XMC7000中如何定时完成多组ADC采样

Raymond Duan

2025 8 23

Table of contents

Table of contents

问题背景

- 客户在实际项目中需要用到多路ADC进行定时采样。由于是首次使用XMC7000系列产品，客户对于该产品外设以及开发工具都不熟悉。而现有demo工程中也没有可供参考的例子，所以不知道如何实现该功能。为此，我实现了满足该功能的例程，本文档是对该例程实现进行详细说明。

➢ 软件开发环境：
  - **ModusToolbox V3.2.0.16028**

➢ 硬件验证环境：
  - **KIT_XMC72_EVK**

Table of contents

使用的外设资源以及参考资料

– 本例程用到的MCU 外设资源如下：

➢ 四路TCPWM：

- PWM: TCPWM[1] Group[1] Counter0，定时触发ADC0采样
- PWM1: TCPWM[1] Group[1] Counter1，定时触发ADC1采样
- PWM2: TCPWM[1] Group[1] Counter2，定时触发ADC2采样
- PWM_S: TCPWM[0] Group[0] Counter0，触发另外三路PWM,PWM1,PWM2同步启动
- TCPWM周期：100us

➢ 三组ADC：

- ADC0的6个channel: P6.0,P6.1,P6.2,P6.3,P6.4,P6.5
- ADC1的6个channel: P12.0,P12.1,P12.2,P12.3,P12.4,P12.5
- ADC2的6个channel: P18.0,P18.1,P18.2,P18.3,P18.4,P18.5

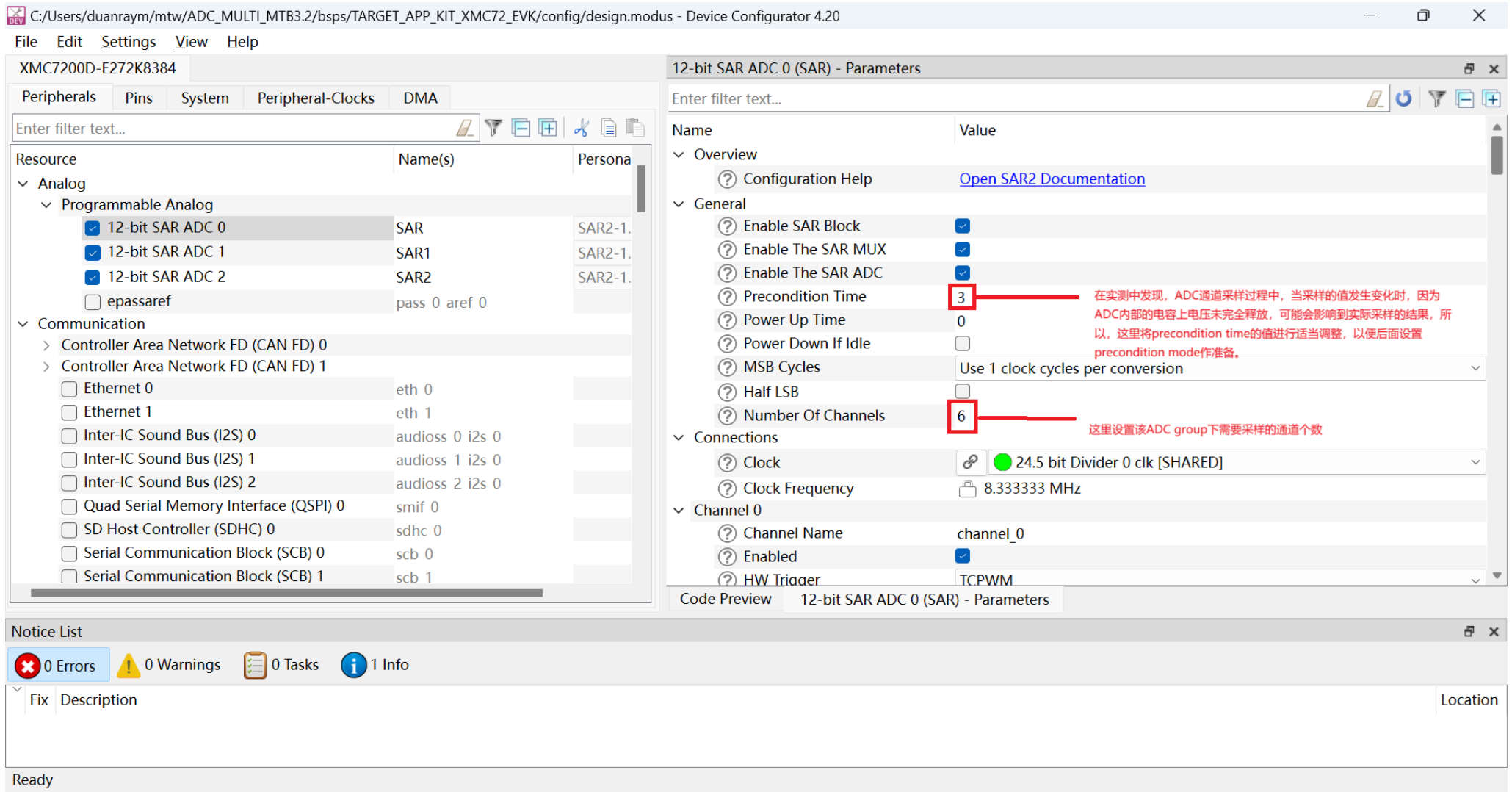
– 例程实现参考官网的以下Appnote:

➢ AN234119 - Timer, Counter, and PWM _TCPWM_ usage in XMC7000 family

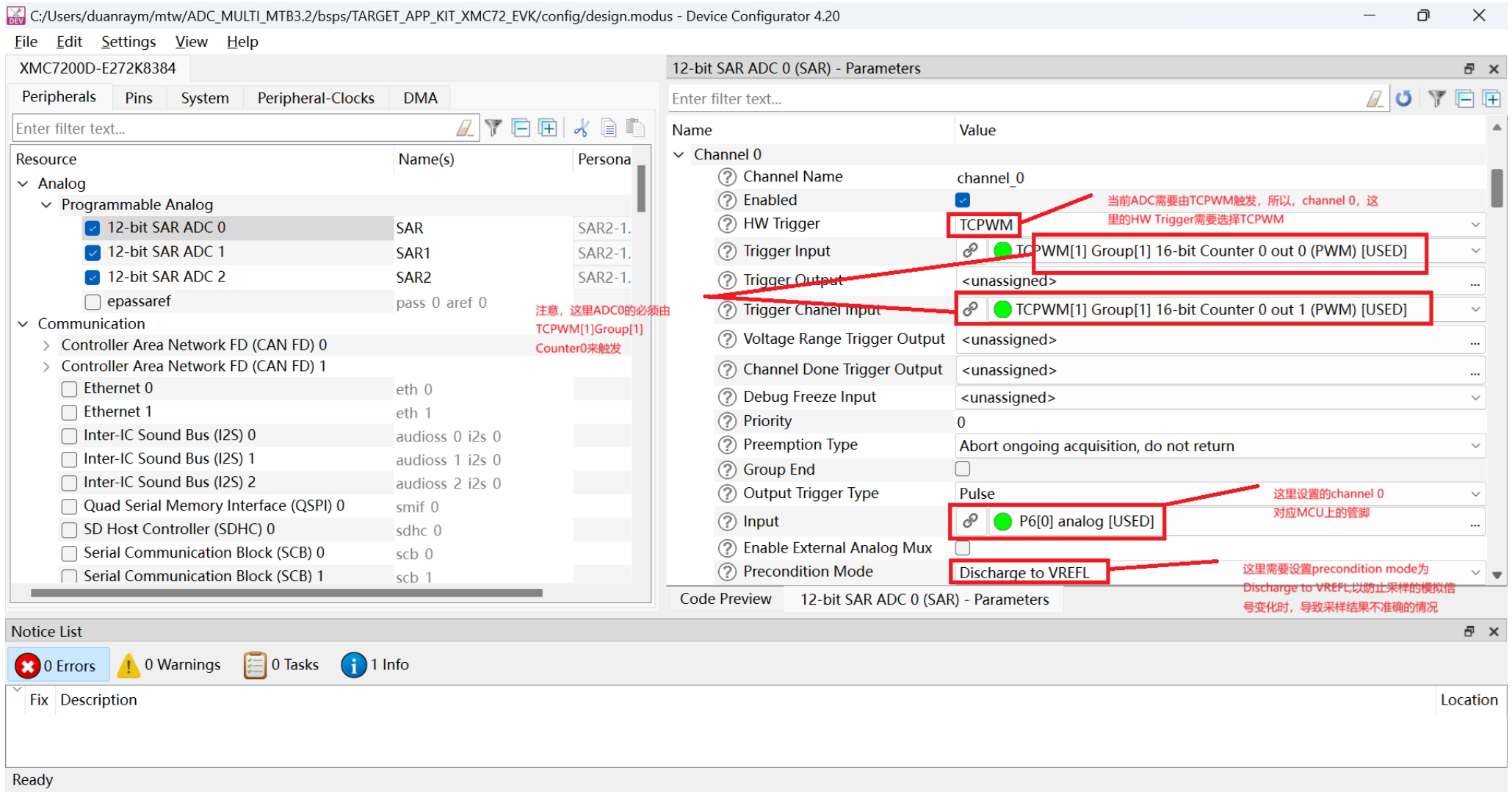
➢ AN234282 - Using a SAR ADC in XMC7000 Family

