

用文本编辑工具打开 mex 文件，可以看到 mex 文件本质上就是用 xml 格式写的配置信息。经过研究 MEX 文件的各个字段，同时对比最新的 2.0.0 D2203 版本，和我举例的 avb demo(旧版本 0.8.0 D2008)，在同样的 S32K344 上，可以发现一些差异，主要不同如下：

1. 版本号、MCU 号命名格式，

本文以 0.8.0 和 1.0.0 问题举例：

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration name="S32K344" xsi:schemaLocation="http://mcuxpresso.nxp.com/XSD/mex_configur
  <common>
    <processor>S32K344</processor>
    <package>S32K344_2578GA</package>
    <mcu_data>PlatformSDK_S32K3_2022_03</mcu_data>
    <cores selected="core0">
      <core name="Cortex-M7" id="core0" description="" />
    </cores>
    <description></description>
  </common>
  <preferences>
    <validate_boot_init_only>true</validate_boot_init_only>
    <generate_extended_information>false</generate_extended_information>
    <generate_code_modified_registers_only>false</generate_code_modified_registers_only>
    <update_include_paths>true</update_include_paths>
  </preferences>
  <tools>
    <pins name="Pins" version="10.0" enabled="true" update_project_code="true">
      <generated_project_files>
        <file path="board/Siul2_Port_Ip_Cfg.c" update_enabled="true"/>
        <file path="board/Siul2_Port_Ip_Cfg.h" update_enabled="true"/>
        <file path="board/Tspc_Port_Ip_Cfg.c" update_enabled="true"/>
        <file path="board/Tspc_Port_Ip_Cfg.h" update_enabled="true"/>
      </generated_project_files>
    </pins>
  </tools>
</configuration>
```

2.0.0

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration name="S32K344" xsi:schemaLocation="http://mcuxpresso.nxp.com/XSD/mex_configu
  <common>
    <processor>S32K344</processor>
    <package>S32K344_2578GA</package>
    <mcu_data>SW32K344_RT_D4_4_080_D2008</mcu_data>
    <cores selected="core0">
      <core name="Cortex-M7" id="core0" description="" />
    </cores>
    <description></description>
  </common>
  <preferences>
    <validate_boot_init_only>true</validate_boot_init_only>
    <generate_extended_information>false</generate_extended_information>
    <generate_code_modified_registers_only>false</generate_code_modified_registers_only>
  </preferences>
  <tools>
    <pins name="Pins" version="8.0" enabled="true" update_project_code="true">
      <generated_project_files>
        <file path="board/Siul2_Port_Ip_Cfg.c" update_enabled="true"/>
        <file path="board/Siul2_Port_Ip_Cfg.h" update_enabled="true"/>
      </generated_project_files>
    </pins>
  </tools>
</configuration>
```

0.8.0

可以看到，XML 对应的标签 <common> 是完全一样的，核心是版本号的不同。

2. Pin 定义

```

</pins_profile>
<pins_profile>
  <processor_version>0.0.0</processor_version>
  <pin_labels>
    <pin_label pin_num="D10" pin_signal="PTF8" label="LED1" identifier="LED1"/>
    <pin_label pin_num="B15" pin_signal="PTF9" label="LED2" identifier="LED2"/>
    <pin_label pin_num="B17" pin_signal="PTF10" label="LED3" identifier="LED3"/>
    <pin_label pin_num="H11" pin_signal="PTF11" label="LED4" identifier="LED4"/>
    <pin_label pin_num="M15" pin_signal="PTA6" label="CAN0_RX" identifier="CAN0_RX"/>
    <pin_label pin_num="M16" pin_signal="PTA7" label="CAN0_TX" identifier="CAN0_TX"/>
    <pin_label pin_num="A11" pin_signal="PTA15" label="XS65_CS" identifier="XS65_CS"/>
    <pin_label pin_num="A12" pin_signal="PTA16" label="MC33879_CS" identifier="MC33879_CS"/>
    <pin_label pin_num="L17" pin_signal="PTA17" label="LPSP13_SOUT" identifier="LPSP13_SOUT;SP
    <pin_label pin_num="C1" pin_signal="PTA19" label="SPI1_SCK" identifier="SPI1_SCK"/>
    <pin_label pin_num="E4" pin_signal="PTA20" label="VPWR_MON" identifier="VPWR_MON"/>
    <pin_label pin_num="D1" pin_signal="PTA21" label="FS26_CS" identifier="FS26_CS"/>
    <pin_label pin_num="C3" pin_signal="PTA22" label="CAN1_RX" identifier="CAN1_RX"/>
    <pin_label pin_num="D3" pin_signal="PTA23" label="CAN1_TX" identifier="CAN1_TX"/>
    <pin_label pin_num="L2" pin_signal="PTA26" label="EMAC_PPS0" identifier="EMAC_PPS0"/>
    <pin_label pin_num="M2" pin_signal="PTA27" label="UART0_TX" identifier="UART0_TX"/>
  </pin_labels>
</pins_profile>

```

2.0.0

```

</pins_profile>
<pins_profile>
  <processor_version>0.0.0</processor_version>
  <pin_labels>
    <pin_label pin_num="B14" pin_signal="PTA1" label="LED" identifier="LED"/>
    <pin_label pin_num="P1" pin_signal="PTA31" label="Gpio_test0" identifier="Gpio_test0"/
    <pin_label pin_num="P3" pin_signal="PTA30" label="Gpio_test1" identifier="Gpio_test1"/
  </pin_labels>
</pins_profile>

```

0.8.0

可以看到 pin 定义部分，xml 标签也是完全一样的，<pin_labels>的定义格式都完全一样，不同仅仅是具体的 pin 配置。所以我们可以大胆的得出结论，CT 工具能够识别出旧版本的 pin 配置，兼容性问题不是这里引起的。

