MCAL应用_CddI2c模块配置及实例(一)

版本

Config Tool Version：1.10.0

ME0 MCAL version：1.3.1

前言

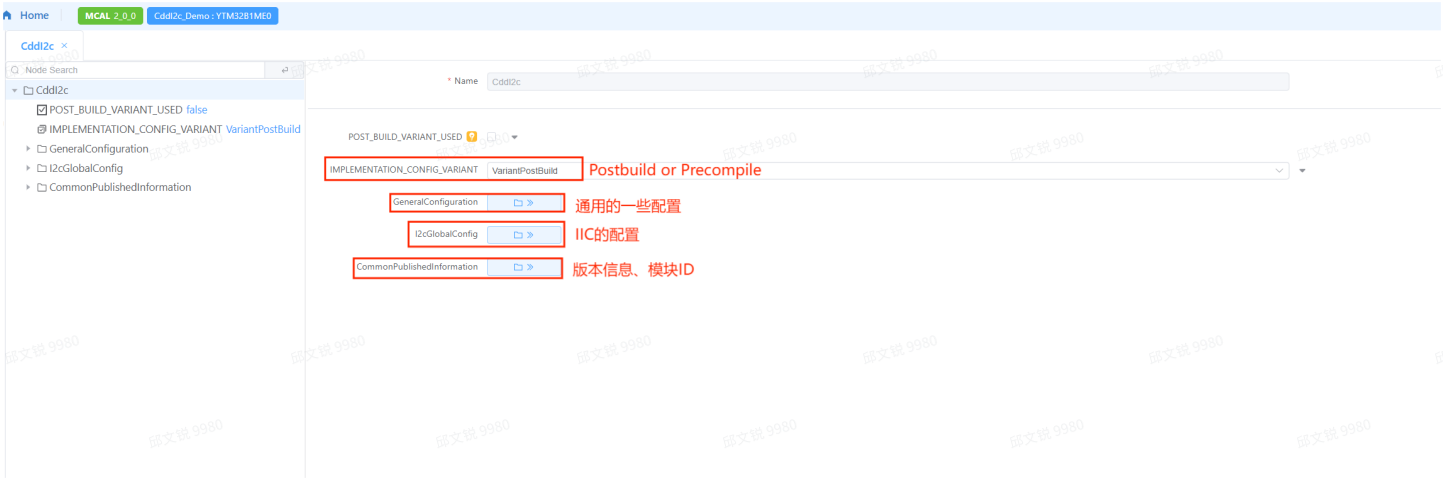
1. CddI2C一般依赖Mcu、Platform和Port模块，Mcu模块里面需要配置外设时钟，Platform里面可以配置外设中断，Port配置使用到的端口资源 ，如果选择DMA方式则需要另外添加CddDMA模块。
2. Port、Mcu、CddDMA和Platform的默认配置参考Port篇、Setup篇、CddDMA篇和Mcu篇介绍。
3. 配置中包含AutoSar规范标准选项和云途自定义选项，左边树选框中粗体字为AutoSar标准选项，细体字为云途自定义选项。

