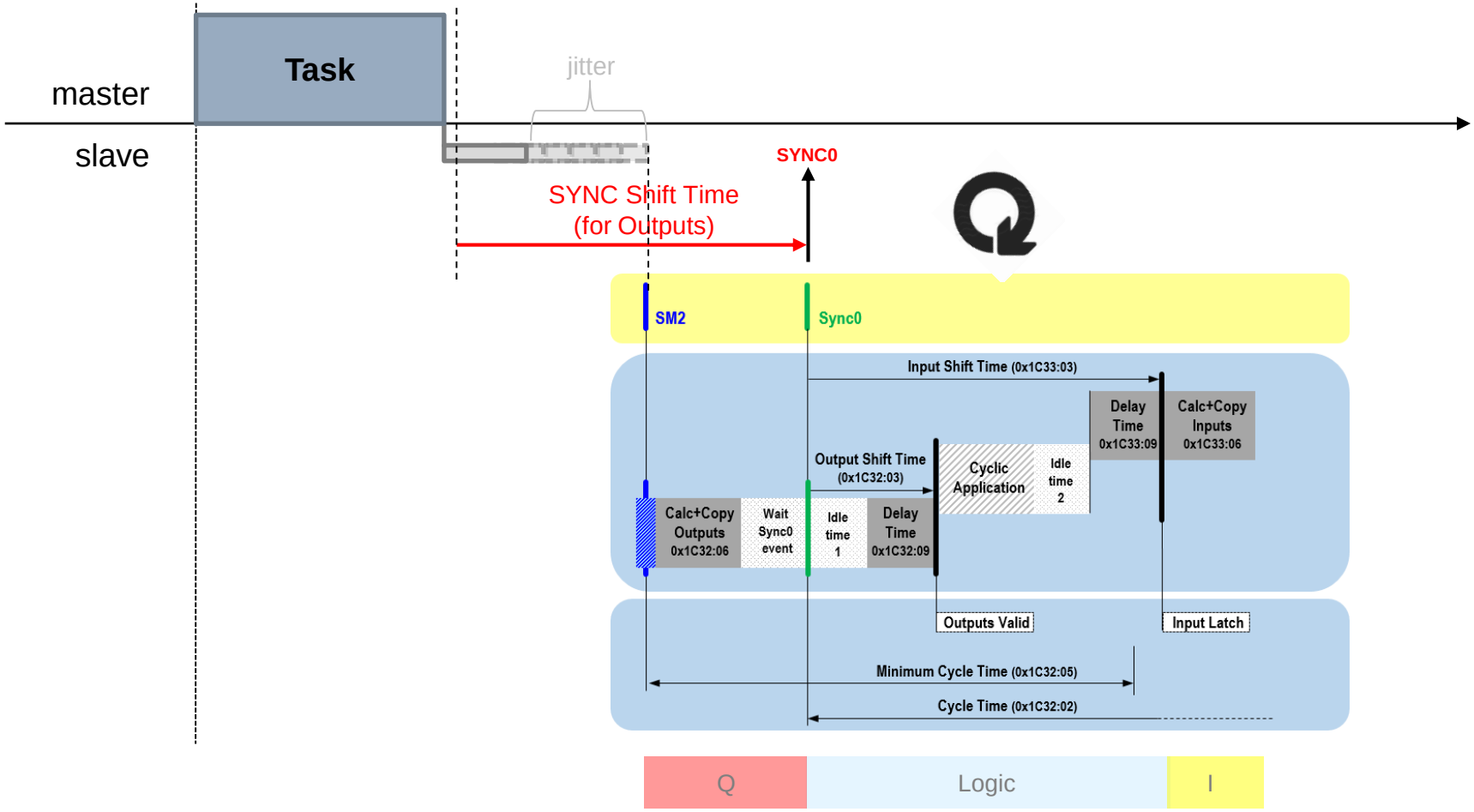
本例基于以下设置：

- 任务配置为 “I/O at Task End” 以及 “Separate Input Update”
- 从 Input 从站读取数据，写入数据到 Output从站



SYNC Shift Time 和 Slave Internal Timings

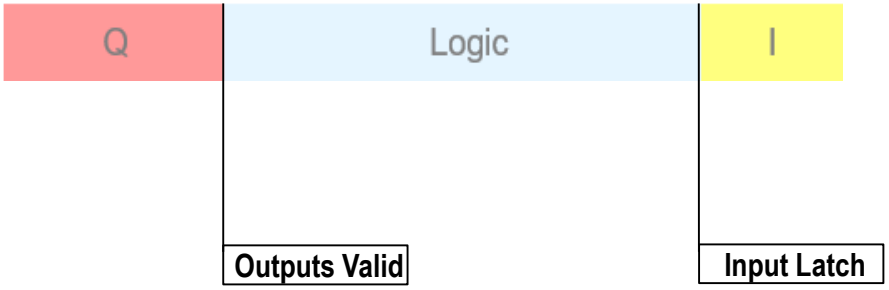
每个 DC-Synchronous 的从站给本地事件（Local Event）定义内部时序（**internal timings**）。本地时序的参考点就是同部事件（SYNC event），通过SYNC Shift Time 依次设定。



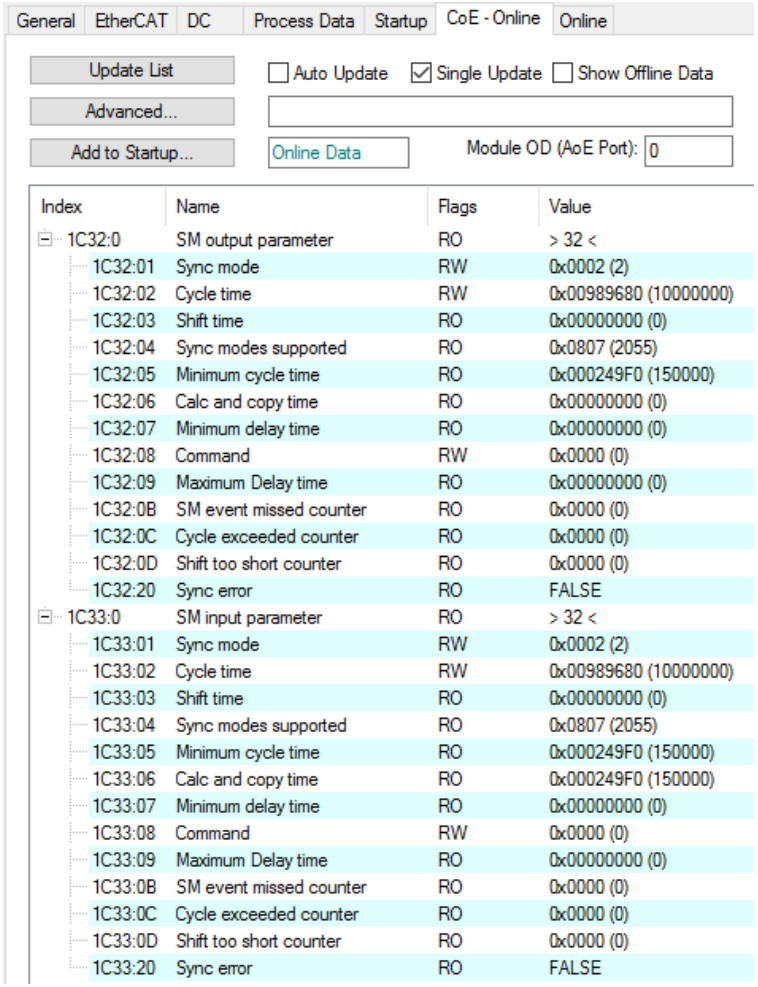
同步对象 0x1C32/33

需要特别说明的是，在从站内部循环里面最重要的两个的本地事件：

- **Output Valid（输出生效）**：从站**设置**现场物理输出信号的**时刻**，输出数据接收自主站上一个通讯周期发来的数据帧。
- **Input Latch（输入锁存）**：从站**采集**现场物理输入信号的**时刻**，输入数据将在下一个通讯周期的数据发送到主站。



从站通过通过标准的CoE对象**0x1C32**和 **0x1C33**描述内部时序。其中**0x1C32**描述Output，**0x1C33**描述Input。



Index	Name	Flags	Value
1C32:0	SM output parameter	RO	> 32 <
1C32:01	Sync mode	RW	0x0002 (2)
1C32:02	Cycle time	RW	0x00989680 (10000000)
1C32:03	Shift time	RO	0x00000000 (0)
1C32:04	Sync modes supported	RO	0x0807 (2055)
1C32:05	Minimum cycle time	RO	0x000249F0 (150000)
1C32:06	Calc and copy time	RO	0x00000000 (0)
1C32:07	Minimum delay time	RO	0x00000000 (0)
1C32:08	Command	RW	0x0000 (0)
1C32:09	Maximum Delay time	RO	0x00000000 (0)
1C32:0B	SM event missed counter	RO	0x0000 (0)
1C32:0C	Cycle exceeded counter	RO	0x0000 (0)
1C32:0D	Shift too short counter	RO	0x0000 (0)
1C32:20	Sync error	RO	FALSE
1C33:0	SM input parameter	RO	> 32 <
1C33:01	Sync mode	RW	0x0002 (2)
1C33:02	Cycle time	RW	0x00989680 (10000000)
1C33:03	Shift time	RO	0x00000000 (0)
1C33:04	Sync modes supported	RO	0x0807 (2055)
1C33:05	Minimum cycle time	RO	0x000249F0 (150000)
1C33:06	Calc and copy time	RO	0x000249F0 (150000)
1C33:07	Minimum delay time	RO	0x00000000 (0)
1C33:08	Command	RW	0x0000 (0)
1C33:09	Maximum Delay time	RO	0x00000000 (0)
1C33:0B	SM event missed counter	RO	0x0000 (0)
1C33:0C	Cycle exceeded counter	RO	0x0000 (0)
1C33:0D	Shift too short counter	RO	0x0000 (0)
1C33:20	Sync error	RO	FALSE

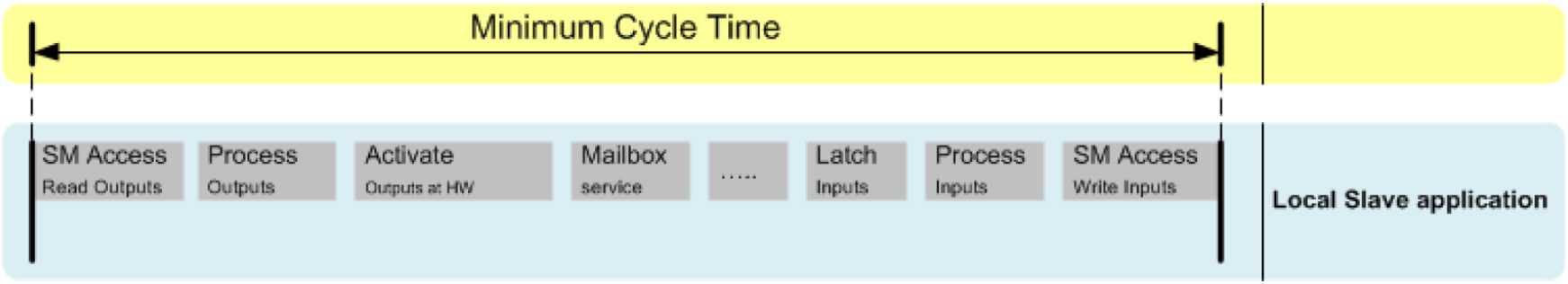
同步对象 0x1C32/33

同步对象 0x1C32/33 中**影响最大的几个SubIndex** 是 (按逻辑顺序，而不是执行顺序排列) ：

Subindex	Name	向主站/用户提供什么信息?	主站/用户 可以怎样设置?
05	Minimum Cycle Time 最小循环时间	要保证从站完成正确操作，最少需要多长的通讯周期？	
03	Shift Time 偏移时间	从站在**主**触发事件和硬件刷新（Output Valid/Input Latch）之间，是否进行**软件偏移**？	如果该数可写，主站/用户就可以修改**主**触发事件和物理硬件刷新之间的软件偏移
08	Get Cycle Time 获得循环时间		如果支持动态的循环时间（见 Subindex :04), 置 1 就会触发从站内部定时的 动态测量 (类似 Minimum Cycle Time).
04	Synchronization Types Supported 支持的同步类型	从站的硬件刷新（Output Valid/Input Latch）支持哪种同步模式和偏移选项（synchronization modes and shifting options）	

最小循环时间 (0x1C32/33:05)

每个从站为了完成所有内部任务需要的最小循环时间，包括过程数据的交换，邮箱通讯，状态机处理，应用相关的功能等等， 最小循环时间取决于多个因素，比如应用的复杂程度以及要交换的过程数据量：

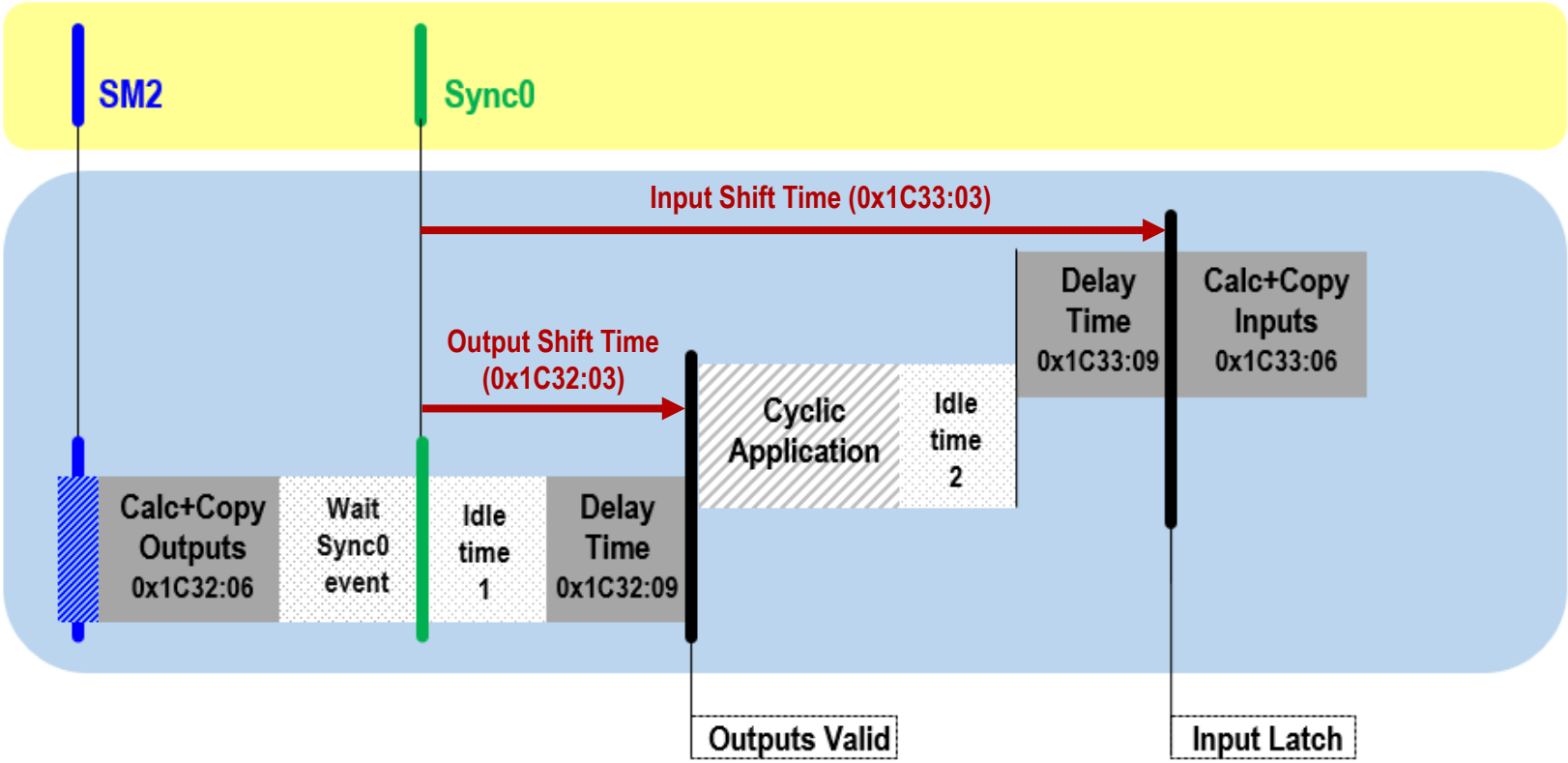


对于链接到 DC-Synchronous 从站的任务，应该确保 主站通讯周期（Communication Cycle Time ）≥（从站最小循环时间）Minimum Cycle Time，否则从站应用就可能跟不上主站通讯的节拍。

如果为了确保主站通讯周期大于所有从站的最小循环周期，不得不严重限制某些DC及非DC从站的通讯周期，以至于需要快速响应的任务不能执行，建议使用周期不同的多个任务，并把每个从站链接到适当周期的任务。

输入/输出偏移时间（Output/Input Shift Time ）(0x1C32/33:03)