3. Clock 配置

```

<clock_sources>
  <clock_source id="FXOSC_CLK.FXOSC_CLK.outFreq" value="16 MHz" locked="false" enabled="t
  <clock_source id="SXOSC_CLK.SXOSC_CLK.outFreq" value="32.768 kHz" locked="false" enable
  <clock_source id="external_clocks.emac_mii_rmii_tx.outFreq" value="50 MHz" locked="fals
</clock_sources>
<clock_outputs>
  <clock_output id="ADC0_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>
  <clock_output id="ADC1_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>
  <clock_output id="ADC2_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>
  <clock_output id="AIPS_PLAT_CLK.outFreq" value="80 MHz" locked="true" accuracy="0.001"/
  <clock_output id="AIPS_SLOW_CLK.outFreq" value="40 MHz" locked="true" accuracy="0.001"/
  <clock_output id="BCTU0_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>
  <clock_output id="CLKOUT_RUN_CLK.outFreq" value="250 kHz" locked="false" accuracy=""/>
  <clock_output id="CLKOUT_STANDBY_CLK.outFreq" value="24 MHz" locked="false" accuracy="
  <clock_output id="CMP0_CLK.outFreq" value="40 MHz" locked="false" accuracy=""/>
  <clock_output id="CMP1_CLK.outFreq" value="40 MHz" locked="false" accuracy=""/>
  <clock_output id="CMP2_CLK.outFreq" value="40 MHz" locked="false" accuracy=""/>
  <clock_output id="CORE_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>
  <clock_output id="CRC0_CLK.outFreq" value="80 MHz" locked="false" accuracy=""/>
  <clock_output id="DCM0_CLK.outFreq" value="40 MHz" locked="false" accuracy=""/>
  <clock_output id="DCM_CLK.outFreq" value="40 MHz" locked="true" accuracy="0.001"/>
  <clock_output id="DMAMUX0_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>
  <clock_output id="DMAMUX1_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>
  <clock_output id="EDMA0_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>
  <clock_output id="EDMA0_TCD0_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>
  <clock_output id="EDMA0_TCD10_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>
  <clock_output id="EDMA0_TCD11_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>
  <clock_output id="EDMA0_TCD12_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>
  <clock_output id="EDMA0_TCD13_CLK.outFreq" value="160 MHz" locked="false" accuracy=""/>

```



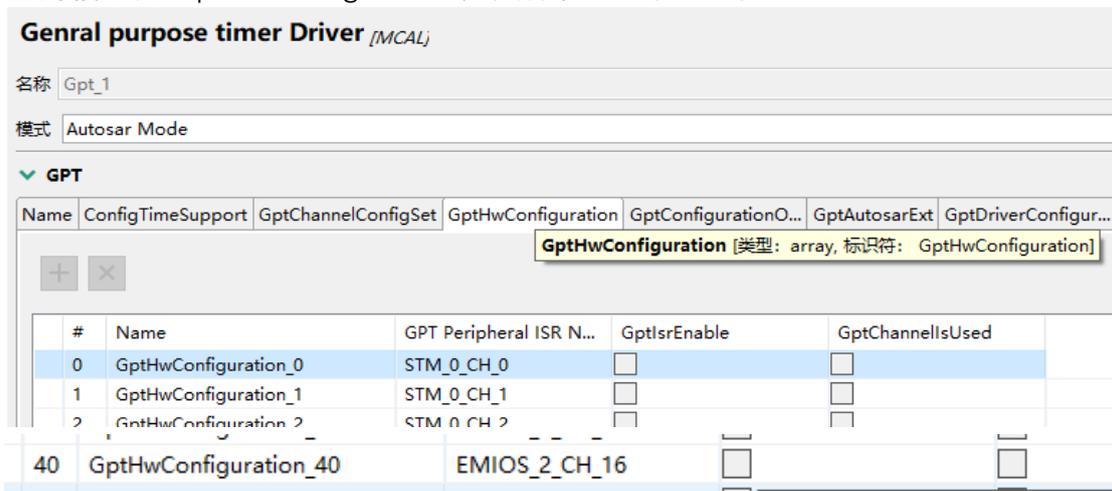
```

        <instance name="C40_lp_1" uuid="7c466fd4-7927-4799-856d-
ce6f4b98af9c" type="C40_lp" type_id="C40_lp" mode="ip" enabled="true" comment=""
custom_name_enabled="false">
            <config_set name="C40_lp"/>
                <setting name="Name" value="Gpt"/>
                    <struct name="ConfigTimeSupport">
                        <setting name="POST_BUILD_VARIANT_USED"
value="false"/>
                            <setting
name="IMPLEMENTATION_CONFIG_VARIANT" value="VARIANT-PRE-COMPILE"/>
                                </struct>
                                    <struct name="43">
                                        <setting name="Name" value=""/>
                                            <setting
name="IsrName"
value="FLASH_0_IRQn"/>
                                                <setting name="IsrEnabled" value="false"/>
                                                    <setting name="IsrPriority" value="0"/>
                                                        </struct>
                                                            <struct name="40">
                                                                <setting
name="Name"
value="GptHwConfiguration_40"/>
                                                                    <setting
name="GptIsrHwId"
value="EMIOS_2_CH_16"/>
                                                                        <setting name="GptIsrEnable" value="false"/>
                                                                            <setting
name="GptChannelsUsed"
value="false"/>
                                                                                </struct>
                                                                                    </instance>
                                                                                        <instance name="IntCtrl_lp_1" uuid="44e0e916-7d18-4eab-96fb-
3e2bd1ce09e4" type="IntCtrl_lp" type_id="IntCtrl_lp" mode="ip"
peripheral="MSCM" enabled="true" comment=""
custom_name_enabled="false">
                            <struct name>
                                </struct>
                            </instance>
                        </functional_group>

```

其中<instance>和<struct>标签是最丰富，每个<instance>直接对应了使用的组件，这里举例了 IntCtrl 和 C40_lp。

Struct 则是该组件中的每一个配置项，比如本例中，`< struct name="40">`，配置的信息是 GptHwConfiguration，具体在 CT 中如下图：



我们在 CT 里面输入的任何内容都会对应过来，对于某些外设<struct>的数量可能会超过 100 个。

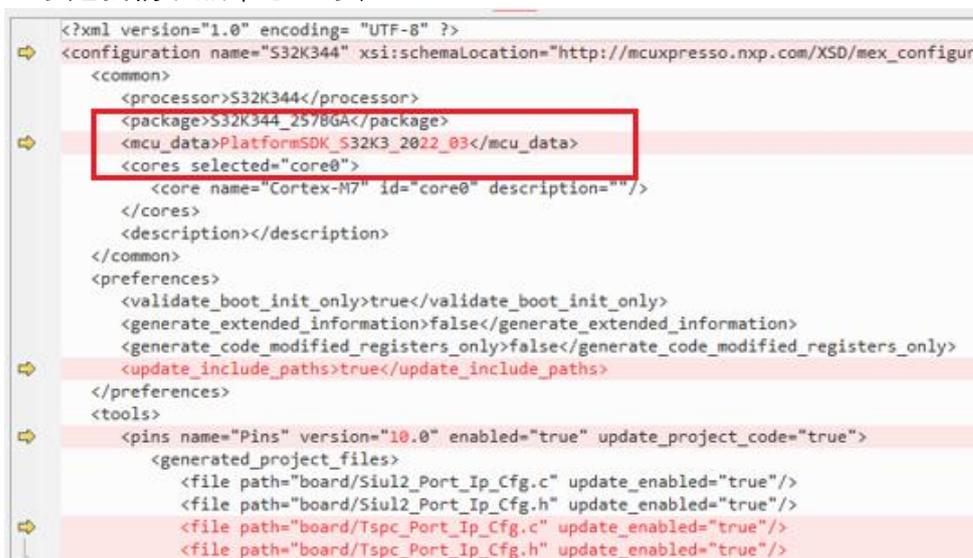
仔细研究这部分的配置信息，尤其是各个外设的配置格式，可以发现新旧版本的定义也是完全一样的，甚至每一个标签中具体的项都一样，理论上也不存在兼容性问题。