Table of contents

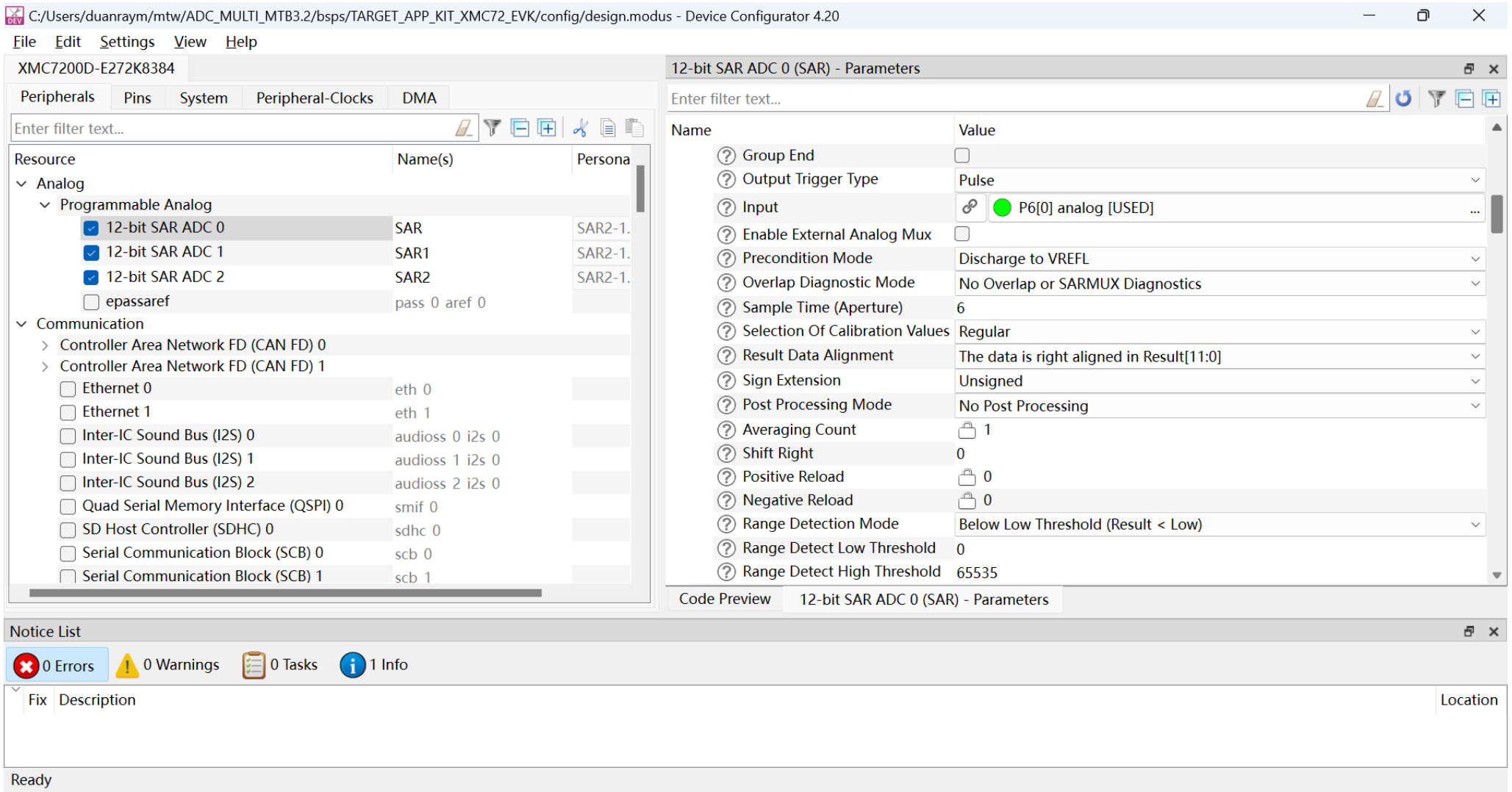
ADC0 配置



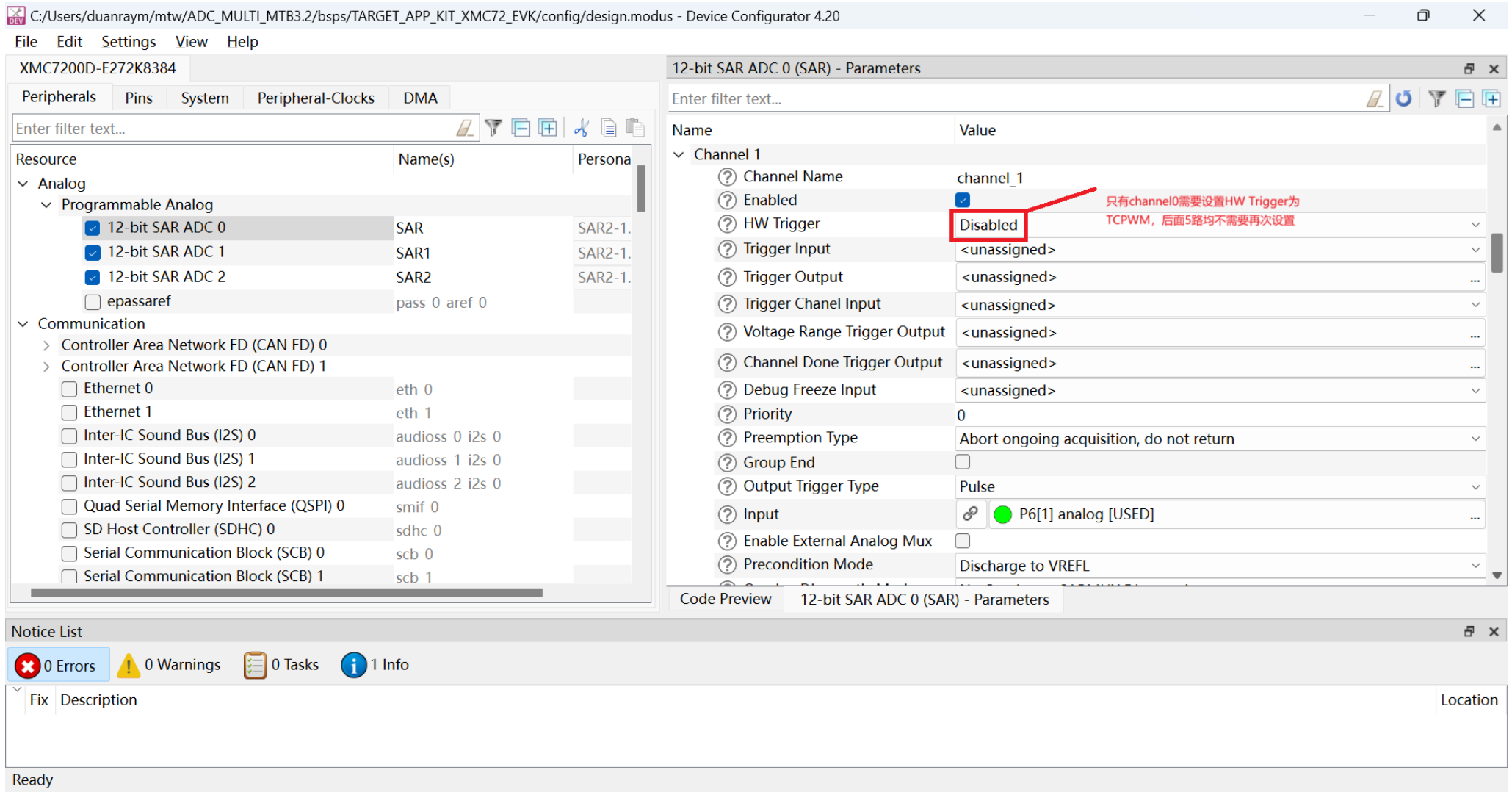
ADC0 channel 0 配置



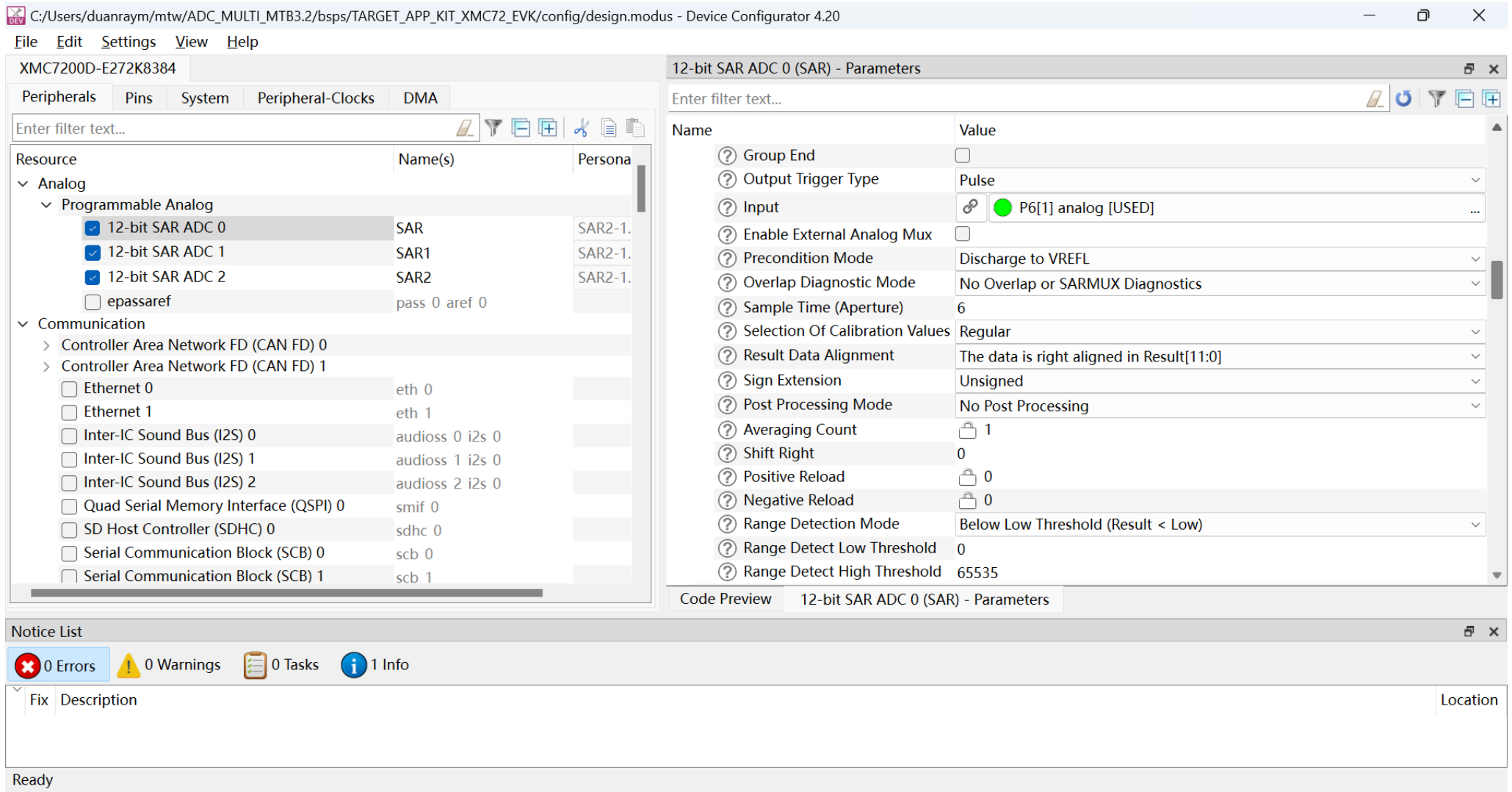
ADC0 channel 0 配置



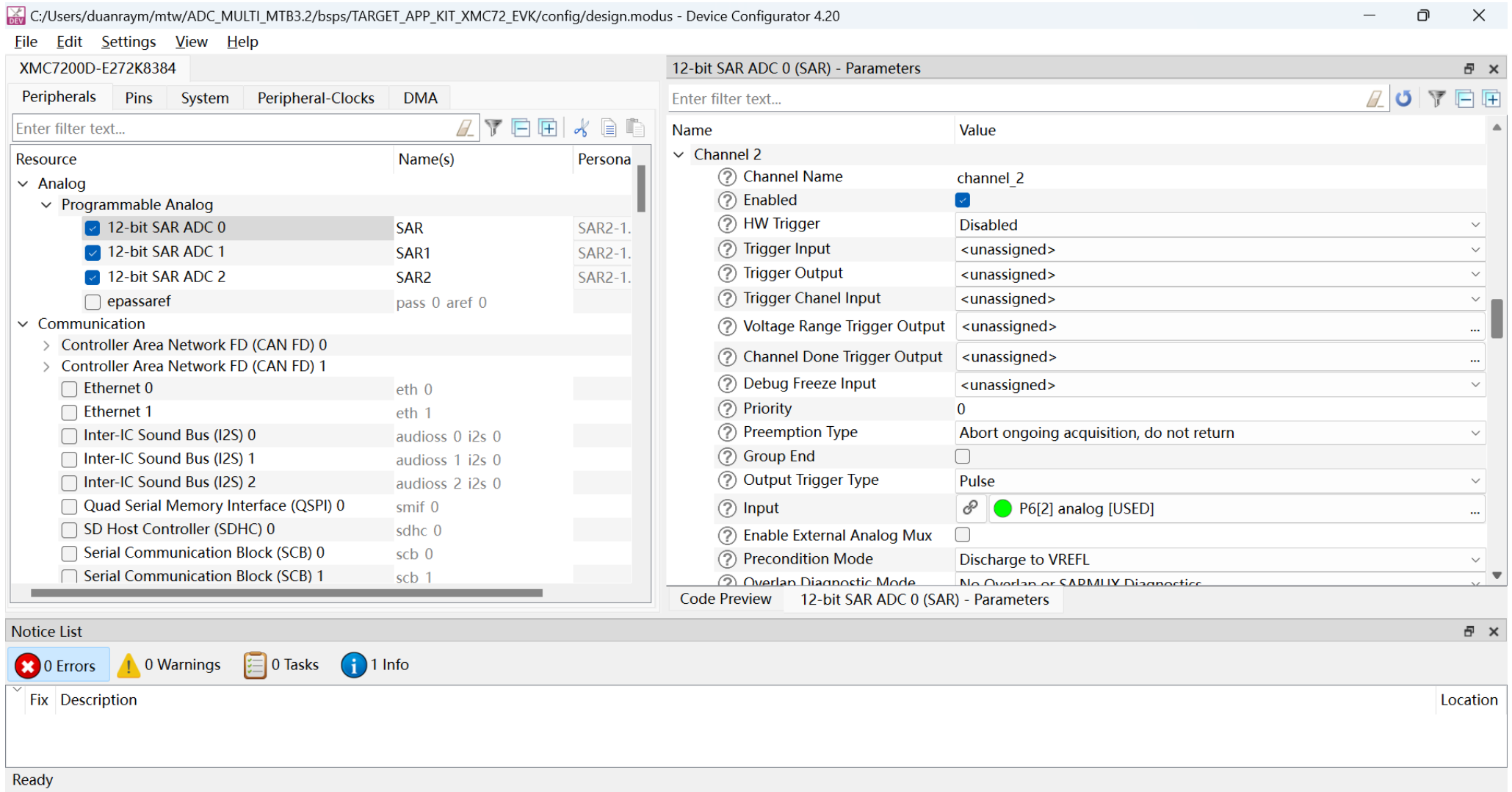
ADC0 channel 1 配置



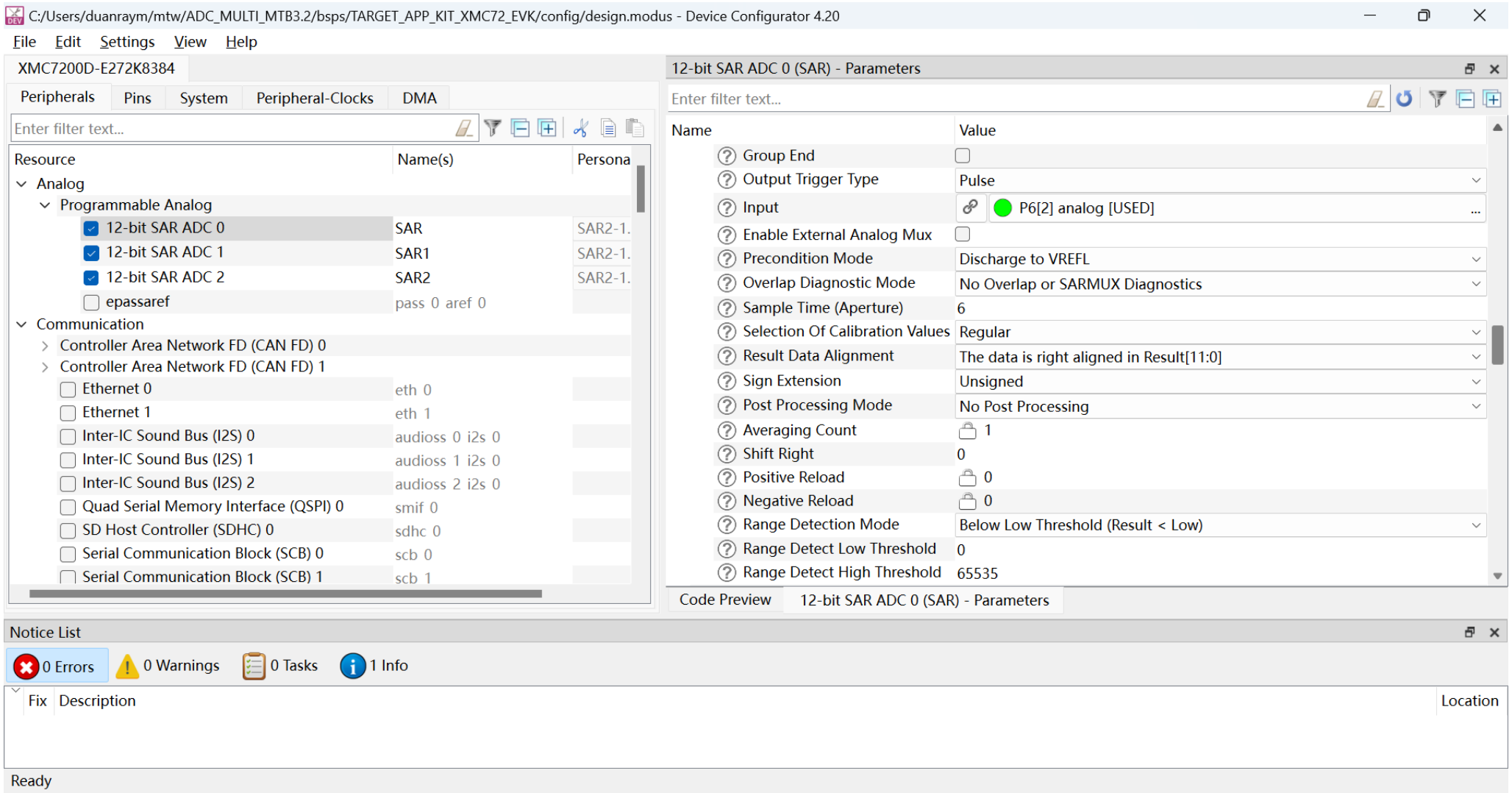
ADC0 channel 1 配置



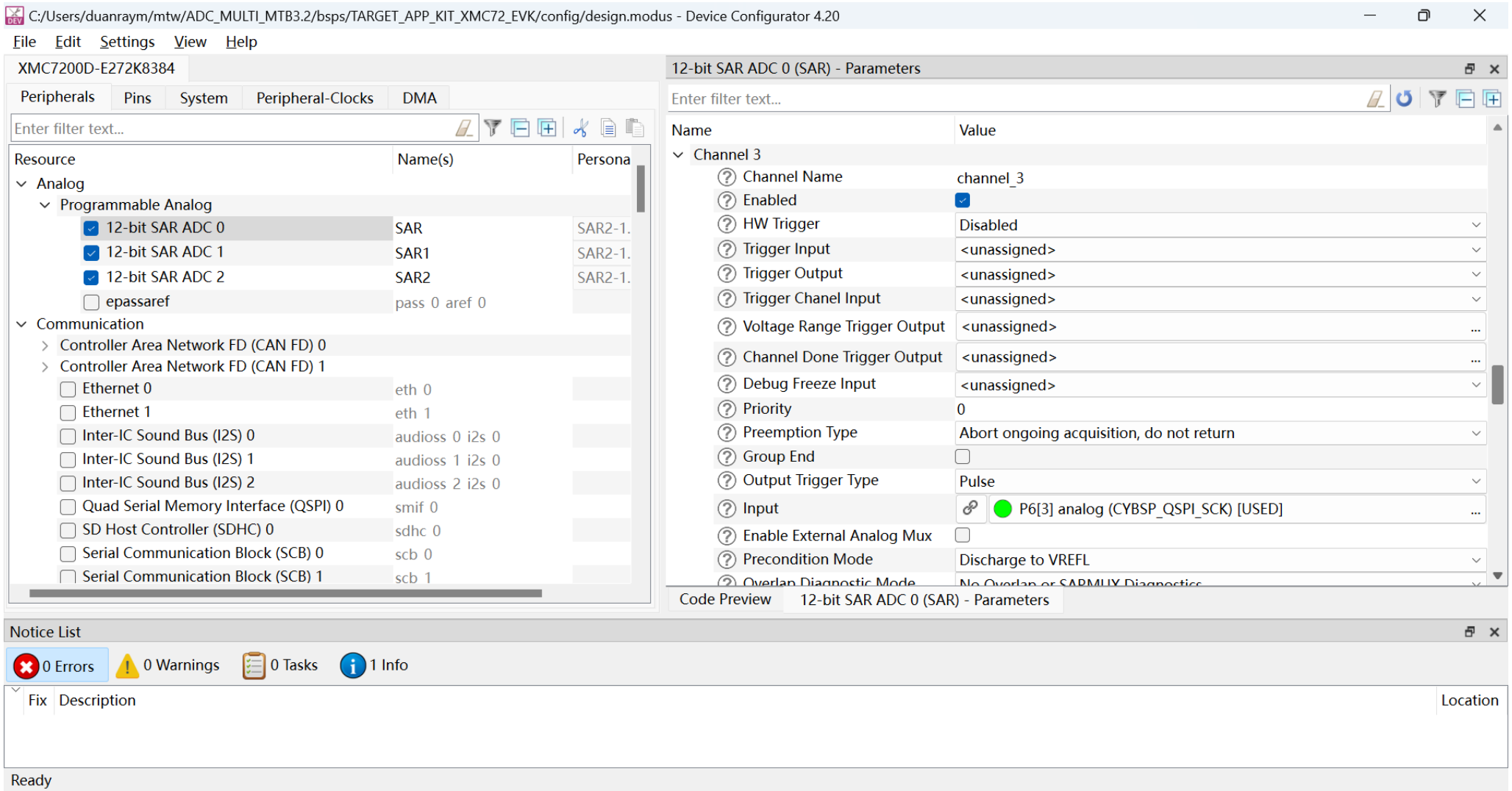
ADC0 channel 2 配置



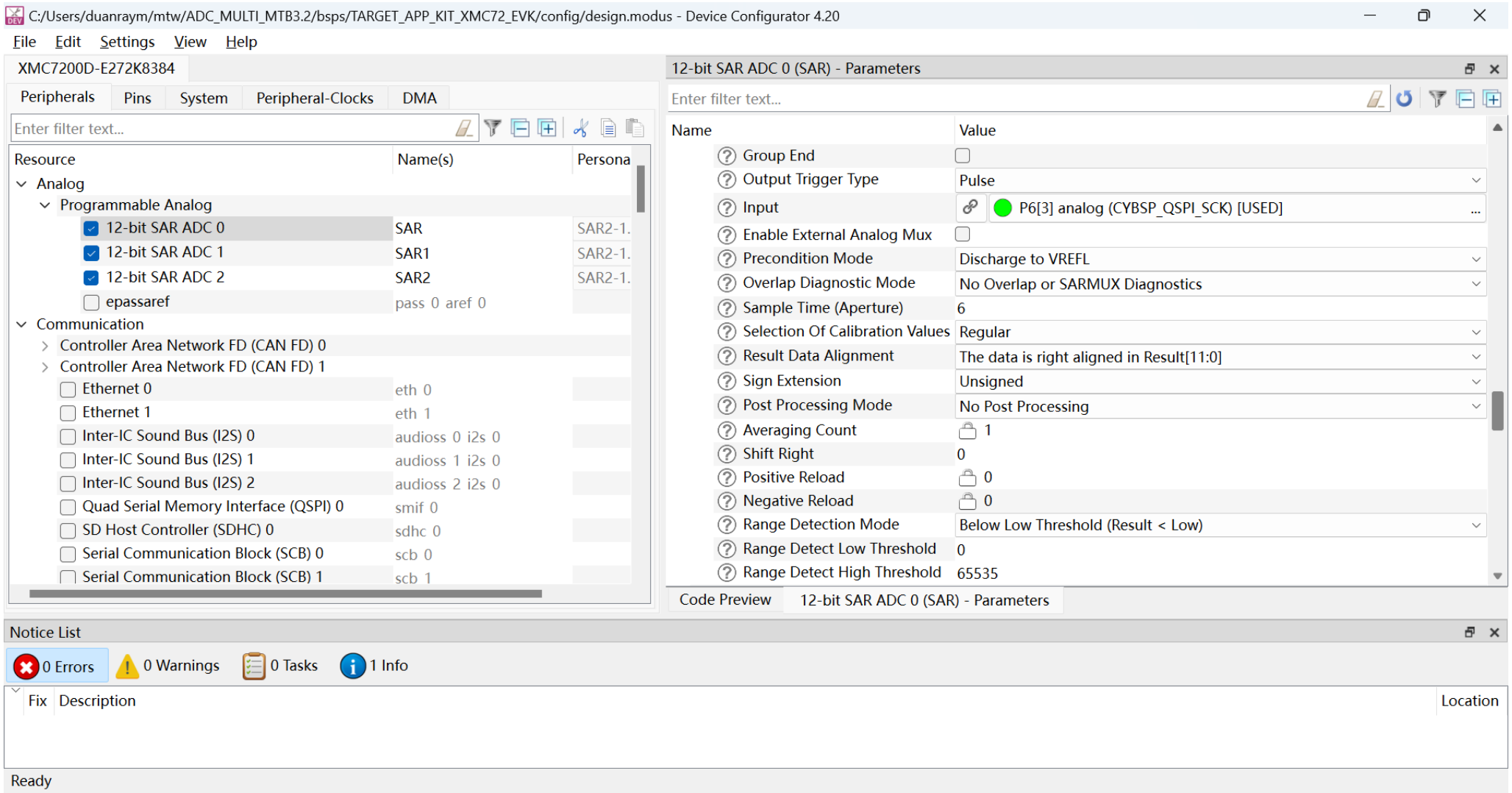
ADC0 channel 2 配置



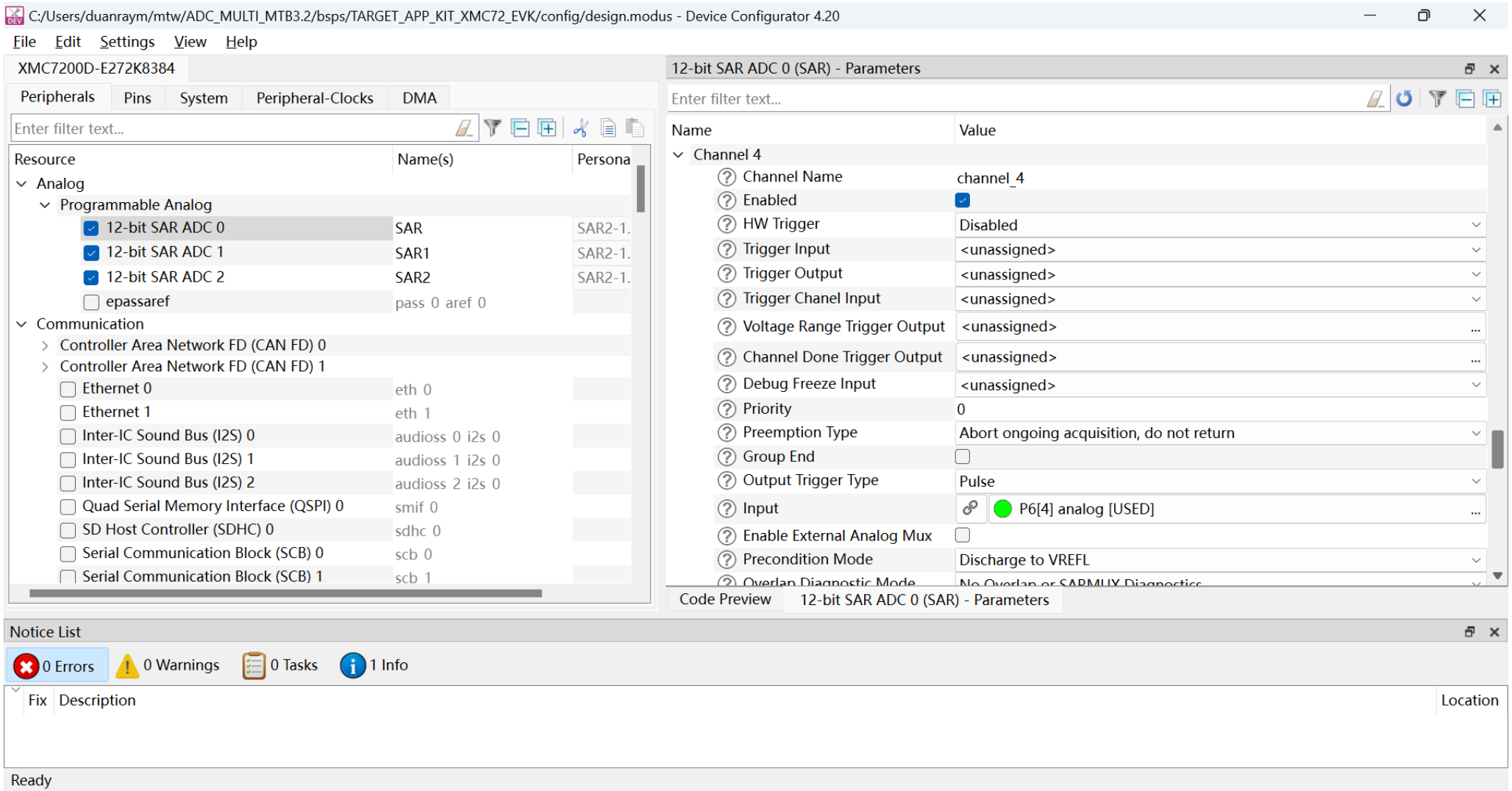
