YTMicro MCAL应用 - Spi模块配置及实例

版本

Config Tool Version：3.0.0

ME0 MCAL version：2.3.0

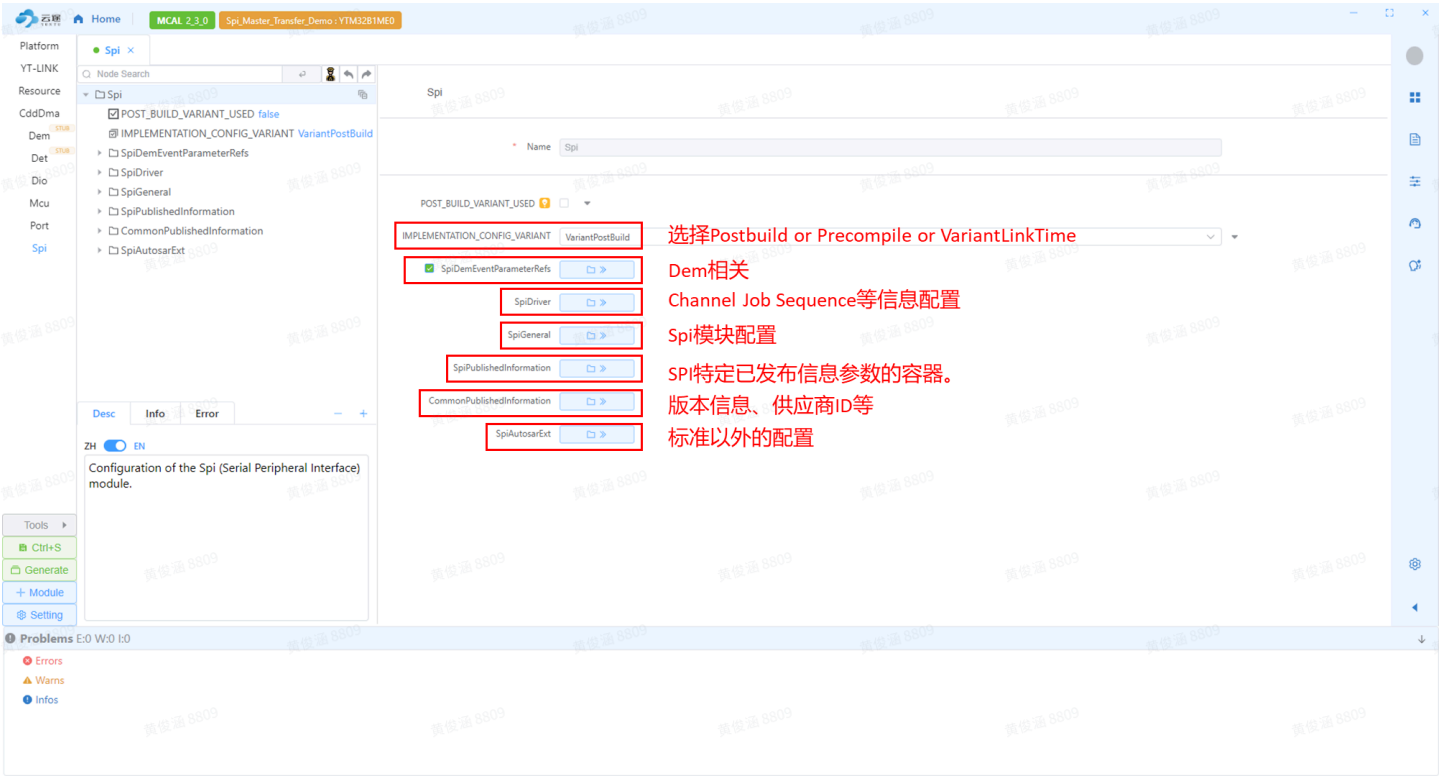
前言

1. Spi 一般依赖 Mcu、Platform 和 Port 模块，Mcu 模块里面需要配置外设时钟，Platform 里面可以配置外设中断，Port 配置使用到的端口资源 ，如果选择 DMA 方式则需要另外添加 CddDMA 模块。
2. Port、Mcu 和 Platform 的配置参考《 MCAL 应用 - 工程建立及工具通用设置》和《 MCAL 应用 - Mcu 模块配置及实例》文档，Port 的配置参考《 MCAL 应用 - Port 模块配置及实例》，CddDMA 的配置参考《 MCAL 应用 - CddDMA 模块配置及实例》.
3. 配置中包含 AutoSar 规范标准选项和云途自定义选项，左边树选框中粗体字为 AutoSar 标准选项，细体字为云途自定义选项。
4. MCAL 应用 Spi 模块配置及实例（一）主要说明 SPI 基本配置，以及作为主机节点的应用实例。