二. 修改方案

CT 这种 GUI 工具，原理就是挨个解析 XML 文件中的标签，然后把每一个内容都转换成图形和文本框在 S32DS 中呈现。

经过上面的分析，可以得出一个重要的结论，不同版本之间的 MEX 文件格式是兼容的，所有配置信息对应的 XML 标签都是一样的，甚至每一个标签内的定义都是一样的，理论上 CT 工具能够识别出不同版本的 MEX 文件，并在新版本上重新 generate code。这就是本方案的理论依据。

于是我们从版本号入手，



这一部分是包含了一些版本信息，核心就是<mcu_data>字段，显然新旧版本这个命名格式出入太大。这块直接对应到配置信息 GUI 界面，CT 就是靠这个字段识别版本信息：



再对比 MEX 文本，以及刚才的结论，很明显可以看出来，xml 的各个配置格式的定义，不同版本都是一样的。于是我大胆假设，只要 CT 能打开 MEX 文件，无论版本是多少，理论上 CT 是可以解析出来的，而且能解析就能 GUI 呈现。

那么我尝试用[文本编辑工具打开 MEX 文件](#)，并手动强制修改版本号，改为当前可以正常使用的版本（本文从 SW32K344_RTD_4_4_080_D2008 改为 PlatformSDK_S32K3_2022_03），再在 S32DS 中用 CT 打开，

可以看到，一切如我预期，IDE 竟然识别出了这样的信息，能够正常打开，甚至能发现这是旧版本建立的：

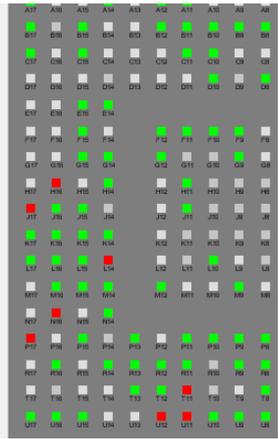
RROR_DETECT)



于是成功打开旧版本的工程，打开后可以看到符合预期，绝大多数配置都正常解析出来了，仅仅存在极少量报错，主要集中在三方面（注：该部分我选取了多个 project 的例子，把我遇到的所有问题都列一下）：

1 . Pin 配置

M3	PTA29			SIUL2.gpio,29	eMIOS_1,ch_h...
N3	PTD17	UART2_RX	UART2_RX	SIUL2.gpio,113...	eMIOS_0,ch_y,18
P3	PTA30	TJA1102_INT	TJA1102_INT	SIUL2.gpio,301...	eMIOS_1,ch_h...
R3	PTE9	RMII_TX_EN	RMII_TX_EN	SIUL2.gpio,137...	eMIOS_0,ch_g...
T3	PTG3	QSPI_CS	QSPI_CS	SIUL2.gpio,67[...	eMIOS_0,ch_g,3
U3	PTB4	RMII_TXD1	RMII_TXD1	SIUL2.gpio,36[...	eMIOS_0,ch_g...
A4	PTE30			SIUL2.gpio,158	
B4	PTE31			SIUL2.gpio,159	
C4	PTE28	CAN3_TX	CAN3_TX	SIUL2.gpio,156	
D4	VSS242				
E4	PTA20	VPWR_MON	VPWR_MON	SIUL2.gpio,20[...	eMIOS_1,ch_h...
F4	PTG8			SIUL2.gpio,200...	
G4	PTG4			SIUL2.gpio,196...	eMIOS_1,ch_y,20
H4	PTG5	TJA1044_1_STB	TJA1044_1_STB	SIUL2.gpio,197...	eMIOS_1,ch_y,21
J4	VSS247				
K4	PTF29	TJA1145_1_CS	TJA1145_1_CS	SIUL2.gpio,189	eMIOS_1,ch_h...
L4	PTD15	UART2_CTS	UART2_CTS	SIUL2.gpio,111...	eMIOS_0,ch_x...



路由详情

引脚 信号 类型筛选文本

BOARD_InitPins 的路由详情 133

#	外设	信号	箭头	已路由的引脚/信号	标签	标识符	方向	Slew Rate	Output Buffer Enable	Safe Mode Co
C15	LPSPI_4	lpspi_sout	->	[C15] PTB9	SPI4_SOUT	SPI4_SOUT	Output	n/a	Enabled	Disable
C17	LPSPI_4	lpspi_sin	<-	[C17] PTB11	SPI4_SIN	SPI4_SIN	Input	n/a	Disabled	Disable
H16	ADC_0	s_in	<-	[H16] PTB13	POT1	POT2	n/a	n/a	Disabled	Disable
J17	ADC_1	s_in	<-	[J17] PTB14	POT1	POT1	n/a	n/a	Disabled	Disable
J16	LPSPI_1	lpspi_sin	<-	[J16] PTB15	SPI1_SIN	SPI1_SIN	Input	n/a	Disabled	Disable

BOARD_InitPins 的路由详情 133

#	外设	信号	箭头
C17	LPSPI_4	lpspi_sin	<-
H16	ADC_0	s_in, 8	<-
J17	ADC_1	s_in, 8	<-
I16	LPSPI_1	ext_in	<-

新版本

引脚	信号	箭头	已路由
15	ADC_1	<-	[P15]
16	CMP_2	->	[R16]
17	CMP_1	<-	[P17]
I16	FlexIO	<-	[N16]
14	ADC_2	<-	[L14]

新版本

旧版本

以 flexIO 和 ADC 为例，可以看到这显然就是新旧版本对于 pinmux 的格式命名差异，导致无法识别。甚至能看出来就是 R&D 在开发的时候犯的小错误，考虑的不够完善。

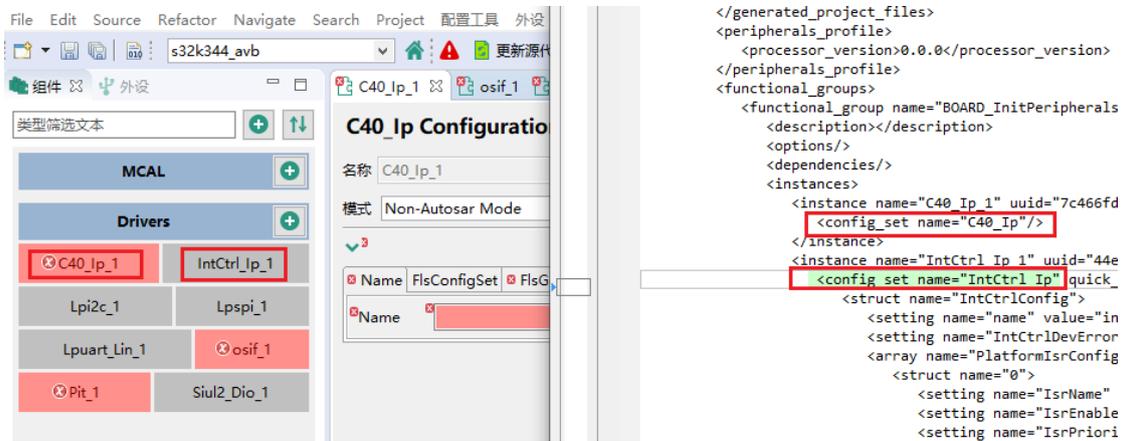
旧版本 FlexIO,
新版本 FLEXIO, 仅仅是大小写差异。

旧版本 s_in
新版本 s_in, 8 对应<pin peripheral="ADC_0" signal="s_in" pin_num="H16"
pin_signal="PTB13">标签。