ADC0 channel 3 配置



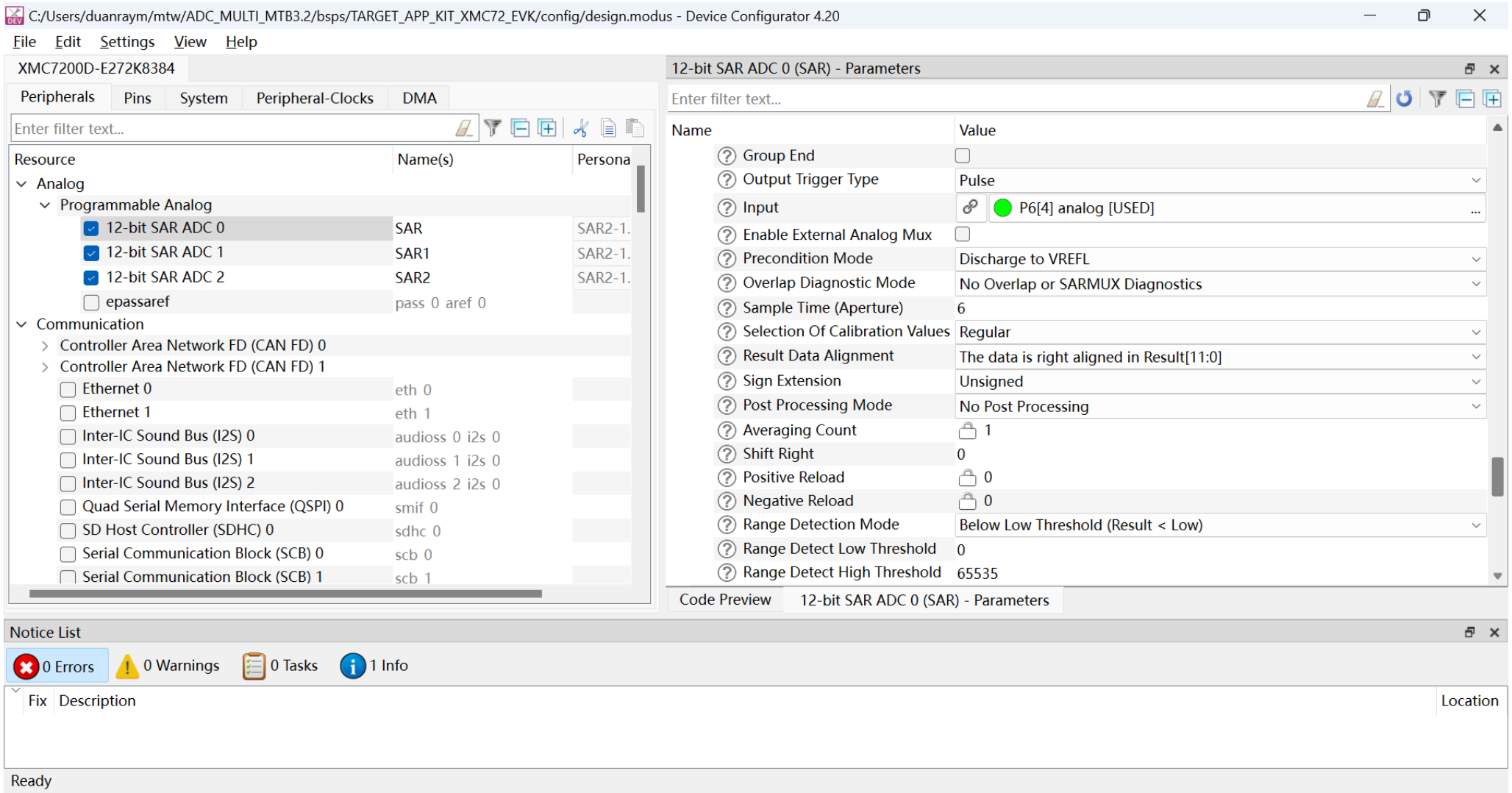
ADC0 channel 3 配置



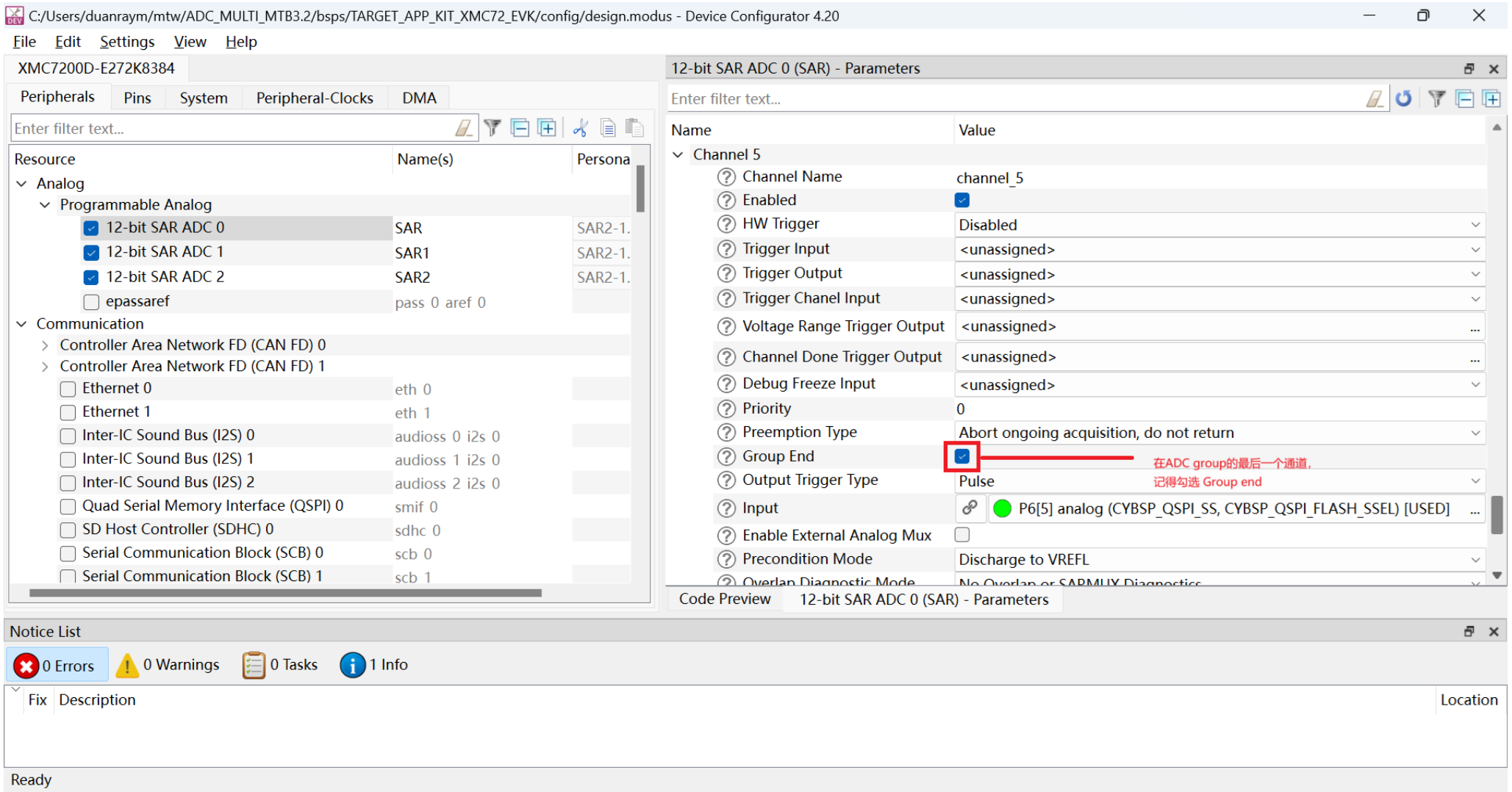
ADC0 channel 4 配置



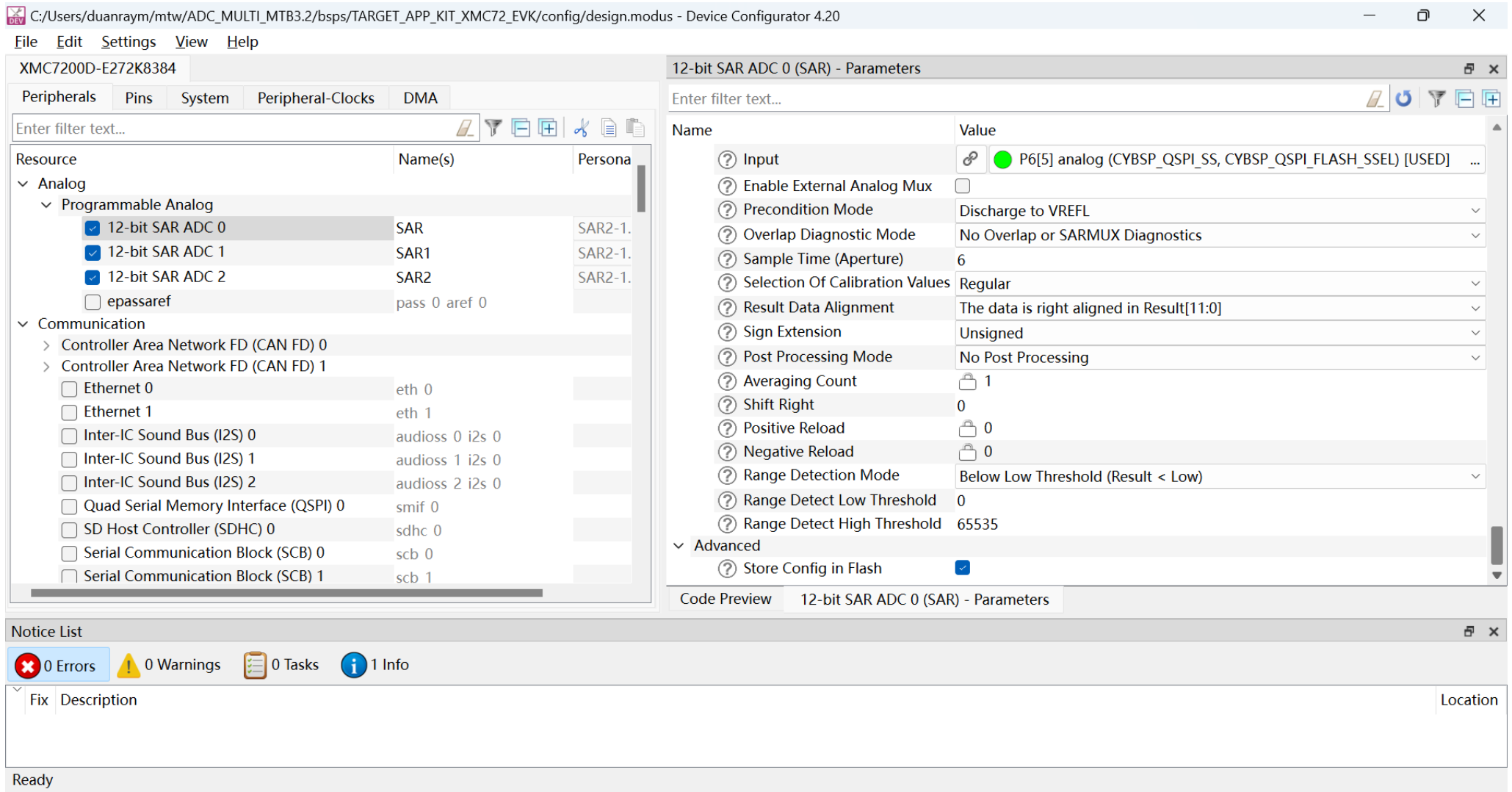
ADC0 channel 4 配置



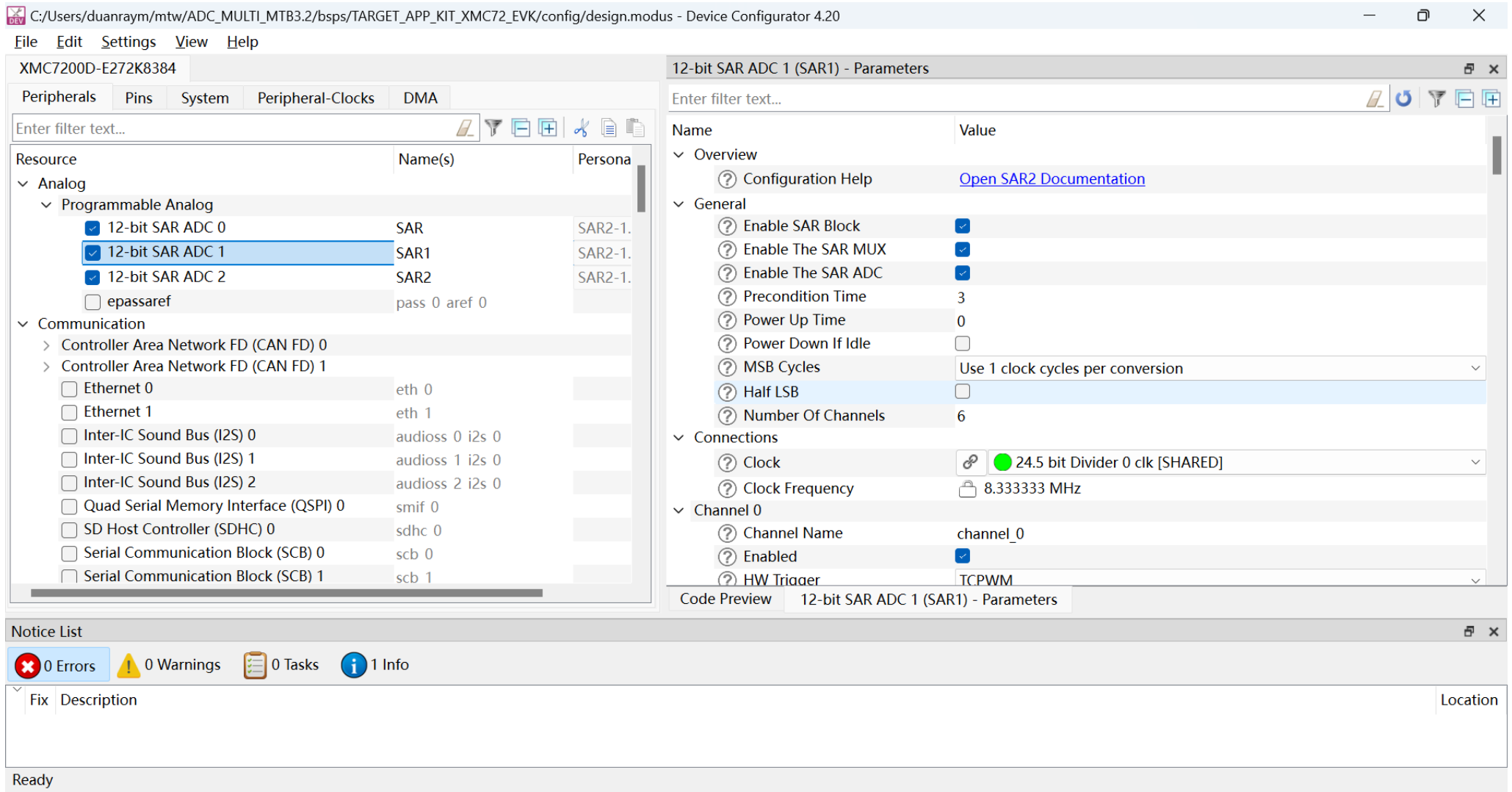
ADC0 channel 5 配置



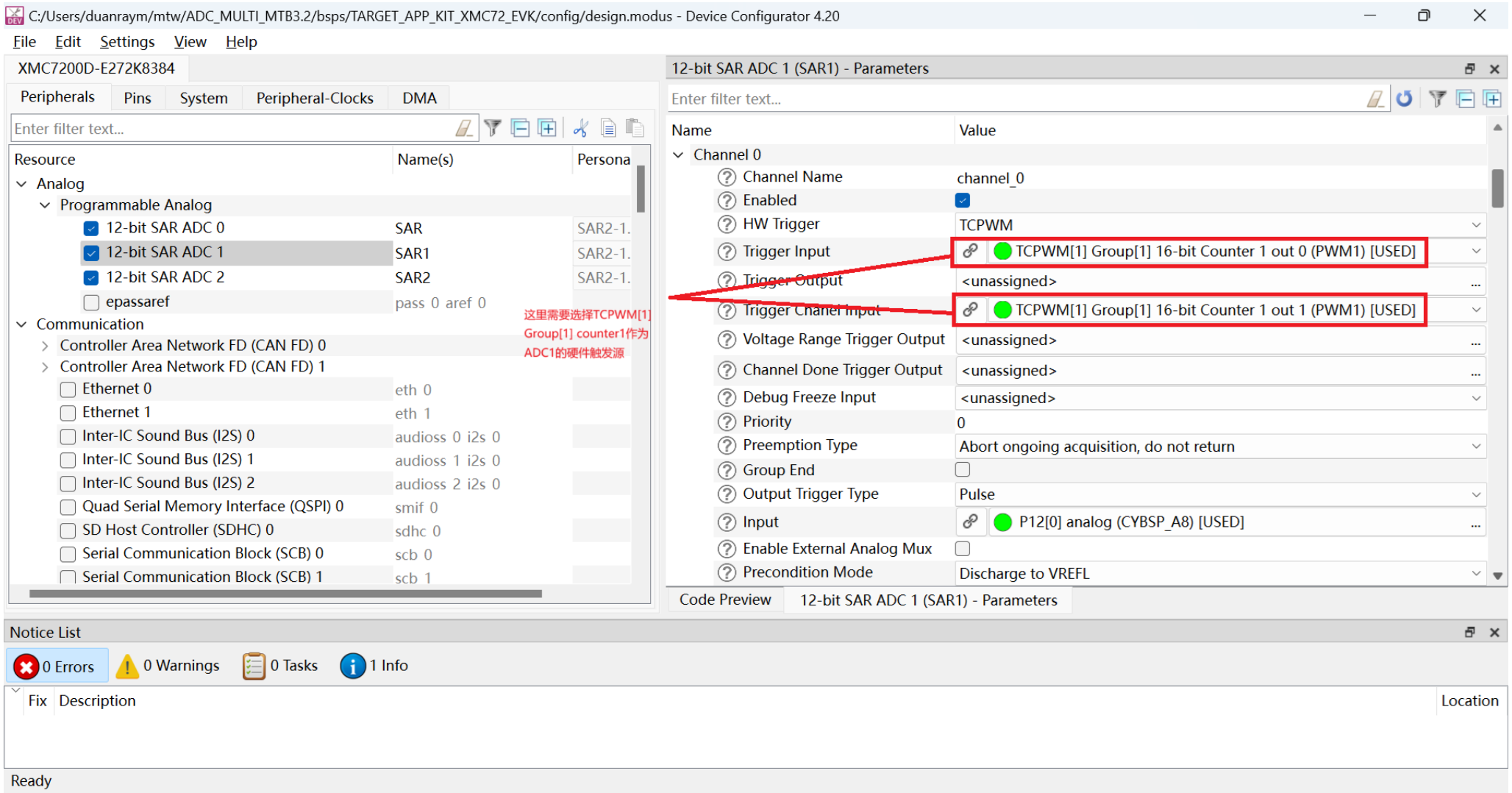
ADC0 channel 5 配置



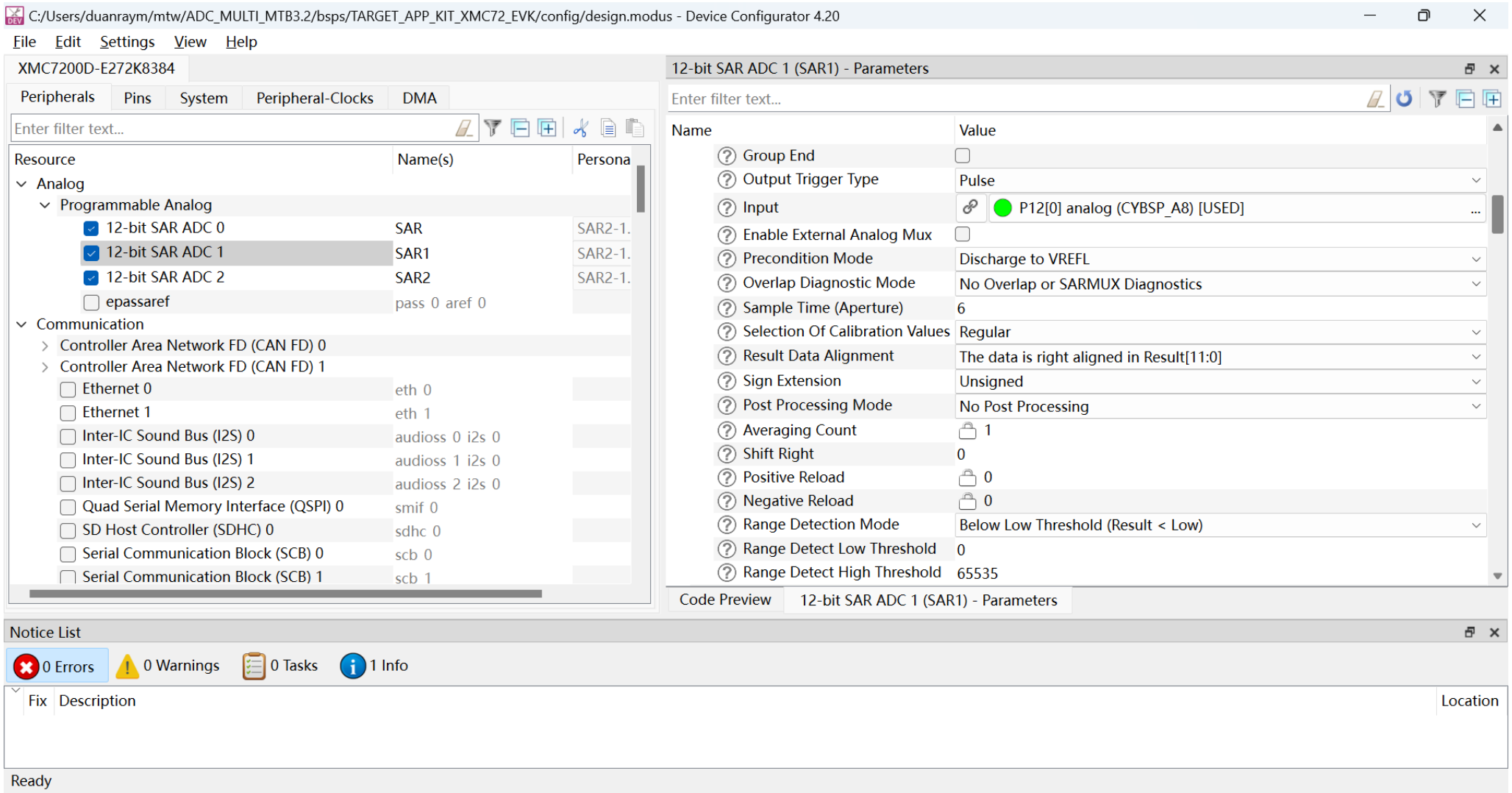
ADC1 配置



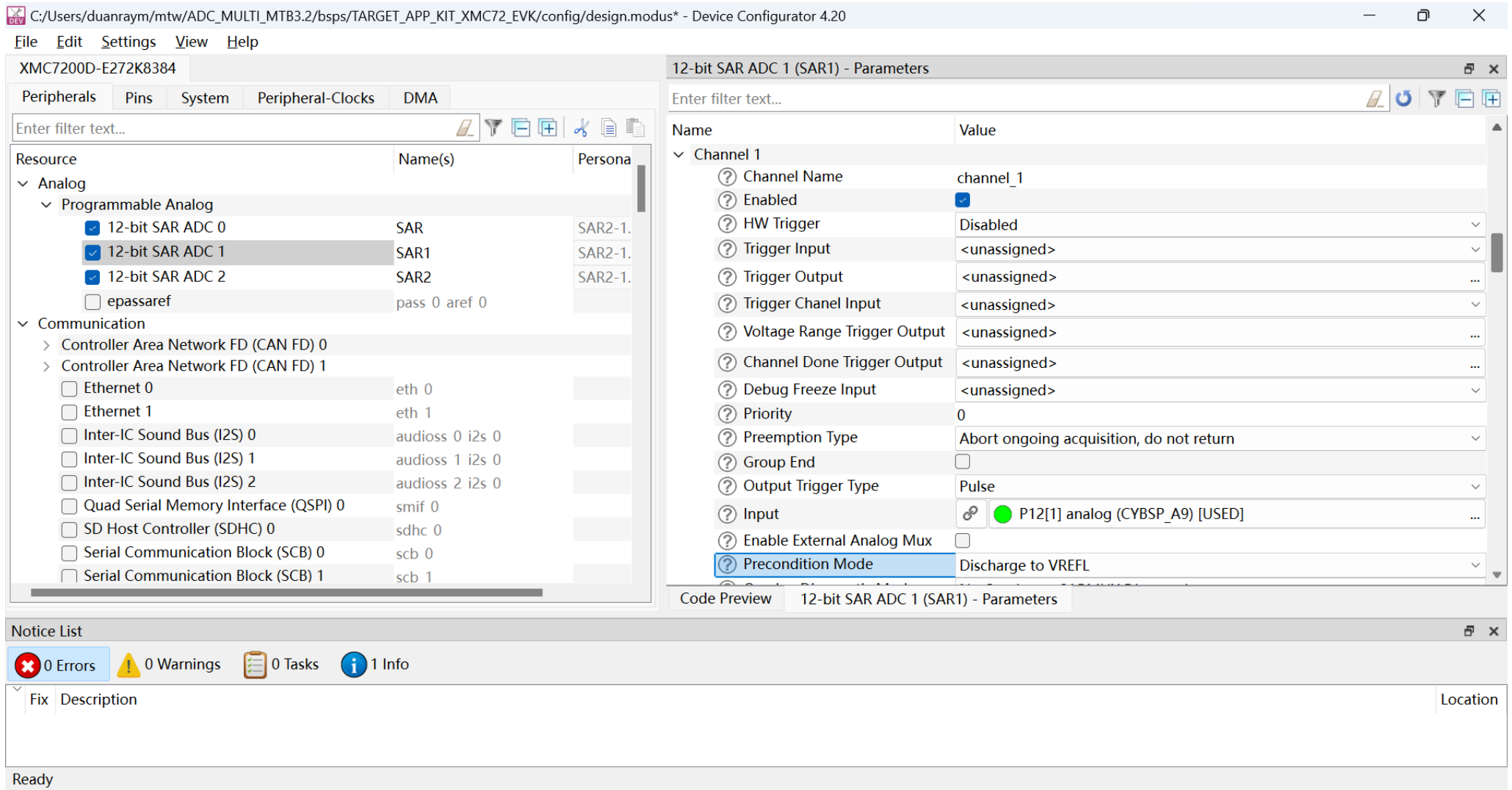
ADC1 channel 0 配置