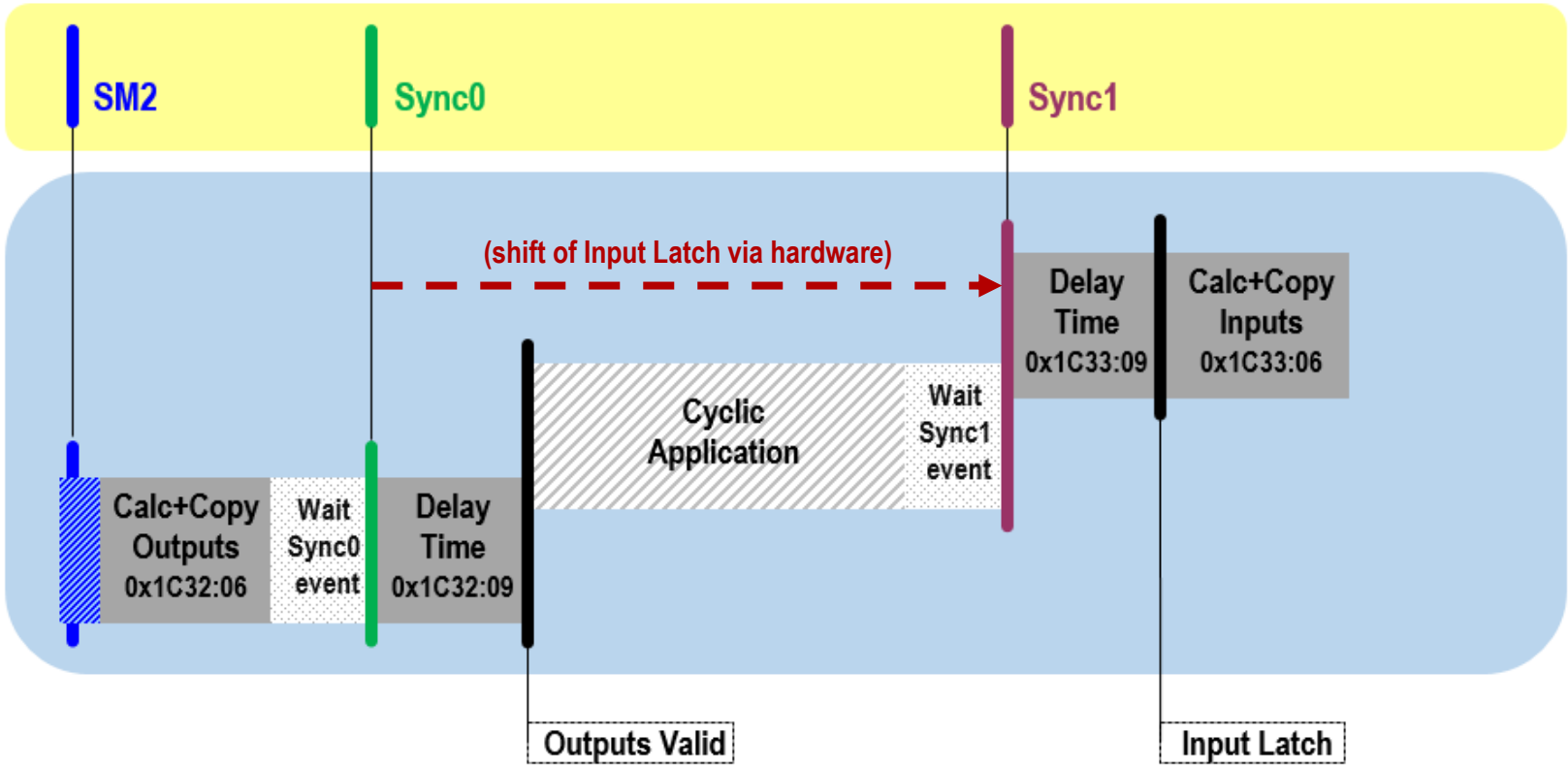
从站可以选择支持**硬件刷新（** Output Valid and/or Input Latch **）**的软件**偏移时间（Shift Time）**



如果这个SubIndex可写，主站/用户就可以在从站周期的范围内对硬件刷新事件（Output Valid and/or Input Latch）的触发时间进行微调。

深入了解 – Input Latch 的硬件偏移

As alternative to the “software” shift through Object 0x1C33:03,
除了通过0x1C33:03 进行软件偏移之外，另一种让从站实现I/O偏移的办法是：使用SYNC1中断，在硬件层面调节 Input Latch 事件的触发时间 。

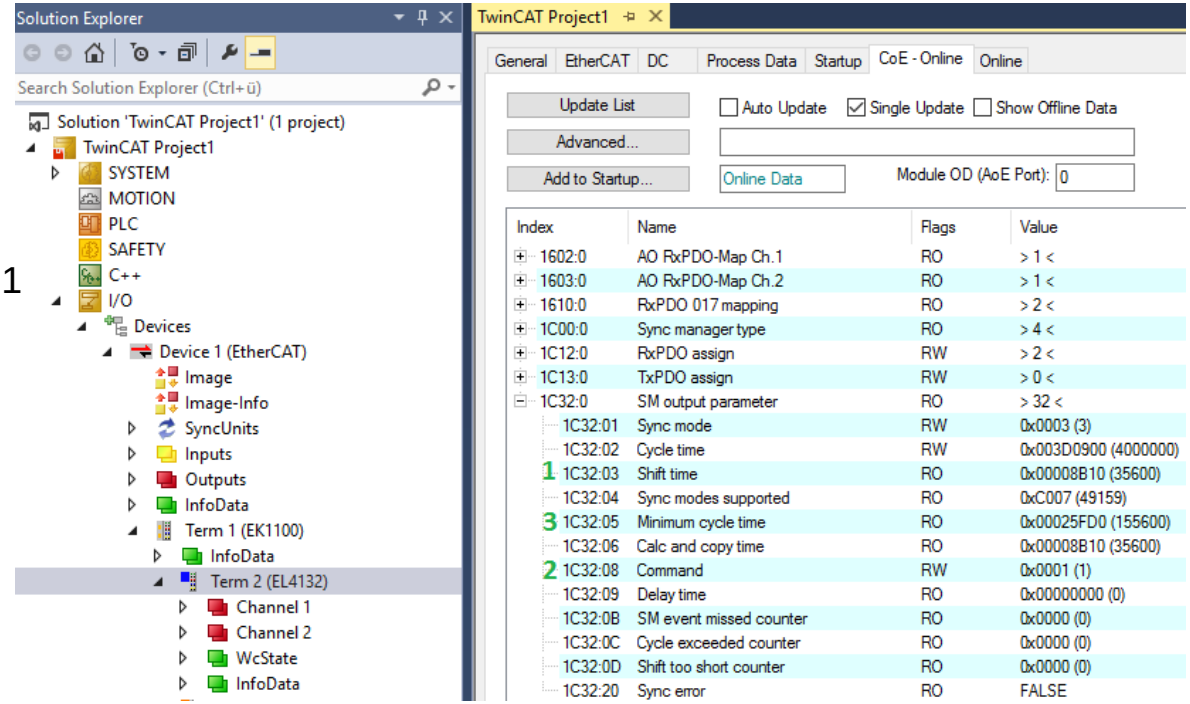


获取循环周期 (0x1C32/33:08)

有的从站可以动态计算它们的内部定时（internal timings）(比如，最小周期 Minimum Cycle Time)，根据当前配置 (此时, 0x1C32/33:04 的Bit 14 会置 True).

命令 (0x1C32/33:08) 置 1 ，就会触发这个计算过程。

1. 检查是否 0x1C32/33:04 的Bit 14 = 1
2. 设置 0x1C32/33:08 = 1
3. 计时 0x1C32/33:05+06+09



对于支持动态周期的从站，推荐在调试阶段触发计算过程，以便获得从站实际的计时。

支持的同步模式 (0x1C32/33:04)

从站支持的同步模式和偏移选项汇总显示在0x1C32/33:04的子索引**Synchronization Types supported** 中：

0x1C32 (outputs)

Synchronization Types supported	UNSIGNED16	Bit 0: Free Run supported Bit 1: Synchronous supported Bit 4:2 : DC Type supported: 000 = No DC 001 = DC Sync0 010 = DC Sync1 100 = Subordinated Application with fixed Sync0
	1	
	2	Bit 6:5: Shift Settings 00 = No Output Shift supported 01 = Output Shift with local timer (Shift Time) 10 = Output Shift with Sync1
		Bit 9…6: Reserved for future use Bit 10: Delay Times should be measured (because they depend on the configuration) Bit 11: Delay Time is fix (synchronization is done by hardware) Bit 13…11: Reserved for future use
	3	Bit 14: Dynamic Cycle Times Times described in 0x1C32 are variable (depend on the actual configuration) This is used for e.g. EtherCAT gateway devices. The slave shall support cycle time measurement in OP state. The cycle time measurement is started by writing 1 to 0x1C32:08. If this bit is set, the default values of the times to be measured (Minimum Cycle Time, Calc And Copy Time, Delay Time) could be 0. The default values could be set in INIT and PREOP state. Bit 14 should only be set, if the slave cannot calculate the cycle time after receiving all Start-up SDO in transition PS.
		Bit 15: Reserved for future use

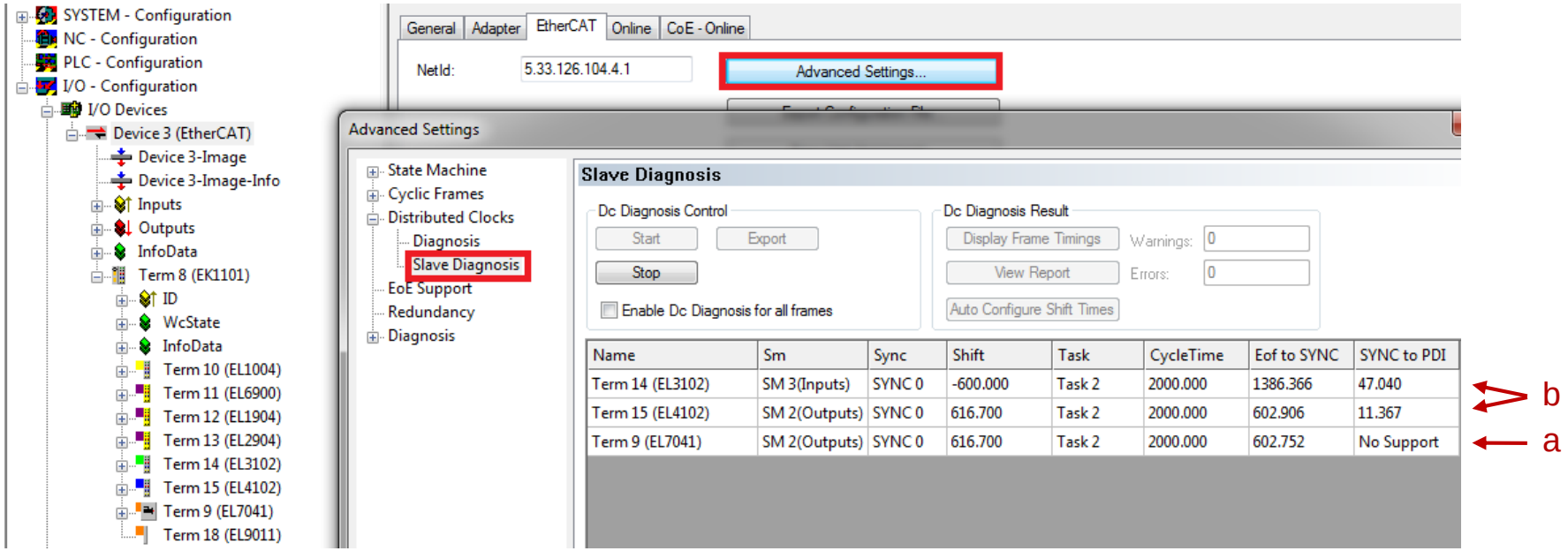
0x1C33 (inputs)

Synchronization Types supported	UNSIGNED16	Bit 0: Free Run supported Bit 1: Synchronous supported Bit 4:2 : DC Type supported: 000 = No DC 001 = DC Sync0 010 = DC Sync1 100 = Subordinated Application with fixed Sync0
	1	
	2	Bit 6:5: Shift Settings 00 = No Input Shift supported 01 = Input Shift with local timer (Shift Time) 10 = Input Shift with Sync1
		Bit 13:6: Reserved for future use
	3	Bit 14: Dynamic Cycle Times Same meaning as in 0x1C32:04
		Bit 15: Reserved for future use

1. 从站应用的主触发事件（标识 Synchronization Mode ）.
2. Shifting options 用于硬件刷新
3. 支持内部计时的动态计算。

时序诊断 (主站)

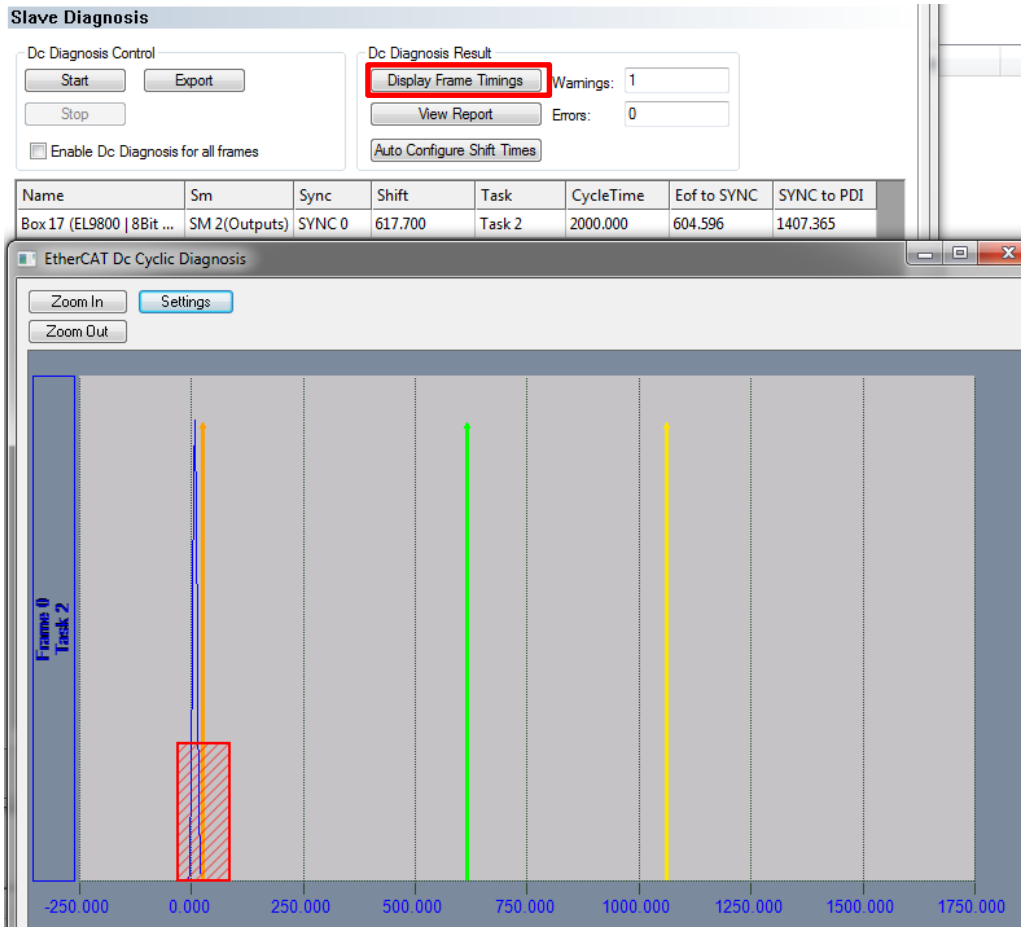
在 TwinCAT 中可以在图形化的界面 **监视**所有DC-Synchronous 从站的**时间**，以确认设置是否正确。 (e.g. 正确的同步偏移时间 SYNC Shift Time):



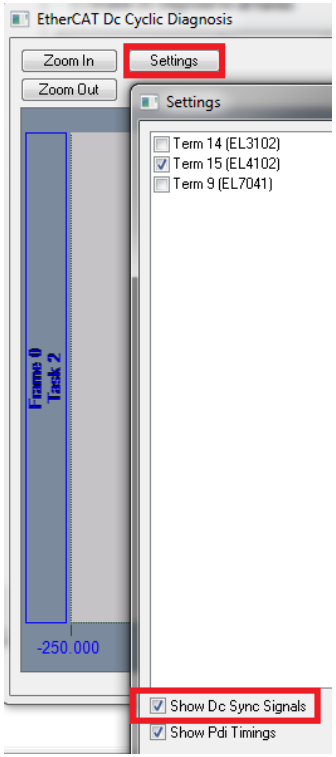
a. 在“SYNC to PDI”列报告 “No Support” 的从站，只能监视 周期性数据帧和同步中断之间的时间关系。

b. 其它从站还能监视 同步中断和 PDI（过程数据接口）访问之间的内部计时。

时序诊断 (主站)

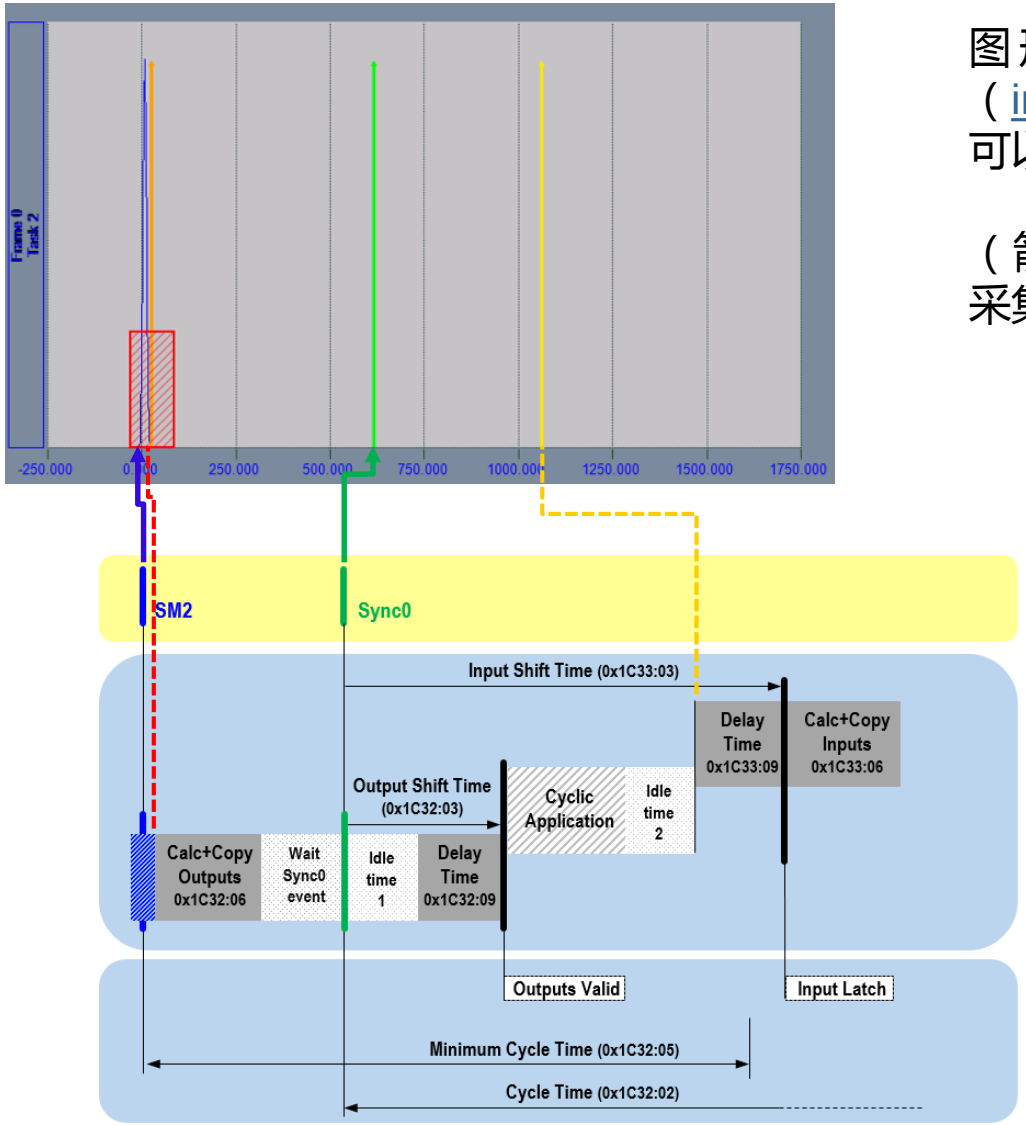


SYNC interrupts的可视化界面需要手动启用：



- 接收数据帧 (with jitter)
- µC 从ESC读取主站的 outputs 数据
- 产生同步中断 SYNC interrupt
- µC 向ESC写入主站的Input数据