实际上是同一回事, 仅仅就这么点命名的差异就导致无法识别, 于是重新配一下 pinmux 即可

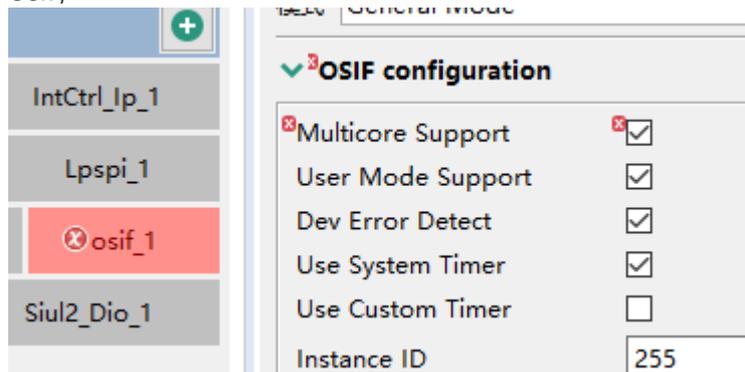
2. 外设配置

外设部分也保留了核心的关键信息, 但同样由于格式差异, 有少部分问题。最复杂的是 C40_Ip 控制, 缺少 name, 对比其他无异常的 MEX 文件, 可以看出来一些信息, 于是手动添加一个 name,



```
</generated_project_files>
<peripherals_profile>
  <processor_version>0.0.0</processor_version>
</peripherals_profile>
<functional_groups>
  <functional_group name="BOARD_InitPeripherals"
  <description></description>
  <options/>
  <dependencies/>
  <instances>
    <instance name="C40_Ip_1" uuid="7c466fd"
      <config_set name="C40_Ip"/>
    </instance>
    <instance name="IntCtrl Ip 1" uuid="44e"
      <config_set name="IntCtrl Ip" quick_
      <struct name="IntCtrlConfig">
        <setting name="name" value="in
        <setting name="IntCtrlDevError
        <array name="PlatformIsrConfig"
          <struct name="0">
            <setting name="IsrName"
            <setting name="IsrEnable
            <setting name="IsrPriori
```

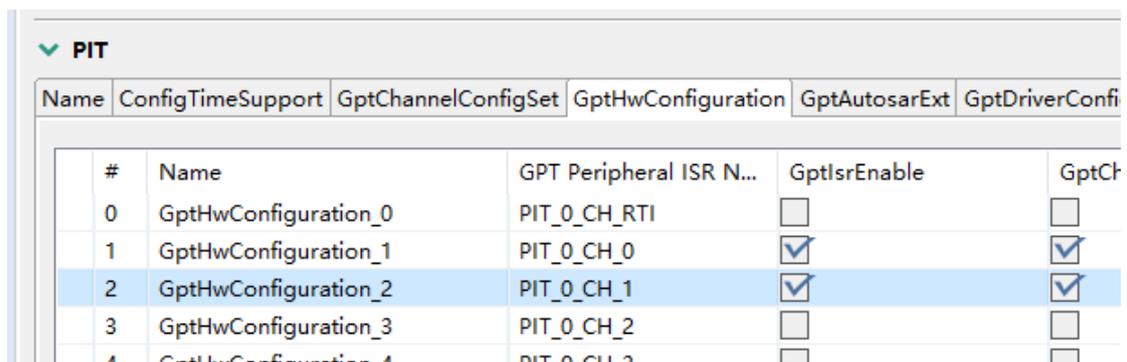
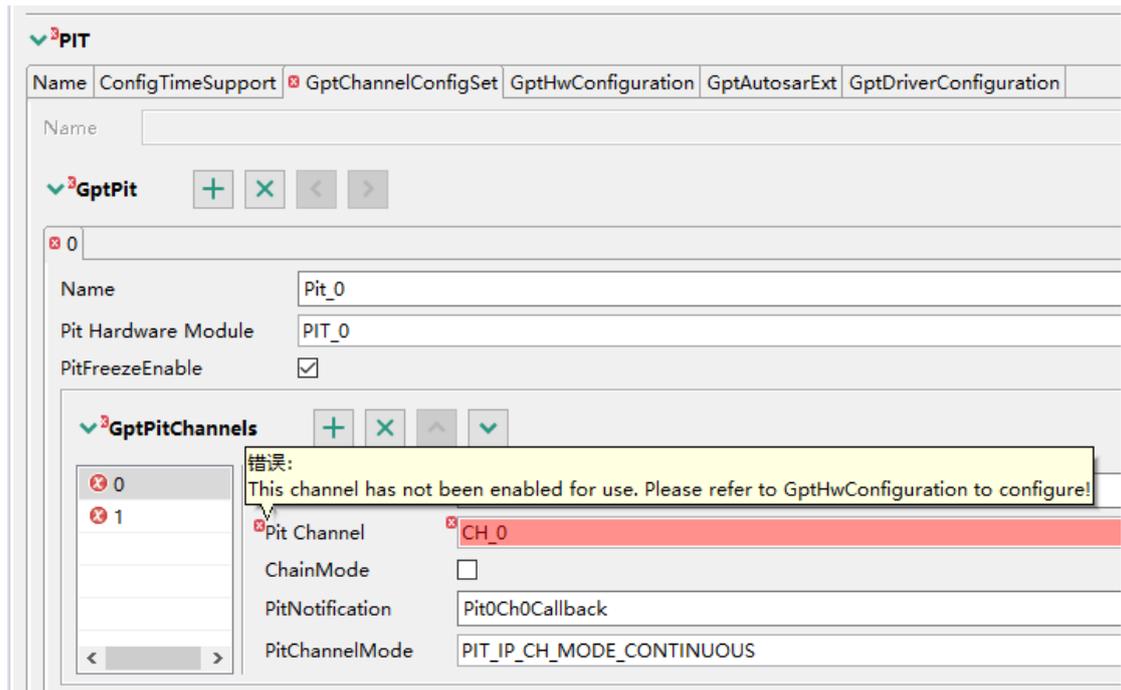
其他几个都非常简单, 总之都是些很小的问题, 重新配一下就好了, 一个是 osif,