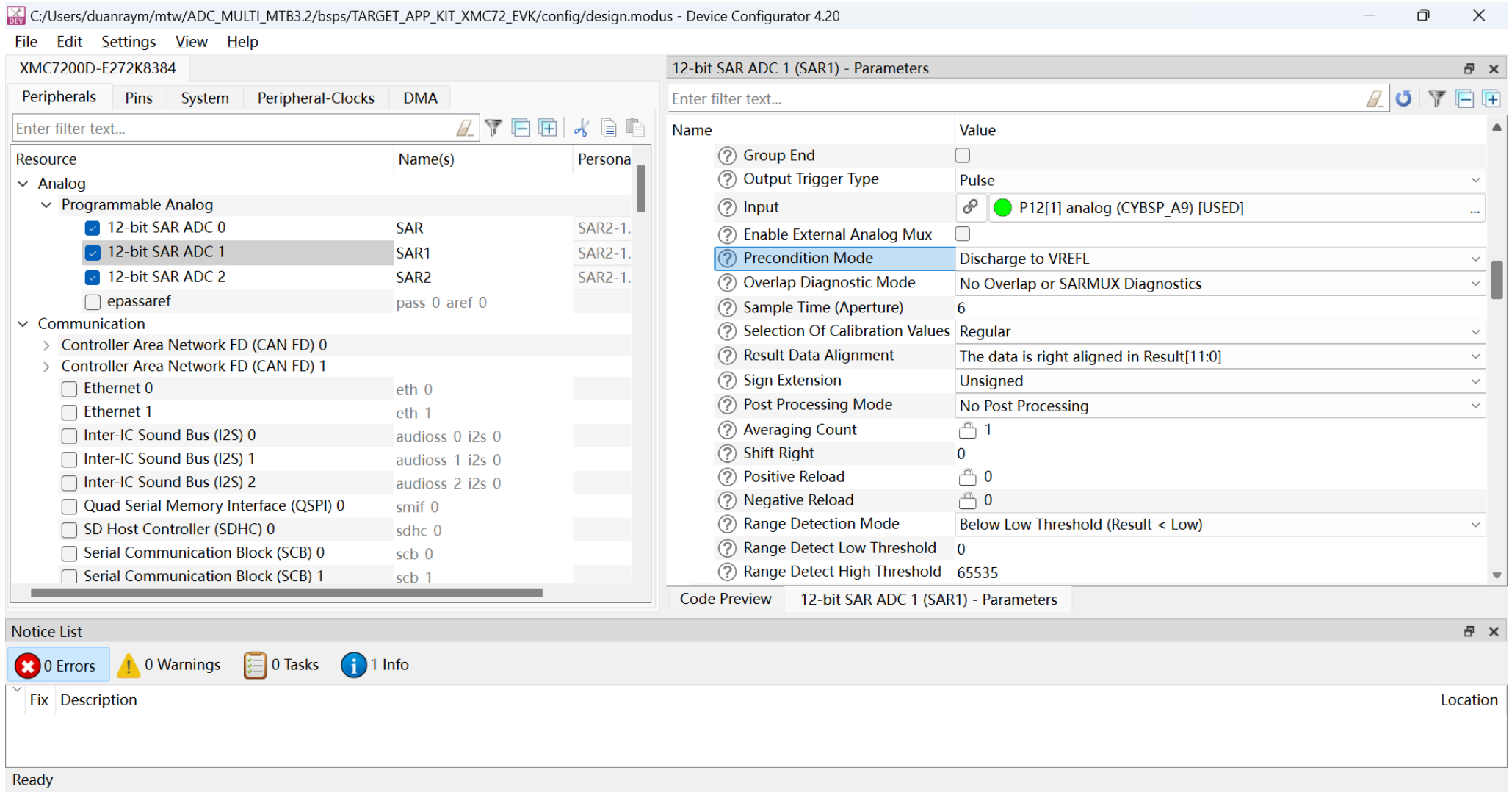
ADC1 channel 0 配置



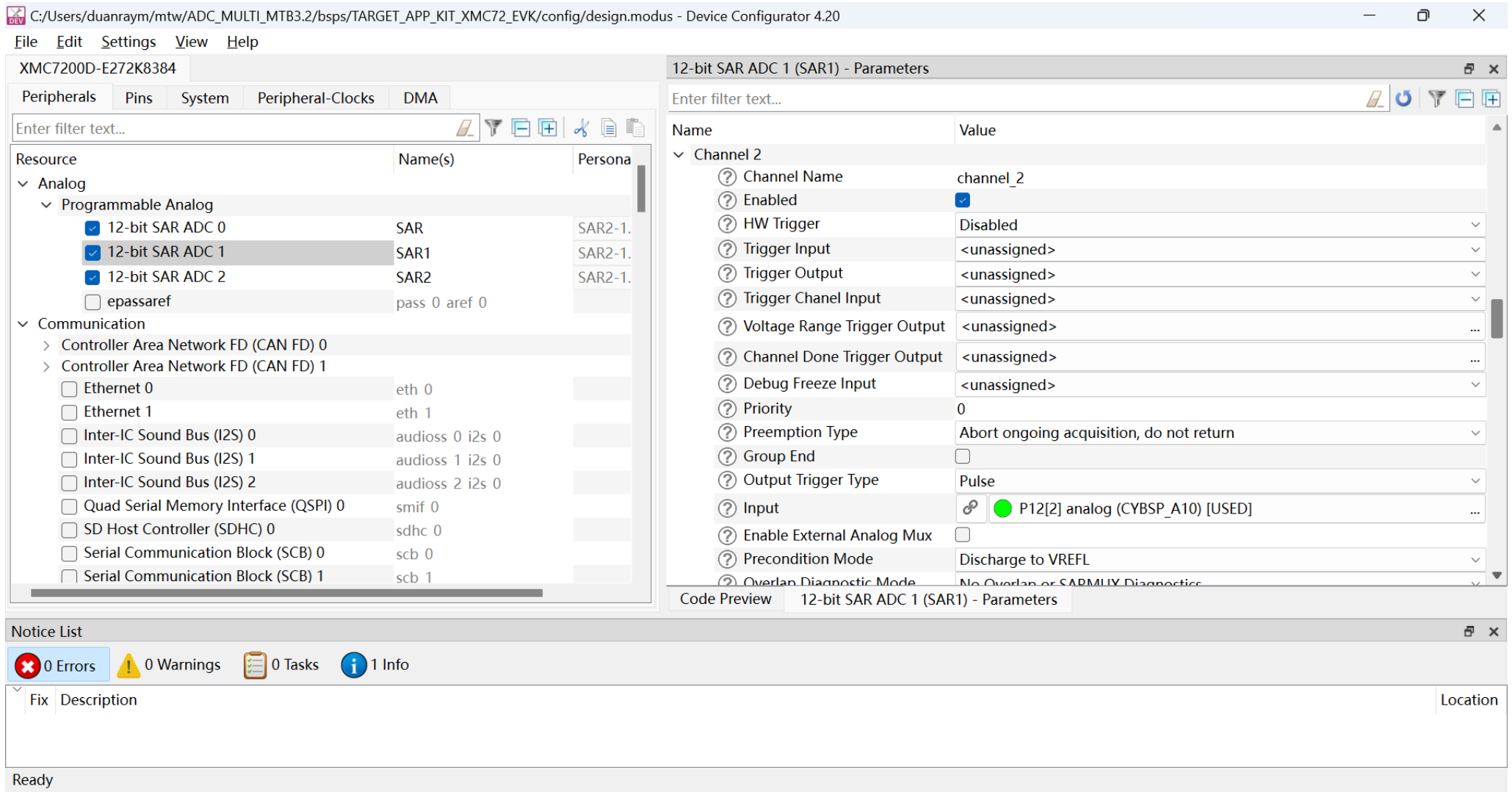
ADC1 channel 1 配置



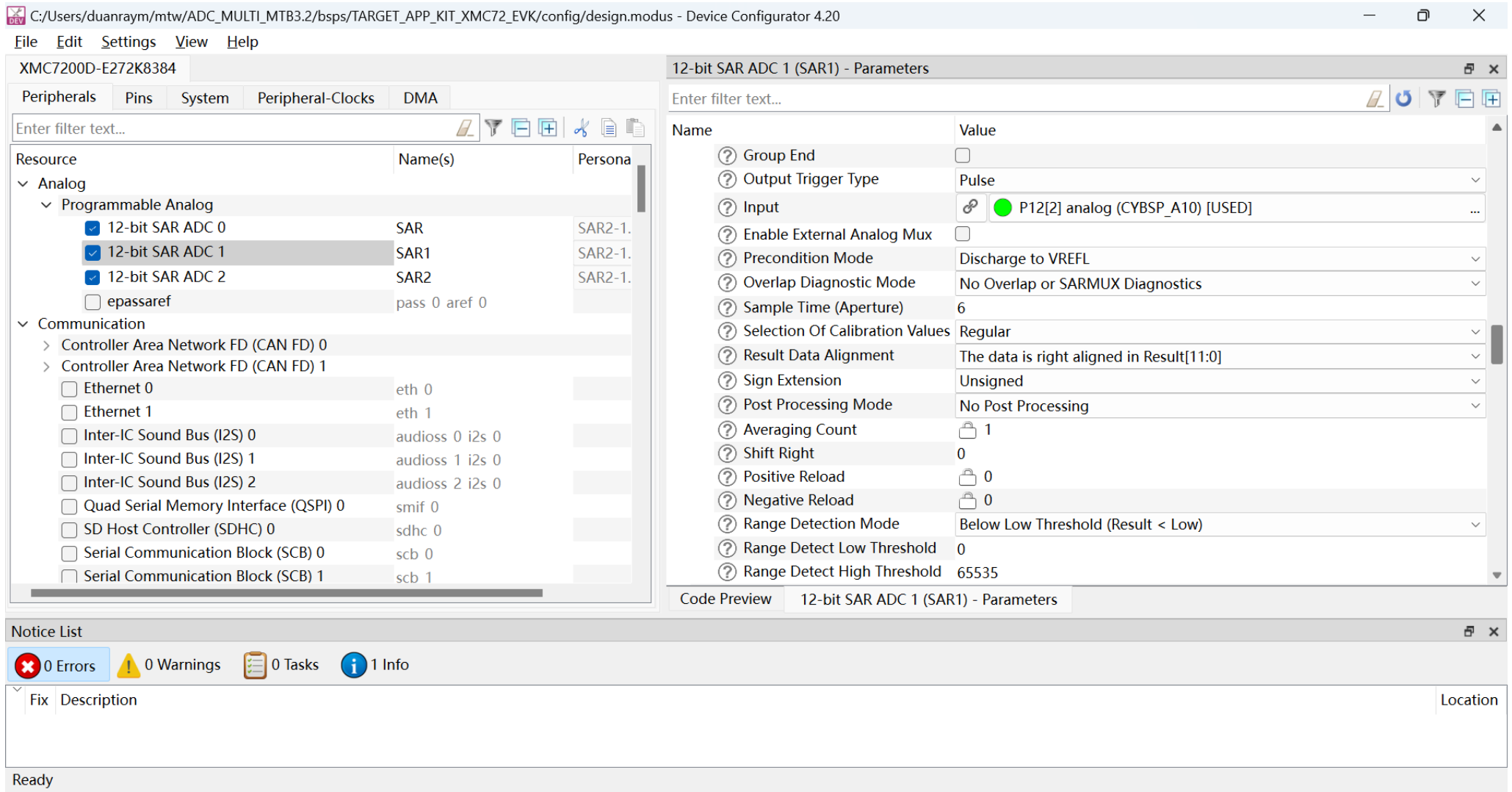
ADC1 channel 1 配置



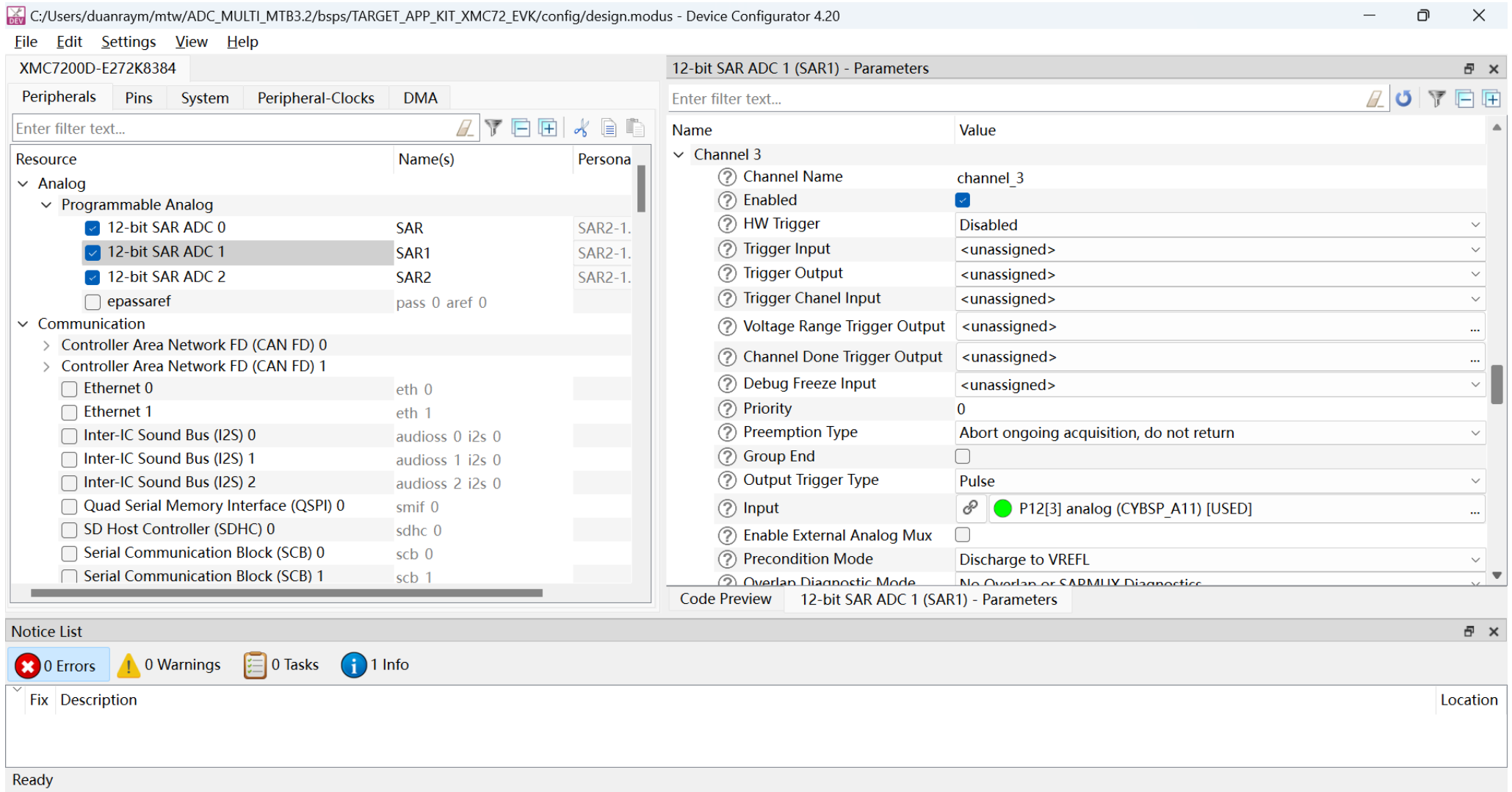
ADC1 channel 2 配置



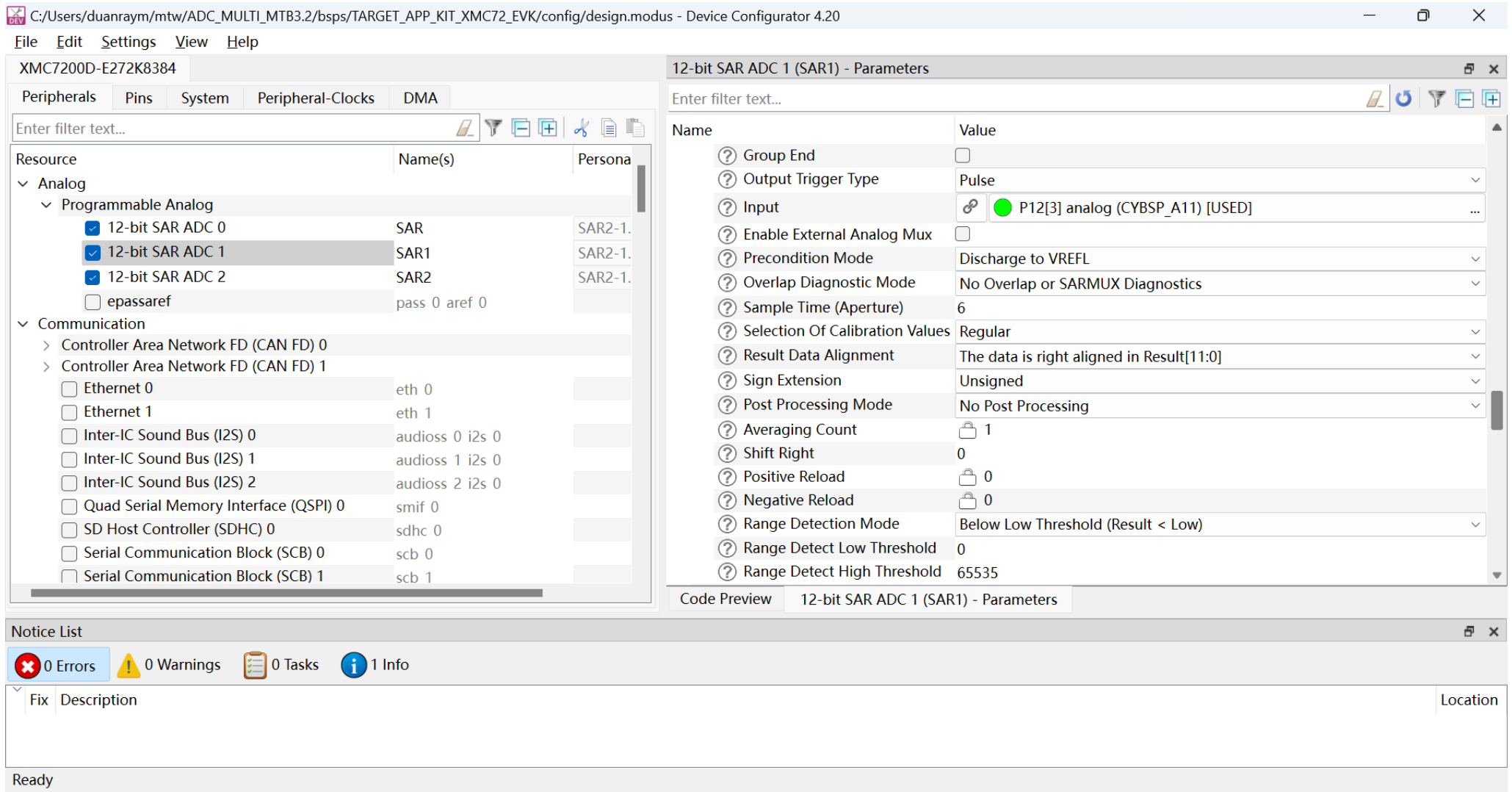
ADC1 channel 2 配置



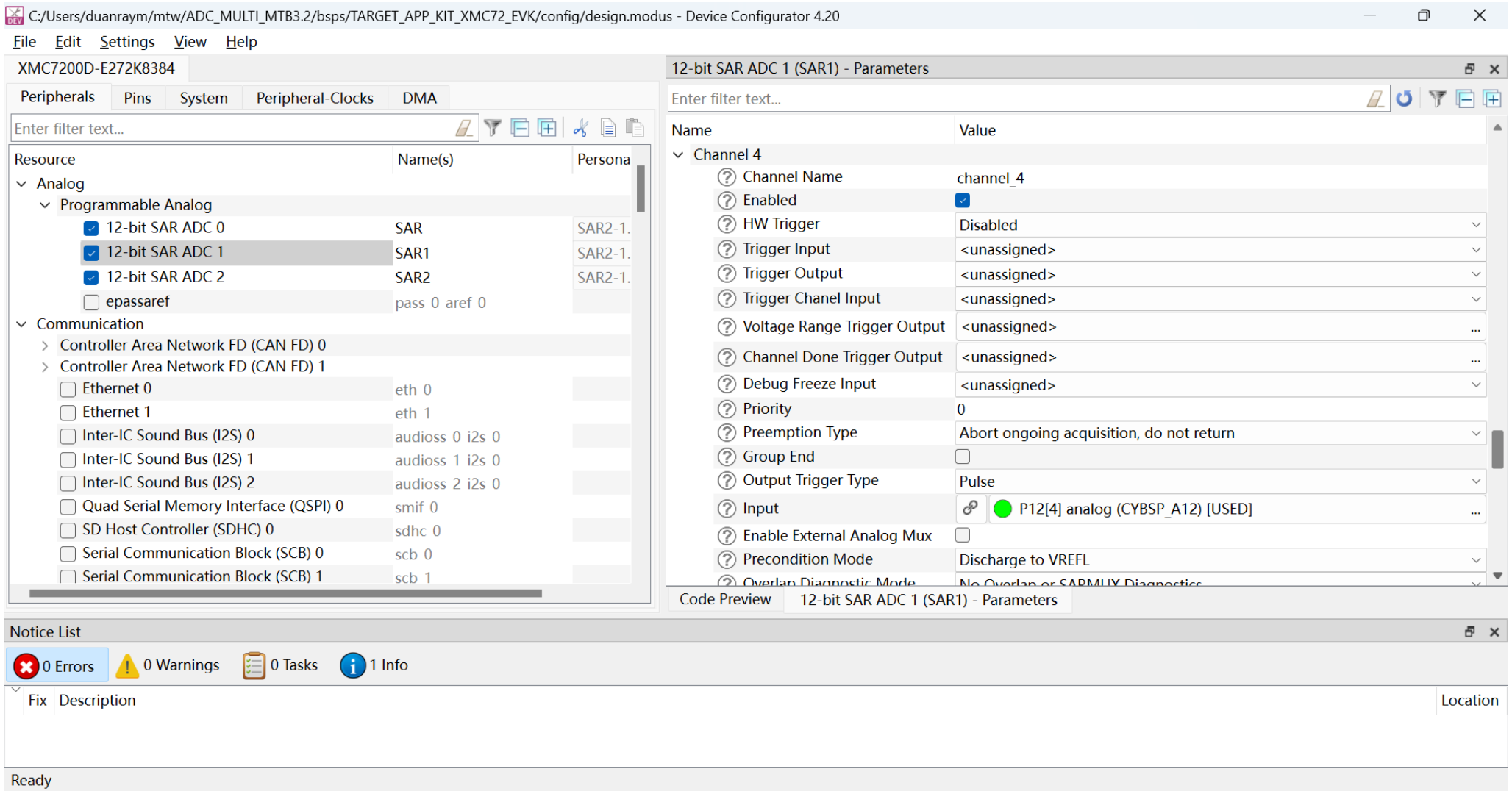
ADC1 channel 3 配置



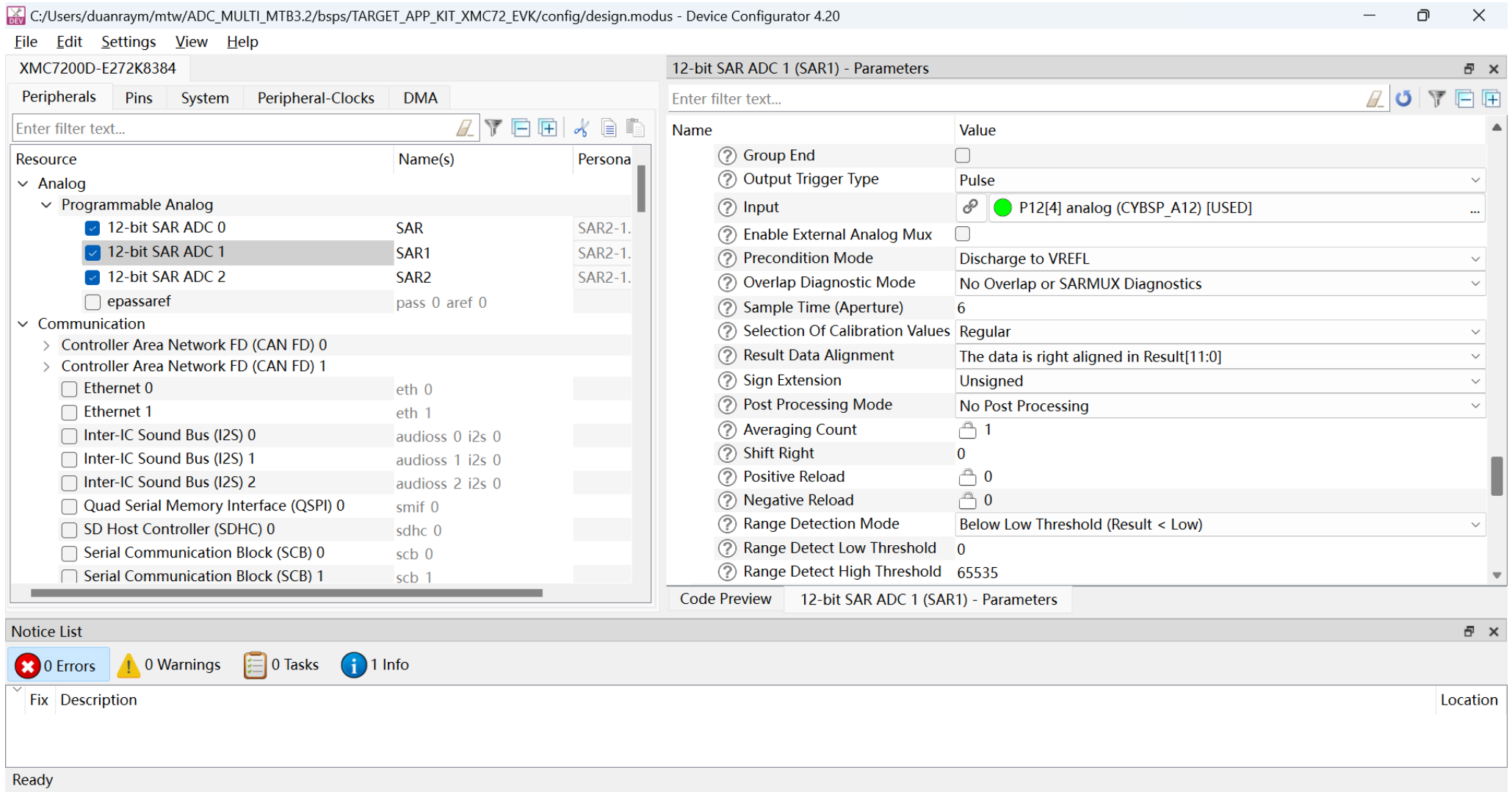
ADC1 channel 3 配置



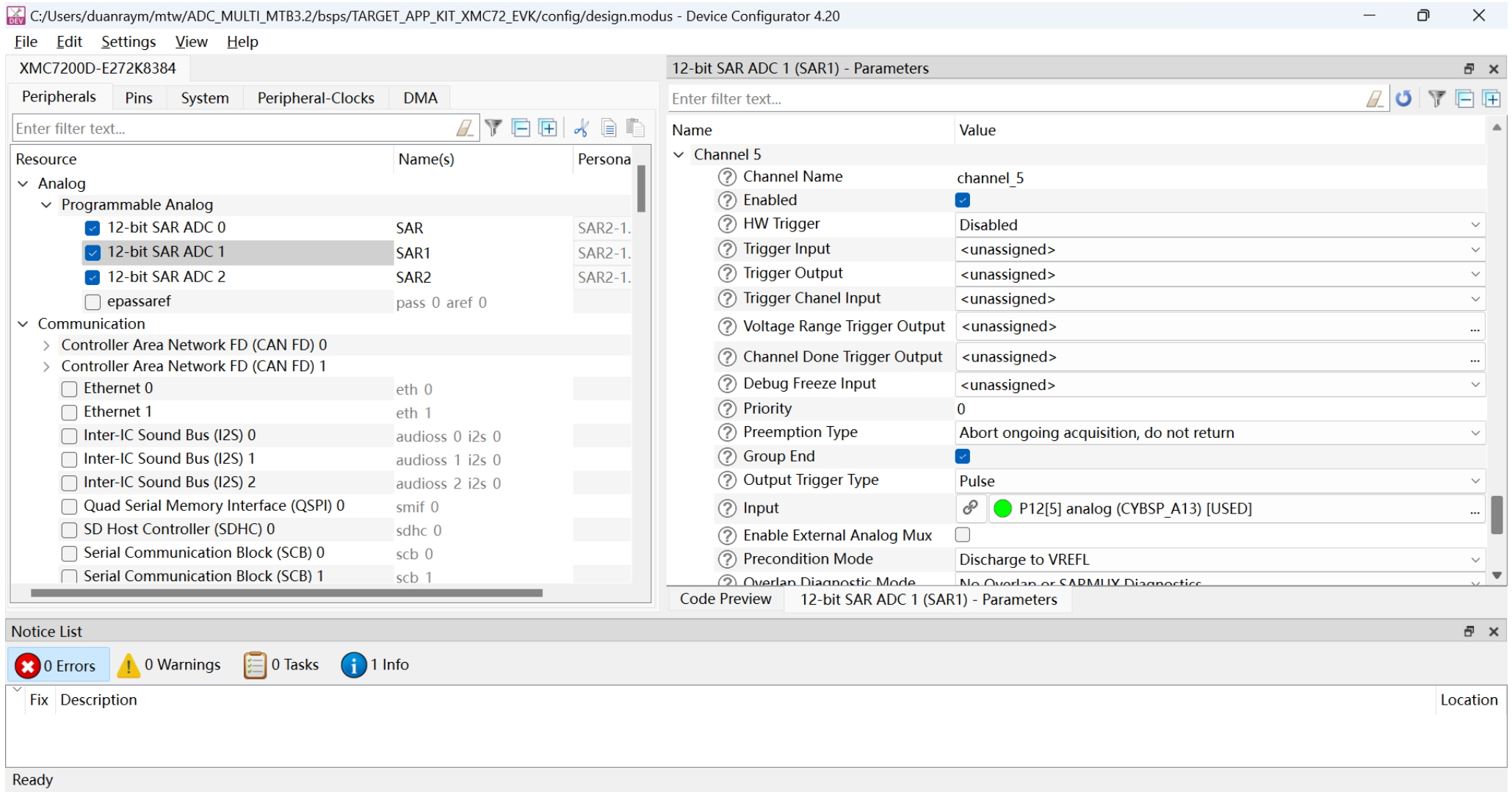
ADC1 channel 4 配置