时序诊断 (主站)



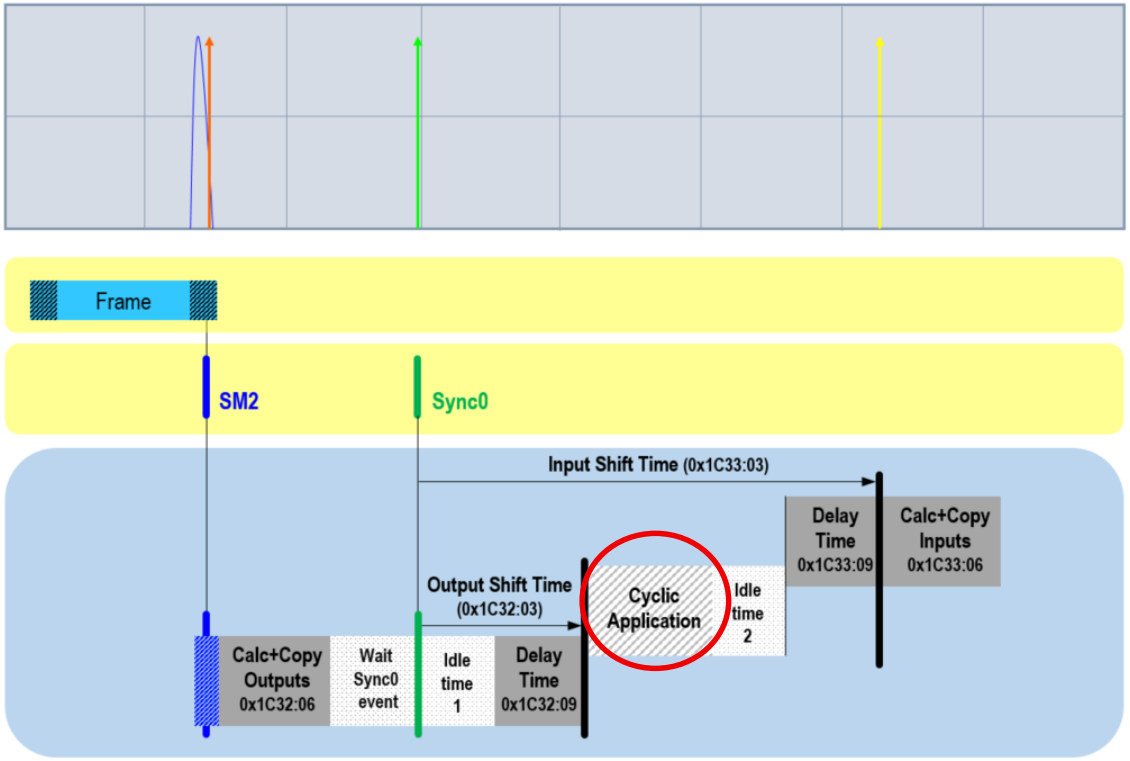
图形化描述与从站设备的内部时序图（internal timing schemes）相对应，并且可以根据后者进行解释。

（箭头并不严格对应输出刷新事件和输入采集事件，但与二者紧密相关）

深入了解 – 不同的时序模式（Timing Patterns）

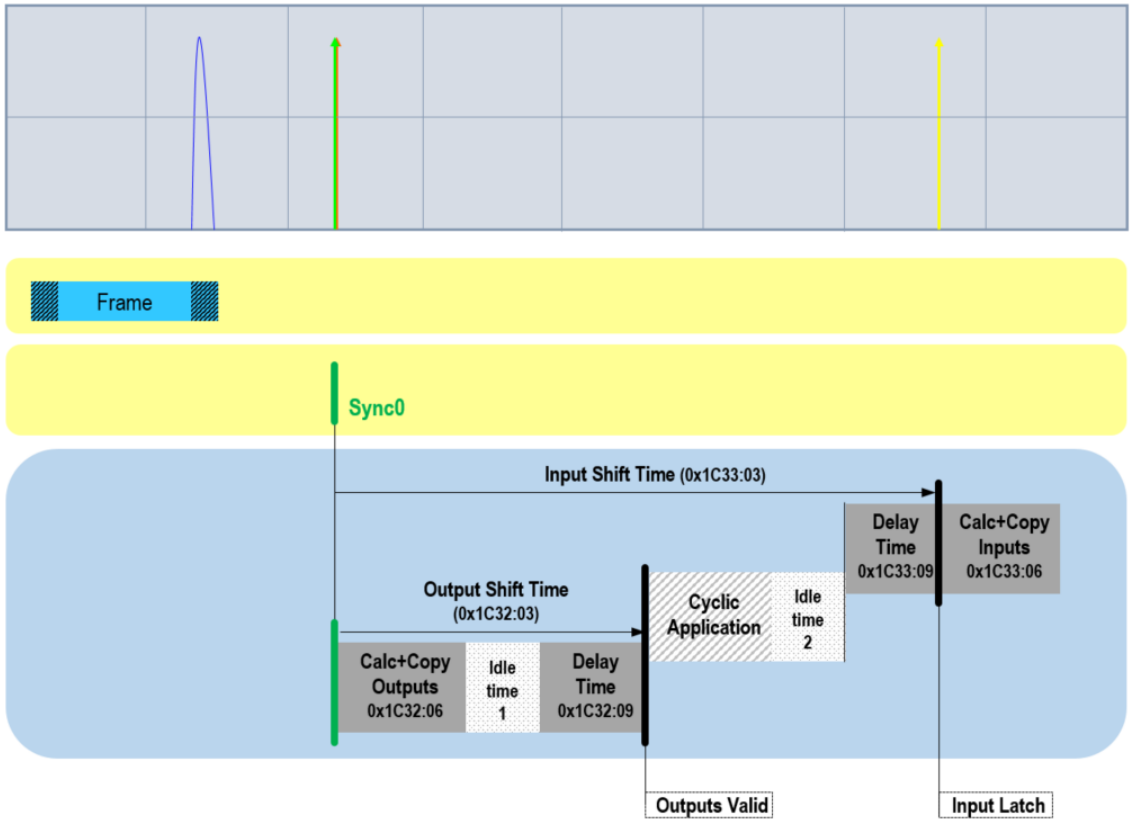
不是所有 DC-Synchronous 从站的内部计时都使用同样的模式（见synchronization modes supported).

有时候，从站在接收到数据帧时就开始从EtherCAT芯片读取周期性输出，以便在SYNC0中断产生的时候可以准备好启动从站的应用。



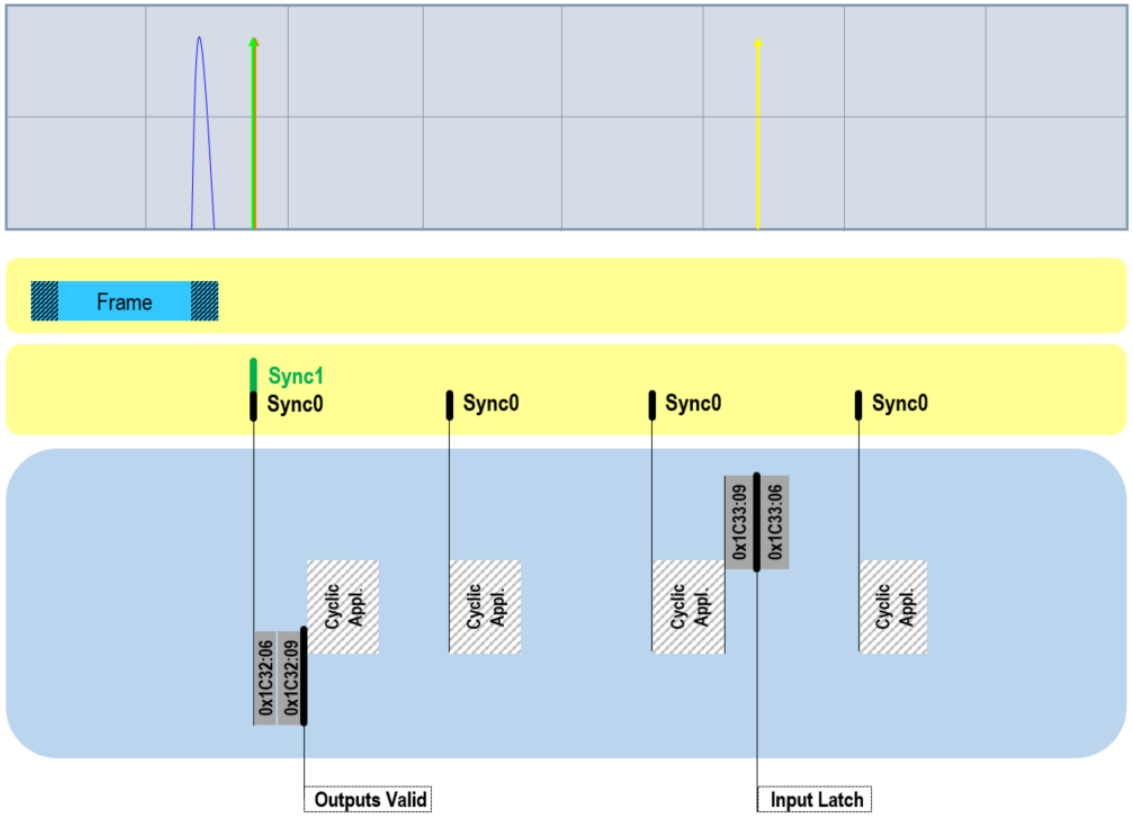
深入了解 – 不同的时序模式（Timing Patterns）

…而其它情况下，数据帧到达从站后，从站要等到SYNC0事件产生时才开始从EtherCAT芯片读取周期性输出（从站的硬件设计中省去了中断线interrupt line），所以绿色箭头（SYNC interrupt）和棕色箭头（从uC读取输出）几乎重叠在一起。



深入了解 – 不同的时序模式（Timing Patterns）

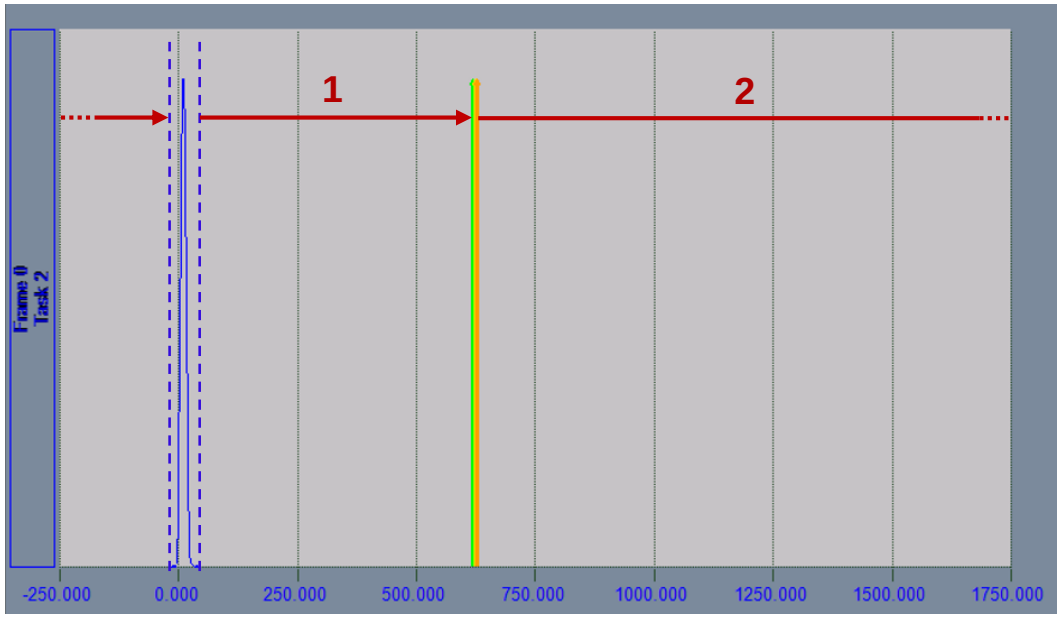
… 还有一些从站（e.g. 大部分伺服驱动器），两个同步信号（SYCN0 和 SYNC1）都启用了，其中SYNC1与通讯周期同步，而SYNC0以更高的频率用于触发内部电流环



时序诊断(主站) 的限制（ Constraints ）

…但是，绝大多数情况下不需要精确了解从站内部时序的模式，也不必进行深入分析。唯一需要确认的重要方面是：

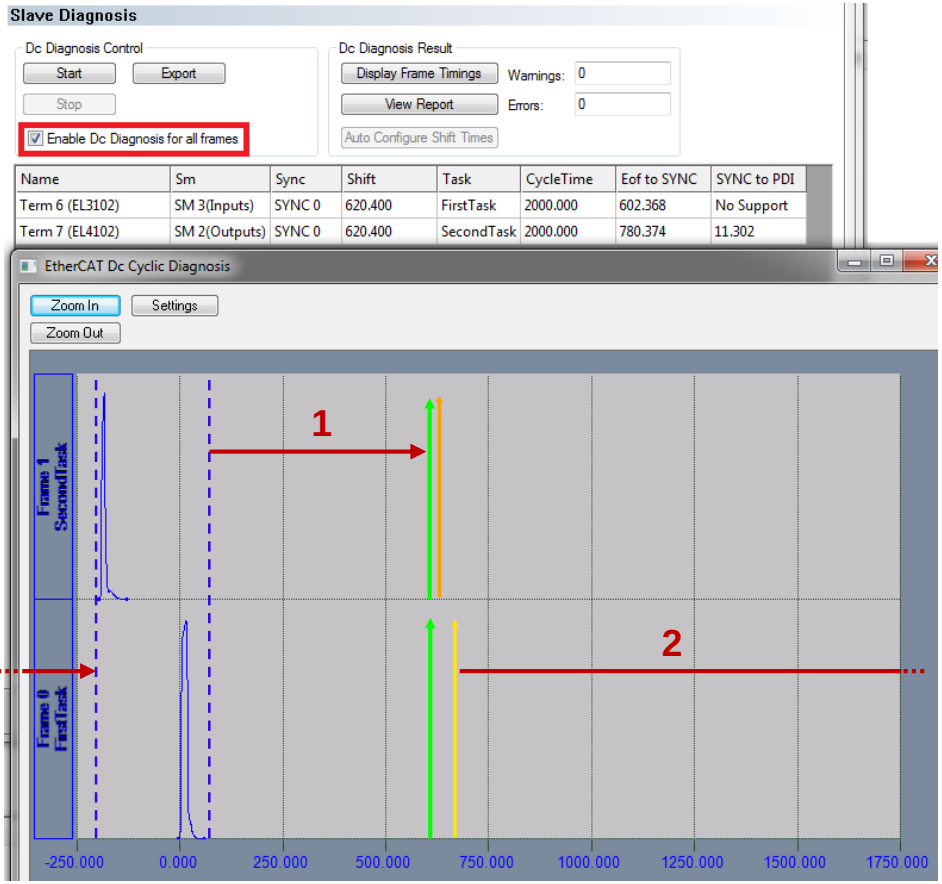
1. 所有DC-Synchronous 从站分别产生同步中断（SYNC interrupt）的时间，都在**接收到**周期性数据帧**之后**。
2. 所有DC-Synchronous 从站执行输入数据的PDI 访问的时间，都在接收到下一个周期性数据帧**之前**.



(时序图的X轴显示范围固定为一个通讯周期，单位us)

时序诊断(主站) 的限制（ Constraints ）

对于包含多个任务的配置（多个DC-Synchronous 从站分别链接到了不同的软件任务），需要检查所有数据帧的时序。



对于包含多个任务的配置:

1. 对于携带ＤＣ从站数据的**最后１个**周期性数据帧（对应多任务配置中的优先级最**低**的任务），应当确保所有从站都在接收到数据帧**之后**产生同步信号

2. 对于携带ＤＣ从站数据的**第１个**周期性数据帧（对应多任务配置中的优先级最**高**的任务）， 应当确保所有从站都在接收到下一个数据帧**之前**完成PDI的输入读取。

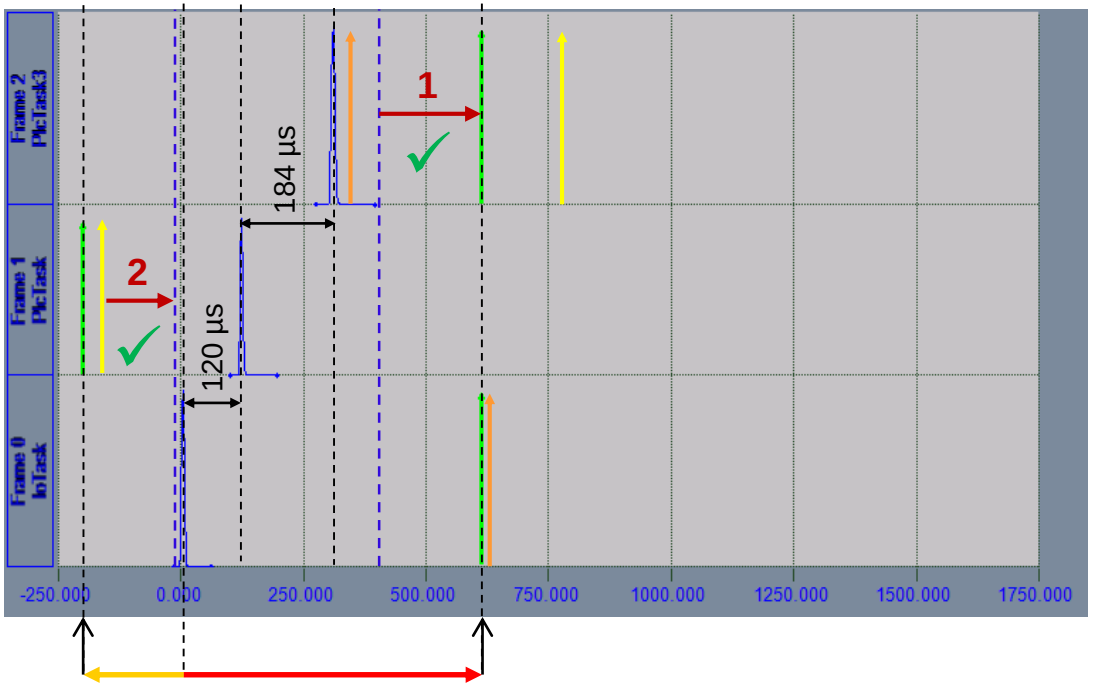
WARNING. 目前还不支持“Separate Input Update”的正确时间显示 (其特殊之处在于，显示获取周期性输入的数据帧时，Pre ticks 没有计算在内).