Option	Checked
Multicore Support	<input checked="" type="checkbox"/>
User Mode Support	<input checked="" type="checkbox"/>
Dev Error Detect	<input checked="" type="checkbox"/>
Use System Timer	<input checked="" type="checkbox"/>
Use Custom Timer	<input type="checkbox"/>

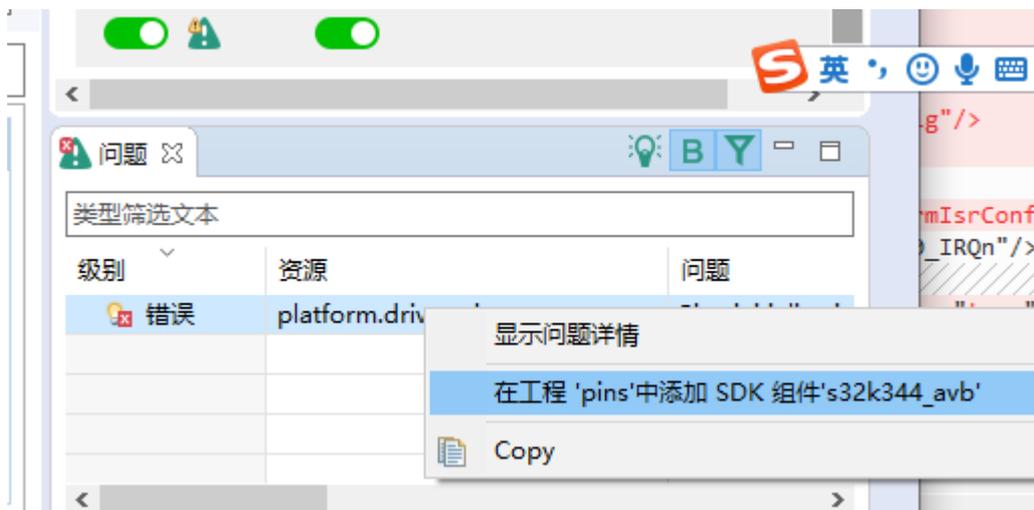
Instance ID: 255

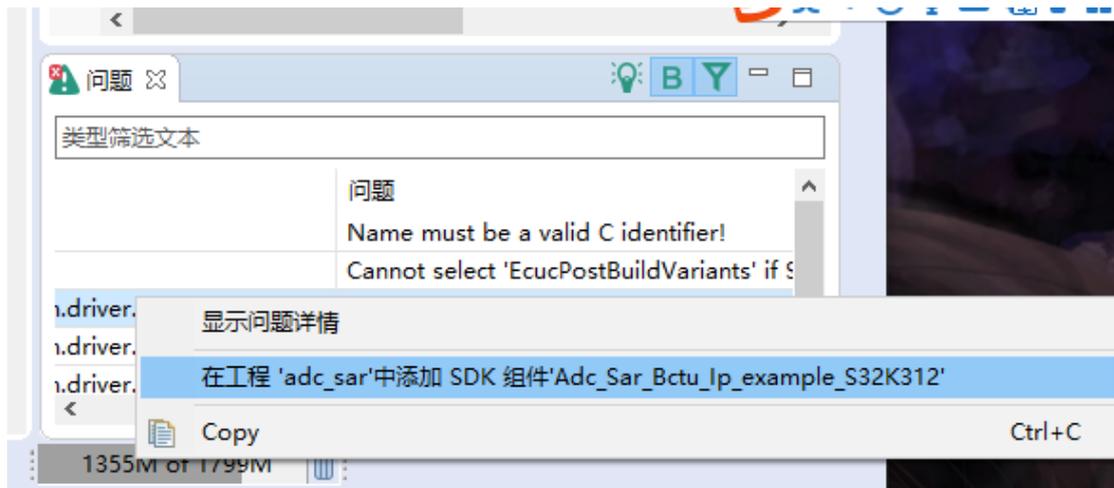
另一个是 PIT, 仅仅是没有使能, 显然也是由于不同版本格式差异导致, 重新使能一下就好了。