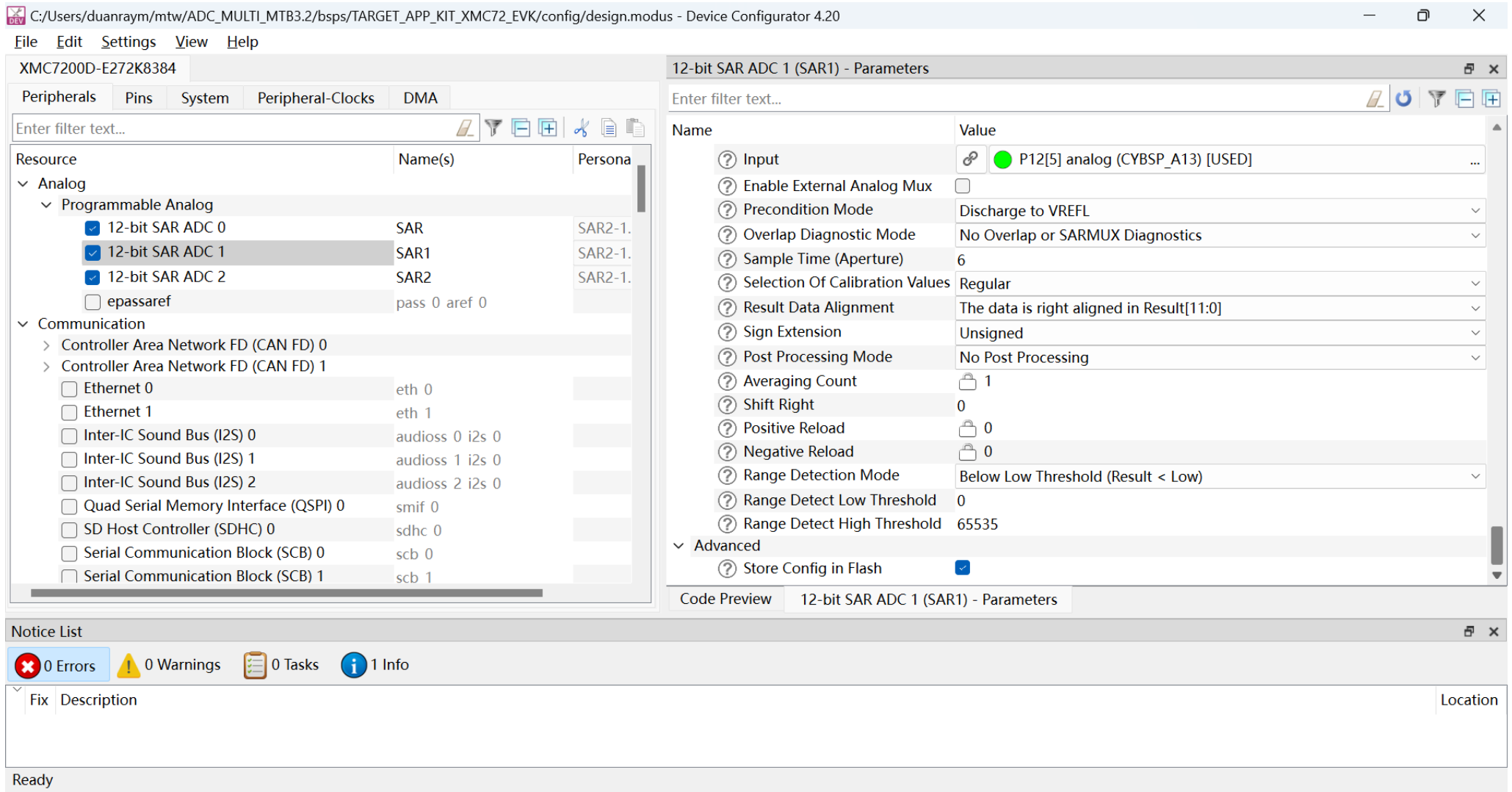
ADC1 channel 4 配置



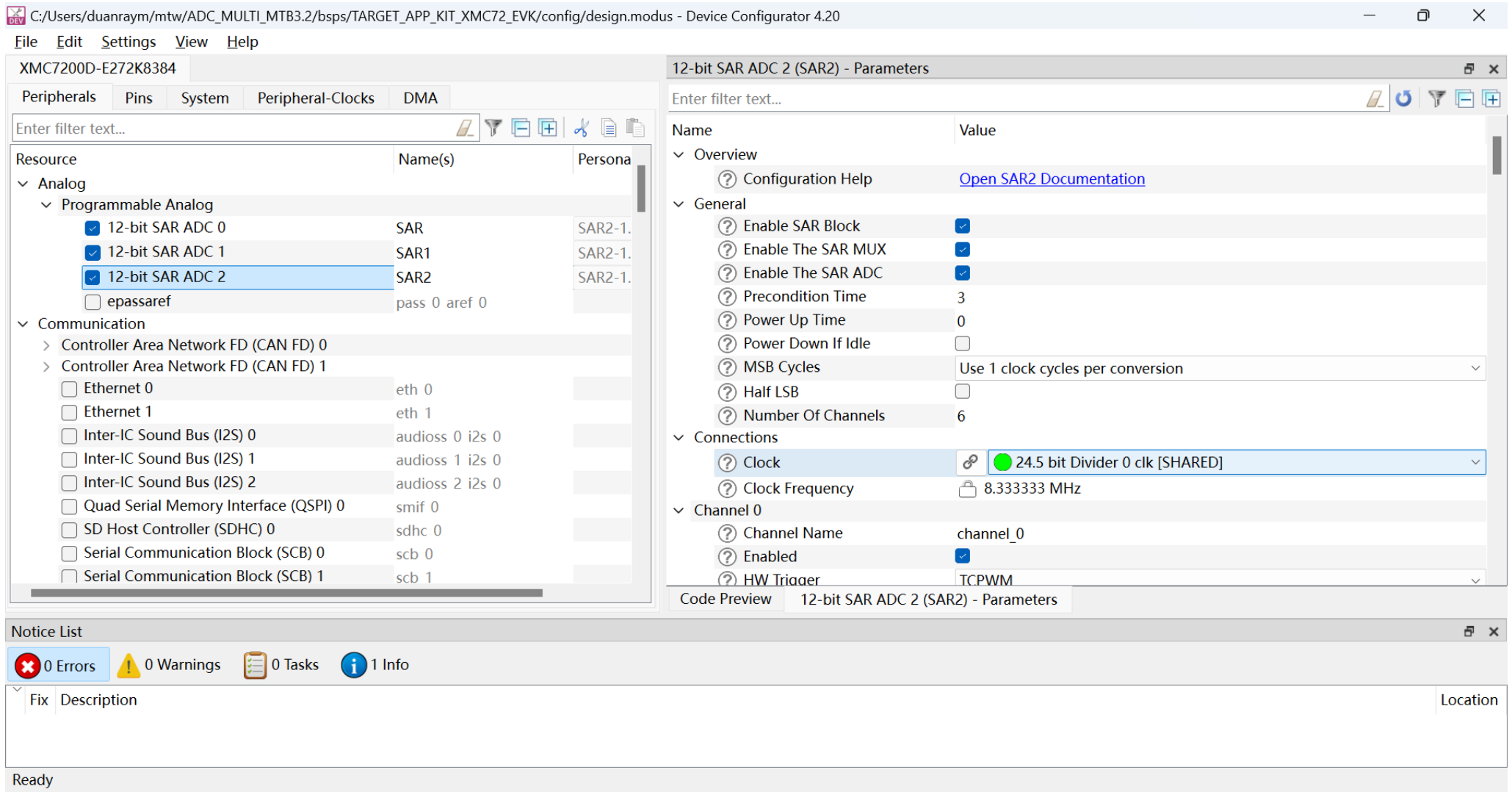
ADC1 channel 5 配置



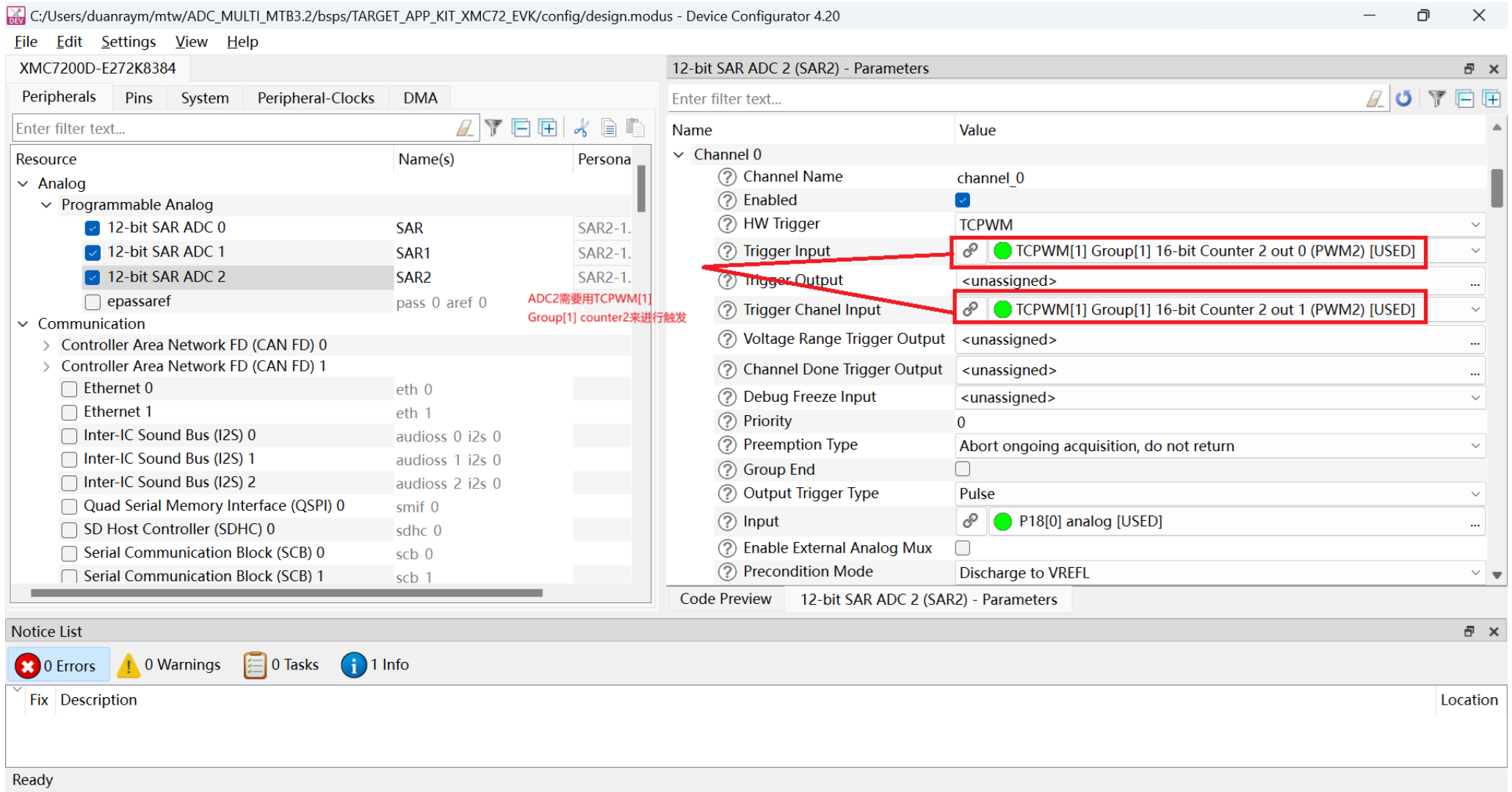
ADC1 channel 5 配置



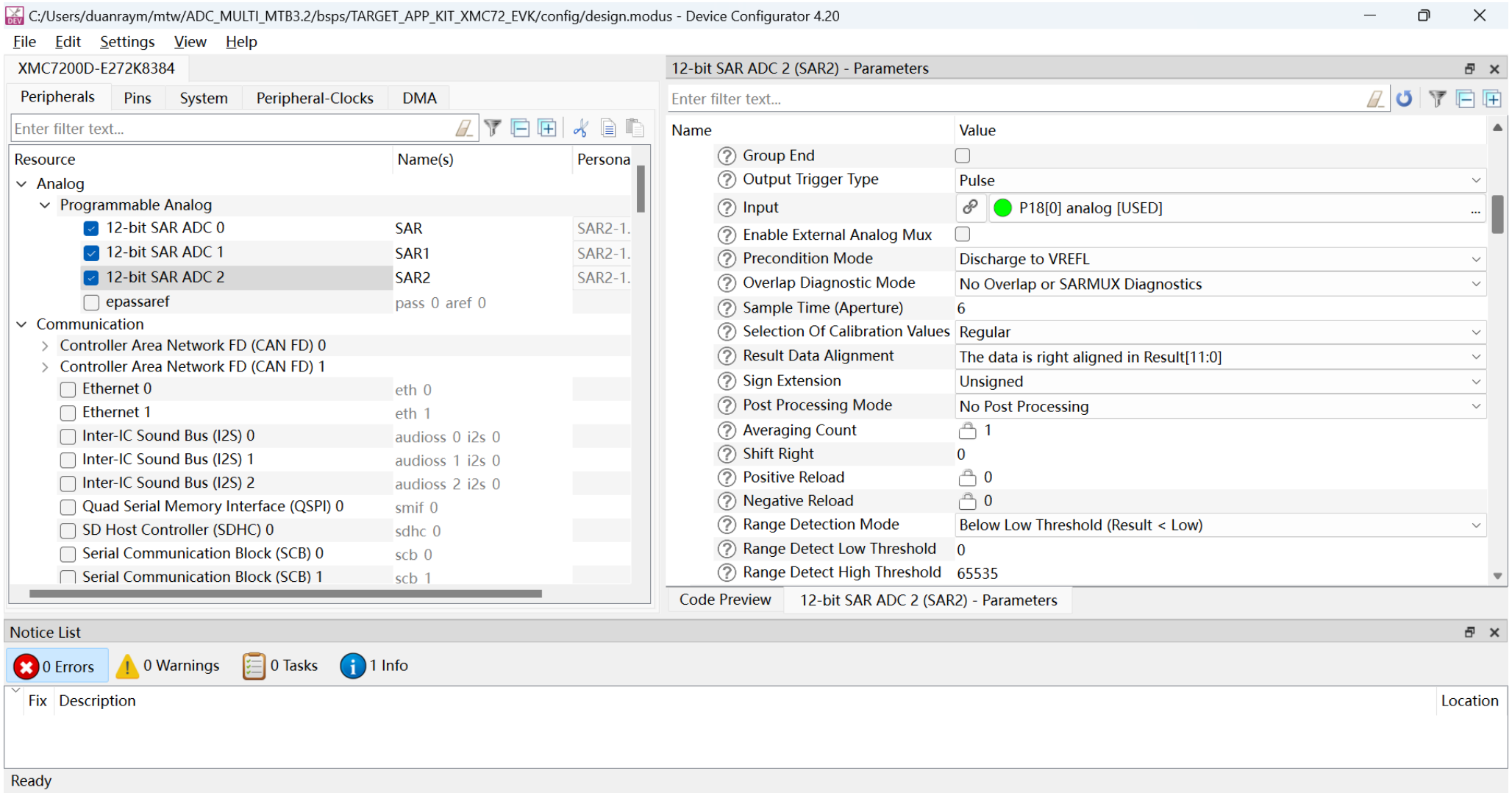
ADC2 配置



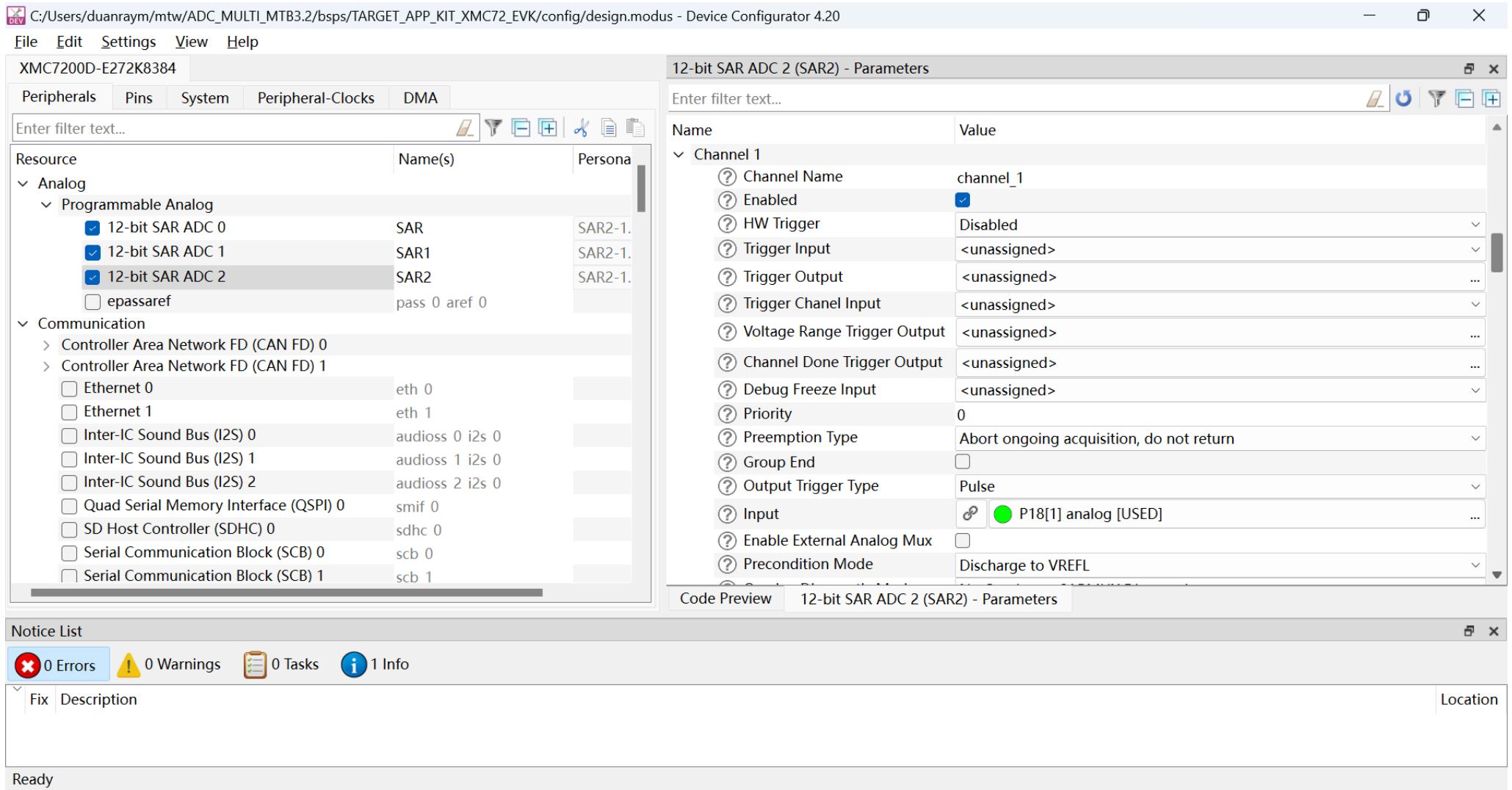
ADC2 channel 0 配置



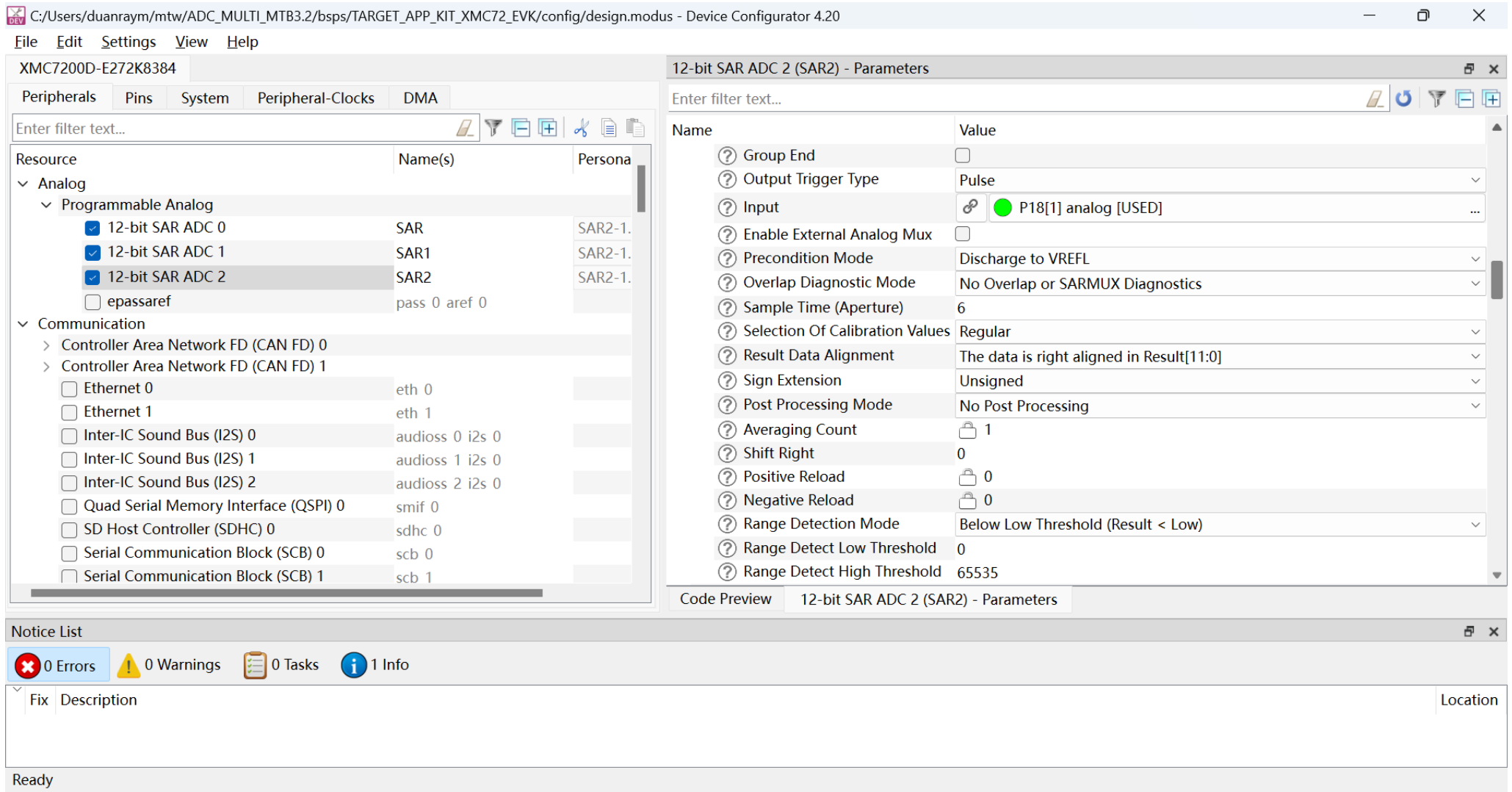
ADC2 channel 0 配置



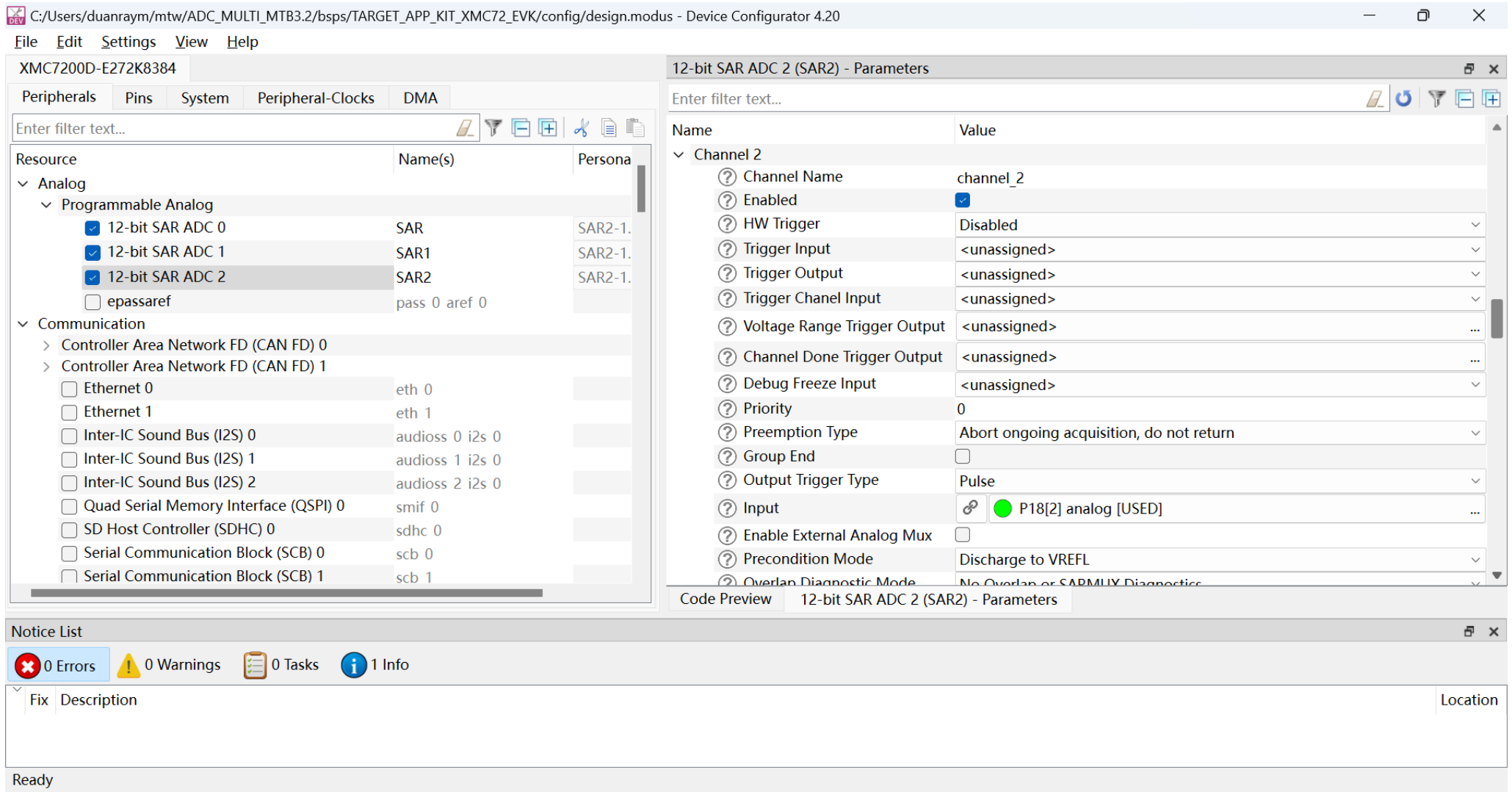
ADC2 channel 1 配置



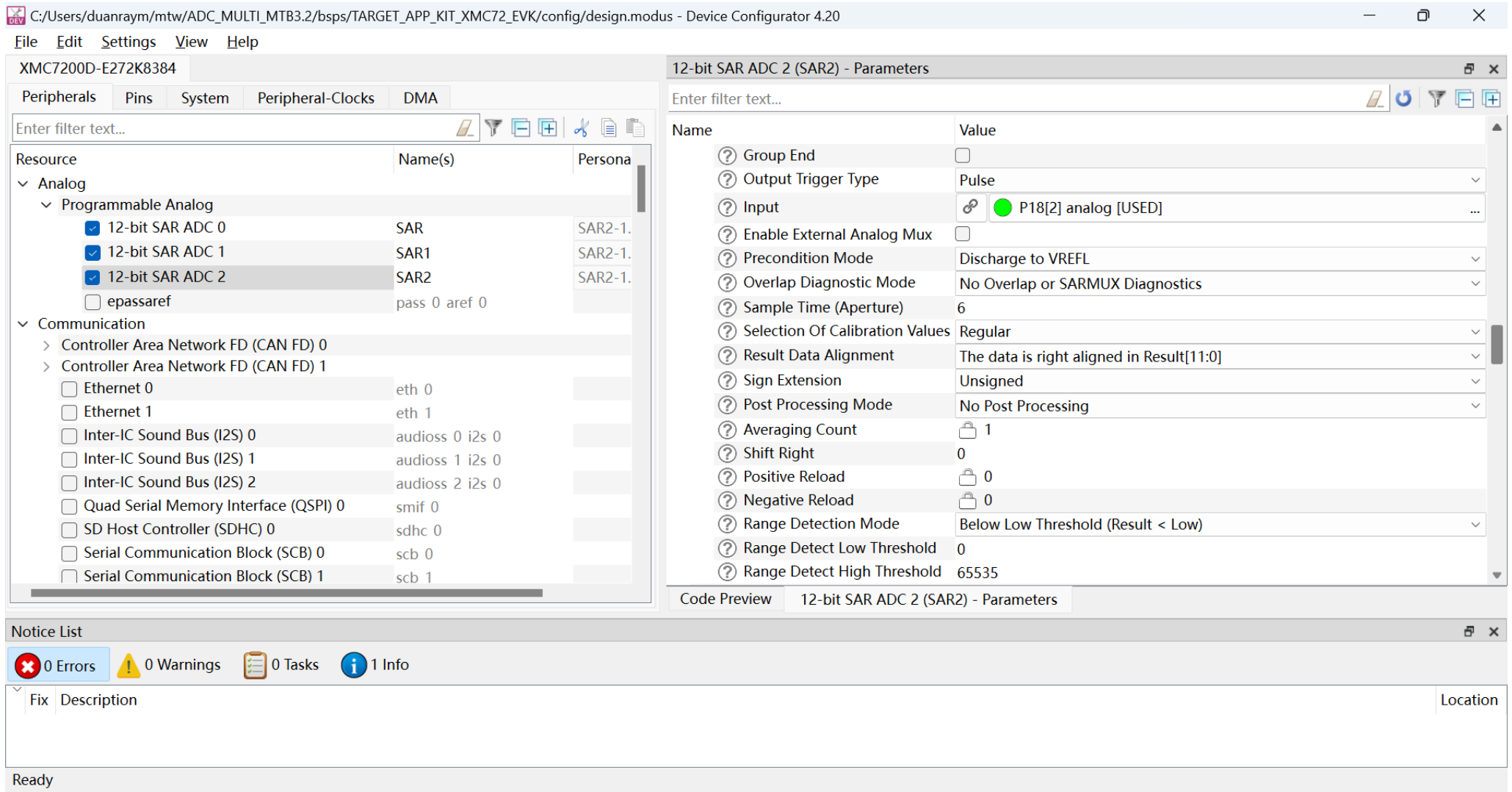