1. CddI2C配置介绍

1.1 如何配置IIC模块

1.1.1 框架说明

此处主要介绍了整个I2C模块的框架，需要我们着重注意的是I2cGlobalConfig。



1.1.2 GeneralConfiguration

此处是通用的一些配置，按照需求配置即可。

1.2 I2cGlobalConfig

1.2.1 I2cChannel

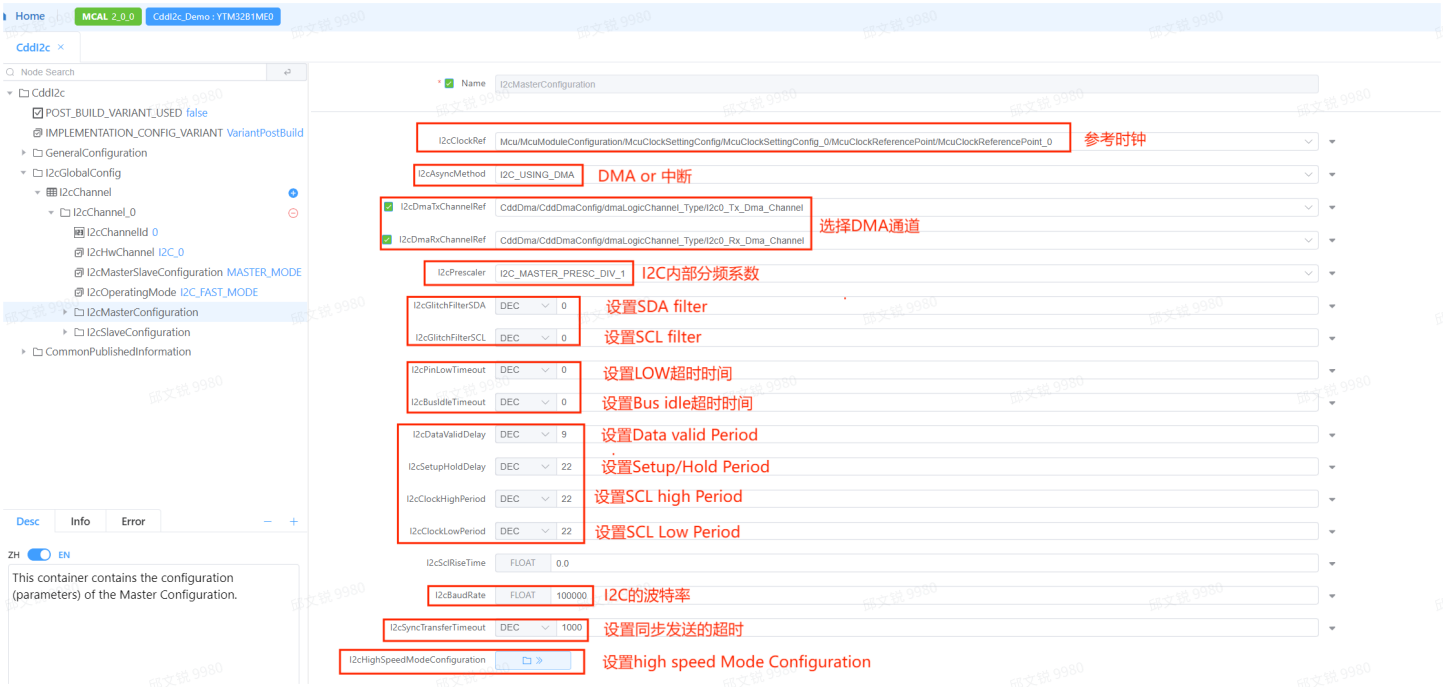
此处是设置I2C模块的具体参数，如果使用多个I2C模块，则建多个IIC即可

Tips:

I2C Operate Mode： I2C_Standard_Mode、I2C_Fast_Mode、I2C_FASTPLUS_MODE、I2C_HighSpeed_Mode

1.2.2 I2cMasterConfiguration

此处配置I2C作为主节点时候的参数（需要在I2Cchannel界面选择主模式）。



Tips:

SDA Filter、SCLFilter对应下图寄存器，小于配置值*模块时钟的信号会被过滤掉

Table 21.8: I2C MFLTCFG Register Description

Field	Function
27 - 24 FLTSCL	**SCL Filter Control** Control digital filter on SCL line SCL digital filter use undivided I2C function clock Noise less or equal FLTSCL * function period will be filtered
19 - 16 FLTSDA	**SDA Filter Control** Control digital filter on SDA line SDA digital filter use undivided I2C function clock Noise less or equal FLTSDA * function period will be filtered

Tips: 如何计算IIC的波特率，按照下图进行计算

```
Calculated as Frequency/(((CLKLO+CLKHI+2)*2^PRESCALER)+ROUNDDOWN((2+FILTSCL)/2^PRESCALER))
```

Tips:

IDLE Timeout、PINLow Timeout对应下图寄存器，计算公式如图所示

Table 21.15: I2C TOCFG Register Description

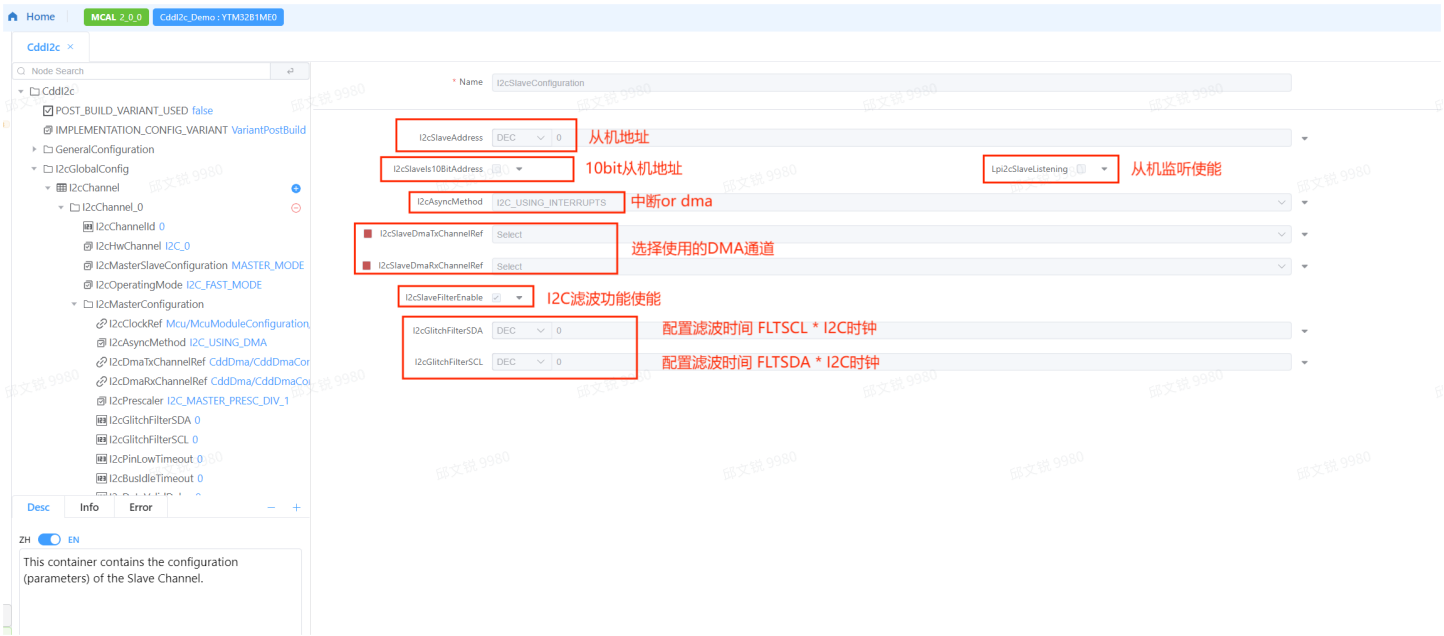
Field	Function
27 - 16 IDLE	**Line Idle Timeout Control** Line idle flag set when SDA and SCL line high for (IDLE) * divided function clock, after line idle, master can send out START.
15 SDA	**SDA Line Low Detect** Set this bit to enable SDA line low detection 0b - Line low only detect SCL 1b - Line low detect SDA and SCL
11 - 0 LOW	**Line Low Timeout Control** Line low period is (LOW * 256) * divided function clock