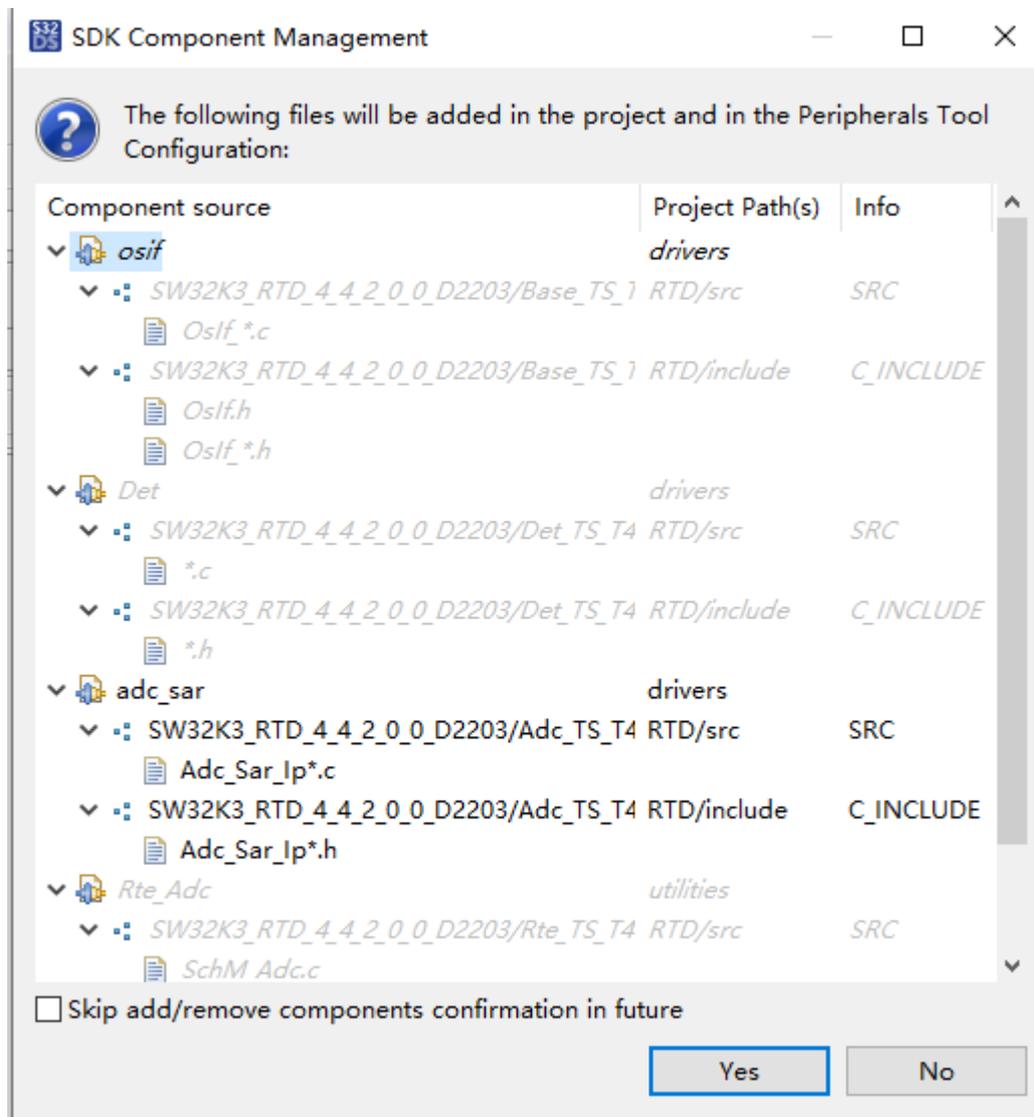
3. 缺部分组件

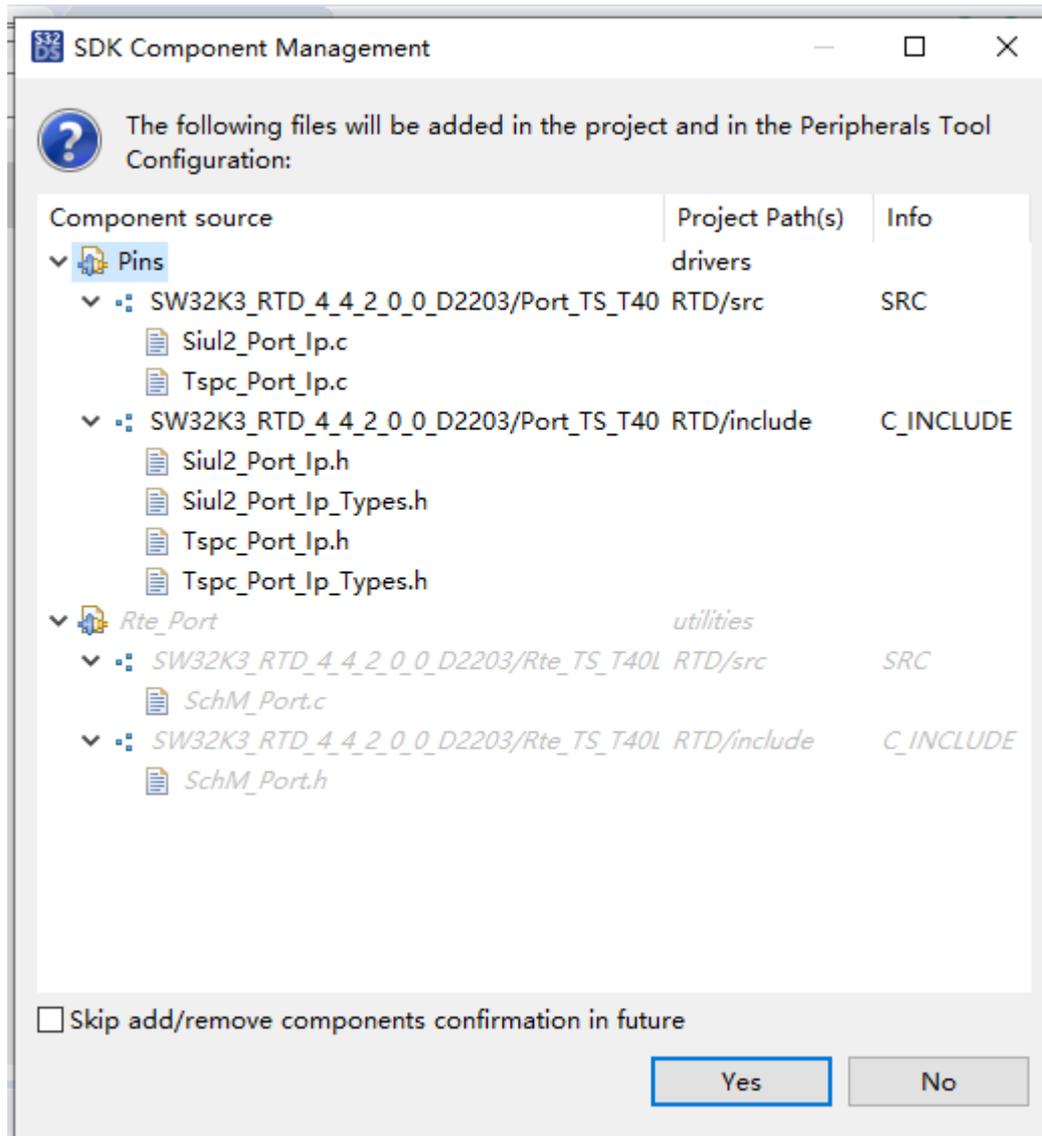
可以手动添加，在右下角界面的错误上点击鼠标右键，会有提示，



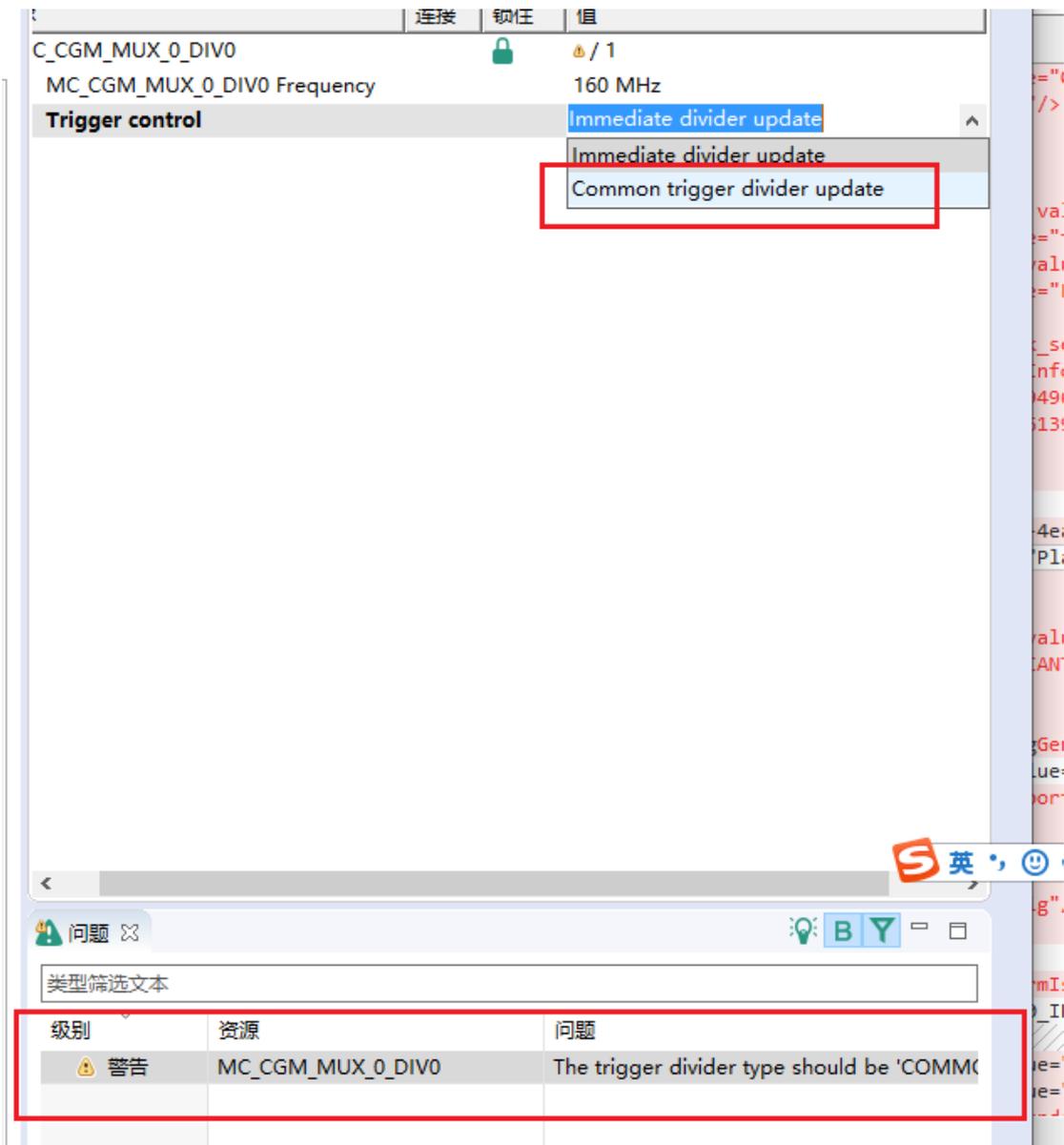


在弹出的对话框中点击确认即可，





另外在别的工程中，clock 部分还有一个 warning，不确定是本身工程的问题还是导入的问题，但重新配置一下可以消除：

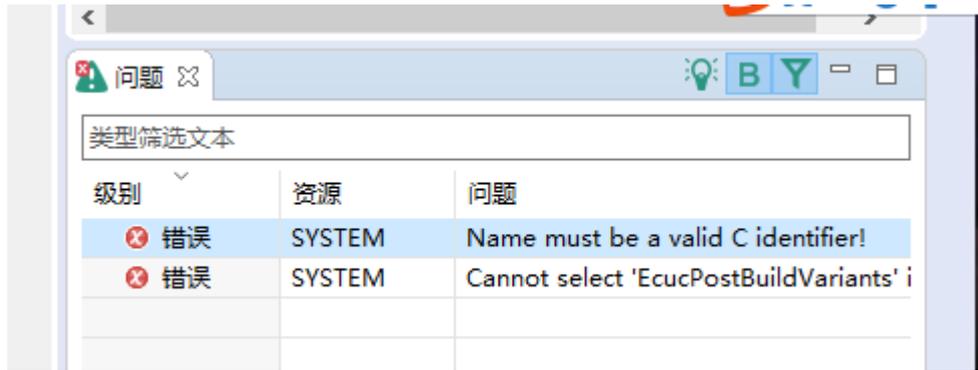


以上内容都可以通过重新配置完成修复。

4. 隐藏问题

最后一个，也是隐蔽最深的。该错误信息对应的配置在 CT 的 GUI 页面中没有显示，但会报错，目前我还不知道是什么原因导致。

但是见招拆招，通过这个文本“[EcucPostBuildVariants](#)”，在可以正常使用的 MEX 中搜索，发现确实有差异。搜索后看到，Name 项缺失，以及某一个配置不对，而且刚好和 CT 中的报错吻合。于是文本编辑手动修改一下（**该步骤的修改需要关闭 S32DS，否则无法生效**）：



```

</functional_groups>
<components>
  <component name="system" uuid="5d808eec-16b8-407e-a20d-6f204dc148fa" type_id="system">
    <config_set_global name="SystemModel">
      <setting name="Name" value=""/>
      <setting name="EcvdGenerationMethod" value="INDIVIDUAL"/>
      <setting name="EcvdOutputPath" value=""/>
      <setting name="EcvdGenerationTrigger" value="Generate Configuration"/>
      <setting name="SyncFunctionalGroups" value="false"/>
      <setting name="IgnoreComponentSuffix" value="false"/>
      <setting name="ComponentGenerationMethod" value="EcucPostBuildVariants"/>