1. 基础模块配置介绍

1.1 框架说明

此处主要介绍了整个SPI的框架，需要我们着重关注的是SpiDriver和SPIGeneral两部分，下文会详细介绍。

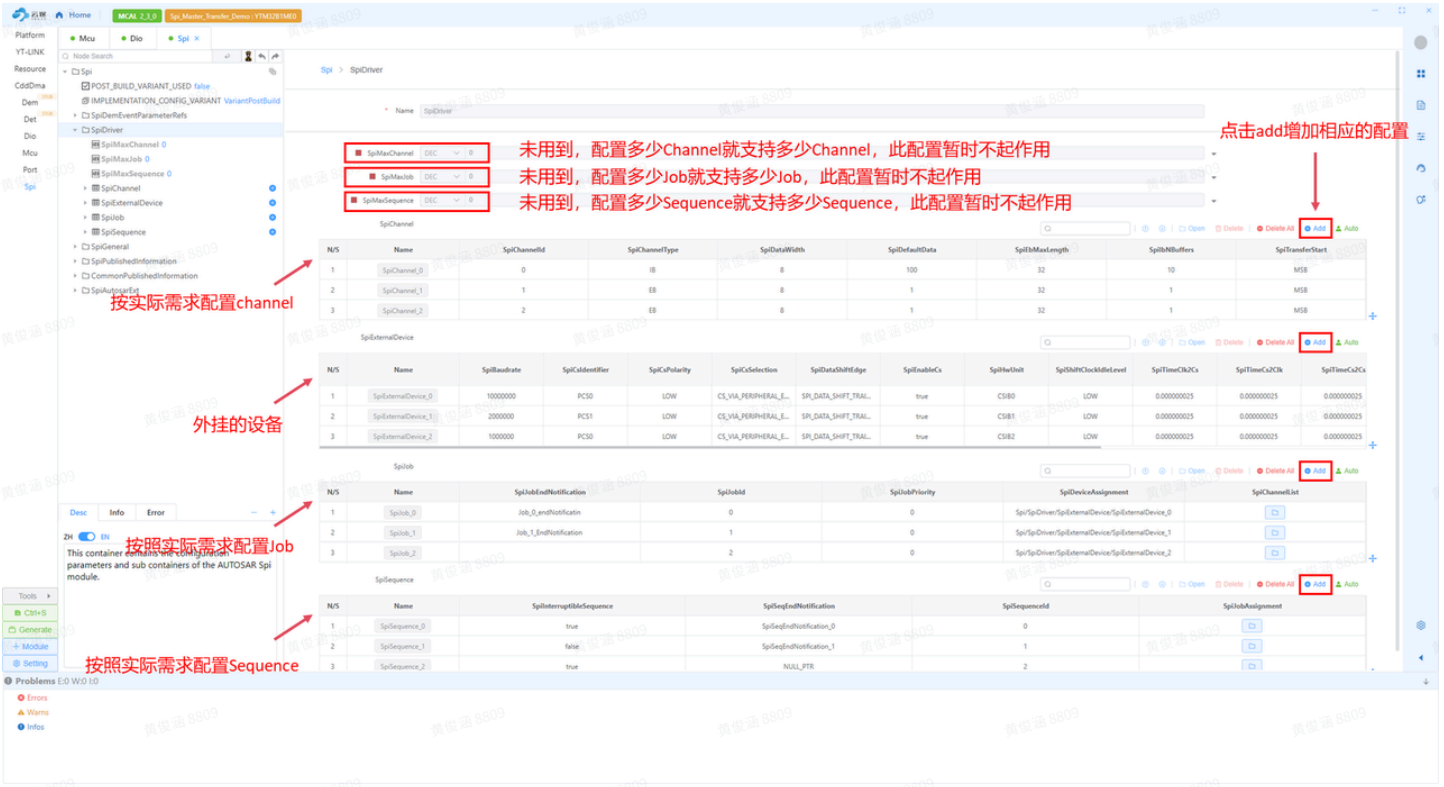


1.2 SpiDemEventParameterRefs



1.3 SpiDriver

此处为SPI的核心配置，涉及Sequence、Job、Channel以及需要通信设备的相关配置信息。



Tips:

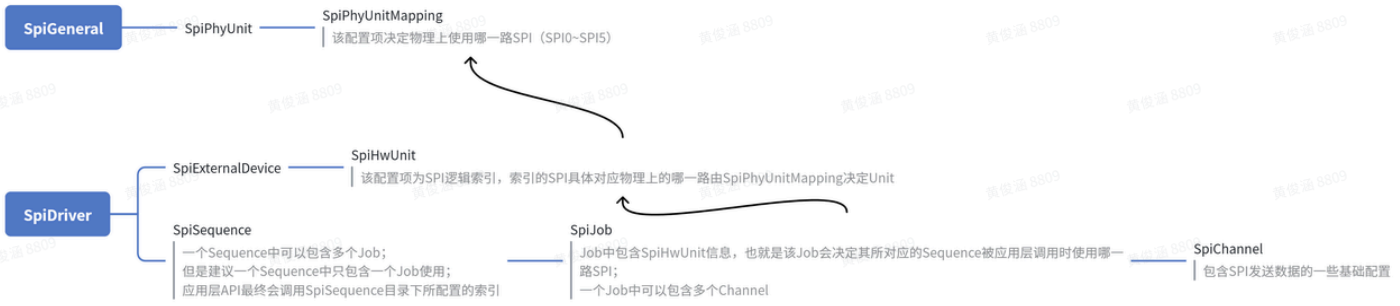
AUTOSAR 提供了三种SPI的抽象。Channel、Job、Sequence。

一个Job 代表了一次完整的SPI通信，一个Job可以由一个或者多个Channel组成，Channel用来装载SPI的发送与接收数据。

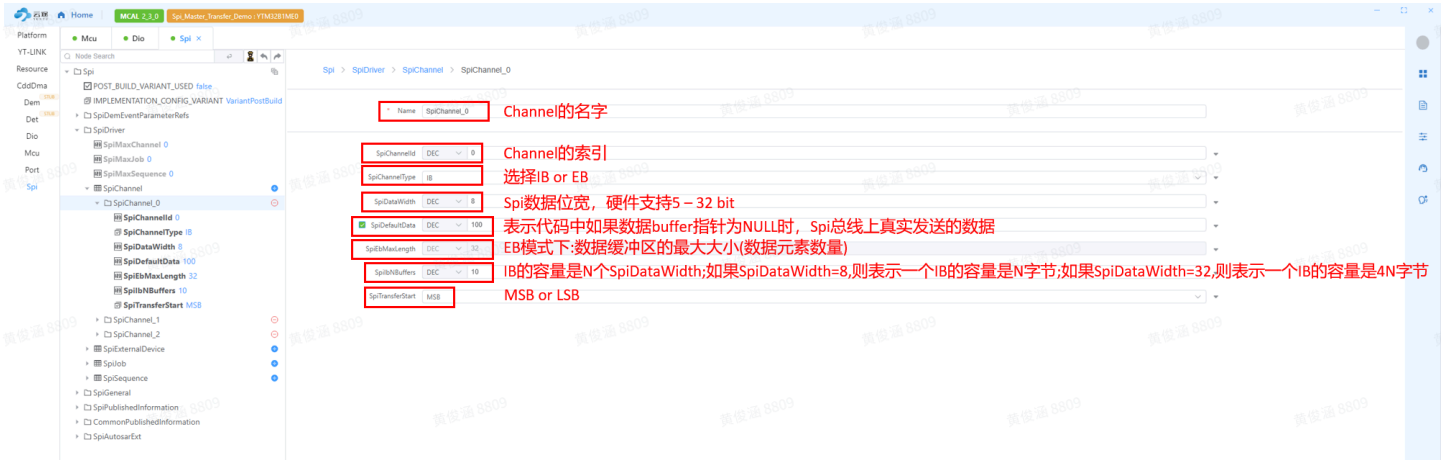
一个Sequence可以由一个或者多个Job,可以将一个操作序列抽象成一个Sequence。

Tips:

实际应用中，常常一个Sequence下挂一个Job，一个Job下挂一个Channel。



1.3.1 SpiChannel

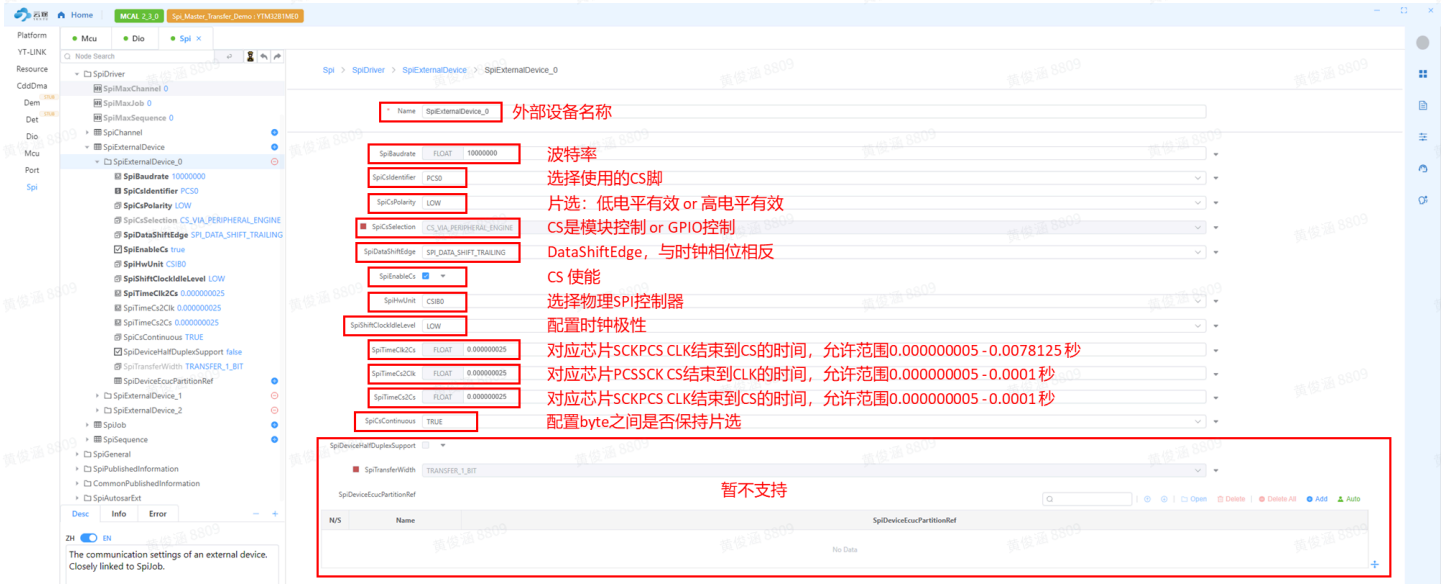


Tips:

IB：静态分配 EB：可以静态分配也可以动态分配，取决于应用场景

IB 类型 buffer 大小是固定的，EB 类型 buffer 可以通过 API 进行设置。

1.3.2 SpiExternalDevice



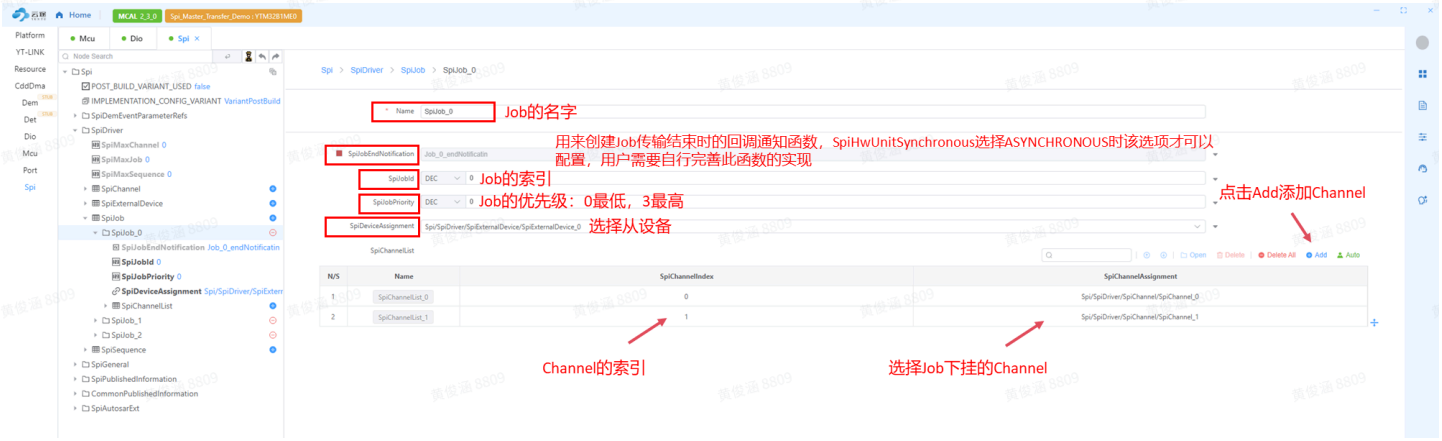
参数	调节的位置	作用
Cs2Clk	CS 拉低 → 第一个 SCK 边沿	选中从机后，等待多久开始发时钟
Cs2Cs	当前字节结束 → 下一个字节开始	控制字节之间的时钟间隔
Clk2Cs	最后一个 SCK 边沿 → CS 拉高	最后一位传完后，继续保持片选多久

代码块

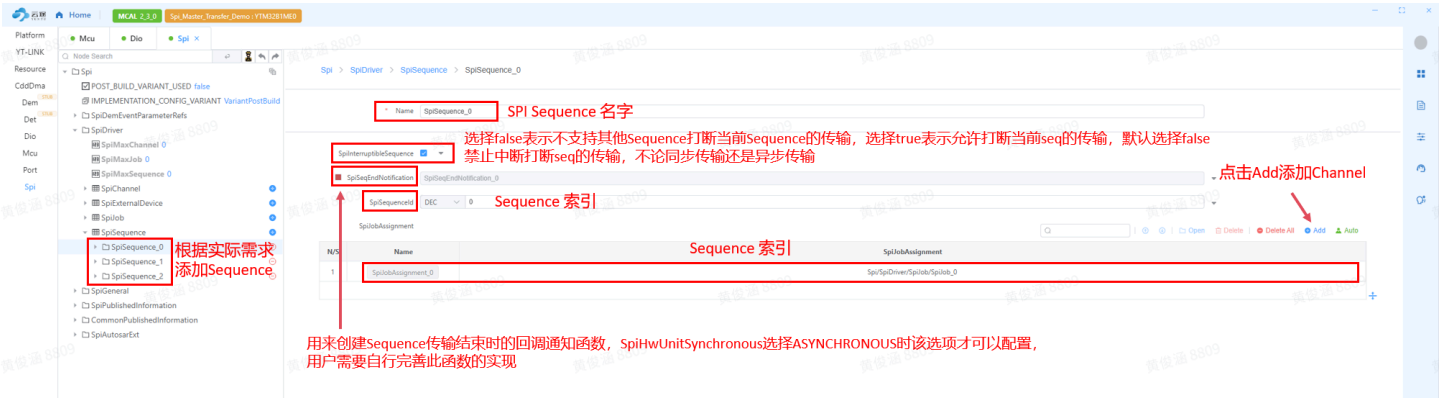
```
CS 拉低
   |
   ├── Cs2Clk ──┤
                第1字节：8个时钟
                              ├── Cs2Cs ──┤
```

```
                                                第2字节：8个时钟
                                                        ...
                                                最后一个时钟
                                                        ┣━
    Clk2Cs ━┫
                                                        ┃
        CS 拉高
```