时序诊断示例 (主站)

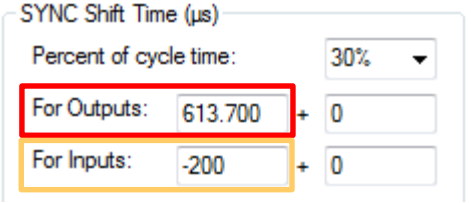
以具有3个任务的配置为例：

- Task 0 (**IoTask**): 周期 2 ms, 优先级19, I/O at Task end, 实际执行时间 16 µs, 链接到 **output-only** 从站
- Task 1 (**PlcTask**):周期 2 ms,优先级20, I/O at Task end,实际执行时间 120 µs, 链接到 **input-only** 从站
- Task 2 (**PlcTask3**):周期 2 ms,优先级21, I/O at Task end,实际执行时间 184 µs, 链接到 **output+input** 从站

Frame	Cmd	Addr	Len	WC	Sync Unit	Cycle (ms)
0	NOP	0x0000 0x0900	4			2.000
0	ARMW	0x0000 0x0910	4			2.000
0	LRD	0x09000000	1			2.000
0	LRW	0x01000000	8	3	<default>	2.000
1	ARMW	0x0000 0x0910	4			2.000
1	LRD	0x02000000	8	1	<default>	2.000
2	ARMW	0x0000 0x0910	4			2.000
2	LRW	0x03000000	16	9	<default>	2.000
2	LWR	0x03000800	4	1	<default>	2.000
2	LRD	0x03001000	3	2	<default>	2.000
2	BRD	0x0000 0x0130	2	8		2.000

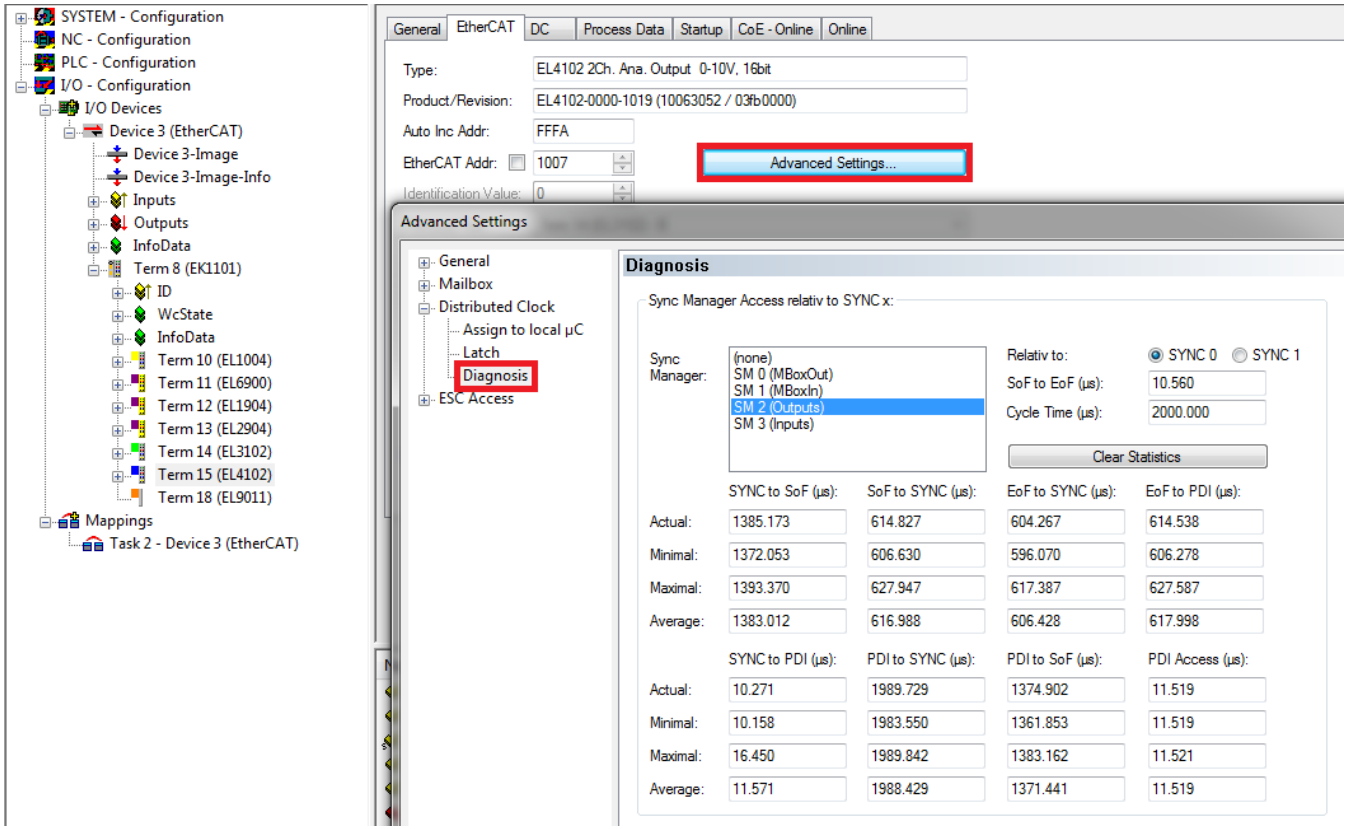


TC3 中 SYNC Shift Time 的标准设置



深入了解 – 时序诊断 Diagnosis (从站)

在从站的 DC diagnosis 界面，可以显示更多的内部计时。
(如果在master DC diagnosis中从站的“SYNC to PDI ”列显示 “No Support”，则本功能不可用)



要使用本功能，要求 Admin Mode (TC2) 或者 SysmanSuperUser (TC3) 权限

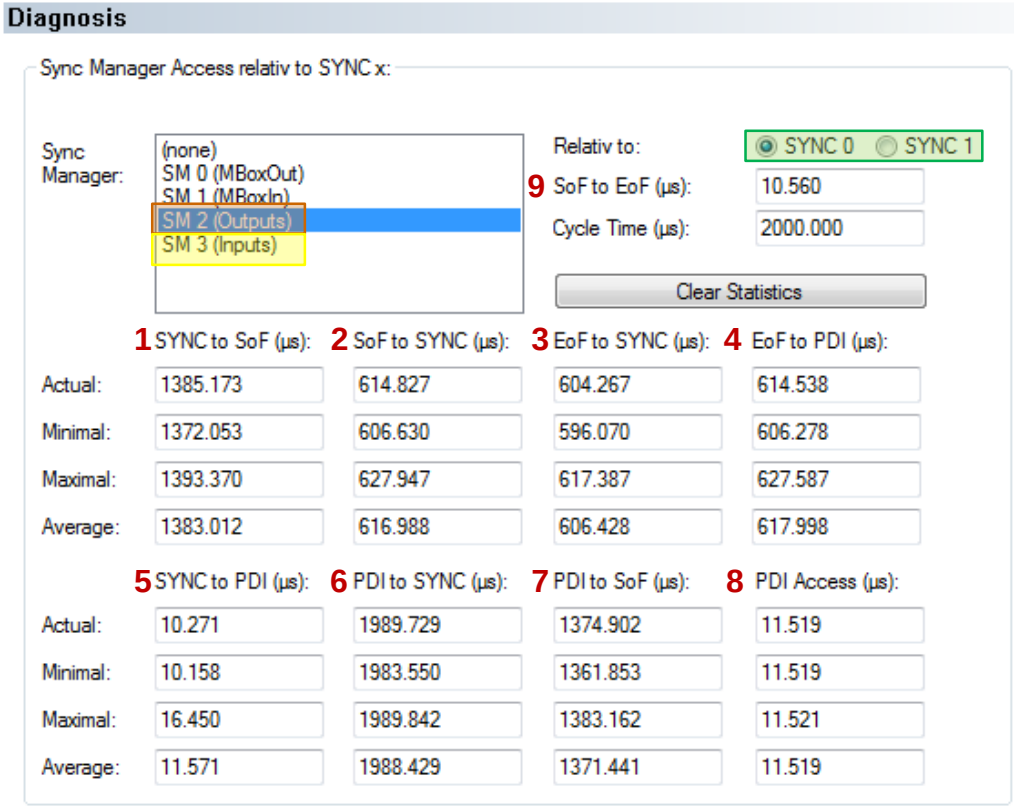
时序诊断 (从站)

- **SoF** (Start of Frame) : Frame 的第1个Bit 到达 ESC
- **EoF** (End of Frame) : Frame 的最后1个Bit 离开 ESC
- **SYNC** : DC单元根据系统时间触发的同步事件
- **PDI** : 本地主控制器访问 ESC的双口内存（ DPRAM ）

选择 **SM 2**, TwinCAT 显示PDI输出访问所对应的时序 (µC 从ESC读取来自主站的Output数据).

选择 **SM 3**, TwinCAT 显示PDI输入访问所对应的时序 (µC 把要送回主站的input数据写入ESC).

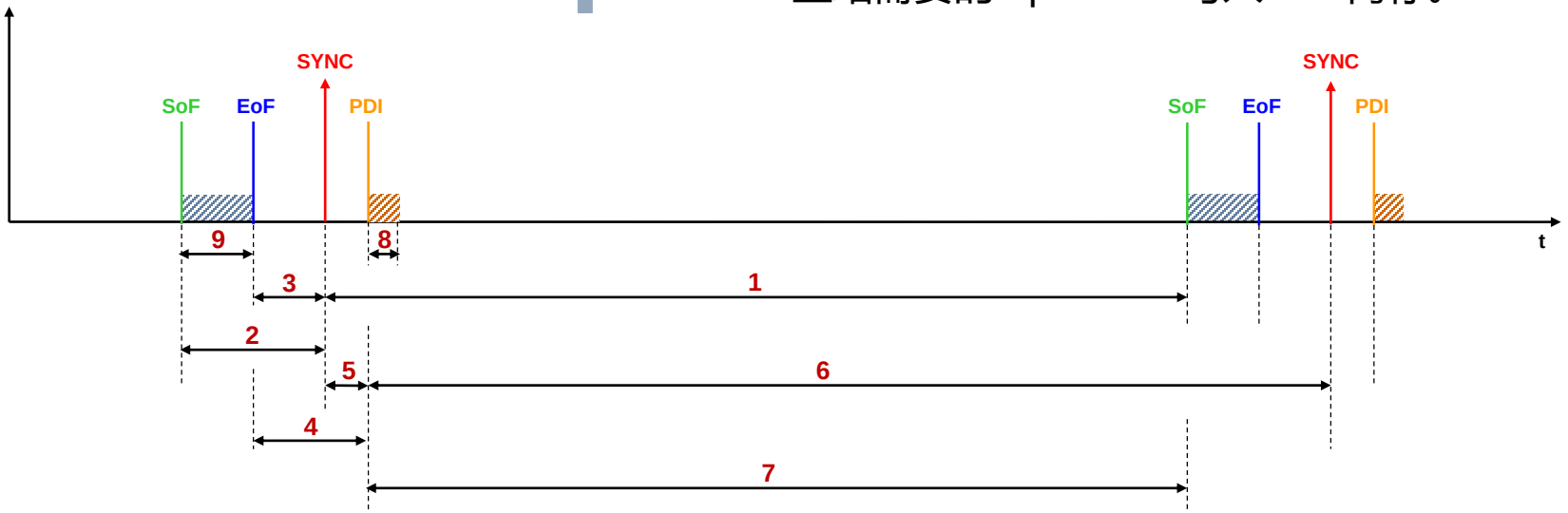
如果从站中两个**SYNC** 信号都启用了，可以选择以SYNC0还是SYNC1为参考点显示时序。



深入了解 – 时序诊断 Diagnosis (从站)

下图用图形表达上页所显示时间的意义何在:

1. 数据帧到达从站，由ESC在线（ “on-the-fly” ）处理。
2. ESC产生同步中断 “SYNC interrupt” 给从站的微处理器（ µC ）使用。
3. µC 经过PDI（Process Data Interface）从ECS的内存读取来自主站的Output Data并将主站需要的Input Data写入ESC内存。

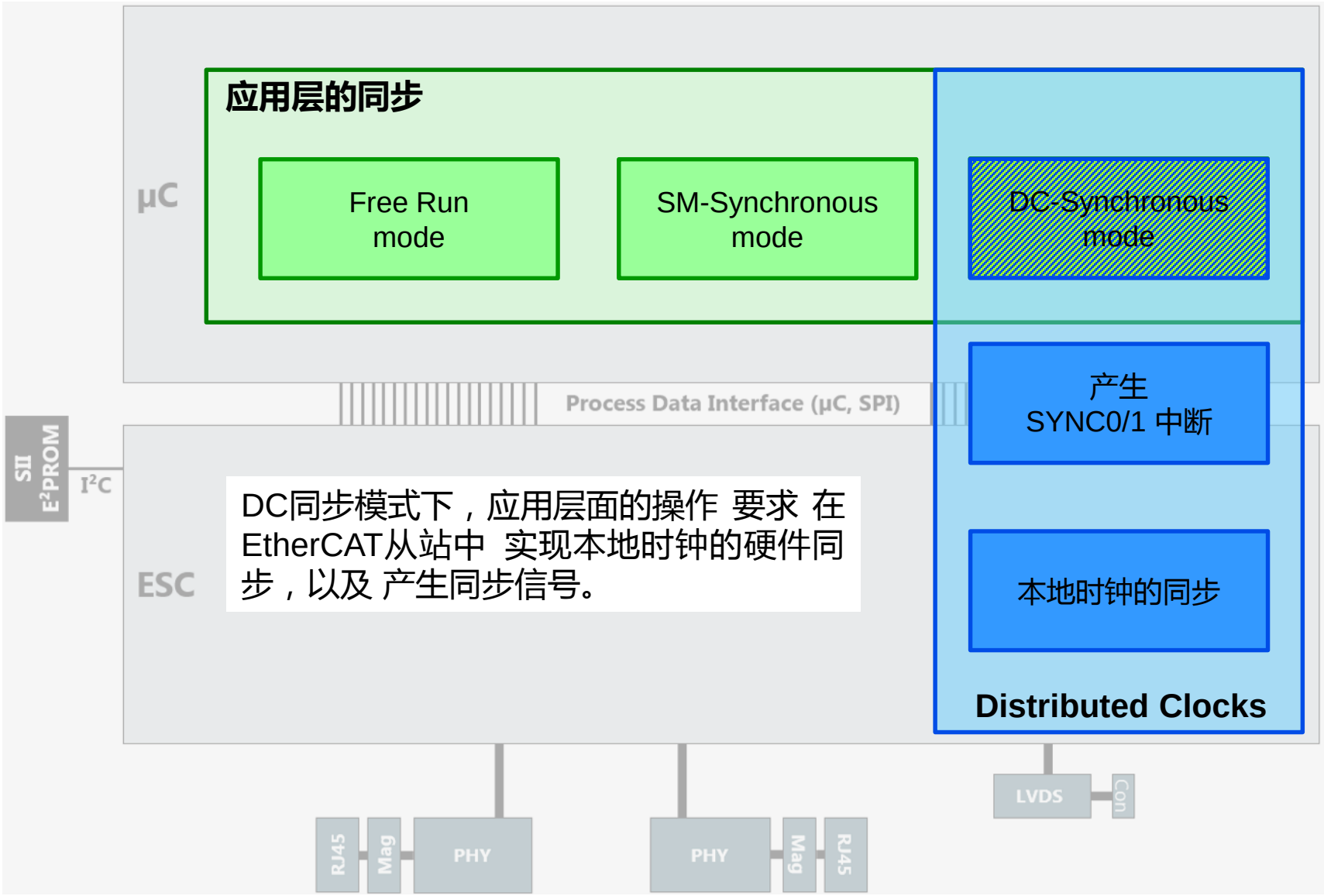


同步丢失时的 AL 状态码

当 DC-Synchronous 从站失去同步，就会退回到 Safe-Operational 并报告以下AL状态码之一：

AL Status Code	Name	Meaning and reliminary checks suggested
0x35 (0x36, 0x37)	DC Invalid SYNC (SYNC0, SYNC1) cycle time DC同步周期无效	主站配置的周期对从站应用来说是无效的。 **Action** → 检查 0x1C32/33:05, 增加所链接的任务的周期时间。如果仍不能解决问题，则检查是否从站允许的周期只支持**几个离散的值**（查看从站的文档），而不支持TC Base Time的任意整数倍。
0x30	Invalid DC SYNC configuration DC同步配置无效	主站下载的 SYNC 设置对于从站应用是无效的。 **Action** →检查 DC SYNC settings 没有手动修改过。如果改过，就删除再手动重新添加配置
0x2D	No SYNC Error 无同步错误	在SafeOP → OP 转换时，没有产生同步信号（SYNC signal） **Action** → 检查 DC SYNC settings 没有手动修改过。如果改过，就删除再手动重新添加配置，然后用Wireshark进行追踪，以检查主站是否正确配置了同步启动时间（SYNC Start Time）
0x32	PLL Error PLL错误	Slave application cannot synchronize itself to the communication cycle. **Action** →检查 master jitter （主站波动）的性能
0x1A	Synchronization Error 同步错误	从站的应用不能把自己同步到通讯周期。 **Action** → 检查 master jitter （主站波动）的性能
0x33	DC Sync IO Error DC同步IO错误	
0x34	DC Sync Timeout Error DC同步超时错误	

应用层的同步


应用层的同步
µC
Free Run mode
SM-Synchronous mode
DC-Synchronous mode
产生 SYNC0/1 中断
本地时钟的同步
Distributed Clocks
Process Data Interface (µC, SPI)
SII E²PROM
I²C
ESC
DC同步模式下，应用层面的操作 要求 在EtherCAT从站中 实现本地时钟的硬件同步，以及 产生同步信号。
RJ45
Mag
PHY
PHY
Mag
RJ45
LVDS
Con