      <setting name="DefaultFunctionalGroup" value="VS_0"/>
      <struct name="PostBuildSelectable" quick_selection="Default">
        <setting name="Name" value="PostBuildSelectable"/>

```

```

<components>
  <component name="system" uuid="5d808eec-16b8-407e-a20d-6f204dc148fa" type_id="sys">
    <config_set_global name="SystemModel">
      <setting name="Name" value="SystemModel"/>
      <setting name="EcvdGenerationMethod" value="INDIVIDUAL"/>
      <setting name="EcvdOutputPath" value=""/>
      <setting name="EcvdGenerationTrigger" value="Generate Configuration"/>
      <setting name="SyncFunctionalGroups" value="false"/>
      <setting name="IgnoreComponentSuffix" value="false"/>
      <setting name="ComponentGenerationMethod" value="FunctionalGroups"/>
      <setting name="DefaultFunctionalGroup" value="VS_0"/>
      <struct name="PostBuildSelectable" quick_selection="Default">
        <setting name="Name" value="PostBuildSelectable"/>
        <array name="PredefinedVariants">

```

经过这样一番处理，CT 中所有的 error 和 warning 都 fix 了，然后在 CT 工具中更新源代码，可以看到，旧的文件删除了，重新生成了匹配的新文件：通过这些文件名，可以猜到这就是不同版本的差异。