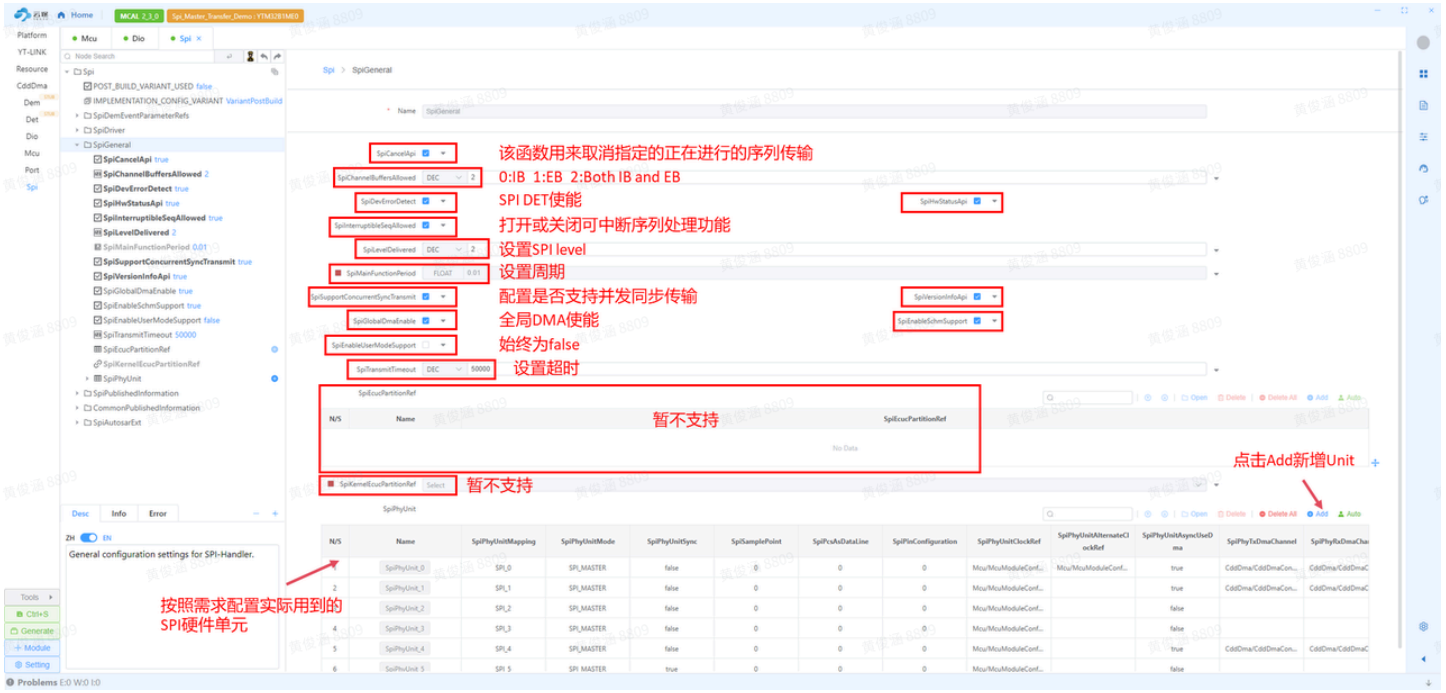
1.3.3 SpiJob



1.3.4 SpiSequence



1.4 SpiGeneral

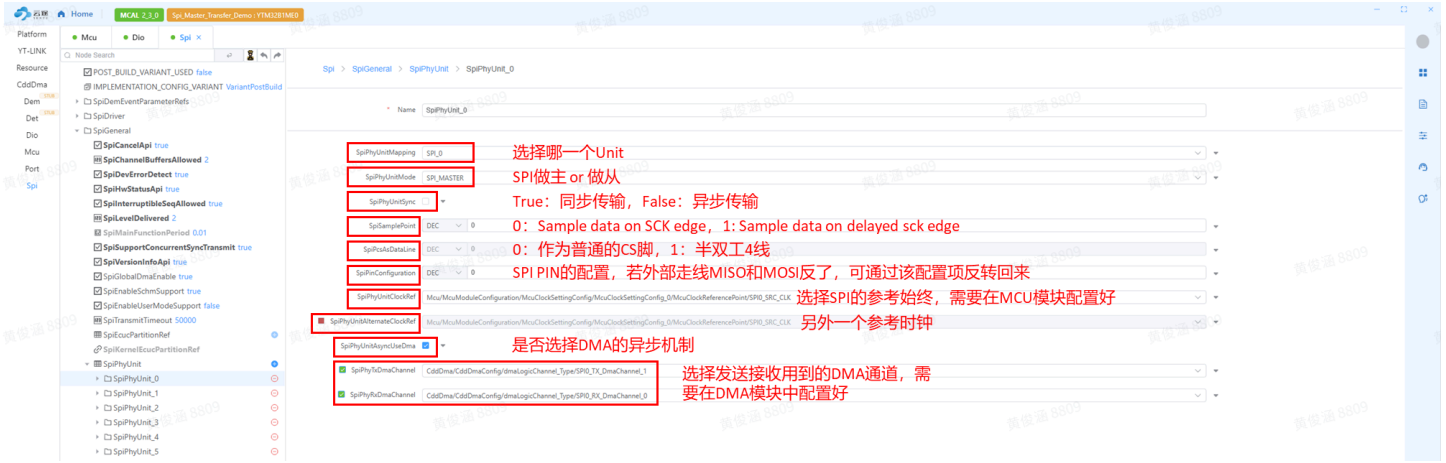


Tips:

SpiDataWidth: 支持 5bit - 32bit

如果 DataWidth 设置成了 24Bit，这种情况下，发送只能发送 24bit，忽略高8位，接收的时候也只接收 24bit，线上的波形是按照 24bit 进行通信，软件层面上需要对 buffer 进行处理。

1.5 SpiPhyUnit



SpiSamplePoint详解：

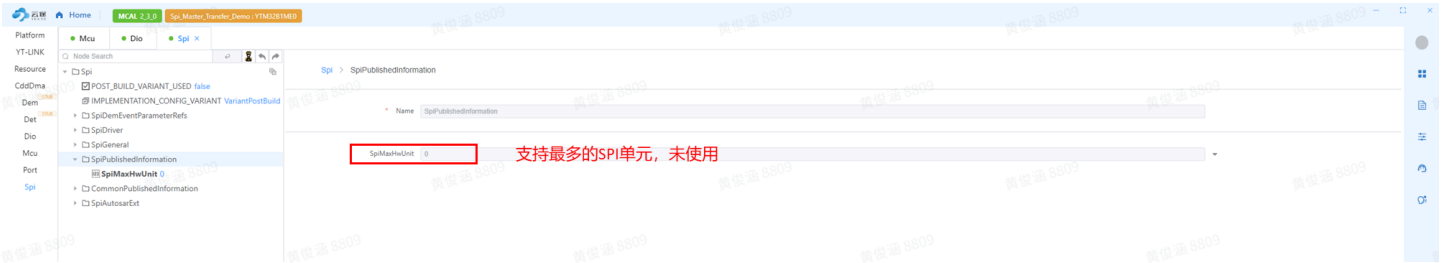
当MISO波形存在边沿缓慢的情况时，可微调此参数延迟采样

代码块

```
MOSI（主机发）  ──┐ ┌─┐ ┌─┐ ┌─┐ ┌──   理想方波
                  │ │ │ │ │ │ │
SCLK              ┘ └─┘ └─┘ └─┘
MISO（从机回）      ╲___╱‾‾‾╲___      从机内部延时 + 线路 RC + 驱动能力
                        ↑
```