Distributed Clocks 分布时钟

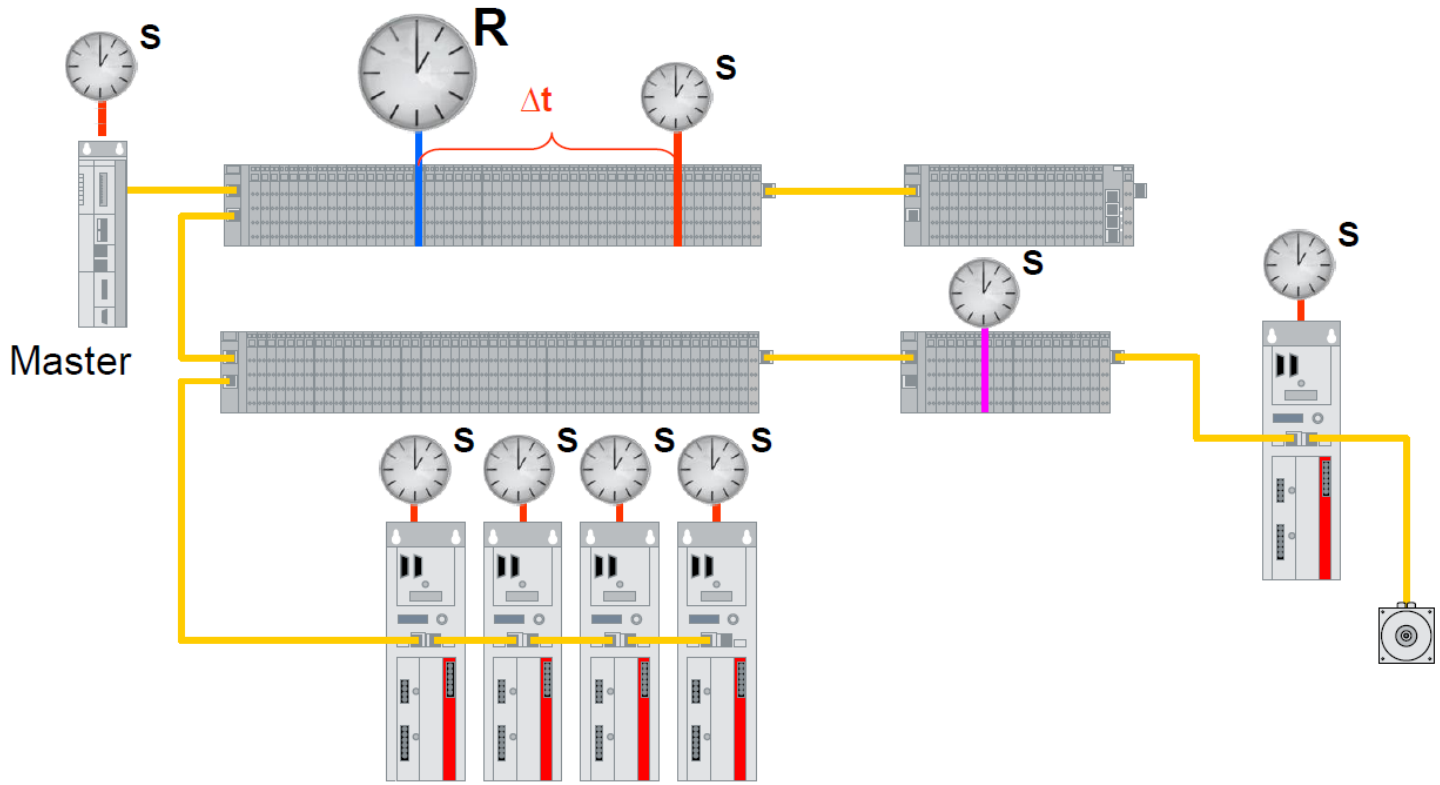
分布时钟技术可以实现EtherCAT网络上所有本地时钟的同步 ——包括主站和所有DC从站——最大偏差 ≤ **100 ns**

- **System Time**: 系统时间，即所有DC同步设备共同使用的时间
  - 从2000年1月1日零点开始。
  - 单位是 ns
  - 长度 64-bit (足可表达超过500年的时间)
  - 低 32-bits 可以表达 4.2 秒 (通常可满足时钟同步和时间戳应用)
- **Reference Clock**: 参考时钟，即决定系统时间 System Time 的**从站**设备, 默认为EtherCAT网络中**首个**配置为DC同步模式的从站 (此设置不可更改)：
  - “**从站**”，这是为了获得完全**基于硬件**的高精度的参考时钟，
  - “**首个**”，这是为了**在一个通讯周期内向**所有其它DC同步模式的网络设备（包括主站和所有DC从站）发布系统时间 System Time

Distributed Clocks 分布时钟

因此，把EtherCAT网络中的 DC-Synchronous 设备归为两类：

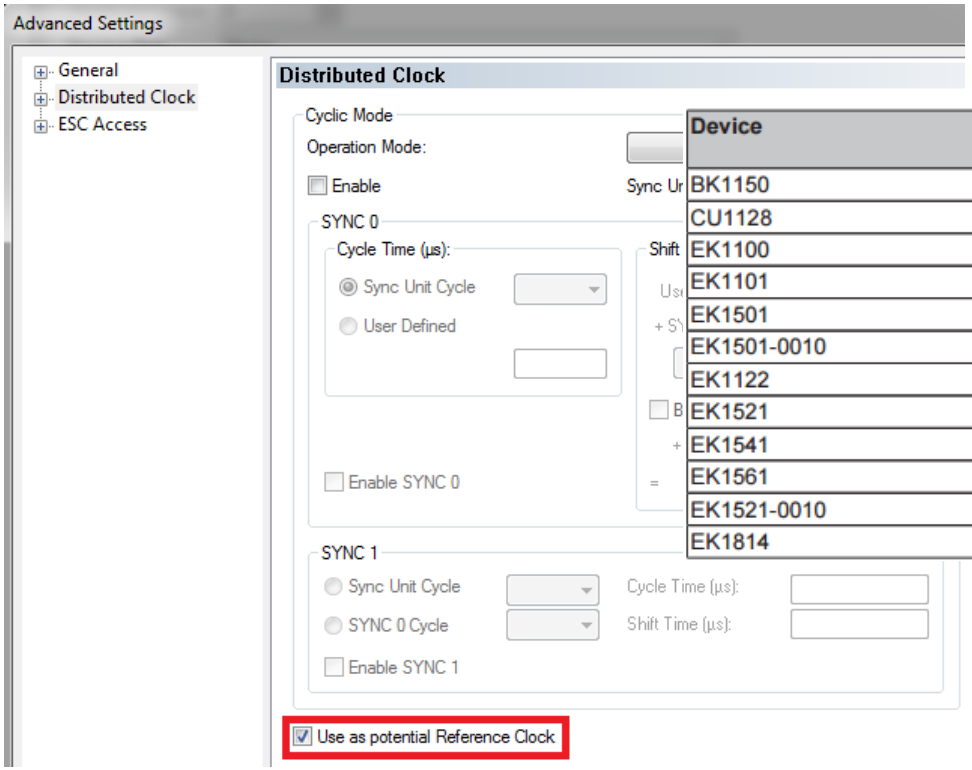
- **Reference Clock**: 首个 DC-Synchronous 从站
- **Slave Clocks**: 所有其它 DC-Synchronous 从站+ (默认) 主站



Potential Reference Clock 潜在的参考时钟？

Reference Clock 参考时钟不是必须要求用网络中的正正第1个DC从站，但建议也不要离主站太远。

考虑到在有的情况下参考时钟的时间基准有可能出现偏差（e.g. 由于多个DC同步的从站报告了同步错误），建议启用网络拓朴中离主站最近的DC从站作为参考时钟。



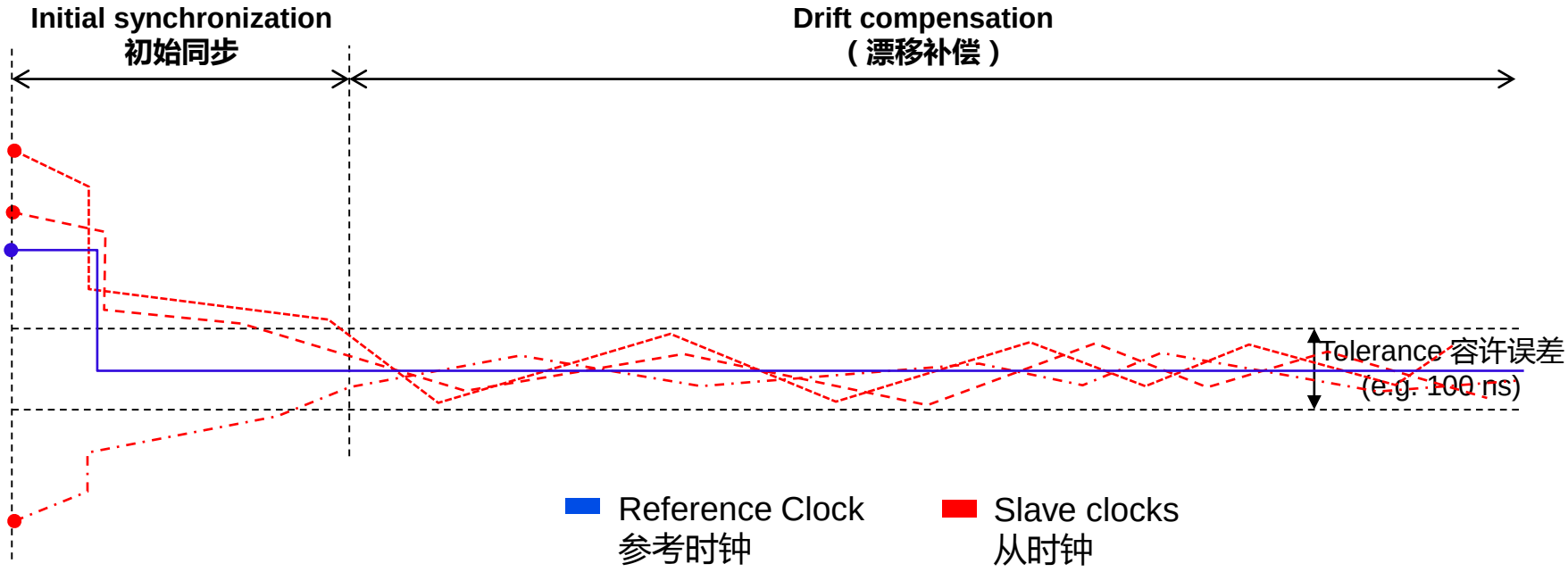
Device	XML revision in the configuration	Serial number of the component
BK1150	from BK1150-0000-0016	from firmware 01: xxxx01yy
CU1128	from CU1128-0000-0000	from firmware 00: xxxx00yy
EK1100	from EK1100-0000-0017	from firmware 06: xxxx06yy
EK1101	from EK1101-0000-0017	from firmware 01: xxxx01yy
EK1501	from EK1501-0000-0017	from firmware 01: xxxx01yy
EK1501-0010	from EK1501-0010-0017	from firmware 02: xxxx02yy
EK1122	from EK1122-0000-0017	from firmware 01: xxxx02yy
EK1521	from EK1521-0000-0018	from firmware 03: xxxx03yy
EK1541	from EK1541-0000-0016	from firmware 01: xxxx01yy
EK1561	from EK1561-0000-0016	from firmware 01: xxxx01yy
EK1521-0010	from EK1521-0010-0018	from firmware 03: xxxx03yy
EK1814	from EK1814-0000-0016	from firmware 00: xxxx00yy

深入了解 – 同步的过程

本地时钟的硬件同步分为两个环节： **2 phases:**

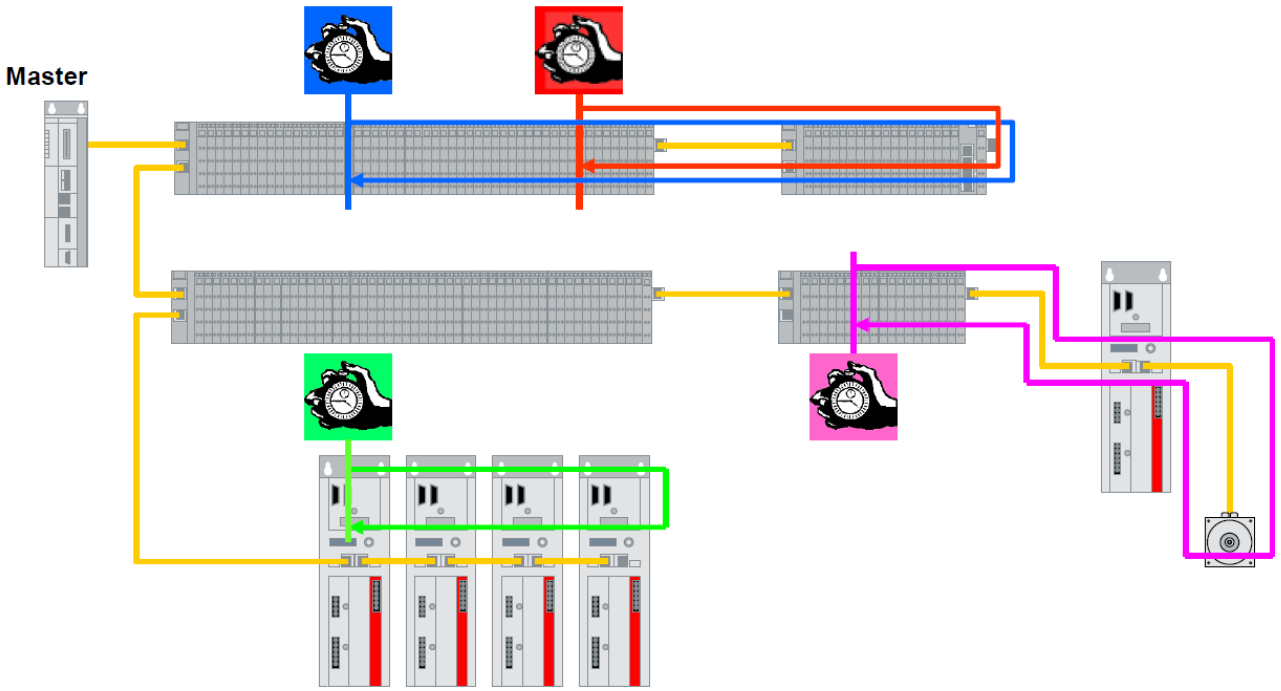
1. **Initial synchronization (初始同步)**：主站对齐所有的DC从站时钟到最大容许误差以内。(典型误差 < 100 ns).

2. **Drift compensation (漂移补偿)**：连续保持本地时钟与参考时钟同步，以抵消时钟漂移的影响 。(时钟漂移来自不同的晶振频率、热效应或者老化程度上的细微差别)



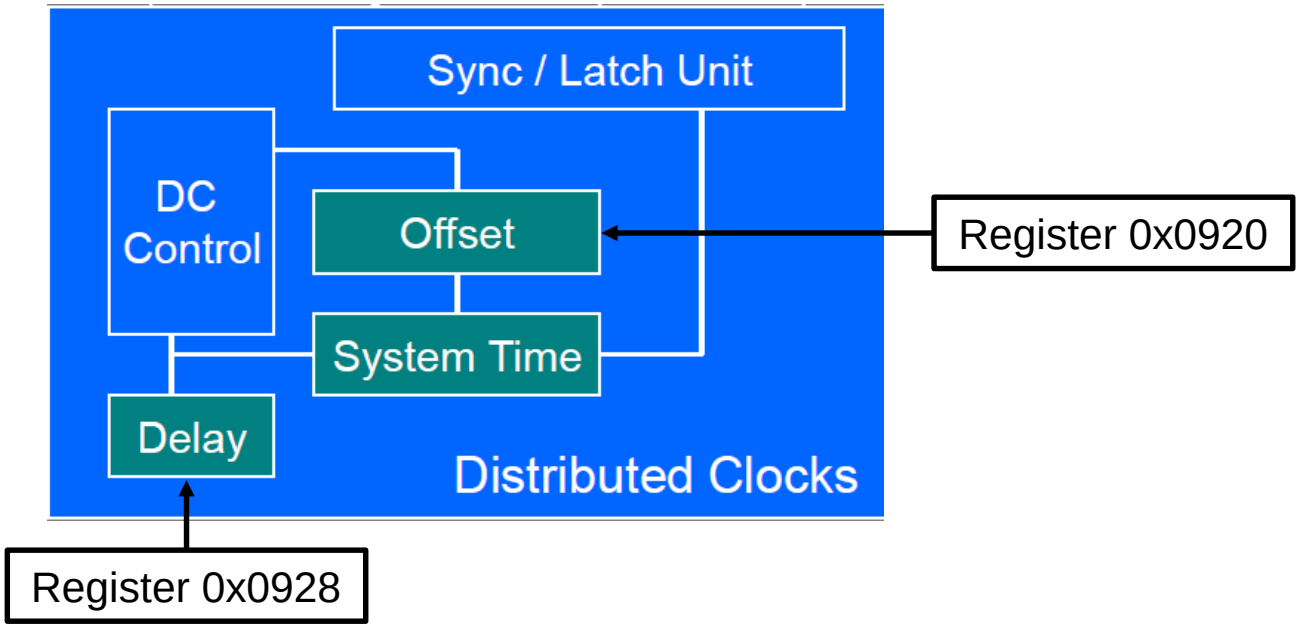
深入了解 – 同步的过程

- **Initial Synchronization (Step 1)**: 初始同步阶段的第1步，主站发送一个广播的写命令到地址0x0900，强制所有从站锁存其每个端口的本地时间，然后从返回的数据帧中读取所有获取到的时间戳。



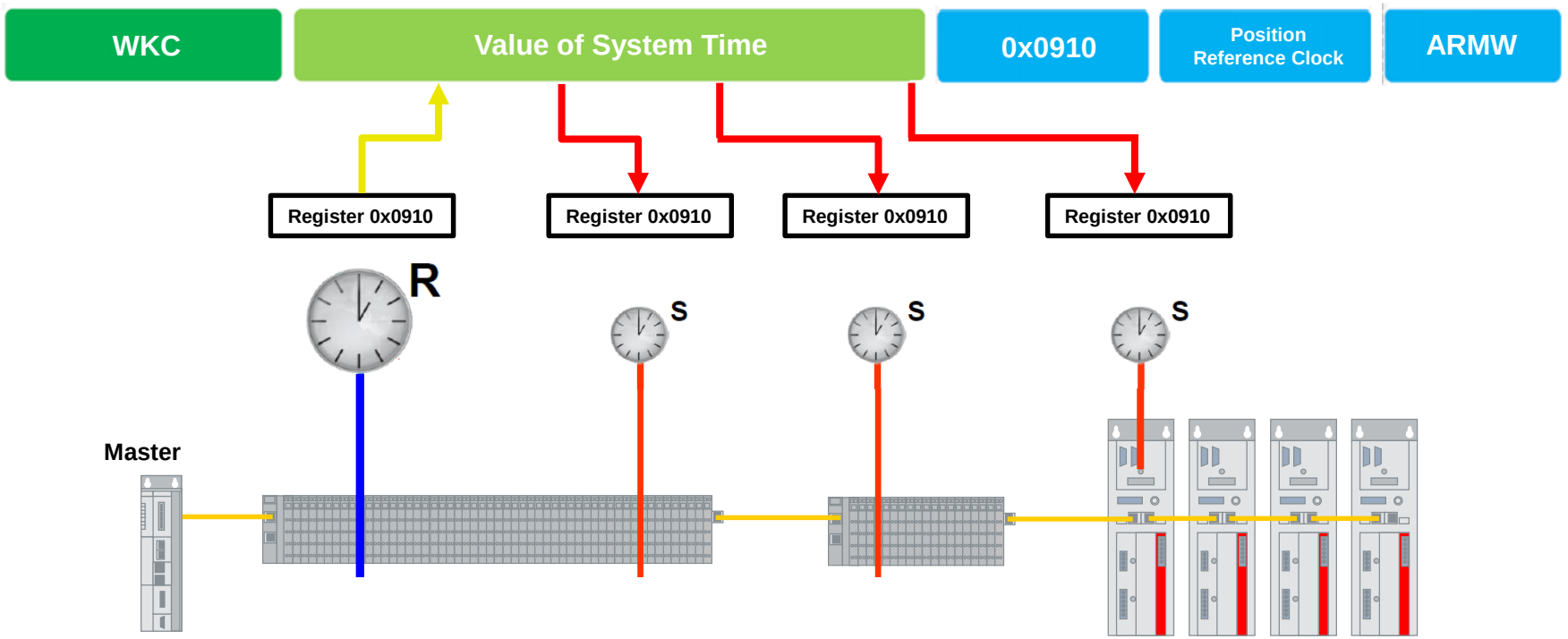
深入了解 – 同步的过程

- **Initial Synchronization (Step 2)：**初始同步阶段的第2步，基于这些收集回来的时间戳，the主站计算每个DC同步的从站的偏移和延时，并把这些值写入DC从站各自的专用内存地址。（从站会在本地时间控制器中使用这些参数）
  - **Offset：**偏移，从站本地时间“0”和绝对系统时间（TwinCAT从TC Time中提取的）的差值。
  - **Delay：**延时，参考时钟和从站之间的传播时间。