更新源文件

生成的文件	状态
<input checked="" type="checkbox"/> 引脚	
<input checked="" type="checkbox"/> board\ <ul style="list-style-type: none"><input checked="" type="checkbox"/> Siul2_Port_Ip_Cfg.c <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> Siul2_Port_Ip_Cfg.h <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> Tspc_Port_Ip_Cfg.c <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> Tspc_Port_Ip_Cfg.h <input type="checkbox"/> 无更改	
<input checked="" type="checkbox"/> 外设	
<input checked="" type="checkbox"/> generate\ <ul style="list-style-type: none"><input checked="" type="checkbox"/> include\<ul style="list-style-type: none"><input checked="" type="checkbox"/> Adc_Sar_Ip_Cfg.h <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> Adc_Sar_Ip_CfgDefines.h <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> Adc_Sar_Ip_PBcfg.h <input type="checkbox"/> 删除<input checked="" type="checkbox"/> Adc_Sar_Ip_VS_0_PBcfg.h <input type="checkbox"/> 创建<input checked="" type="checkbox"/> Bctu_Ip_Cfg.h <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> Bctu_Ip_CfgDefines.h <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> Bctu_Ip_PBcfg.h <input type="checkbox"/> 删除<input checked="" type="checkbox"/> Bctu_Ip_VS_0_PBcfg.h <input type="checkbox"/> 创建<input checked="" type="checkbox"/> IntCtrl_Ip_Cfg.h <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> IntCtrl_Ip_CfgDefines.h <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> modules.h <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> Oslf_ArchCfg.h <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> Oslf_Cfg.h <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> Siul2_Port_Ip_Defines.h <input type="checkbox"/> 无更改<input checked="" type="checkbox"/> generate\<ul style="list-style-type: none"><input checked="" type="checkbox"/> src\<ul style="list-style-type: none"><input checked="" type="checkbox"/> Adc_Sar_Ip_PBcfg.c <input type="checkbox"/> 删除<input checked="" type="checkbox"/> Adc_Sar_Ip_VS_0_PBcfg.c <input type="checkbox"/> 创建<input checked="" type="checkbox"/> Bctu_Ip_PBcfg.c <input type="checkbox"/> 删除<input checked="" type="checkbox"/> Bctu_Ip_VS_0_PBcfg.c <input type="checkbox"/> 创建<input checked="" type="checkbox"/> IntCtrl_Ip_Cfg.c <input type="checkbox"/> 无更改	

点击 ok 后，生成代码，此时所有的 CT 配置问题都已经解决，代码也全部生成，没有任何错误。

另外注意到一点，CT 配置的都是 Ip 层，也就是 SDK 中直接操作寄存器的 drivers，就是以前的 LLD，目前没看到设计 IP Wrapper Layer 的。

5. SDK include

先别高兴，点击编译，可以看到仍然会报错，提示找不到相关文件，

```

=====*/
#include "StandardTypes.h"
#include "Adc_Sar_Ip_CfgDefines.h"
=====*/
/*
 * SOURCE FILE VERSION INFORMATION
 *=====*/
#define ADC_SAR_IP_VENDOR_ID_TYPES 43
#define ADC_SAR_IP_AR_RELEASE_MAJOR_VERSION_TYPES 4
#define ADC_SAR_IP_AR_RELEASE_MINOR_VERSION_TYPES 4
#define ADC_SAR_IP_AR_RELEASE_REVISION_VERSION_TYPES 0
#define ADC_SAR_IP_SW_MAJOR_VERSION_TYPES 2
#define ADC_SAR_IP_SW_MINOR_VERSION_TYPES 0
#define ADC_SAR_IP_SW_PATCH_VERSION_TYPES 0
=====*/
/*
 * FILE VERSION CHECKS
 *=====*/
/* Check if Adc_Sar_Ip_Types.h file and Adc_Sar_Ip_CfgDefines.h file are of the same vendor */
#if (ADC_SAR_IP_VENDOR_ID_TYPES != ADC_SAR_IP_VENDOR_ID_CFGDEFINES)
#error "Adc_Sar_Ip_Types.h and Adc_Sar_Ip_CfgDefines.h have different vendor ids"
#endif

/* Check if Adc_Sar_Ip_Types.h file and Adc_Sar_Ip_CfgDefines.h file are of the same Autosar version */
#if ((ADC_SAR_IP_AR_RELEASE_MAJOR_VERSION_TYPES != ADC_SAR_IP_AR_RELEASE_MAJOR_VERSION_CFGDEFINES) |
(ADC_SAR_IP_AR_RELEASE_MINOR_VERSION_TYPES != ADC_SAR_IP_AR_RELEASE_MINOR_VERSION_CFGDEFINES) |
(ADC_SAR_IP_SW_PATCH_VERSION_TYPES != ADC_SAR_IP_SW_PATCH_VERSION_CFGDEFINES))
#error "Adc_Sar_Ip_Types.h and Adc_Sar_Ip_CfgDefines.h have different Autosar versions"
#endif
=====*/

```

CDT Build Console [Adc_Sar_Bctu_Ip_example_S32K312]

```

from ../generate/src/Adc_Sar_Ip_VS_0_PbCfg.c:42:
../RTD/include/Adc_Sar_Ip_Types.h:45:10: fatal error: StandardTypes.h: No such file or directory
45 | #include "StandardTypes.h"
    | ^~~~~~

```

不过到这里已经没啥难度，这些缺失的都是 RTD 标准的库函数，只是当前工程默认指向的旧版本的 SDK 目录，在 2.0.0 的 RTD include 添加一下：

```

Include paths (-I)
../RTD/include
"G:\NXP\S32DS.3.4\S32DS\software\PlatformSDK_S32K3_2022_03\SW32K3_RTD_4.4.2.0_0_D2203\Base_TS_T40D34M20I0R0\include"
"G:\NXP\S32DS.3.4\S32DS\software\PlatformSDK_S32K3_2022_03\SW32K3_RTD_4.4.2.0_0_D2203\Base_TS_T40D34M20I0R0\header"
"G:\NXP\S32DS.3.4\S32DS\software\PlatformSDK_S32K3_2022_03\SW32K3_RTD_4.4.2.0_0_D2203\Platform_TS_T40D34M20I0R0\include"
"G:\NXP\S32DS.3.4\S32DS\software\PlatformSDK_S32K3_2022_03\SW32K3_RTD_4.4.2.0_0_D2203\Platform_TS_T40D34M20I0R0\startup\include"
$(ProjDirPath)/generate/include
$(ProjDirPath)/RTD/include

```

然后再编译：

```

CDT Build Console [Adc_Sar_Bctu_Ip_example_S32K312]
Invoking: Standard S32DS File to Build
arm-none-eabi-size --format=berkeley Adc_Sar_Bctu_Ip_example_S32K312.elf
text data bss dec hex filename
38752 1040 10608 50400 c4e0 Adc_Sar_Bctu_Ip_example_S32K312.elf
Finished building: Adc_Sar_Bctu_Ip_example_S32K312.siz

00:42:00 Build Finished. 0 errors, 1 warnings. (took 13s.635ms)

```

大功告成，编译成功了!!!

总结：

不同版本的 MEX 文件，内部各个标签定义的格式是完全一样的，CT 可以识别，且手动编辑版本后用 CT 可以打开，绝大多数信息都能正确识别。
少量的报错，只需要根据具体的报错信息，CT 重新配置一下即可。
对于其他的问题，根据具体报错信息修复。
由于手头测试 project 无法包含所有问题，可能会有遗漏的，后续继续补充。