ADC2 channel 1 配置



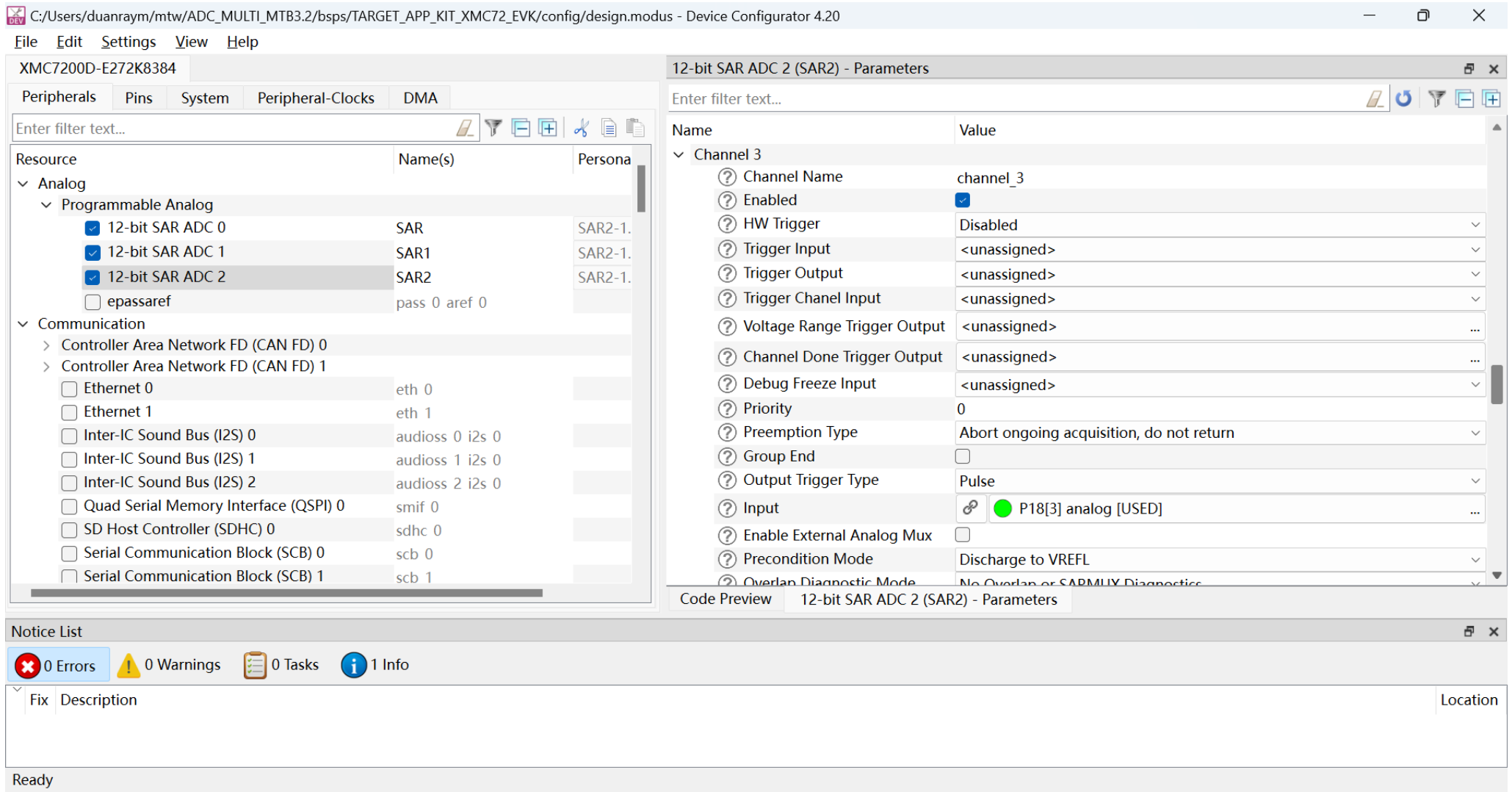
ADC2 channel 2 配置



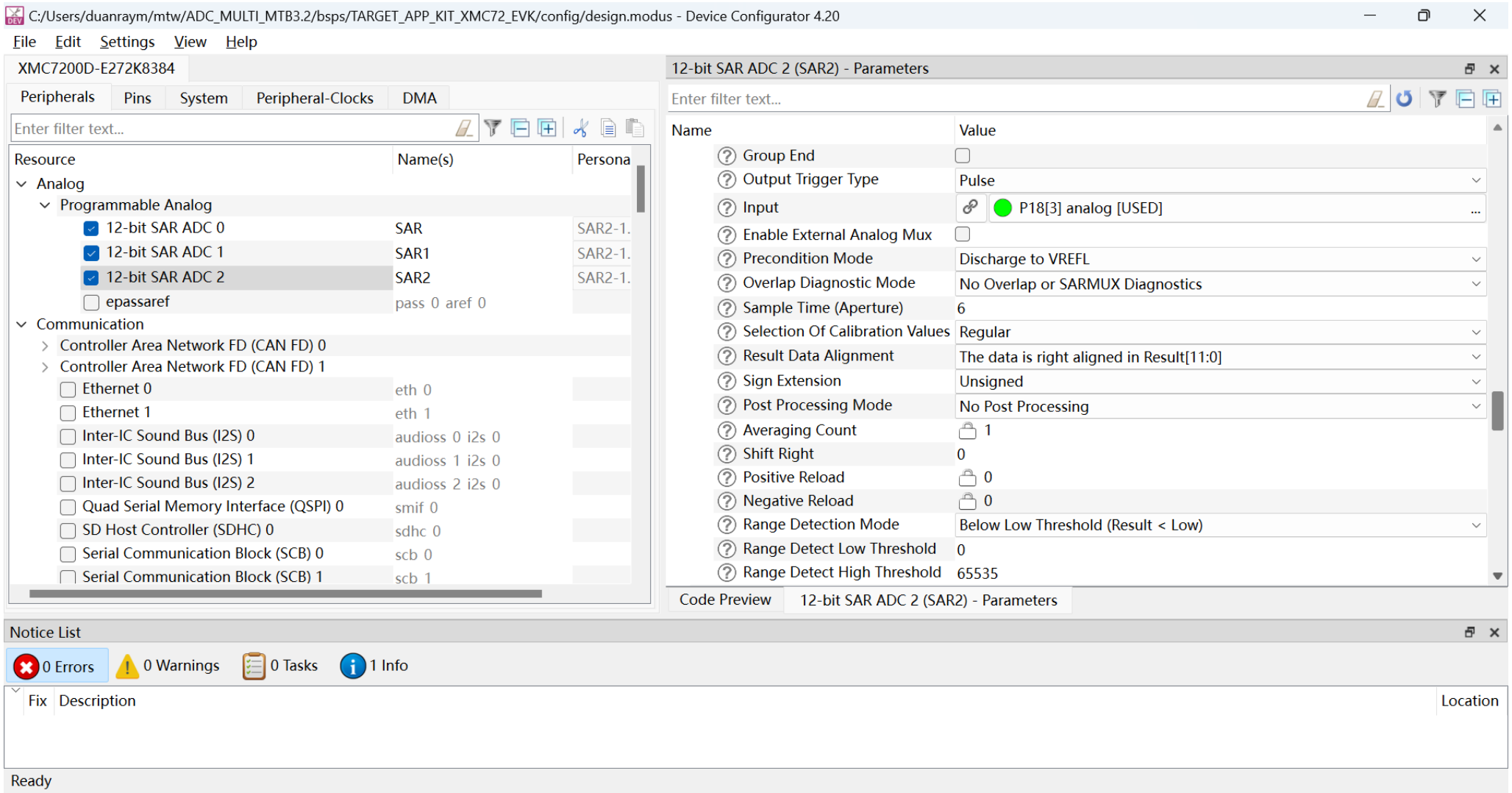
ADC2 channel 2 配置



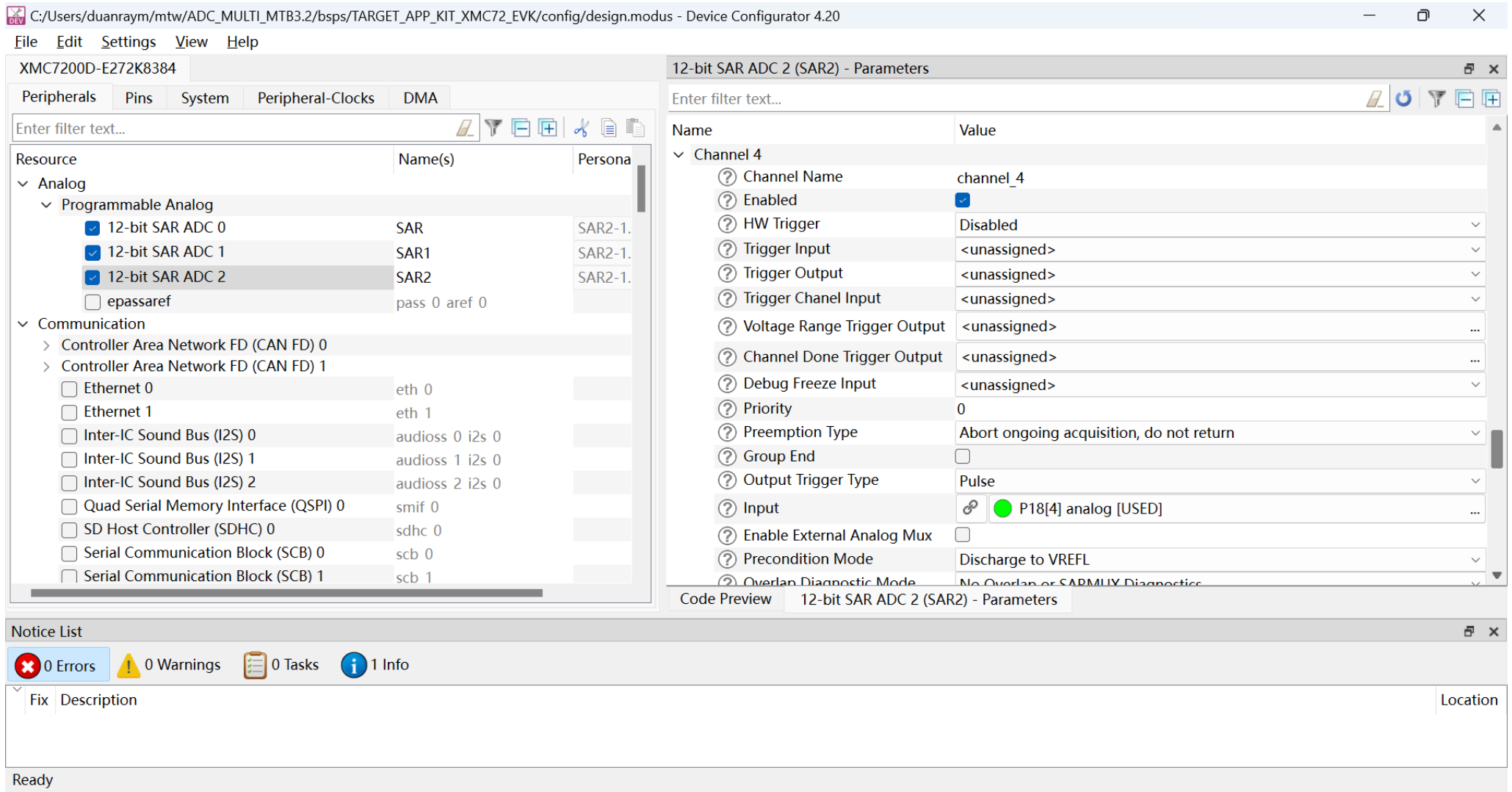
ADC2 channel 3 配置



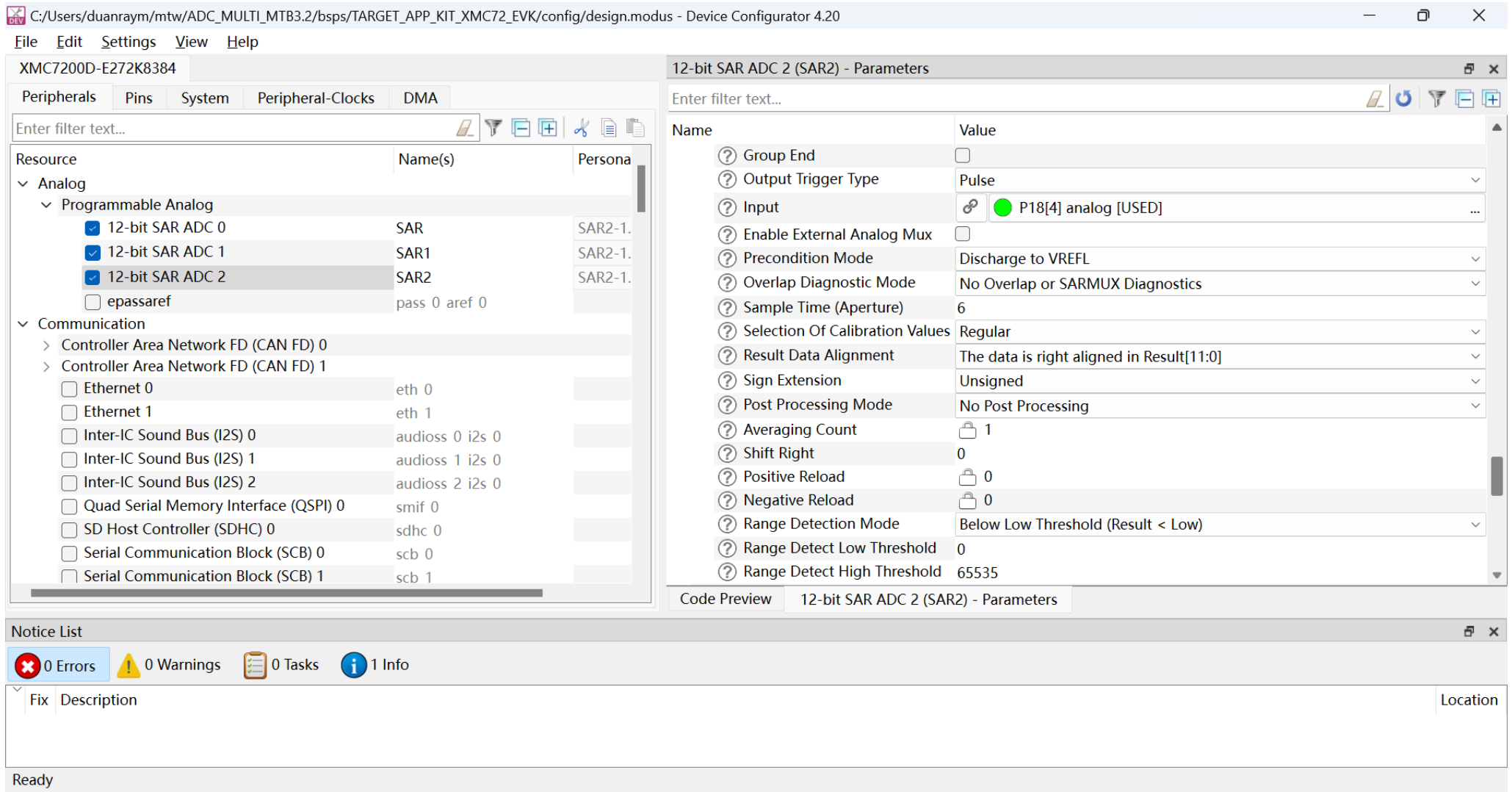
ADC2 channel 3 配置



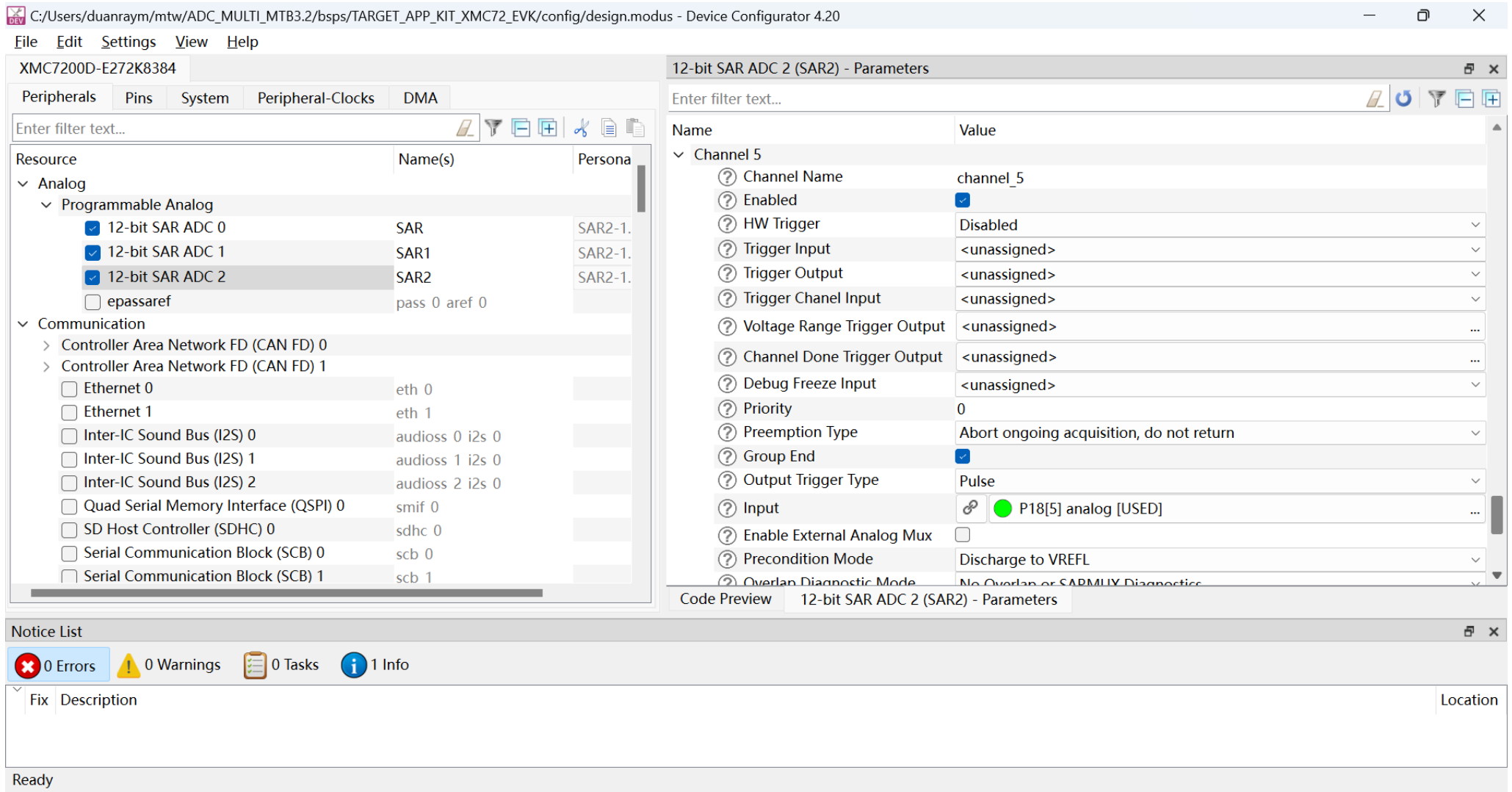
ADC2 channel 4 配置



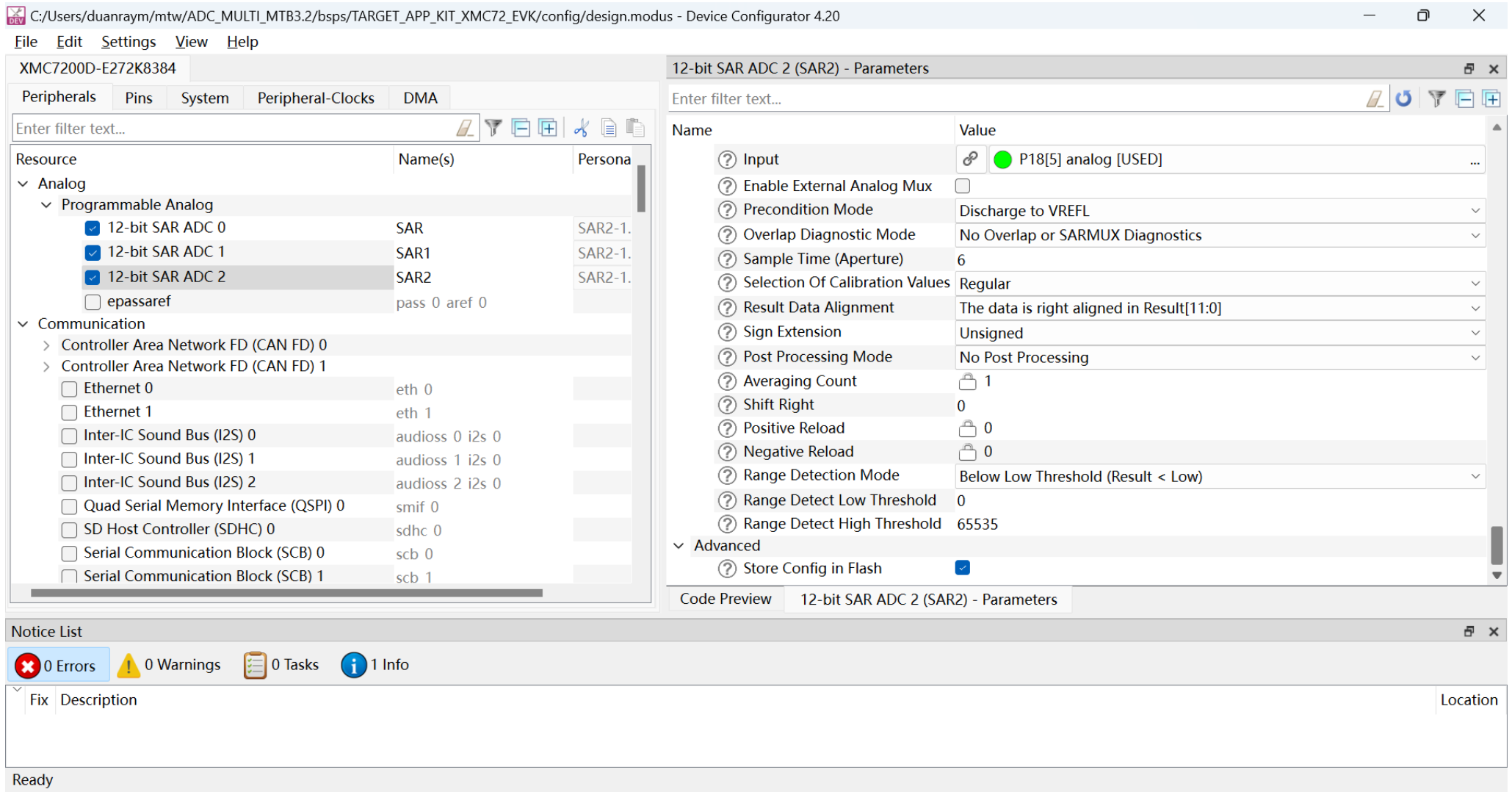
ADC2 channel 4 配置



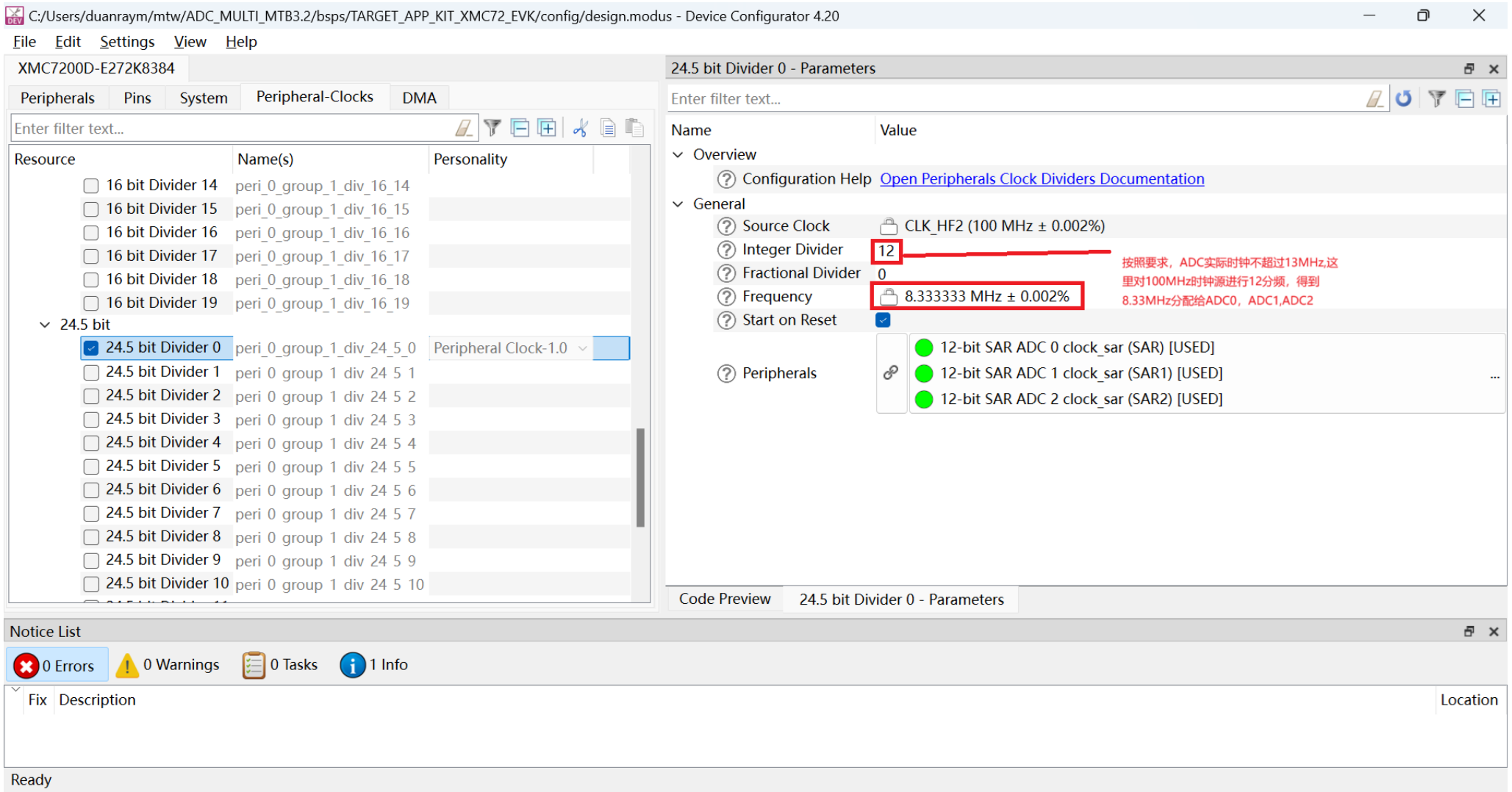
ADC2 channel 5 配置