7	实际数据比时钟晚到 → 按默认边沿采样可能采在"跳变沿"上

1.6 SpiPublishedInformation



1.7 CommonPublishedInformation



1.8 SpiAutosarExt



2. 示例

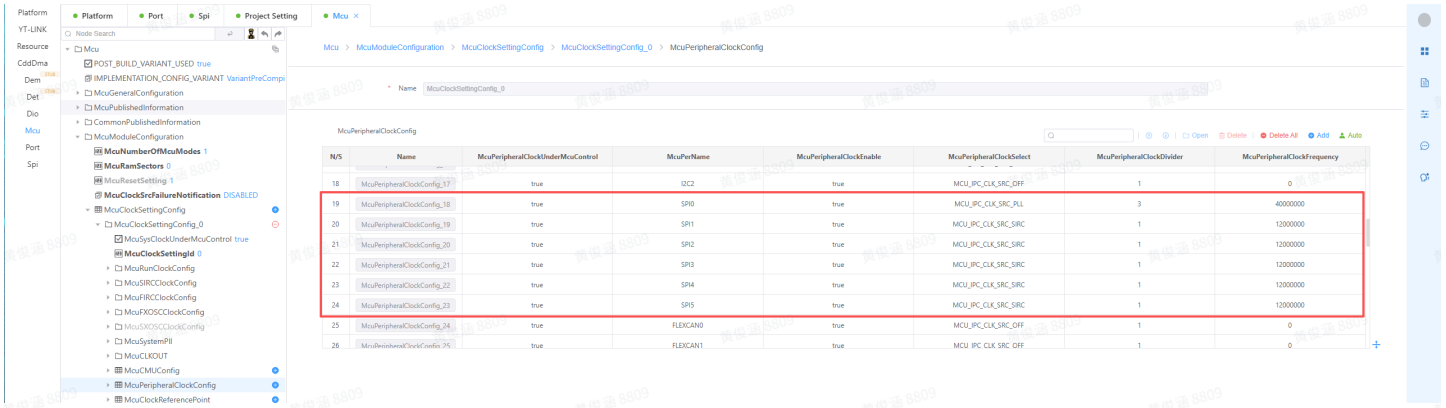
2.1 配置步骤

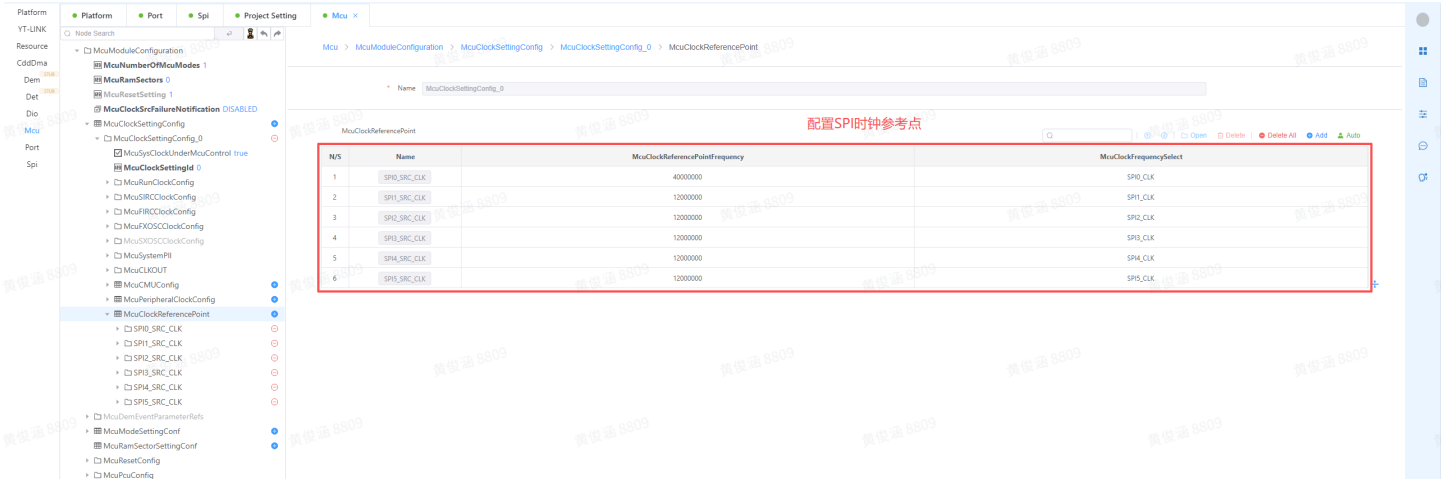
示例主要完成以下功能展示：

1. 使用 DMA 的异步传输
2. 不使用 DMA 的异步传输
3. Polling 模式的异步传输
4. 同步传输

SPI 做主机，使用上述传输模式发送 {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09}

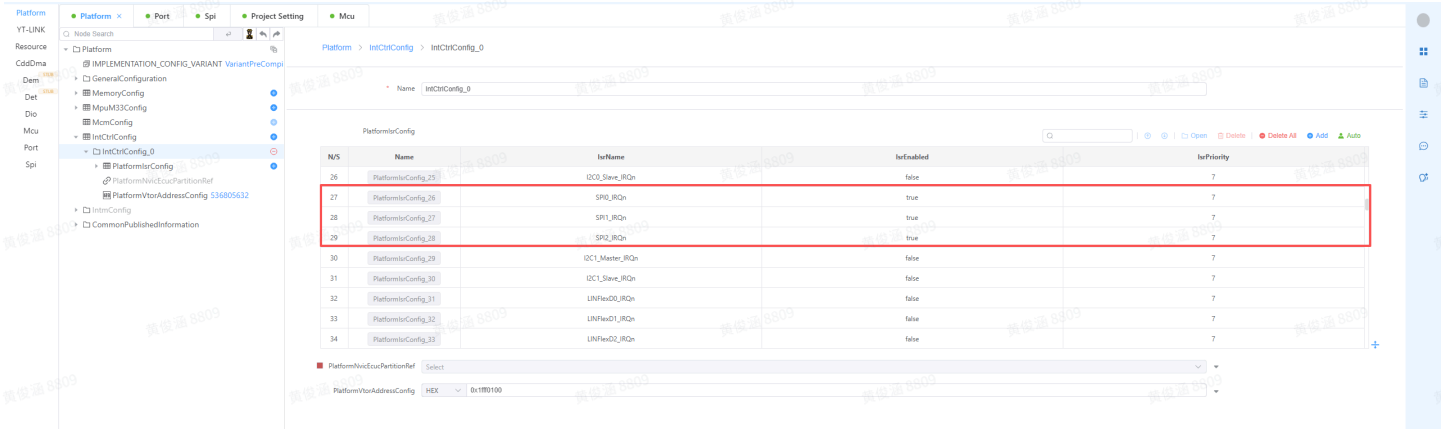
Step1：时钟配置



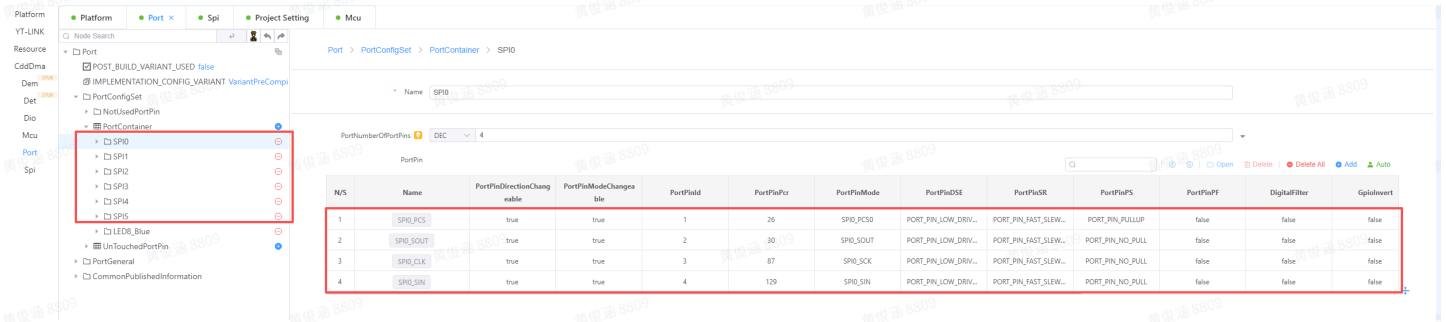


Step2：中断配置

使能对应SPI的中断，设置中断优先级

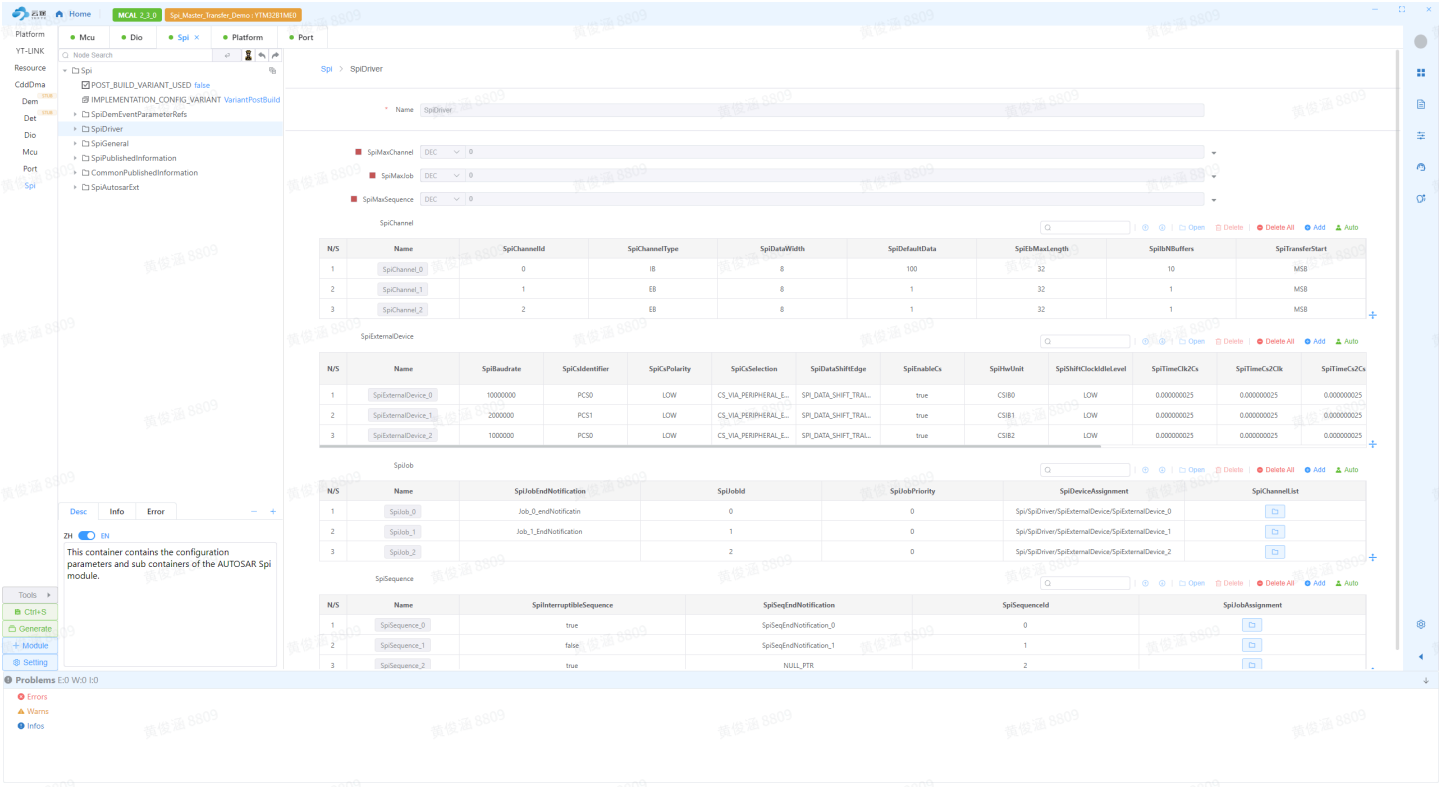


Step3：Port配置



Step4：配置SPI模块