1.2.2.1 I2cHighSpeedModeConfiguration

此处配置HIghSpeedMode时的波特率，需要在I2cChannel界面选择HIghSpeedMode。

1.2.3 I2cSlaveConfiguration

此处配置I2C作为从节点时候的参数（需要在I2Cchannel界面选择从模式）。



1.3 CommonPublishedInformation

此处主要是版本信息、AutoSar标准、模块ID、供应商ID信息。需要注意的是，I2C并不是标准的MCAL模块，属于

复杂驱动，这里我们设置的ModuleID为255。

2. I2C模块应用实例

2.1 应用场景

- 基于ME0 Demo板，使用I2C0和Demo板上的Eerpom进行通信。

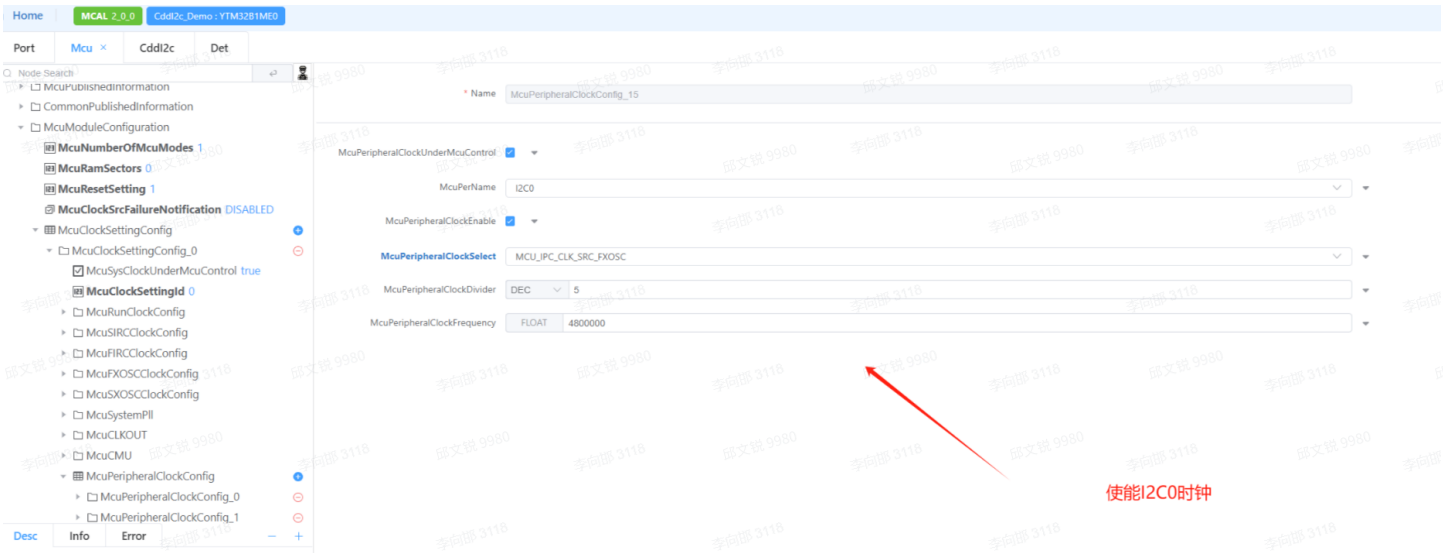
- 设置通信波特率为100K。从地址0x50，写入数据为0x18, 0x20, 0x31, 0x47, 0x5, 0x6, 0x7
- Port上选择