深入了解 – 同步的过程

- **Initial Synchronization (Step 3)：**初始同步阶段的第3步，为了让参考时钟Reference Clock和所有DC从站时钟都共享系统时间System Time，主站发送许多个ARMW 命令到从站的内存地址 0x0910，触发每个从站的本地时间控制器微调本地时钟，以尽量缩小和系统时钟的差值，(把 偏移Offeset 和 延时Delay 计算在内)



深入了解 – 同步的过程

- **Initial Synchronization (Step 4)：**初始同步阶段的第3步， 主站读取每个从站的内存地址0x092C，以检查所有分布时钟与系统时钟的差值都在最大容许误差以内 (e.g. < 100 ns) ：

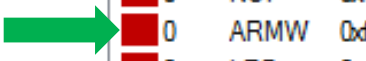
Register System Time Difference (0x092C:0x092F)

Bit	Description	ECAT	PDI	Reset Value
30:0	Mean difference between local copy of System Time and received System Time values	r/-	r/-	0
31	0: Local copy of System Time greater than or equal received System Time 1: Local copy of System Time smaller than received System Time	r/-	r/-	0

如果没有达到要求的同步精度，则重复第3步和第4步。

深入了解 – 同步的过程

- **Drift Compensation** ：漂移补偿，为了使各个从站的本地时钟在运行过程中保持对齐，主站在周期性数据帧中增加一个 ARMW（**对时**）命令，和初始同步阶段发送的ARMW命令一样：

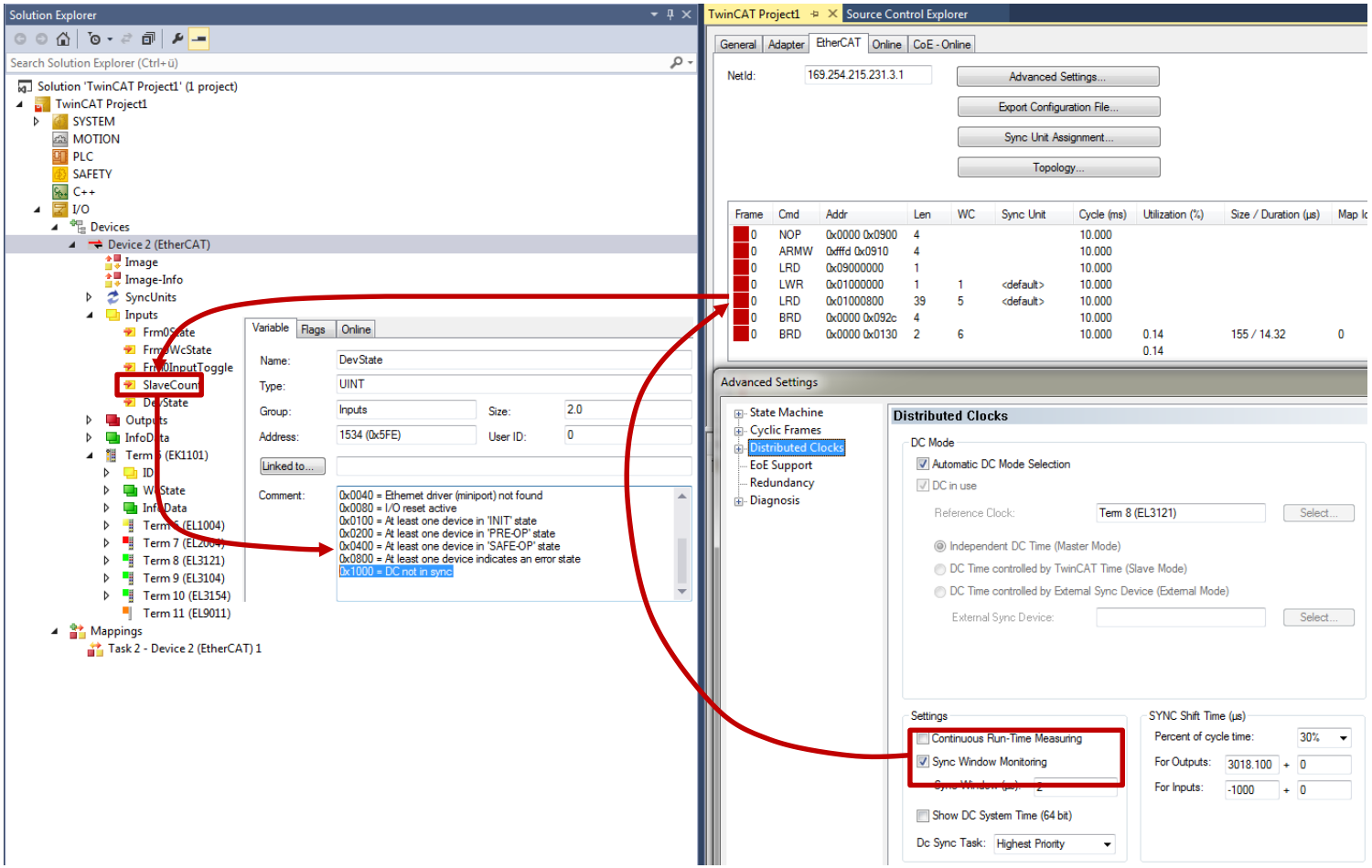


Frame	Cmd	Addr	Len	WC	Sync Unit	Cycle (ms)	Utilization (%)	Size / Duration (µs)	Map Id
0	NOP	0x0000 0x0900	4			2.000			
0	ARMW	0xffff 0x0910	4			2.000			
0	LRD	0x09000000	1			2.000			
0	LRW	0x01000000	32	12	<default>	2.000			
0	BRD	0x0000 0x0130	2	5		2.000	0.57	119 / 11.44	1
							0.57		

每次接受到ARMW（**对时**）命令，从站就会调整它的本地时钟的速度，所以这个命令应当以足够高的频率发送，才能保证良好的漂移补偿。由于ARMW命令是和周期性数据帧一起发送的，所以它的发送频率取决于任务周期。链接到所有DC同步从站的一个或者多个任务中，最高优先级的那个任务周期，不应当超过**2 - 4ms**。

同步监视

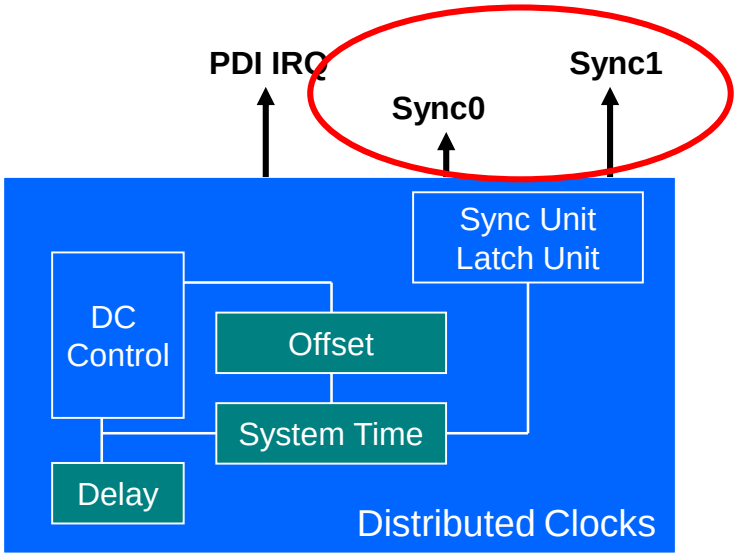
在EtherCAT主站的Advanced Settings 中的Distribut Clock选项里勾选 “Sync Window Monitoring”标记后，激活配置。如果网络中DC从站的本地时钟的最大偏差超过了配置的阀值，主站的周期性变量**DevState**的 Bit 12 就会变成 1.

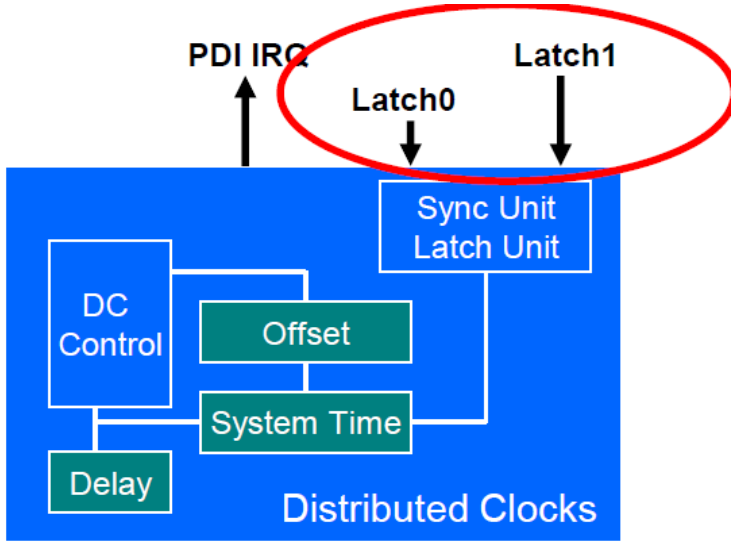


SYNC 和 LATCH 信号

本地时钟同步到系统时间后，每个DC同步的从站就可以同步处理“数字事件” **digital events**:

- **SYNC signals**: 同步信号，这是供EthrCAT芯片使用的数字输出信号，只在指定的系统时间触发。 (前面已经介绍了把同步信号作为硬件中断， 用于同步 DC同步 模式下的从站应用)
- **LATCH signals**: 锁存信号，这是供EtherCAT芯片使用的数字输入信号，用于锁存该信号触发瞬间的系统时间值（该值可以通过周期性数据或者非周期性访问发送给主站)

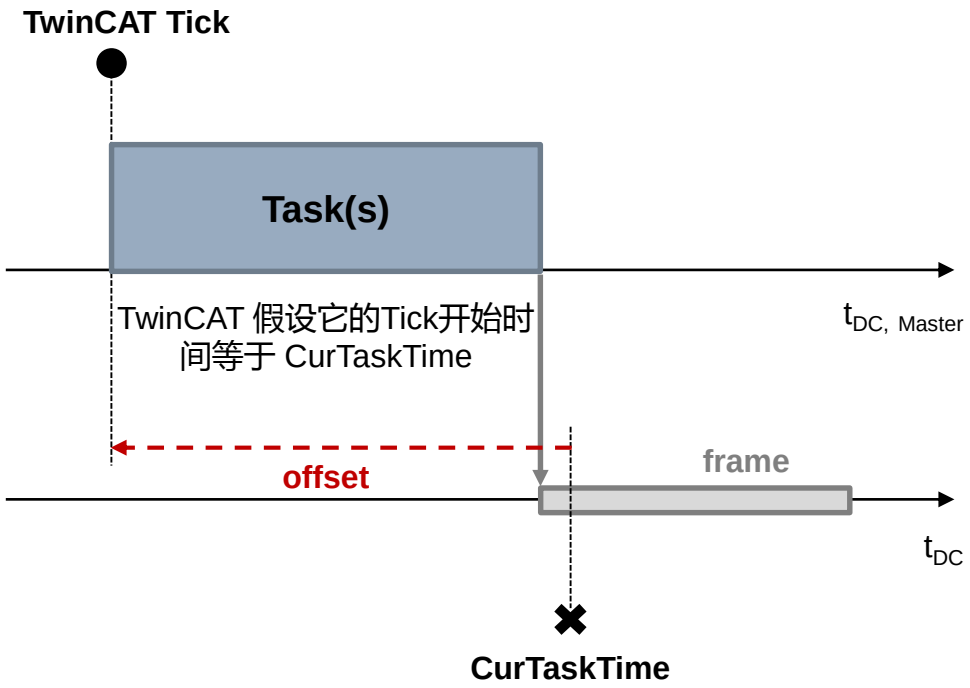




深入了解 - TwinCAT Base Time 和 CurTaskTime

按默认设置，一旦配置成了DC同步的EtherCAT网络，TwinCAT 就会调整它的**实时时钟**（real-time clock ）到系统时间，但这两个时间的参考点并不一致。

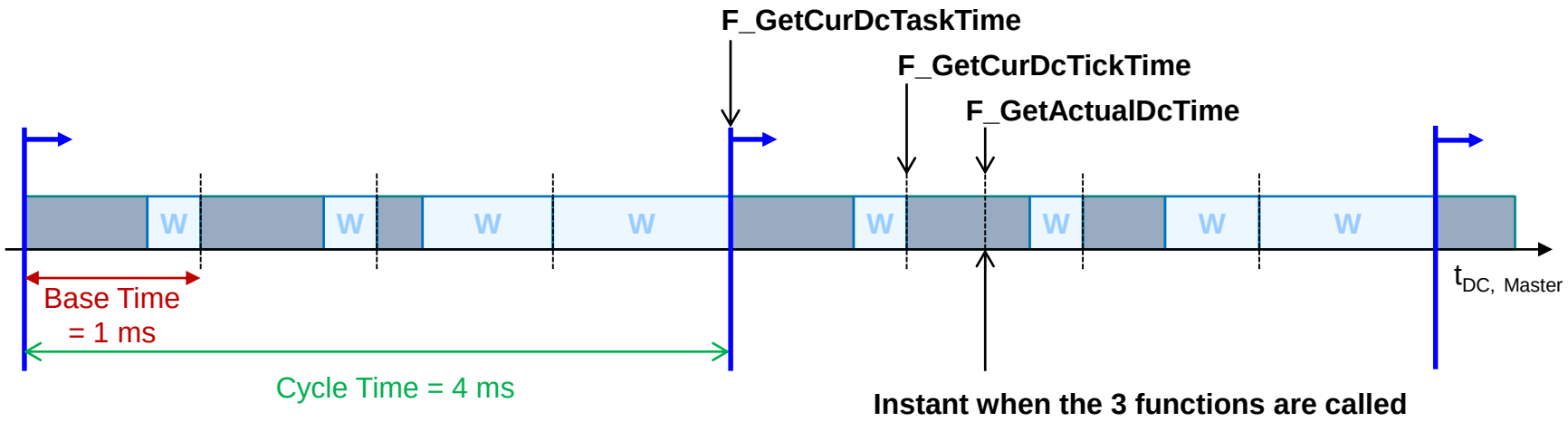
为了使计算简单，在TwinCAT内部先假定软件任务开始的时间等于 CurTaskTime，这样就会在硬件的 DC time (t_{DC}) 和 调整后的 TwinCAT DC time ($t_{DC, Master}$)之间形成一个偏移量：

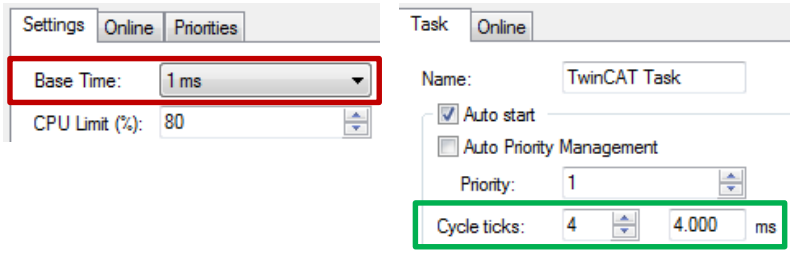


深入了解 - TwinCAT PLC 时间功能

使用**TcEtherCAT.lib**库中的函数，PLC程序对 DC times进行以下处理：

- **F_GetCurDcTaskTime**：返回当前任务周期开始的DC时间
- **F_GetCurDcTickTime**：返回上个TwinCAT Base Time Tick（TC 时基刻度）的DC时间
- **F_GetActualDcTime**：返回调用本函数瞬间的DC时间