根据需求参考上述“1.基础模块配置介绍”进行配置。



2.2 测试用例

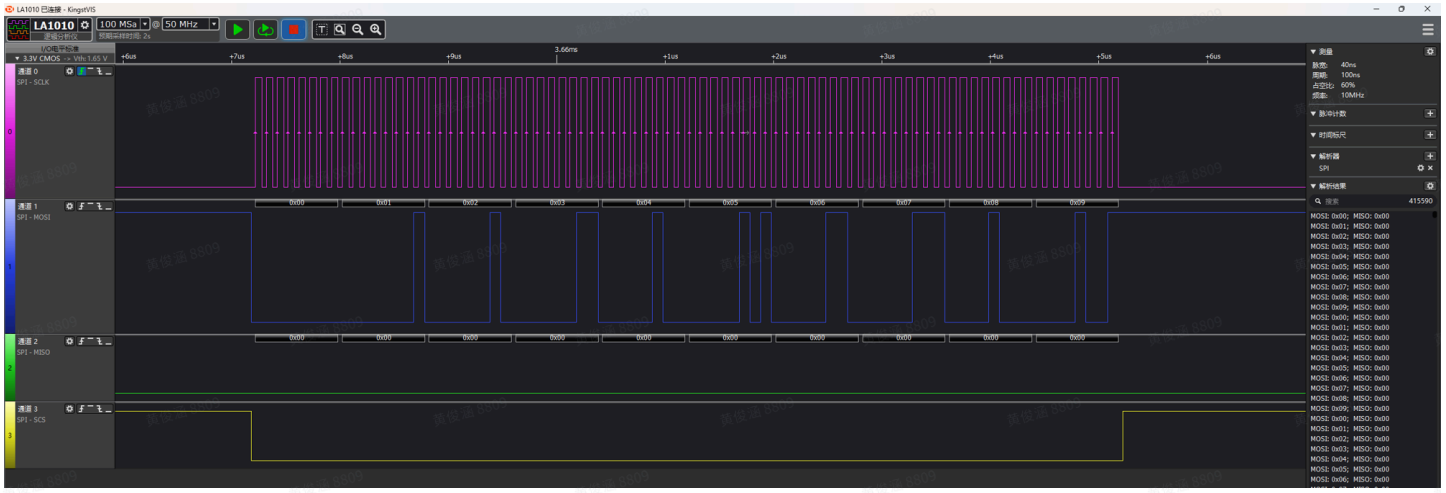
2.2.1 使用 DMA 的异步传输

以 SPI 波特率 10MHz 进行测试，主要对比在高速通信情况下使用 DMA 和不使用 DMA 时总线波形区别。

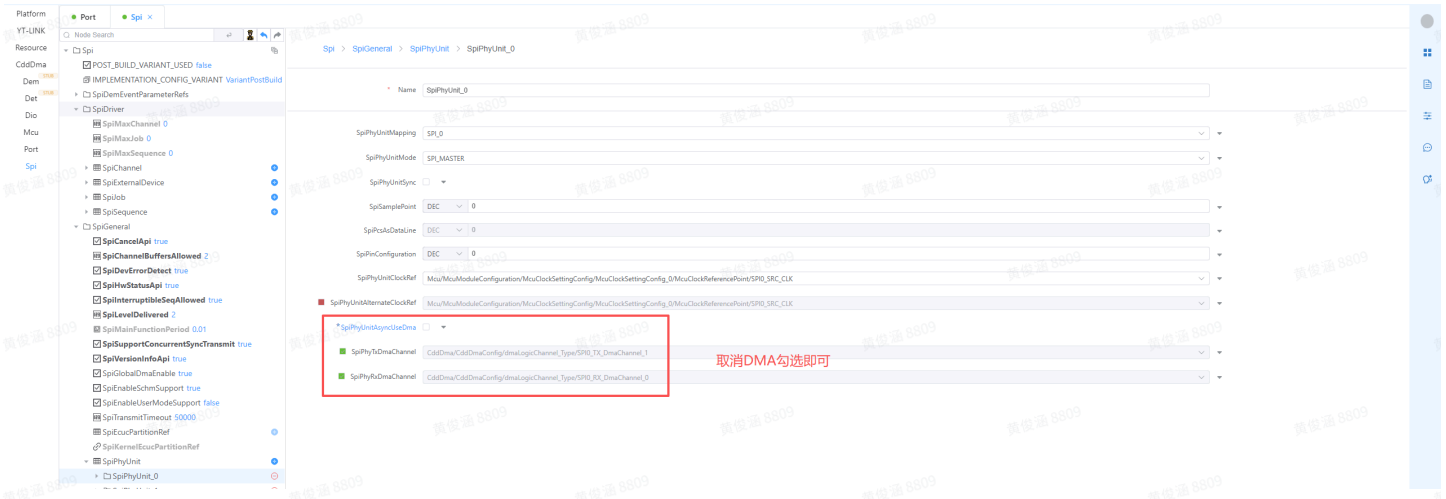



```
/*Test case for Leve2 API Asynctransmit with interrupt mode*/
Spi_SetAsyncMode(Mode: SPI_INTERRUPT_MODE);
Spi_WriteIB(Channel: SpiConf_SpiChannel_SpiChannel_0, DataBufferPtr: &IbSrcDataBuffers[0]);
Spi_AsyncTransmit(Sequence: SpiConf_SpiSequence_SpiSequence_0);
while (Spi_GetSequenceResult(Sequence: SpiConf_SpiSequence_SpiSequence_0) != SPI_SEQ_OK);
```