  PTA3 -- I2C0_SCL

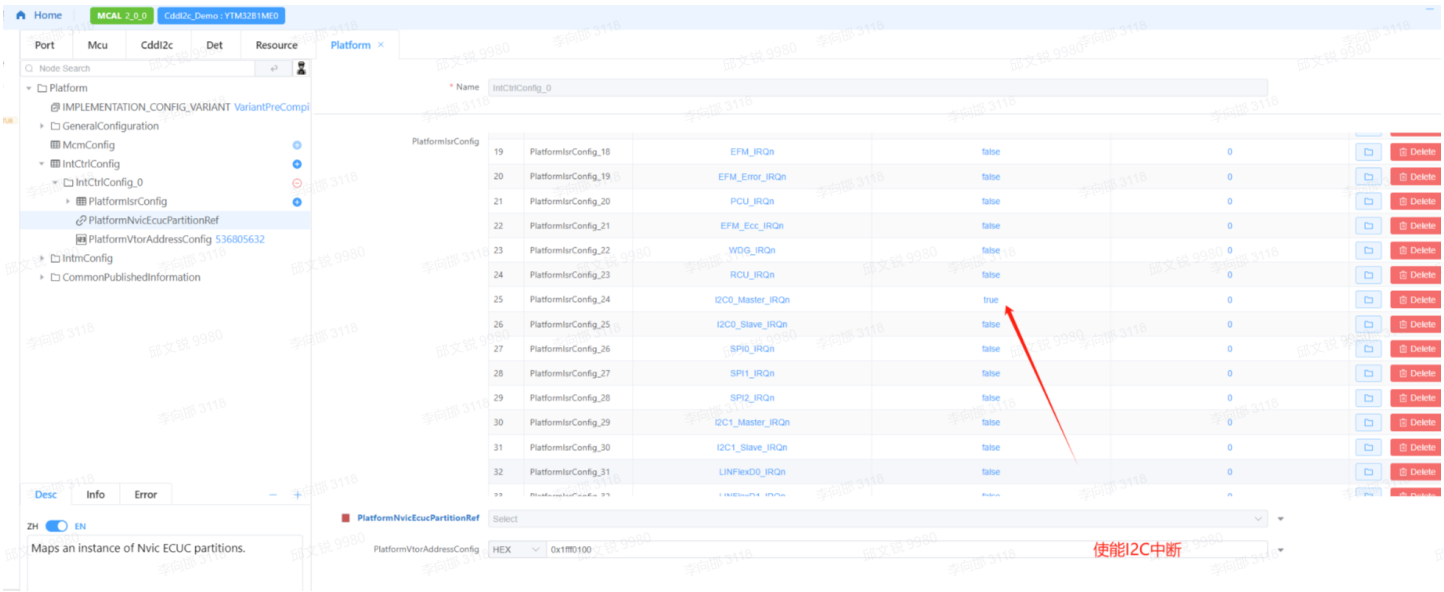
  PTA2 -- I2C0_SDA

  PTB11 -- EEP_WP Eeprom芯片唤醒脚
- 逻辑分析仪抓取PTA3 、PTA2上的波形，软件验证读写buffer一致。

2.1.1 时钟配置

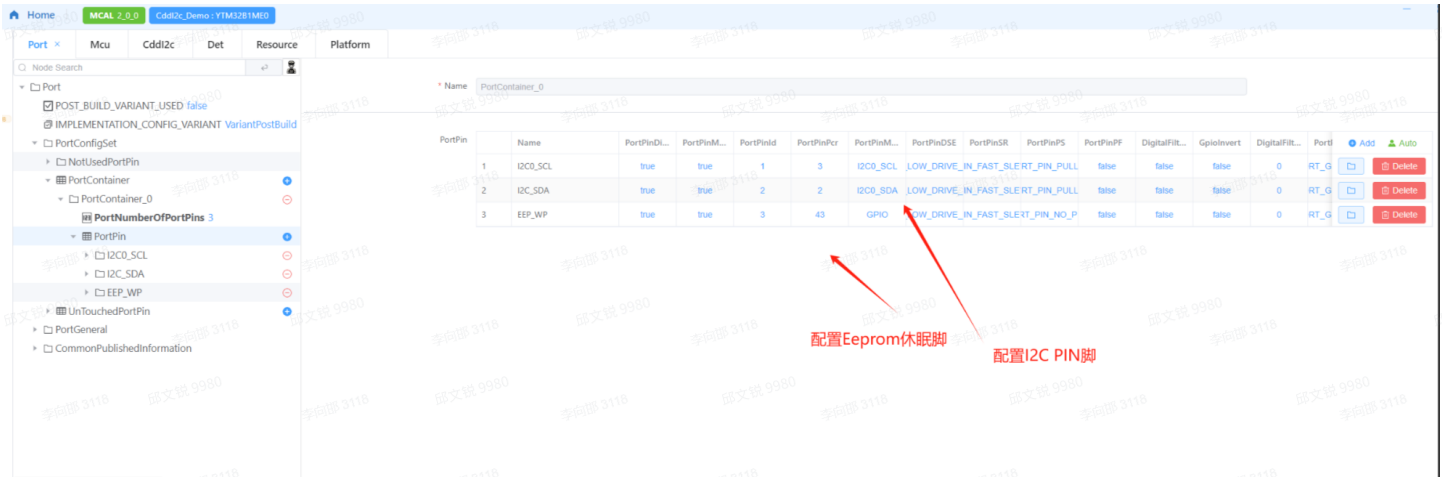


2.1.2 中断配置

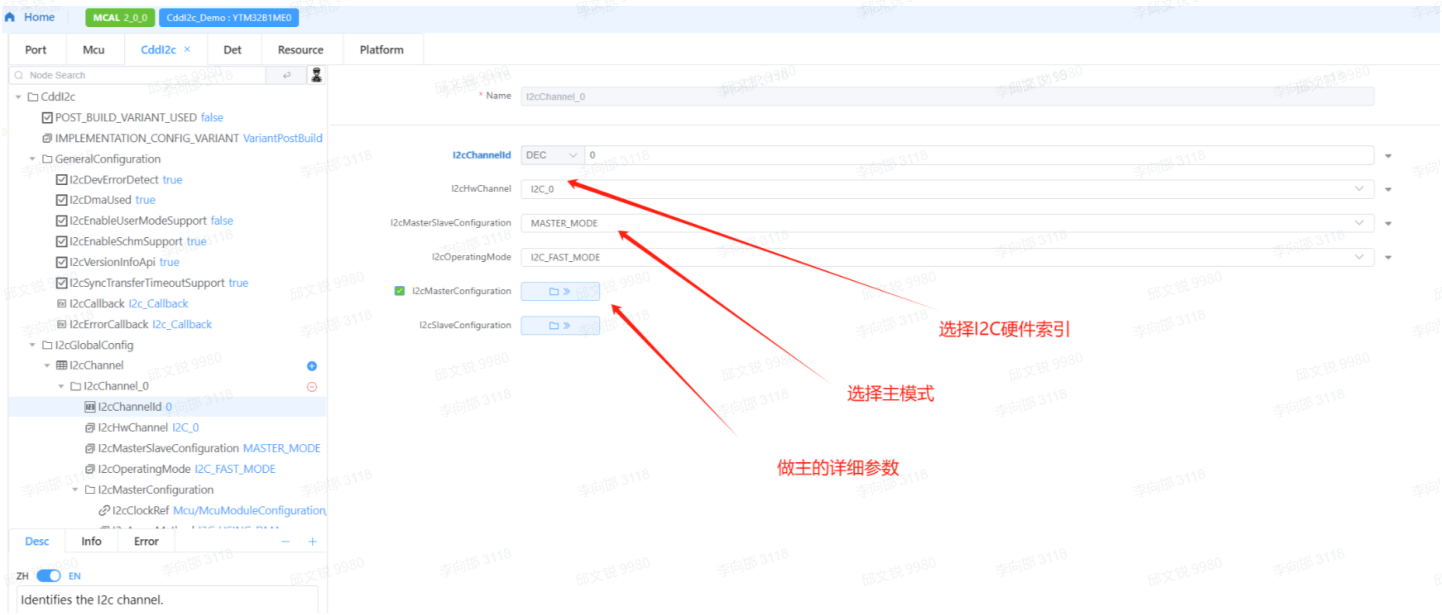


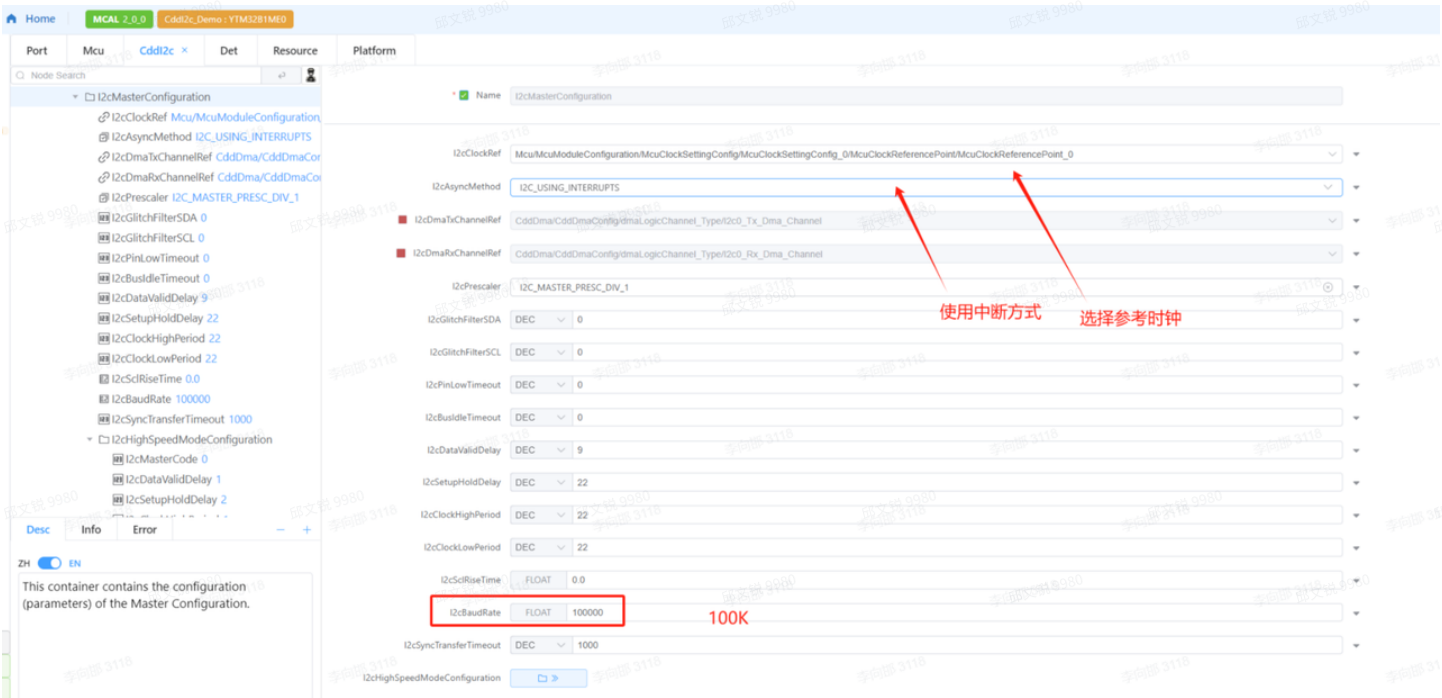
2.1.3 Port配置

这里配置PTA3、PTA2、PTB11。



2.1.4 配置I2C模块





2.2 API说明

这里我们用同步接口往Eeprom写数据，用异步接口读取之前写到的数据。

```
#endif
    /* USER CODE END 1 */
    Board_Init();
    /* USER CODE BEGIN 2 */
    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    /*write data to EEPROM in EVB with the I2C0 */
#if (I2C_TX_MODE == I2C_TRANS_IN_SYNCMODE)
    CddI2c_SyncModeTransfer(CddI2cConf_I2cChannel_I2cChannel_0, &CDDI2cRequestTx);