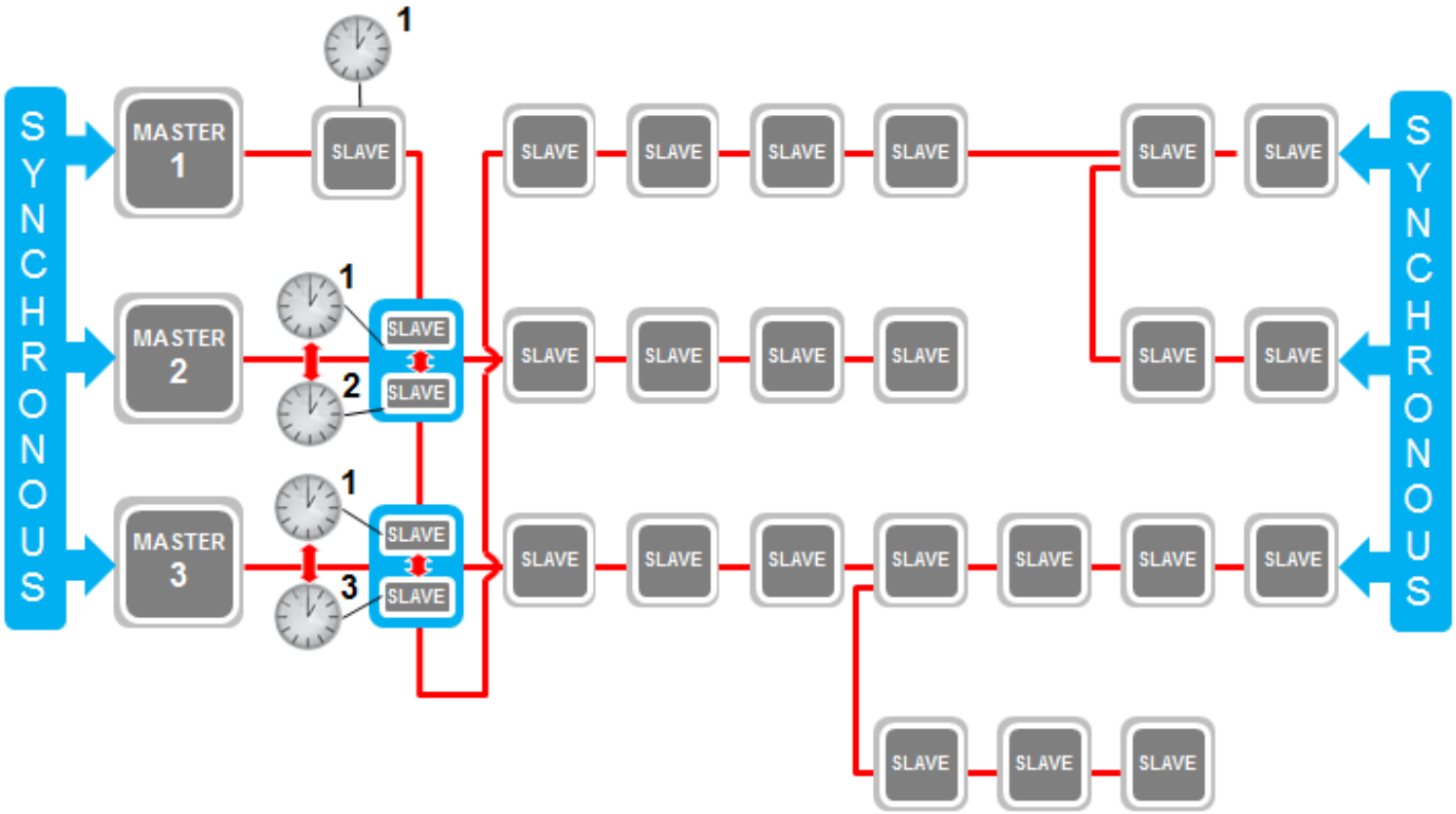




这3个函数返回的时间信息都是基于$t_{DC,\ Master}$的时间基准

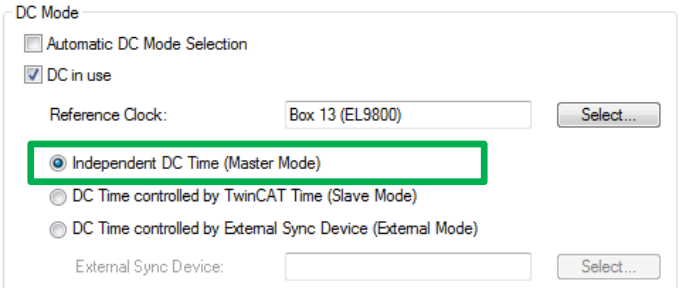
不同网络之间的 DC 同步

除了在一个EtherCAT网络中实现同步之外，分布时钟技术还能在不同的EtherCAT网络之间实现同步，以及把EtherCAT网络的DC时钟同步到一个外部 IEEE 1588 参考时钟。



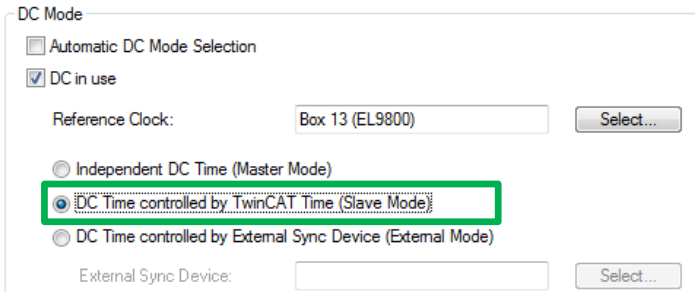
不同网络之间的 DC 同步

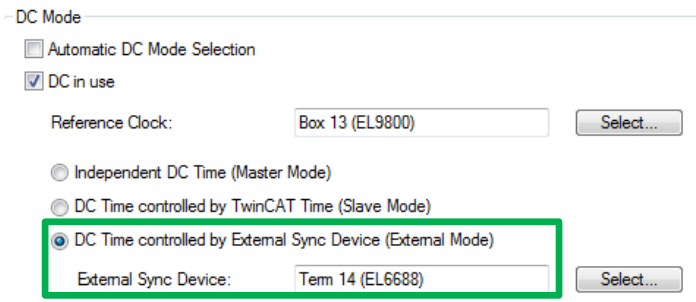
一个网络的参考时钟独立运行，不受干扰 (默认一个TwinCAT项目中只配置一个EtherCAT网络), 它的DC工作模式就是**主时钟模式（Master Mode）**.



而有时，参考时钟可能被另一个时钟源调整：

- **Slave Mode**: 从时钟模式，影响它的时钟源是主站设备的时钟 (i.e. TwinCAT clock).
- **External Mode**，外部时钟模式，影响它的时钟源（e.g. 另一个网络的参考时钟或者一个 IEEE1588 时钟信号) 通过一个专门的从站设备提供。





深入了解 – Multiple 同步 和 周期性数据帧

Master Mode 的EtherCAT网络只在周期性数据帧中包含1个ARMW（对时命令）周期性命令，以实现共享本地系统时间：

Frame	Cmd	Addr	Len	WC	Sync Unit	Cycle (ms)	Utilization (%)	Size / Duration (µs)	Map Id
0	NOP	0x0000 0x0900	4			1.000			
0	ARMW	0xffff 0x0910	4			1.000			
0	LRD	0x09000000	1			1.000			
0	LRW	0x01000000	22	1	<default>	1.000			
0	LWR	0x01000800	1	1	<default>	1.000			
0	BRD	0x0000 0x0130	2	3		1.000	1.16 1.17	122 / 11.68	1

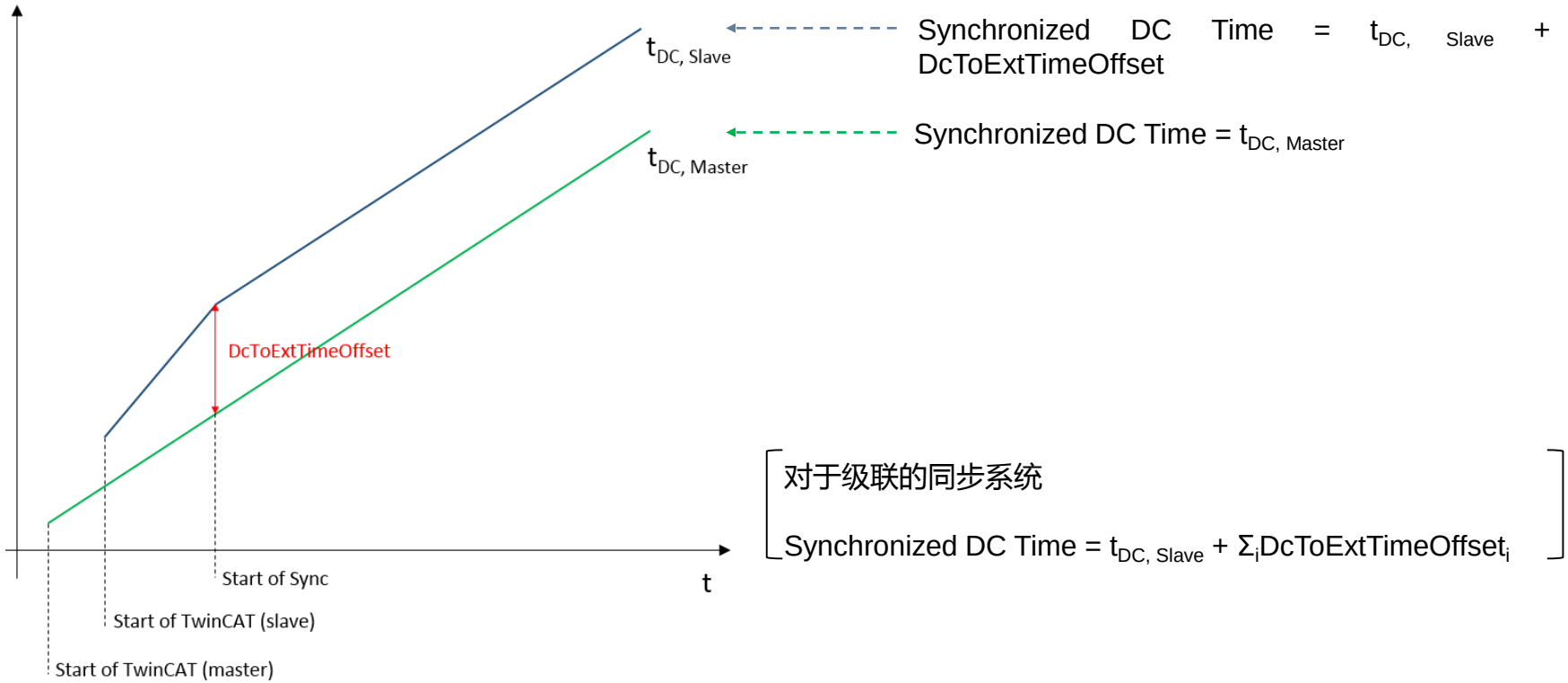
在 **Slave Mode** 或者 **External Mode** 的EtherCAT网络还要在周期性数据帧中额外增加一个APWR 命令，用于根据外部时钟源调整参考时钟

Frame	Cmd	Addr	Len	WC	Sync Unit	Cycle (ms)	Utilization (%)	Size / Duration (µs)	Map Id
0	NOP	0x0000 0x0900	4			1.000			
0	ARMW	0xffff 0x0910	4			1.000			
0	APWR	0xffff 0x0910	4			1.000			
0	LRD	0x09000000	1			1.000			
0	LRW	0x01000000	22	1	<default>	1.000			
0	LWR	0x01000800	1	1	<default>	1.000			
0	BRD	0x0000 0x0130	2	3		1.000	1.29 1.30	138 / 12.96	1

深入了解 - DcToExtTimeOffset

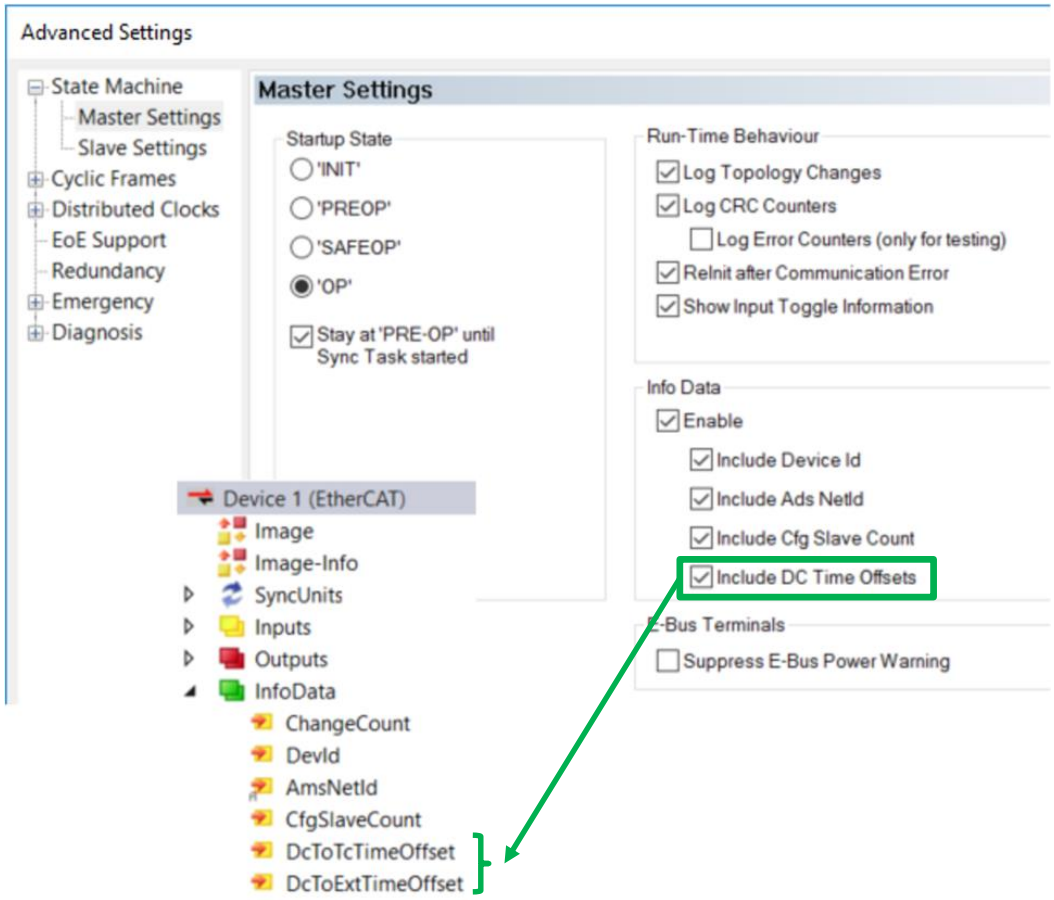
一旦EtherCAT网络的本地参考时钟调整到了一个外部时钟源，**frequencies** 就被同步了。一般来说，本地系统时钟的绝对值相对于外部时间源的时间绝对值，就会有一个偏移。

如果时间信息要基于同一个时间基准来使用，就不能不考虑本地时钟与外部时钟源的偏移量。



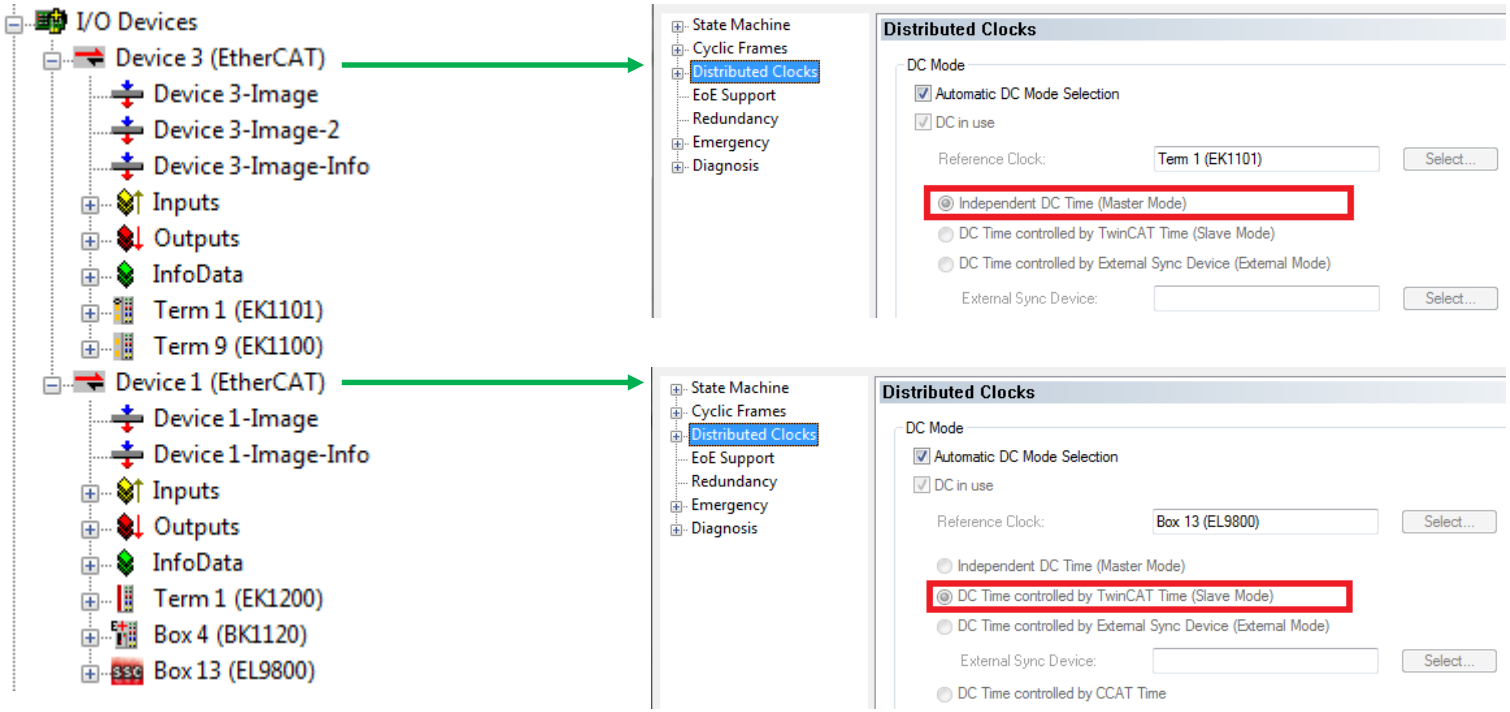
深入了解 - DcToExtTimeOffset

TwinCAT 在EtherCAT网络的InfoData变量中包含了 **DcToExtTimeOffset，**which可以链接到PLC程序进行时间计算 (需要在Master Setting 中手动设置，InfoData 中才会出现这个变量).