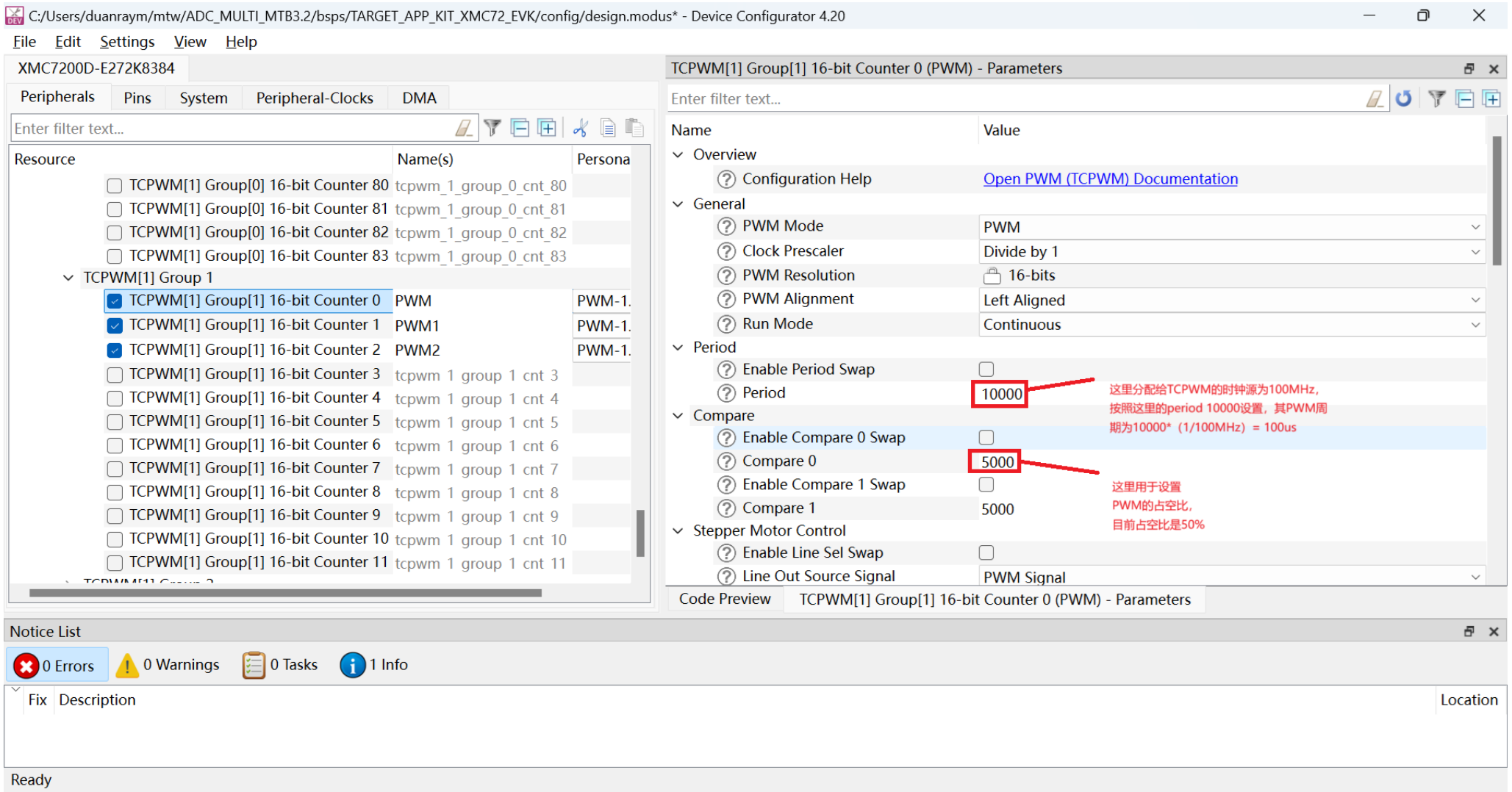
ADC2 channel 5 配置



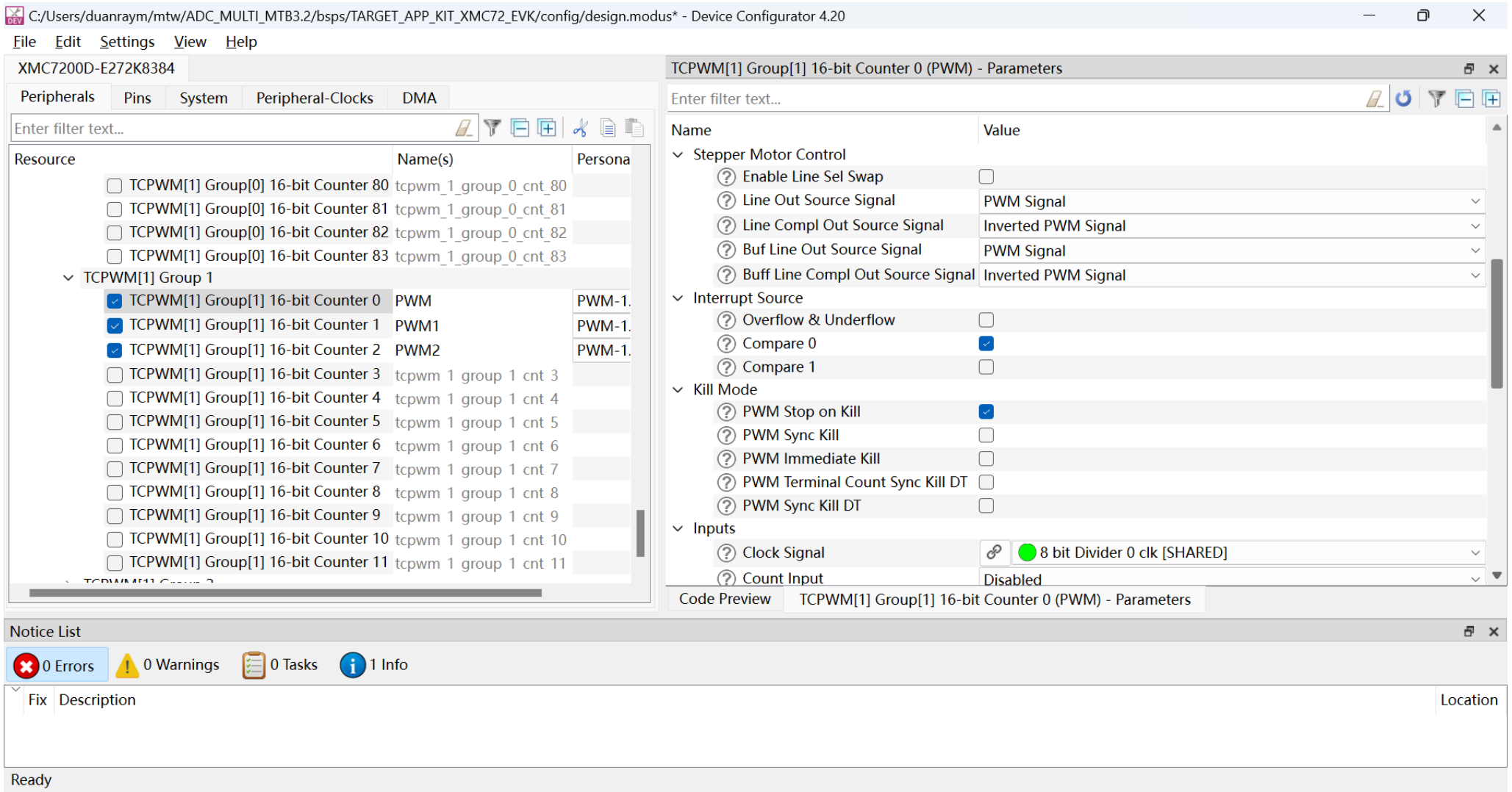
ADC时钟配置



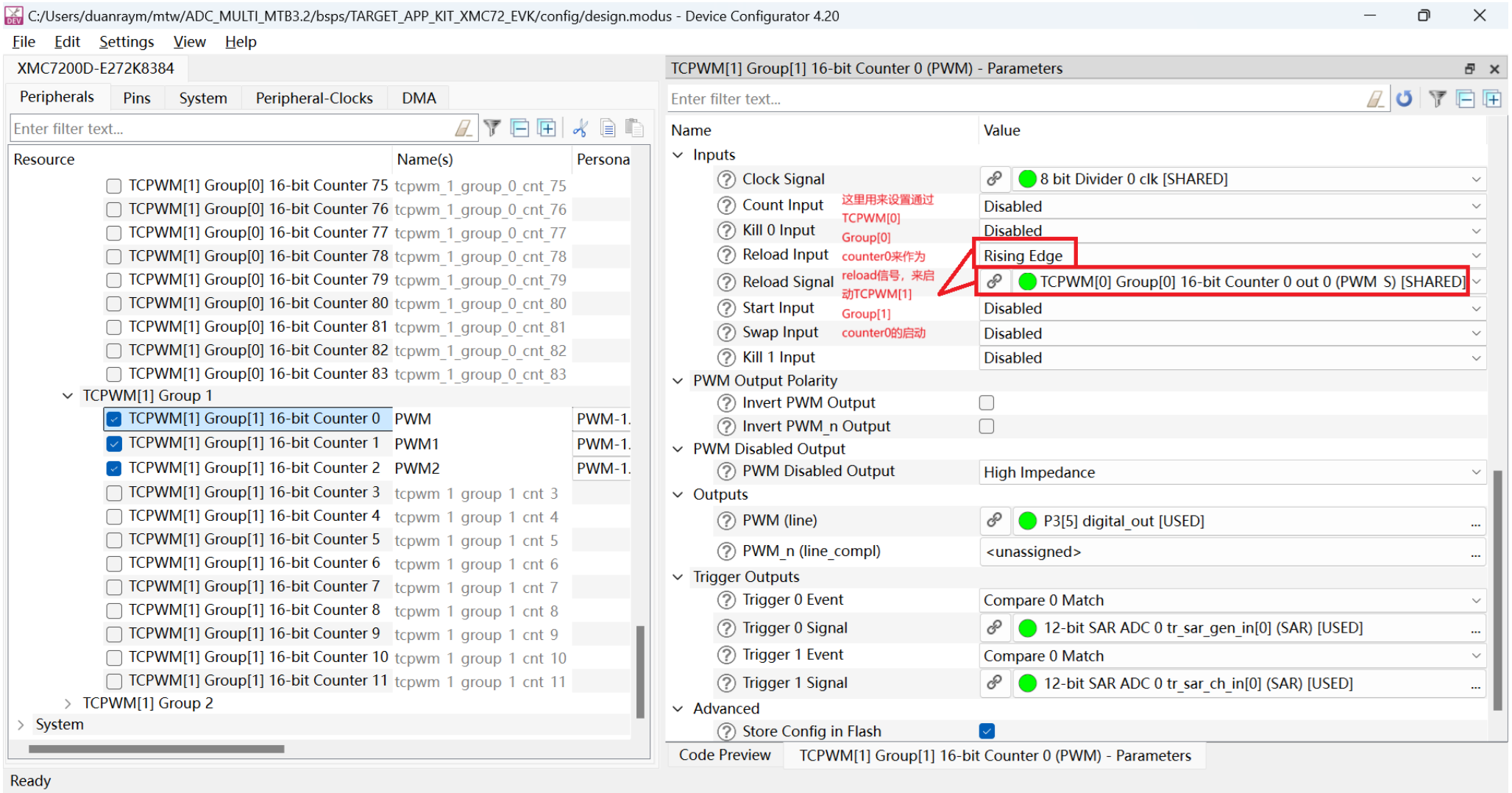
TCPWM[1] Group[1] Counter0配置



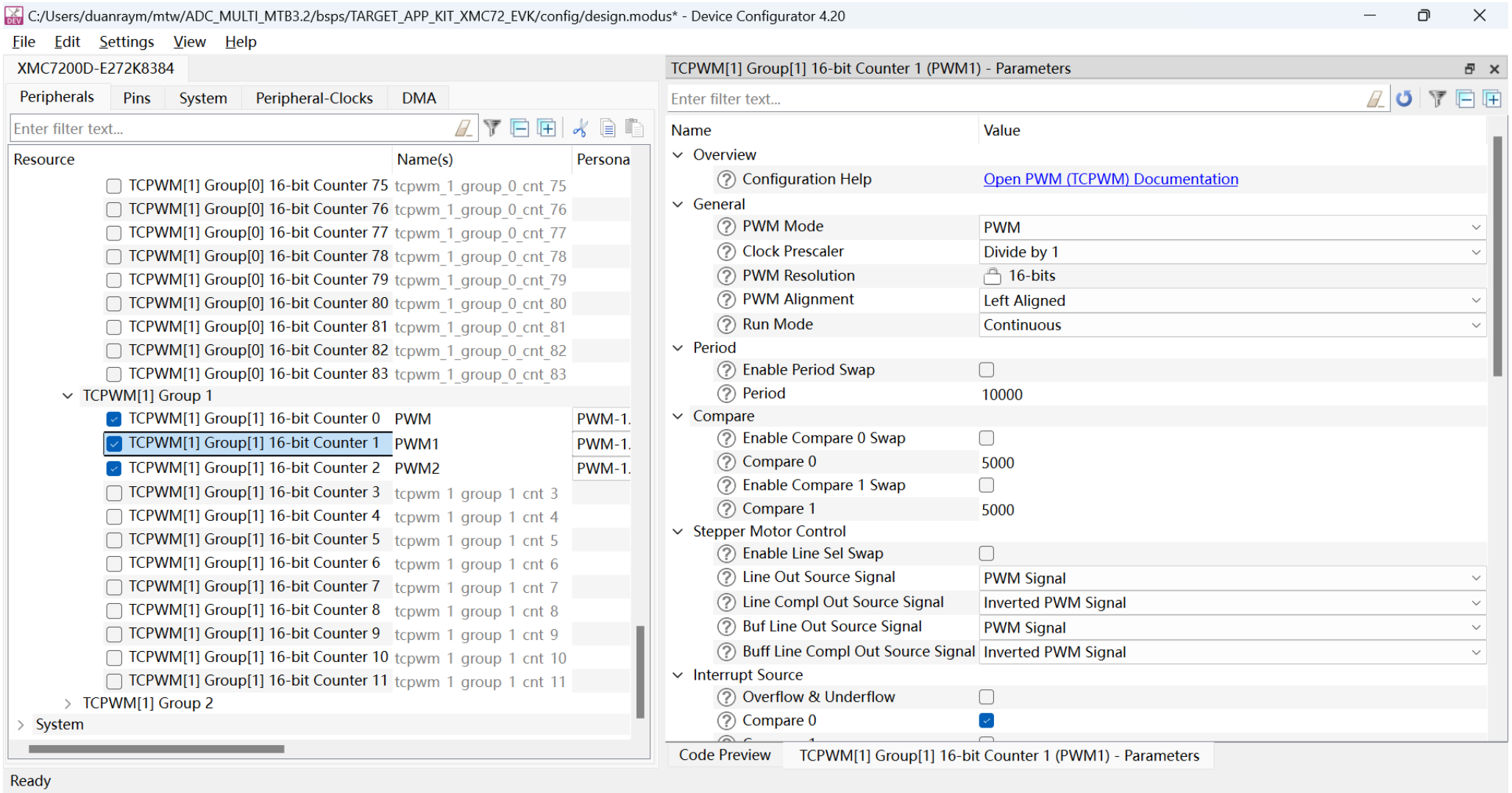
TCPWM[1] Group[1] Counter0配置



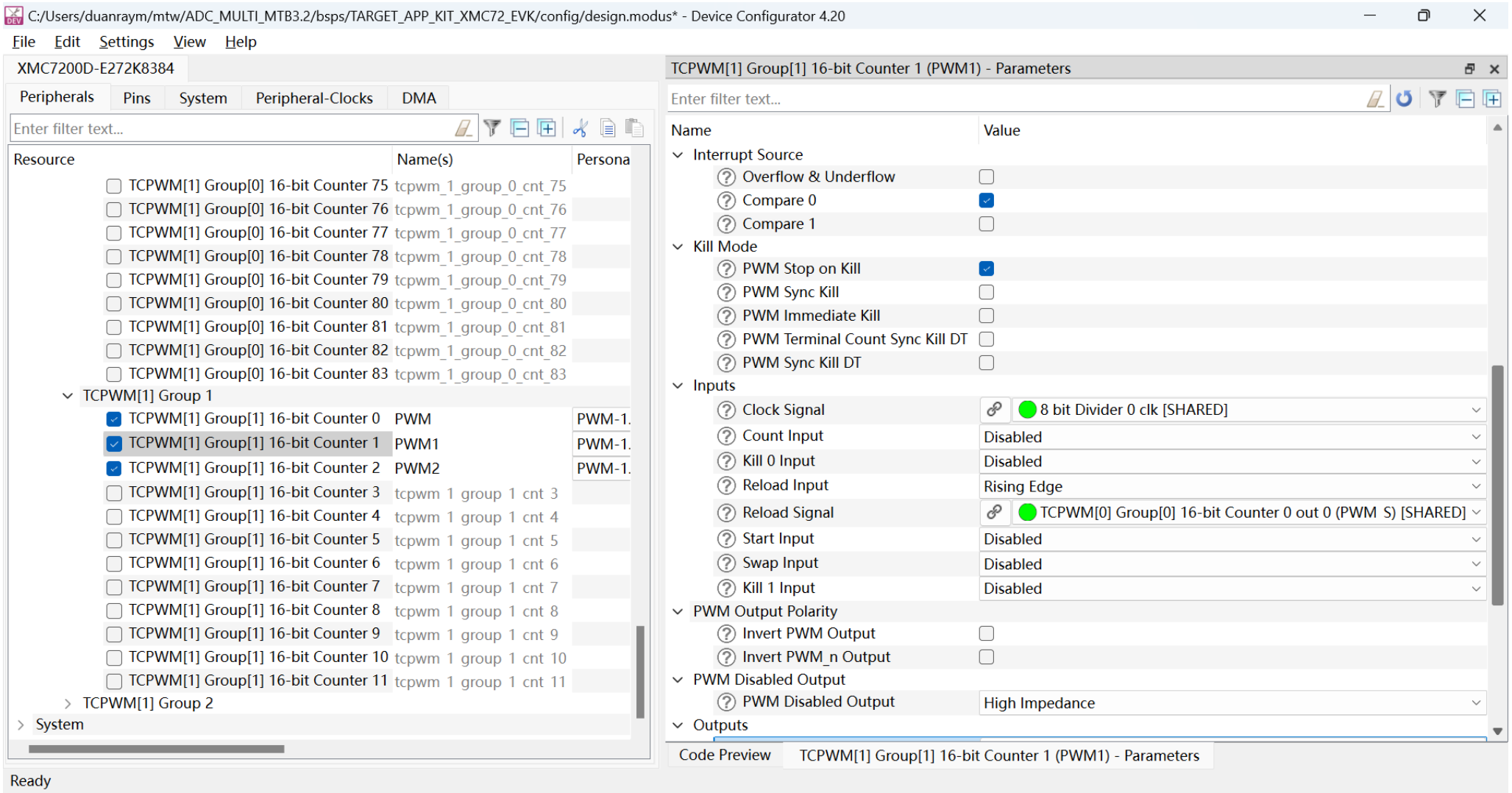
TCPWM[1] Group[1] Counter0配置



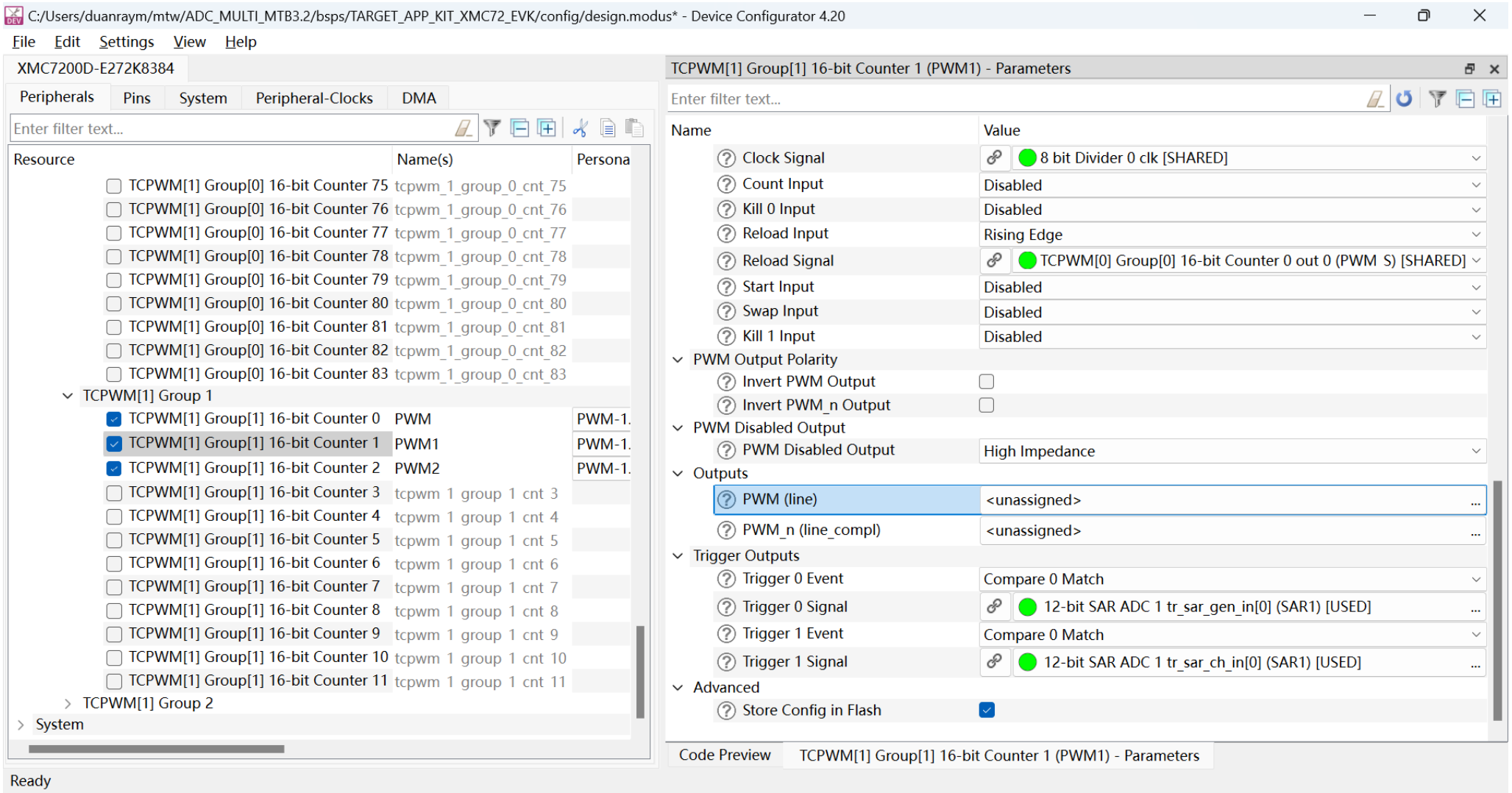
TCPWM[1] Group[1] Counter1 配置



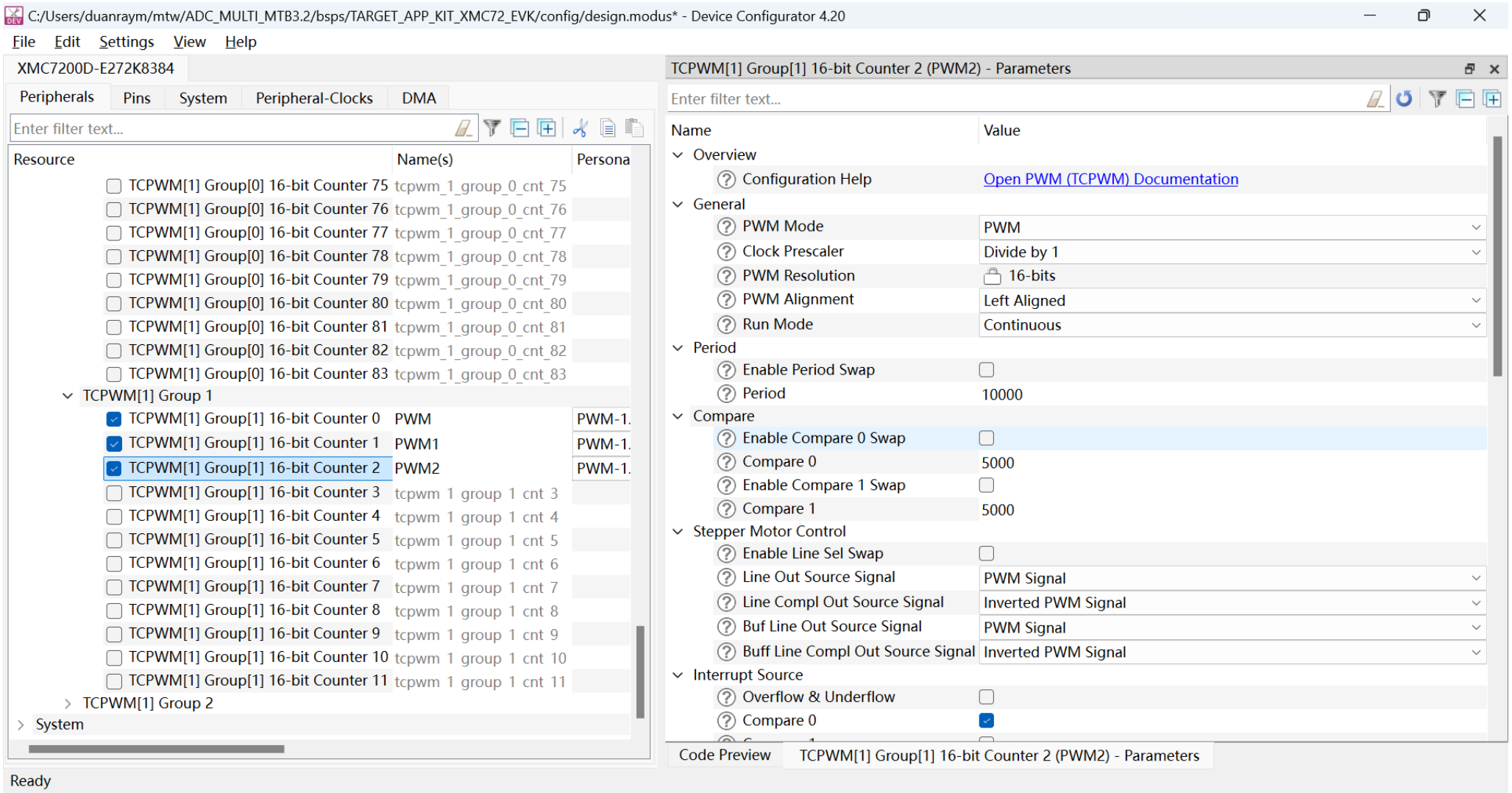
TCPWM[1] Group[1] Counter1 配置