    TempRav = CddI2c_GetTransferStatus(CddI2cConf_I2cChannel_I2cChannel_0);
#else
    CddI2c_AsyncModeTransfer(CddI2cConf_I2cChannel_I2cChannel_0, &CDDI2cRequestTx);
    do
    {
        TempRav = CddI2c_GetTransferStatus(CddI2cConf_I2cChannel_I2cChannel_0);
    } while (I2C_CHN_COMPLETED != TempRav);
#endif

    /*delay 10ms*/
    i = 50000;
    while (i--);

    /*Read data from EEPROM in EVB with the I2C0*/
#if (I2C_RX_MODE == I2C_TRANS_IN_ASYNCMODE)
    CDDI2cRequestTx.BufferSize = 1;
    CddI2c_AsyncModeTransfer(CddI2cConf_I2cChannel_I2cChannel_0, &CDDI2cRequestTx);
    do
    {
        TempRav = CddI2c_GetTransferStatus(CddI2cConf_I2cChannel_I2cChannel_0);
    } while (I2C_CHN_COMPLETED != TempRav);

    CddI2c_AsyncModeTransfer(CddI2cConf_I2cChannel_I2cChannel_0, &CDDI2cRequestRx);
    do
    {
        TempRav = CddI2c_GetTransferStatus(CddI2cConf_I2cChannel_I2cChannel_0);
    } while (I2C_CHN_COMPLETED != TempRav);
#else
    CDDI2cRequestTx.BufferSize = 1;
    CddI2c_SyncModeTransfer(CddI2cConf_I2cChannel_I2cChannel_0, &CDDI2cRequestTx);
    CddI2c_SyncModeTransfer(CddI2cConf_I2cChannel_I2cChannel_0, &CDDI2cRequestRx);

    TempRav = CddI2c_GetTransferStatus(CddI2cConf_I2cChannel_I2cChannel_0);

#endif
    /* Verify the recieve data is right or not*/
    while (BufferCompare(&TestTxBuffer[1], &TestRxBuffer[0], 7) == FALSE);

    CddI2c_DeInit();
```