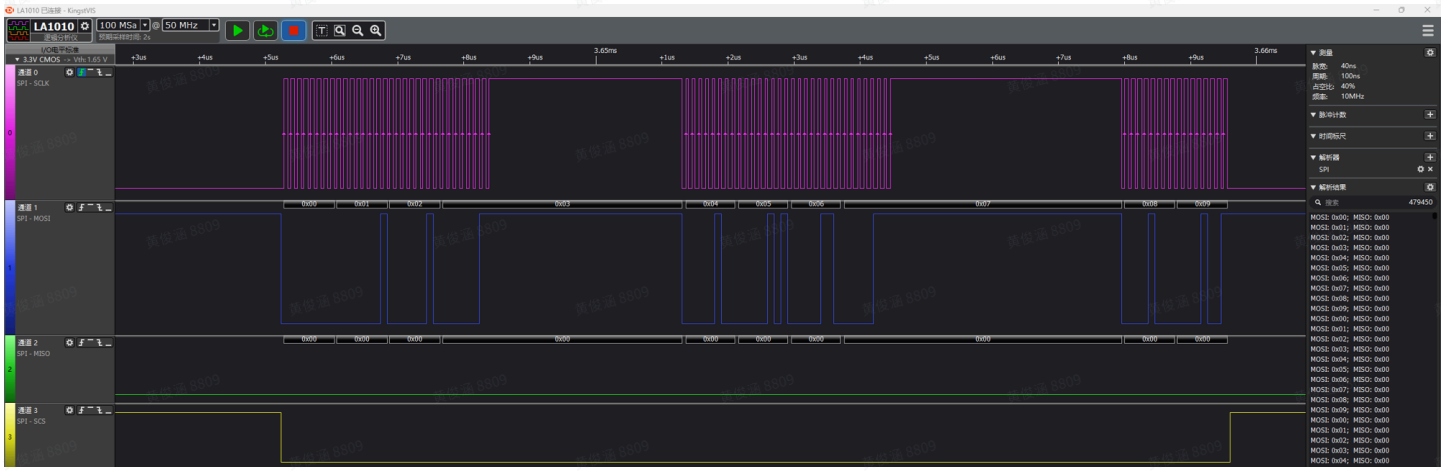
使用 10MHz 波特率通信时，波形连续。



2.2.2 不使用 DMA 的异步传输



同样使用 10MHz 波特率传输，不使用 DMA 时，同一帧数据字节之间会产生较大间隔，若从机无法容忍这种间隔则会通信故障。

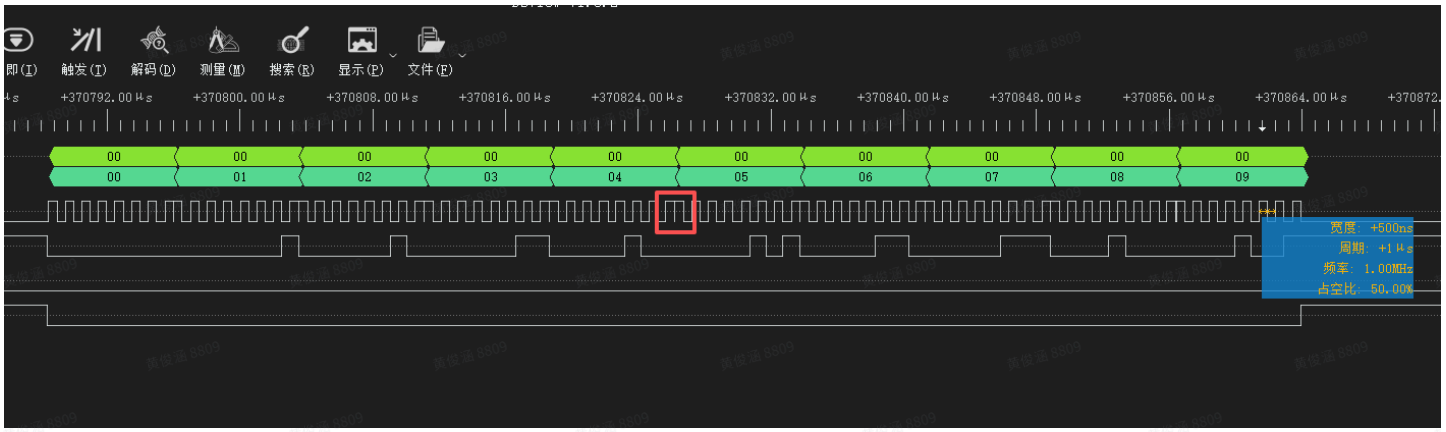


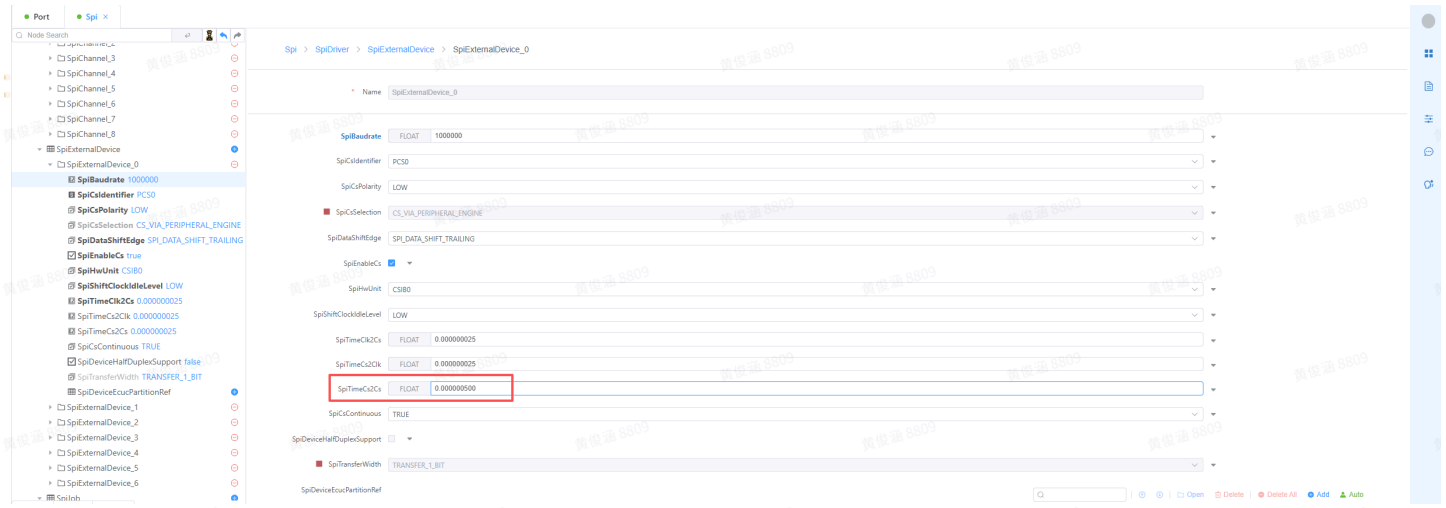
2.2.3 Polling 模式的异步传输

情况一：SPI 以波特率 1MHz 进行测试

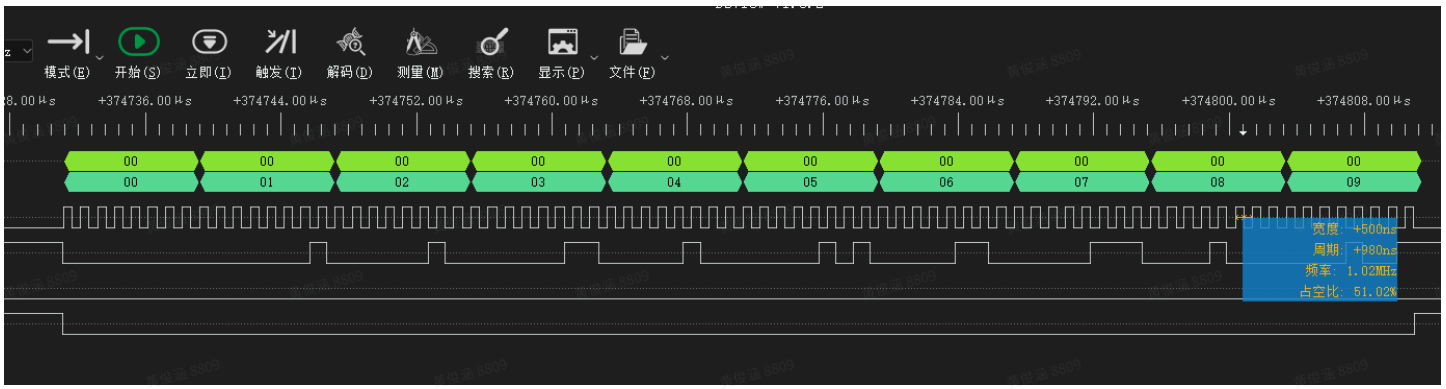
```
// /*Test case for Leve2 API Asynctransmit with polling mode*/
Spi_SetAsyncMode(Mode: SPI_POLLING_MODE);
Spi_AsyncTransmit(Sequence: SpiConf_SpiSequence_SpiSequence_0);
while (Spi_GetSequenceResult(Sequence: SpiConf_SpiSequence_SpiSequence_0) != SPI_SEQ_OK)
{
    Spi_MainFunction_Handling();
}
```

字节之间时钟可能会贴的很近，通常不影响通信，如果对此有要求则缩放SpiTimeCs2Cs的值来调节字节间的间距。

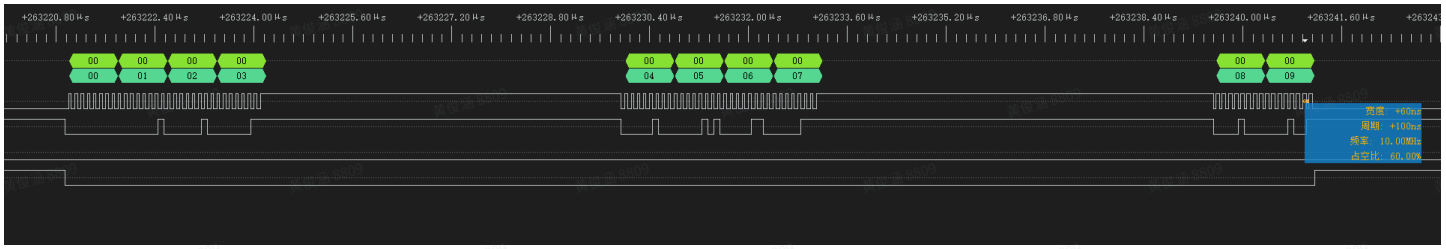




调整后如下图，字节间隔被加大。



情况二：以 10Mhz 进行测试