同一PC上多个网络之间的同步

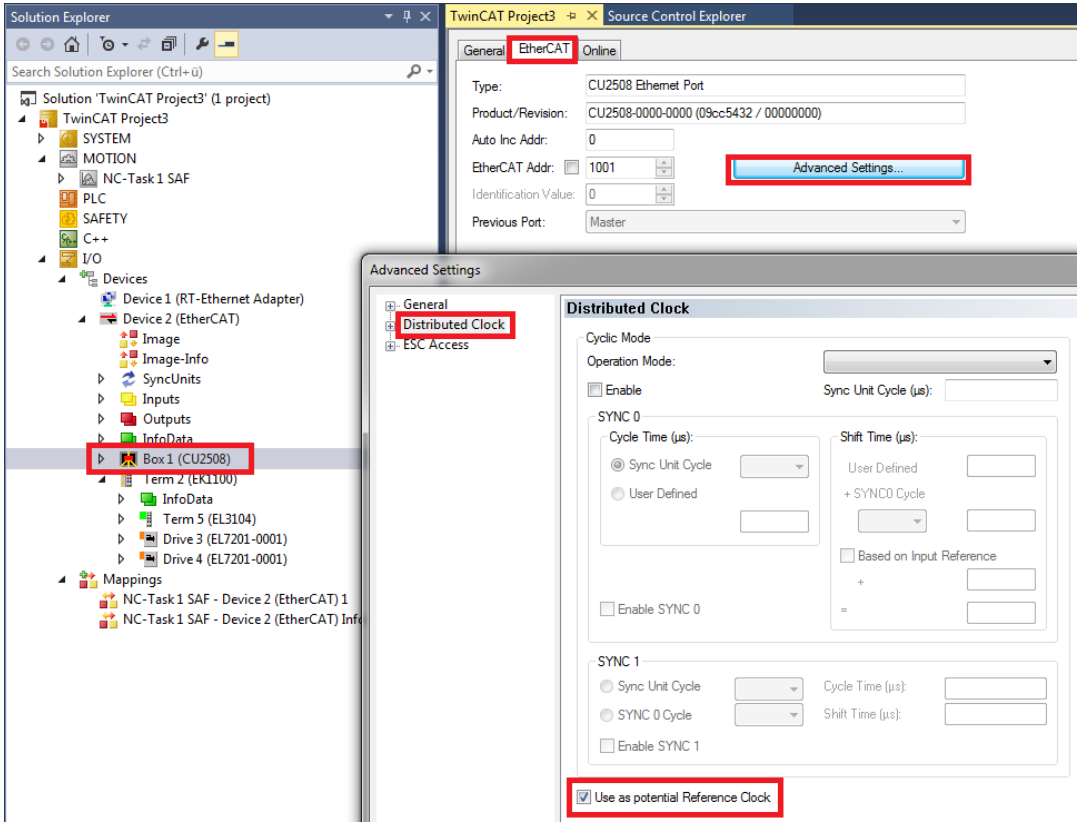
如果一个TwinCAT控制器上运行了多个DC同步网络，几个网络之间的时钟同步是自动实现的。(不需要另加的硬件同步模块).



第1个添加的网络会自动工作在主时钟模式，其它网络工作在从时钟模式。

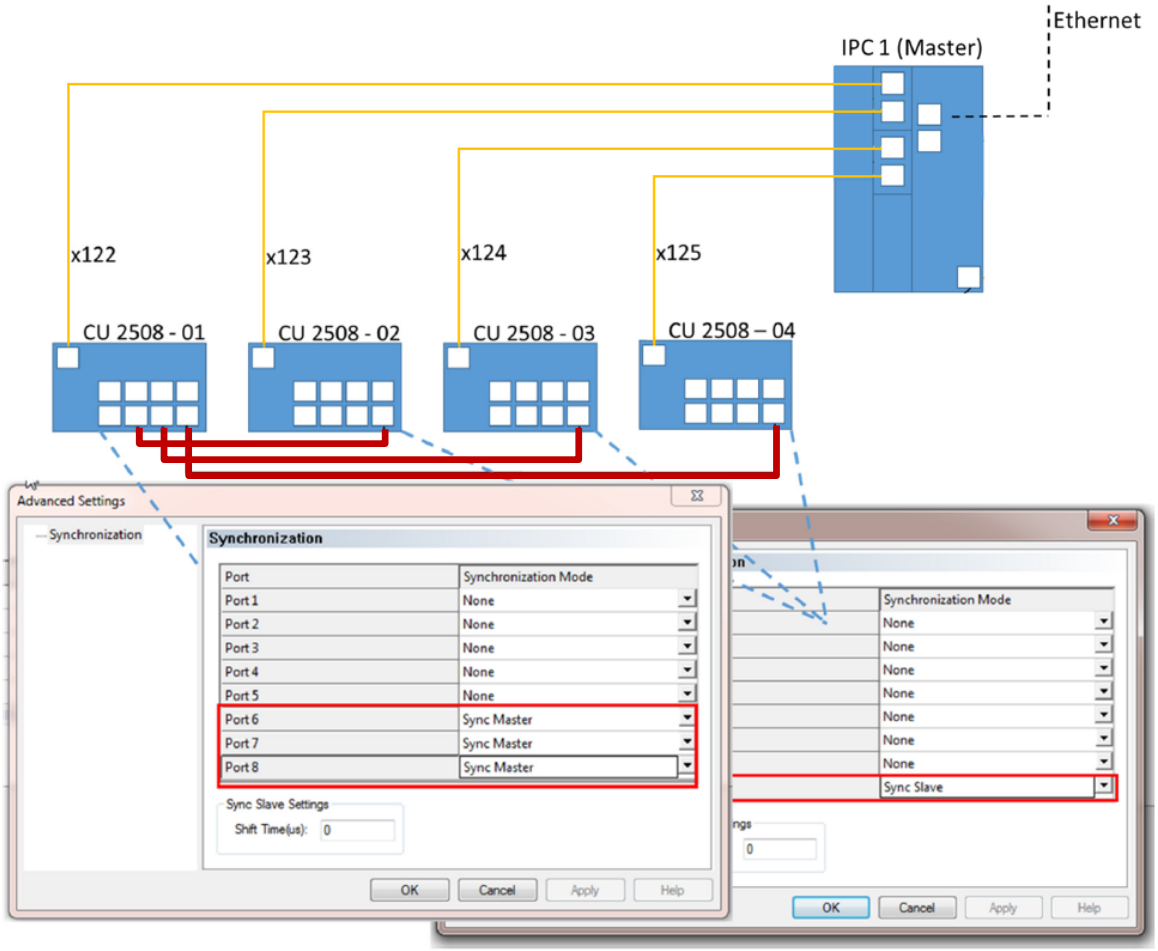
深入了解 – 经网络倍增器同步

为了使经过端口倍增器连接到控制器的多个EtherCAT网络之间实现时钟同步，在各个EtherCAT网络中，都应该把**CU2508** 手动配置为参考时钟**Reference Clock** （必须使用CU2508，才能同时启用DC 同步和电缆冗余功能！）



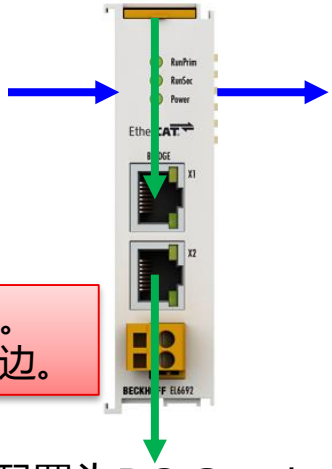
深入了解 – 多个网络倍增器CU2508的口同步

如果项目中使用了多个CU2508 (以XTS应用为例), 不同 CU2508 的时钟也应该互相同步。



经 EtherCAT 桥接模块 (EL6692/EL6695) 同步

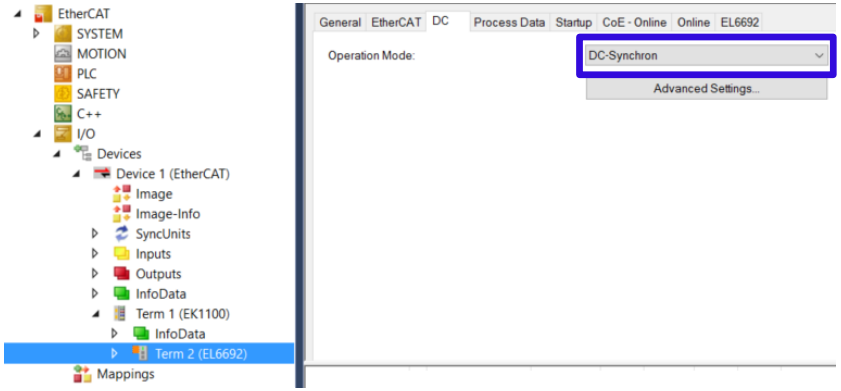
EL6692/EL6695 桥接模块同时是两个EtherCAT网络中的从站：

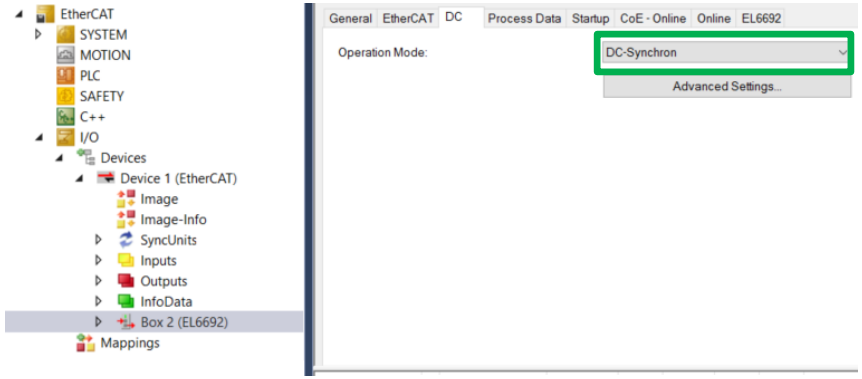


- **Primary Side**：EBUS
- **Secondary Side**：RJ45

模块物理安装所在的系统，称为原边。
通过网线连接到模块的系统，称为副边。

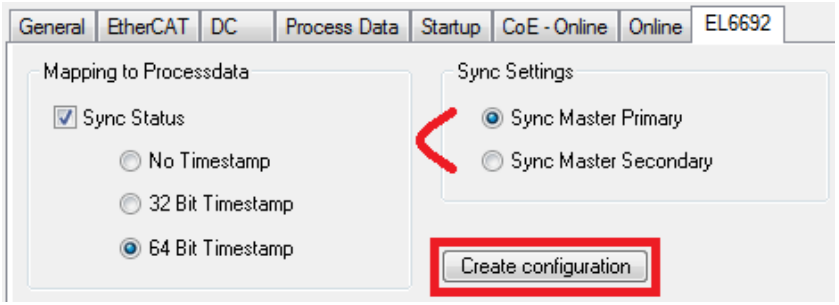
为了实现DC同步，桥接的两侧都应当配置为DC-Synchronous 模式:

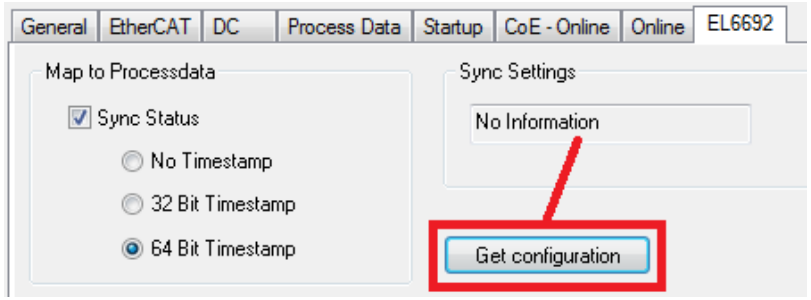




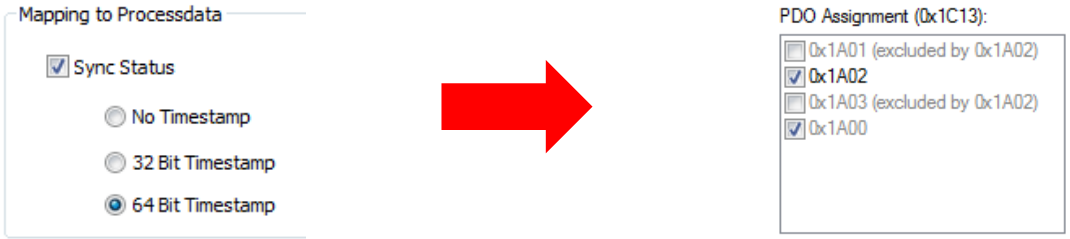
经 EtherCAT 桥接模块 (EL6692) 同步

对于 EL6692，配置原边（Primary Side，即模块经EBUS安装所在的系统）时，必须定义以原边还是以副边为**同步的主时钟（Sync Master）**(i.e. 同步主时钟的本地系统时间不调整). 副边（应稍后进行配置）可以从原边上传这个配置 （按钮 **Get configuration** ）



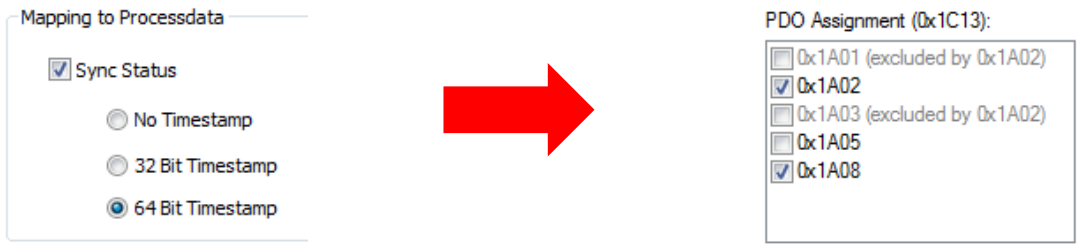


在原边和副边的系统中，当选择时间戳格式时，都会自动设置传输时间信息所需要的PDO assignment 。

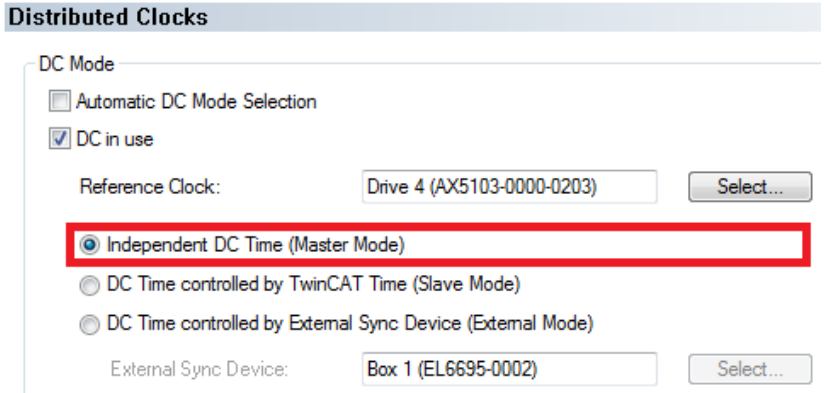


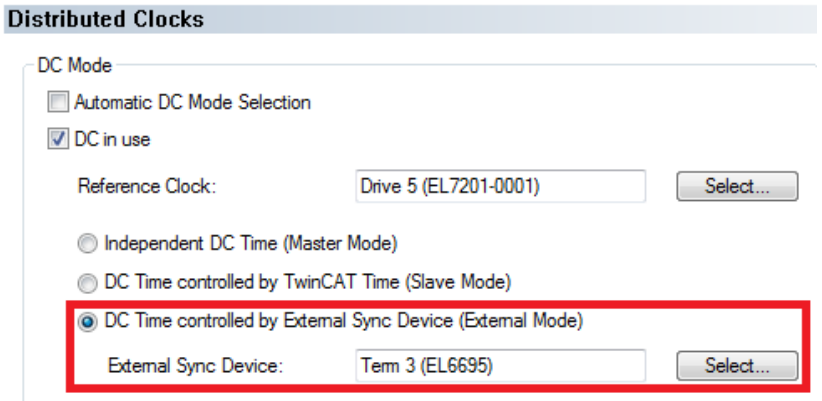
经 EtherCAT 桥接模块 (EL6695) 同步

对于 EL6695, 只要在专门的页面设置Timestamp的格式，要求的PDOs 就会自动映射：



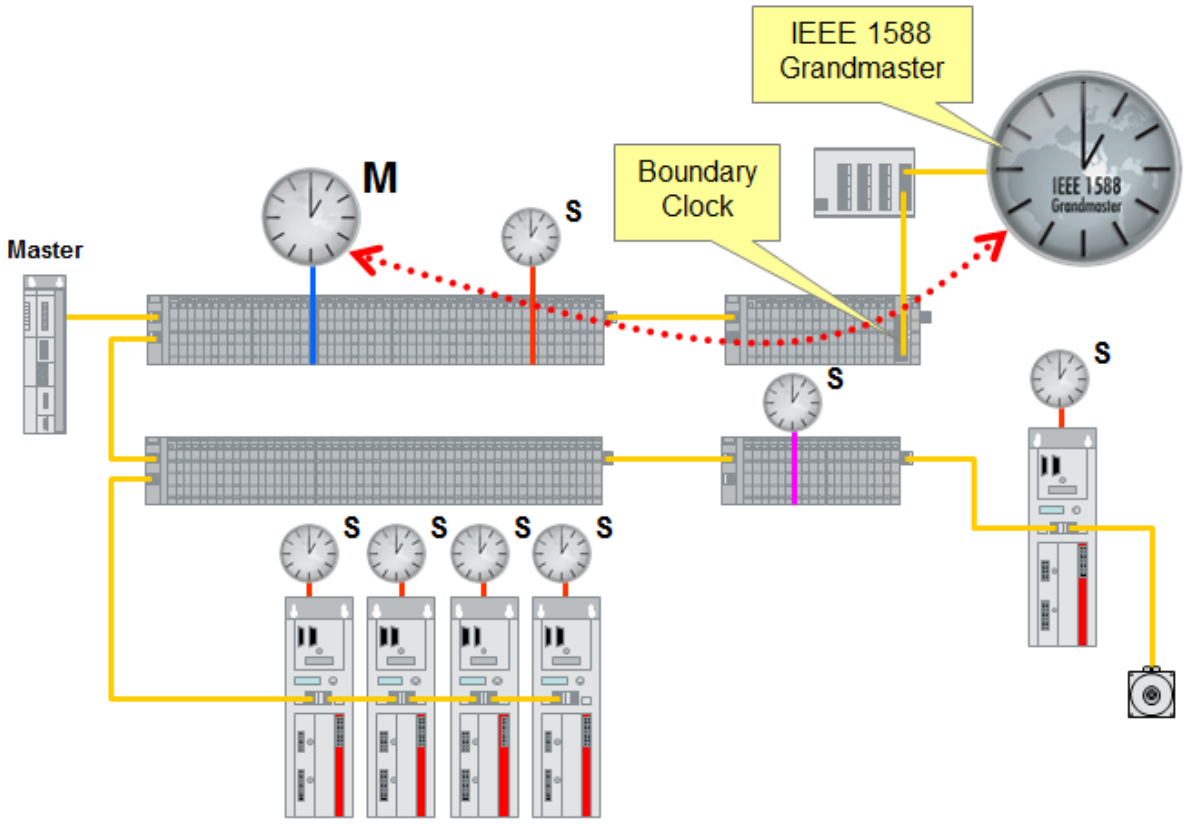
至于哪边作同步主时钟而哪边做从时钟，应该直接在各自EtherCAT主站的Advanced Settings中进行设置。





经 IEEE 1588 网络时钟模块 (EL6688) 同步

EtherCAT 网络的参考时钟可以同步到外部的 IEEE 1588 网络时钟 **Grandmaster Clock**.



经 IEEE 1588 网络时钟模块 (EL6688) 同步

TwinCAT 项目中插入了EL6688模块，DC 模式就**自动**配置为**External Mode** (不需要手动设置). 本地参考时钟就会周期性调整到外部的 IEEE 1588 时钟源 (两个时钟基准之间的偏移量，会在变量 DcToExtTimeOffset 中显示).