delay一段时间

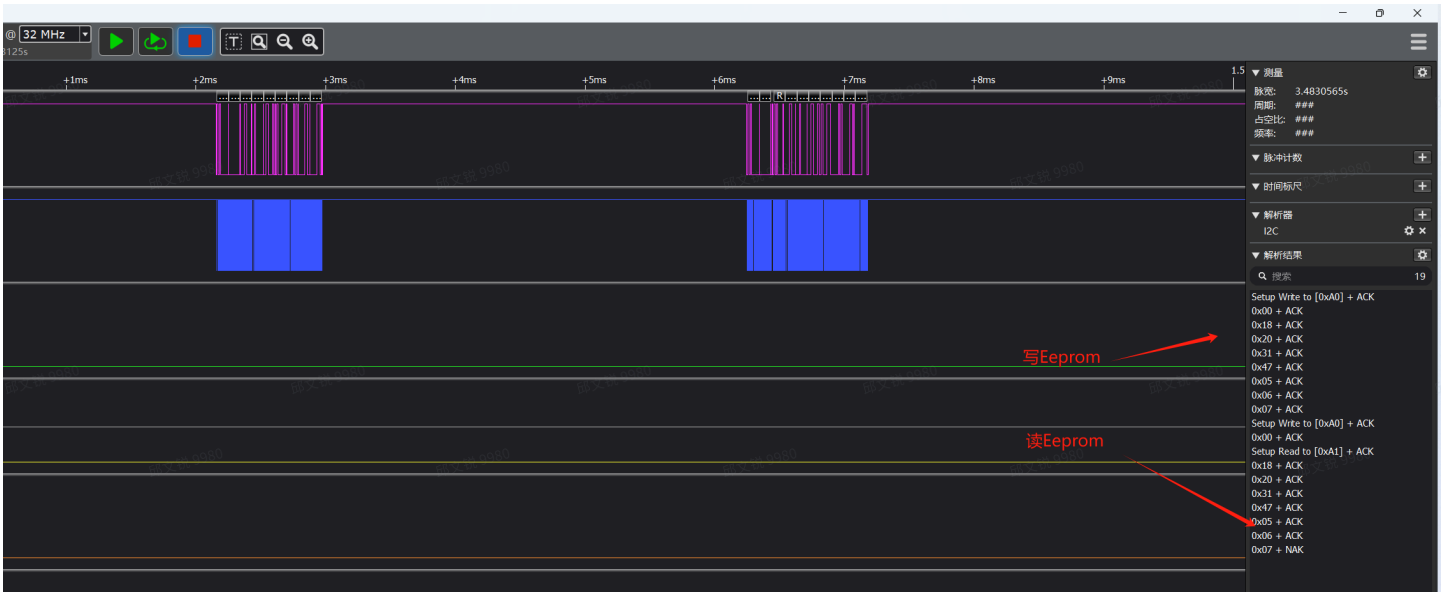
使用同步接口写数据

使用异步接口读数据

校验读写buffer一致

2.3 现象

线上抓到的数据，与预期一致



文档历史

版本号	日期	修订记录
V1.0	2024.01.04	初始版本