这是 Polling 补充 TX FIFO 不及时造成的停顿，SPI FIFO 物理宽度为4字节，底层会将发送数据组成4字节再填入 FIFO 提升发送效率。故在 10 MHz 下，一个字节只需约 **0.8 µs**，4 字节约 **3.2 µs**。CPU 如果没在这段时间内补充数据，SCK 就会暂停；图中的间隙约 **5.84 µs**。

所以这个间隔是无法优化的，高速SPI建议使用DMA。

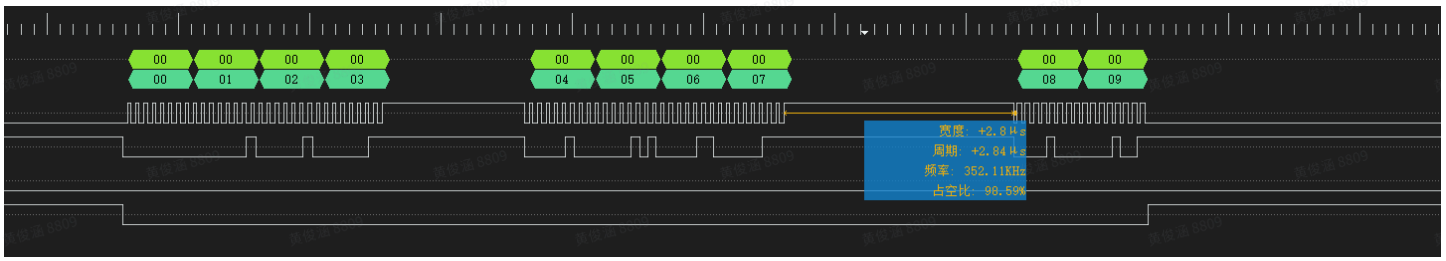
```
CPU 填入一批数据 → 硬件发送 → FIFO 耗尽 → 等 CPU 再填入 → 继续发送
                   00～03                  04～07
```

2.2.4 同步传输

以 SPI 波特率 10MHz 进行测试

同步模式下以高速传输时每4个字节间仍会有间隔，间隔约 2.84us。





```
// /*Test case for Level0 API Synctransmit*/
Spi_WriteIB(Channel: SpiConf_SpiChannel_SpiChannel_8, DataBufferPtr: &IbSrcDataBuffers[10]);
Spi_SyncTransmit(Sequence: SpiConf_SpiSequence_SpiSequence_5);
while (Spi_GetSequenceResult(Sequence: SpiConf_SpiSequence_SpiSequence_5) != SPI_SEQ_OK);
```

注意：

同步模式发送函数为：Spi_SyncTransmit()

异步模式发送函数为：Spi_AsyncTransmit()

文档历史

版本号	日期	修订记录
V1.0	2024.01.04	初始版本
V2.0	2026.05.11	1. 更新配置工具YCT版本为3.0.0及MCAL版本为2.3.0； 2. 增加SpiDriver层级关联介绍
v3.0	2026.11.06	1. 针对部分配置项增加更详细的描述 2. 增加5个测试用例