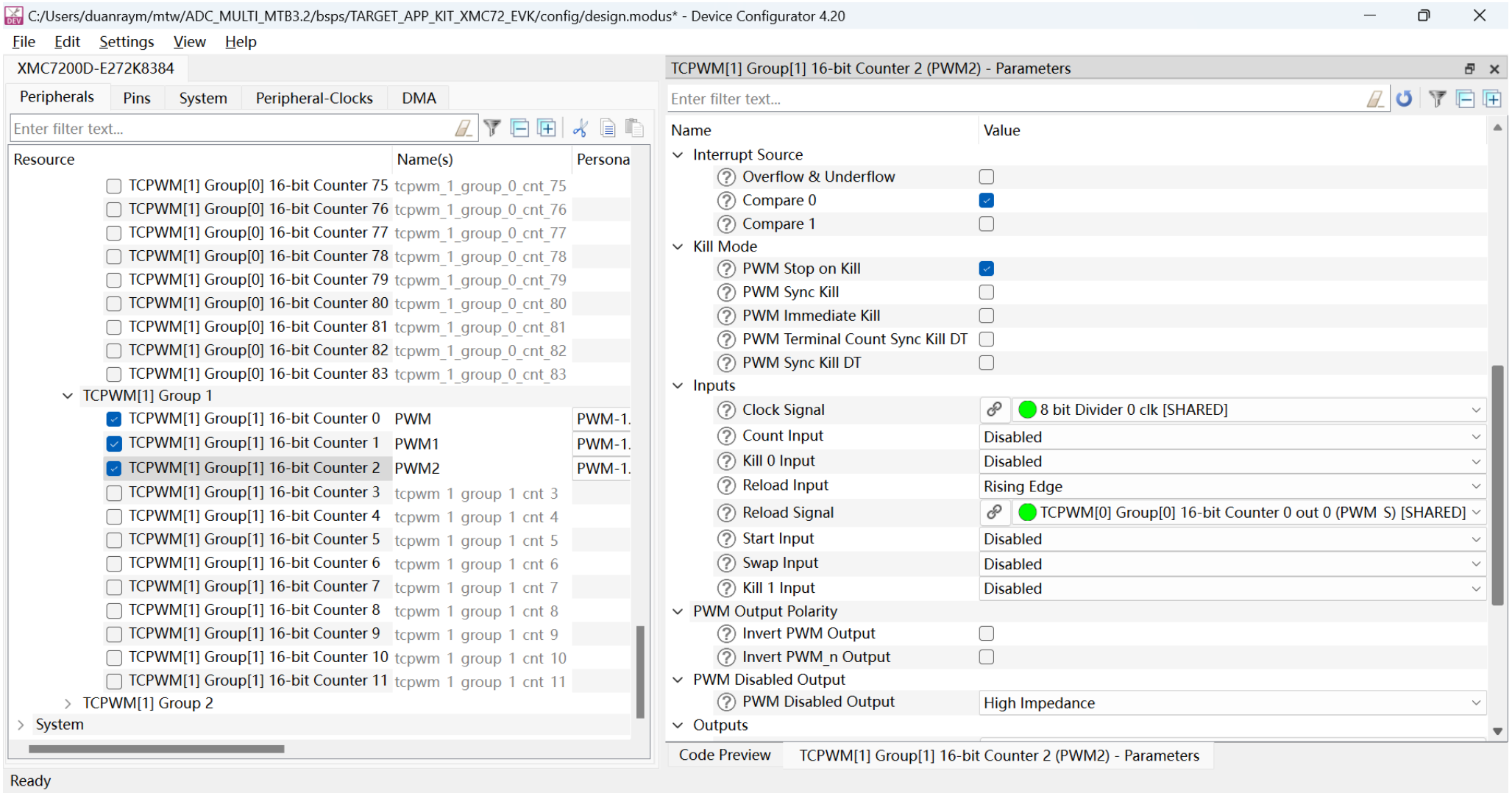
TCPWM[1] Group[1] Counter1 配置



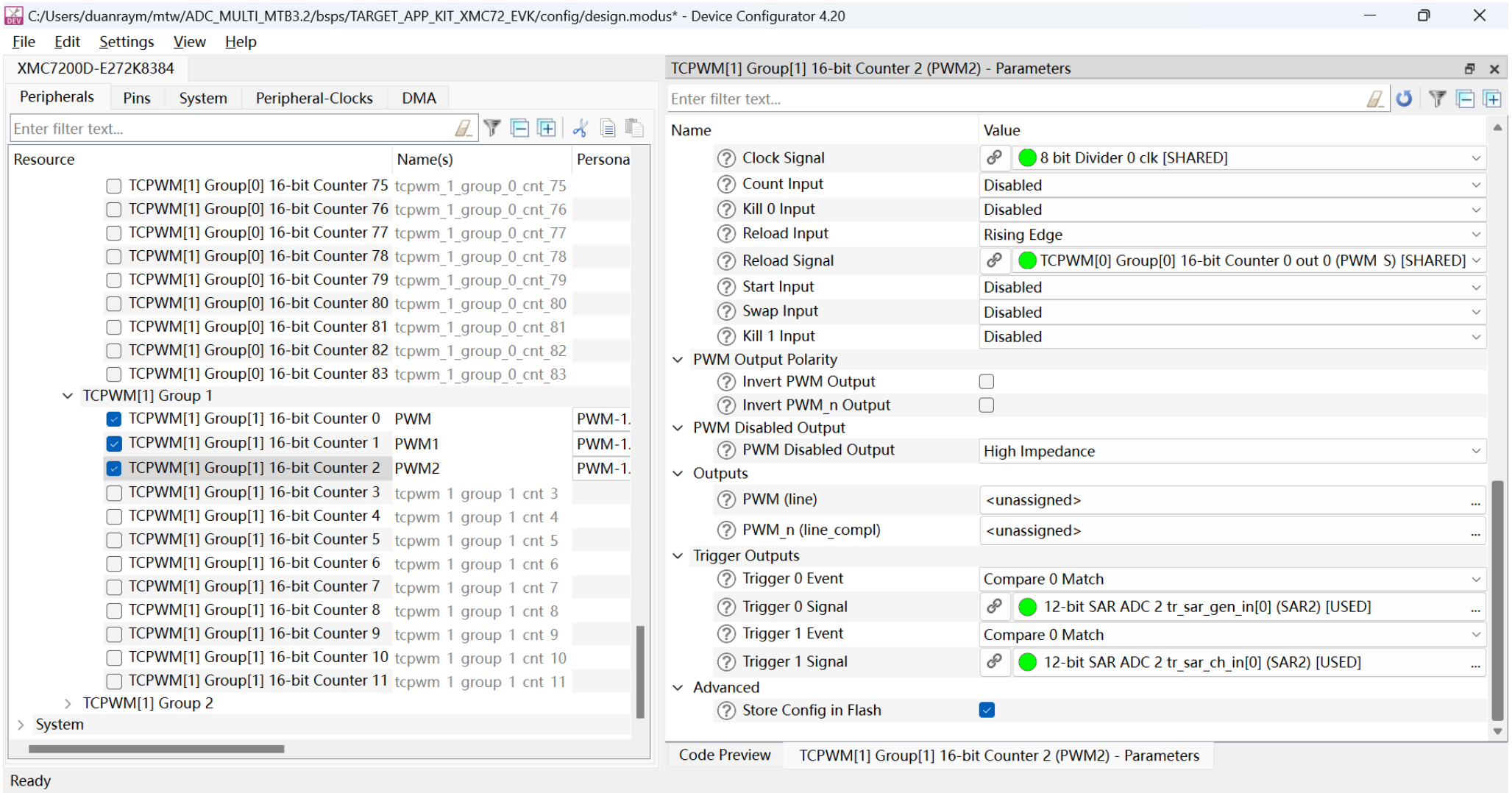
TCPWM[1] Group[1] Counter2 配置



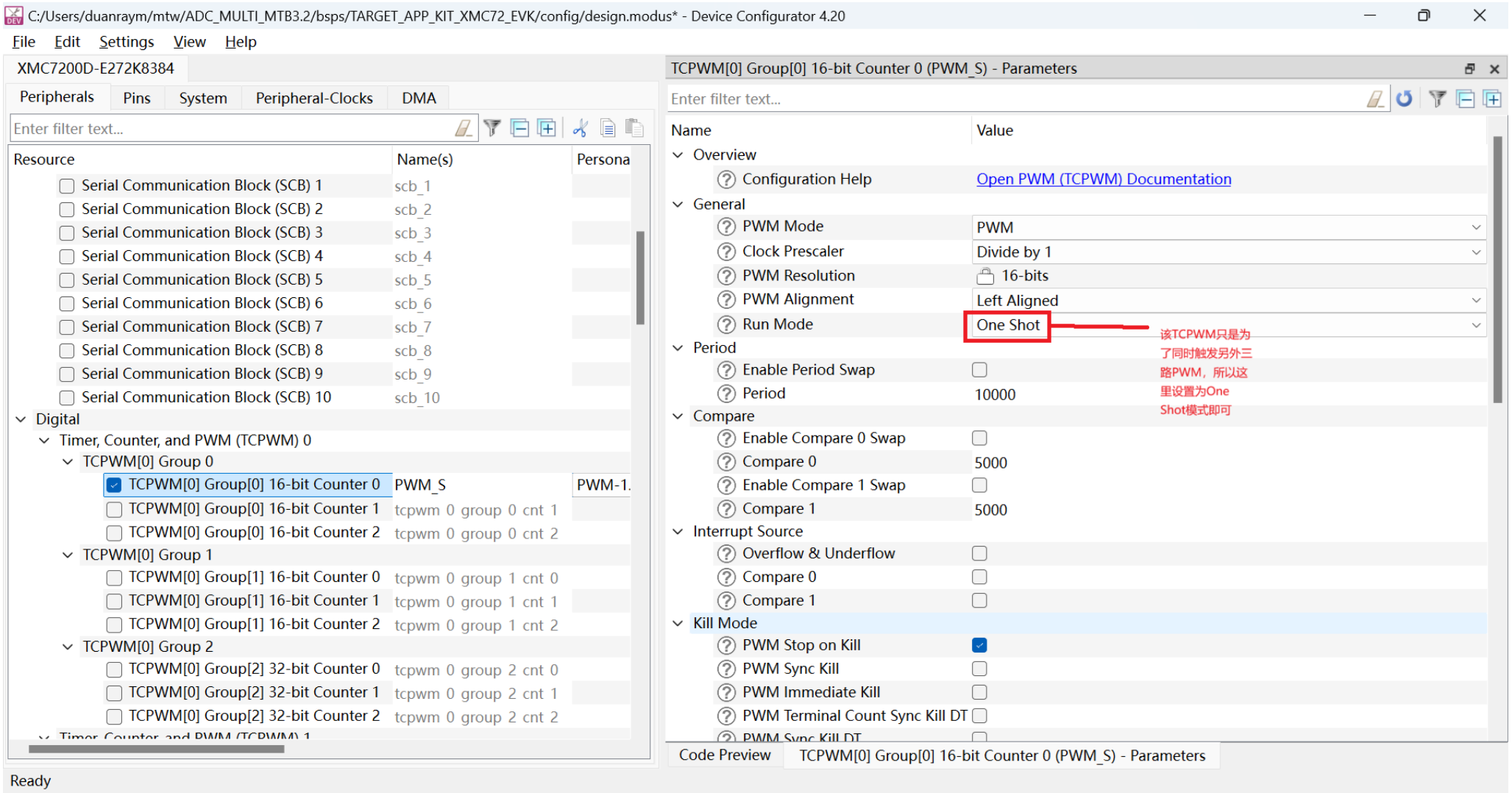
TCPWM[1] Group[1] Counter2 配置



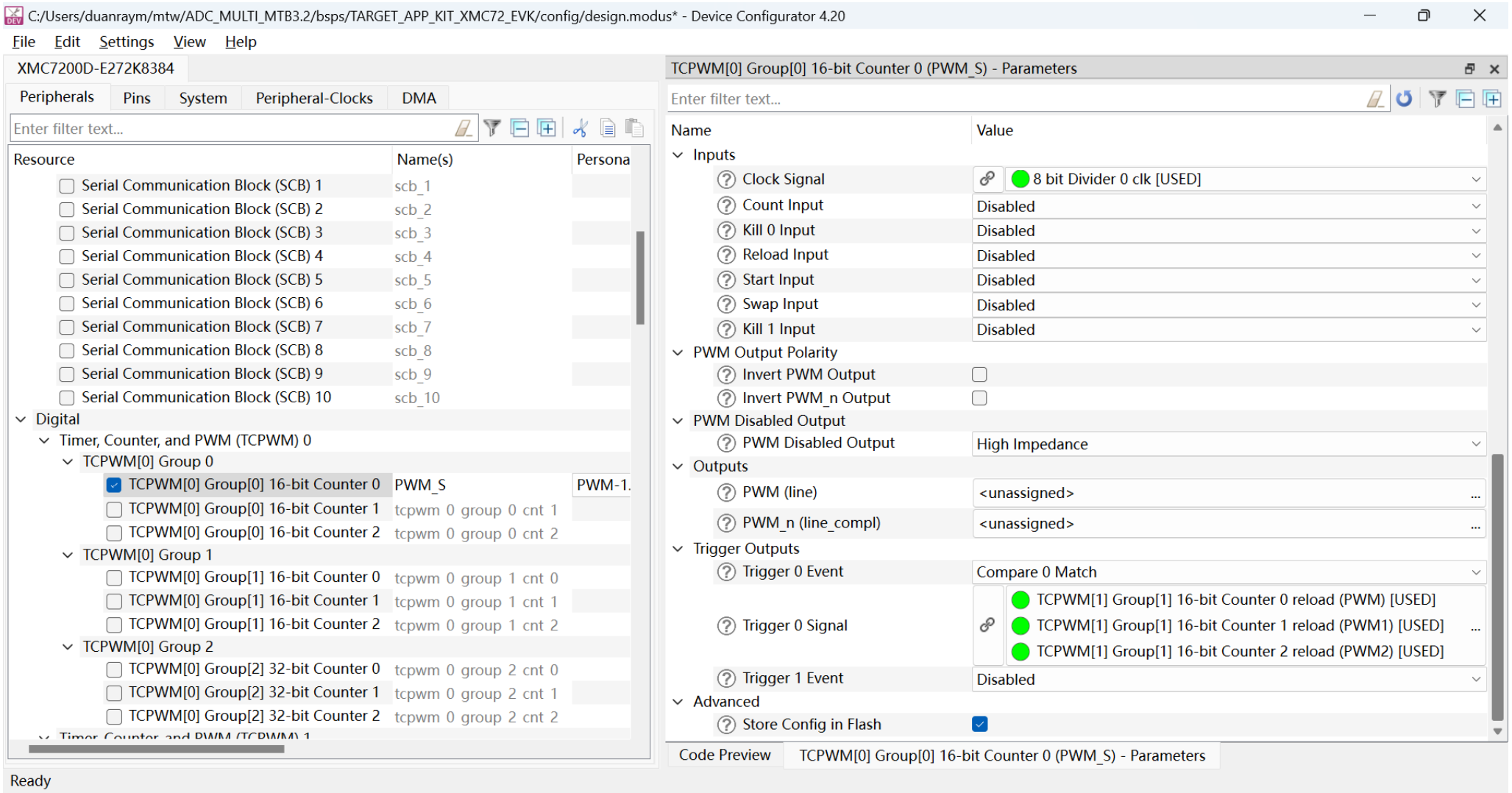
TCPWM[1] Group[1] Counter2 配置



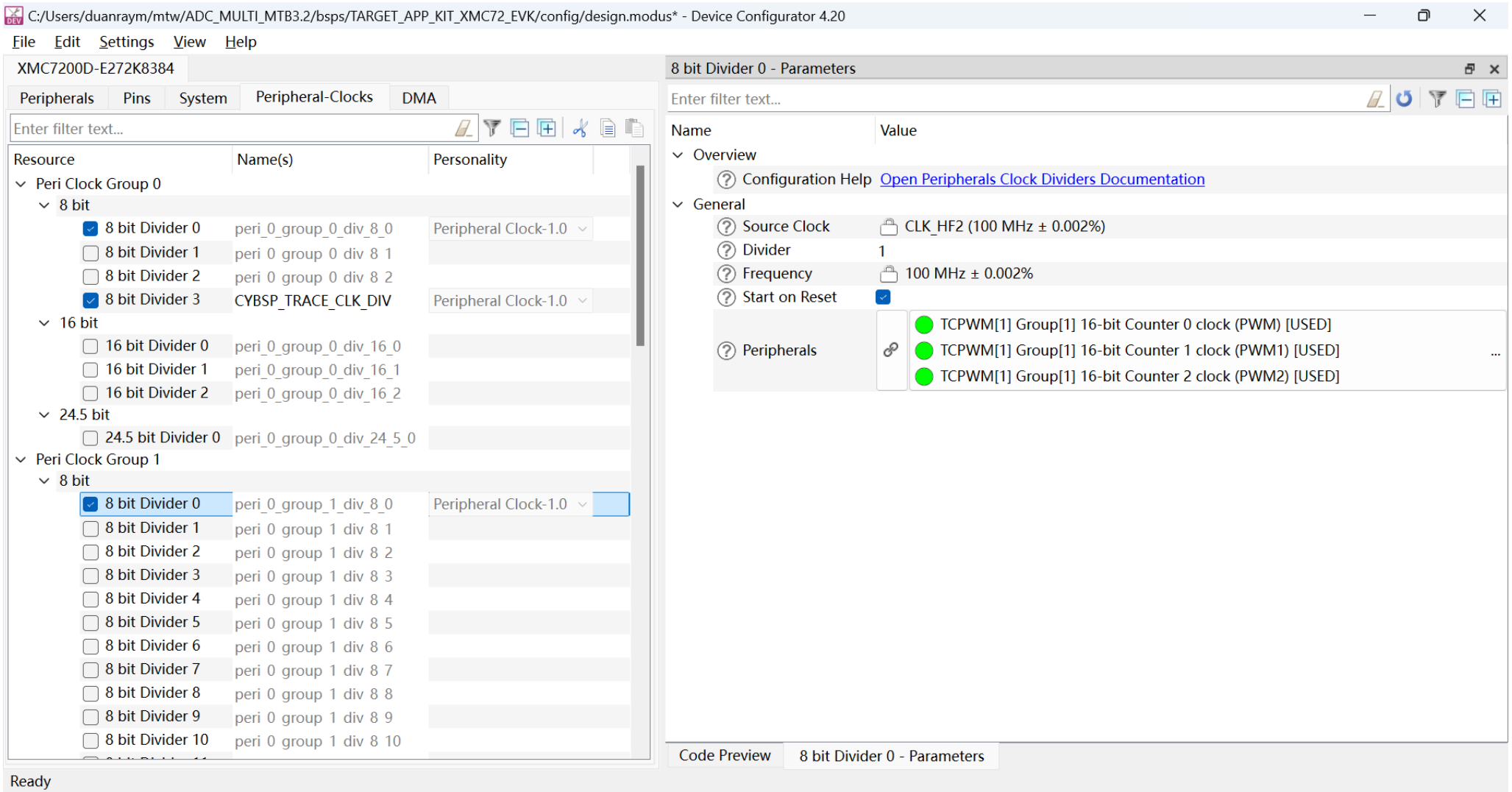
TCPWM[0] Group[0] Counter0 配置



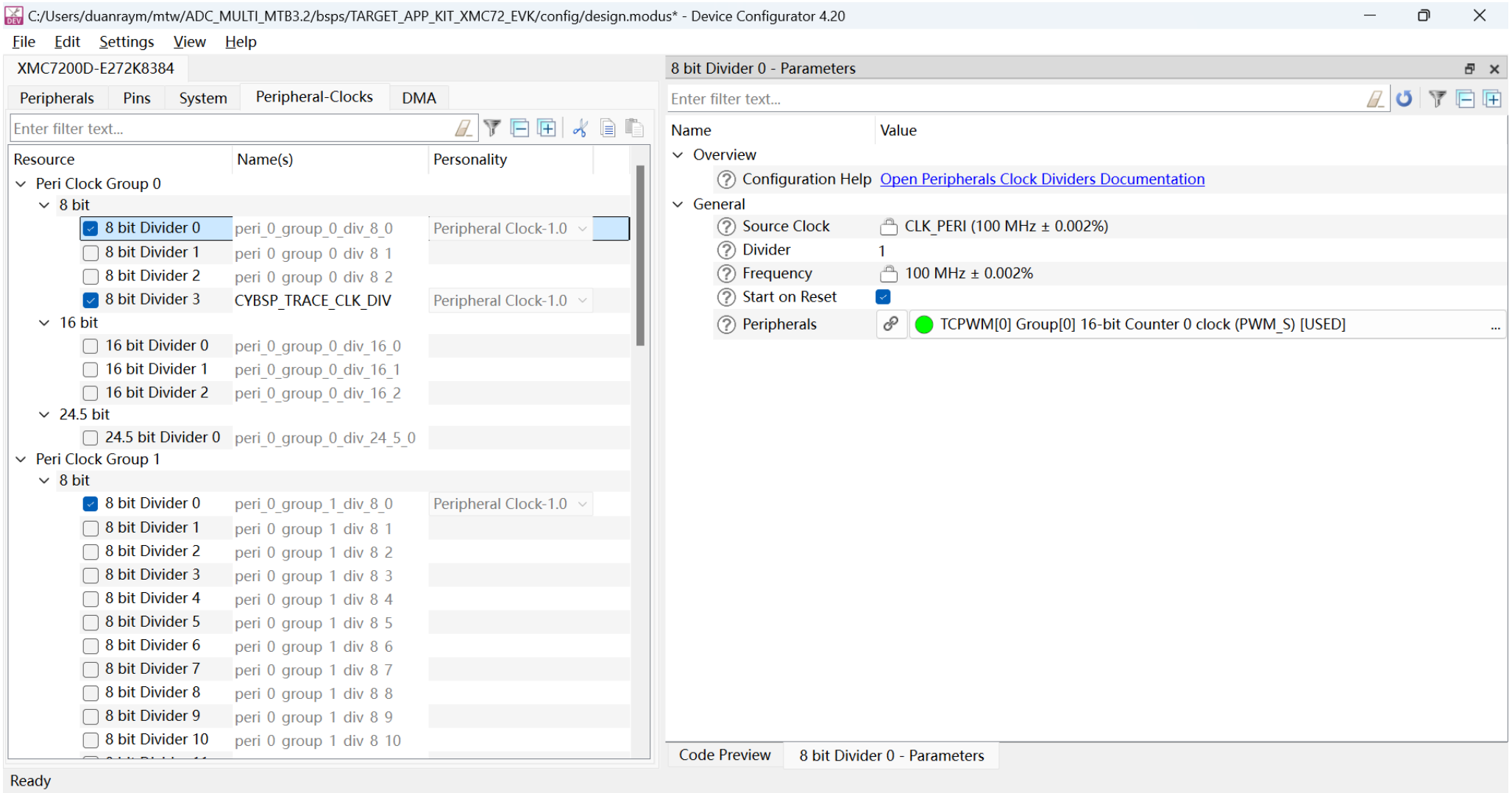
TCPWM[0] Group[0] Counter0 配置



TCPWM时钟配置



TCPWM时钟配置



采样通道PIN脚配置

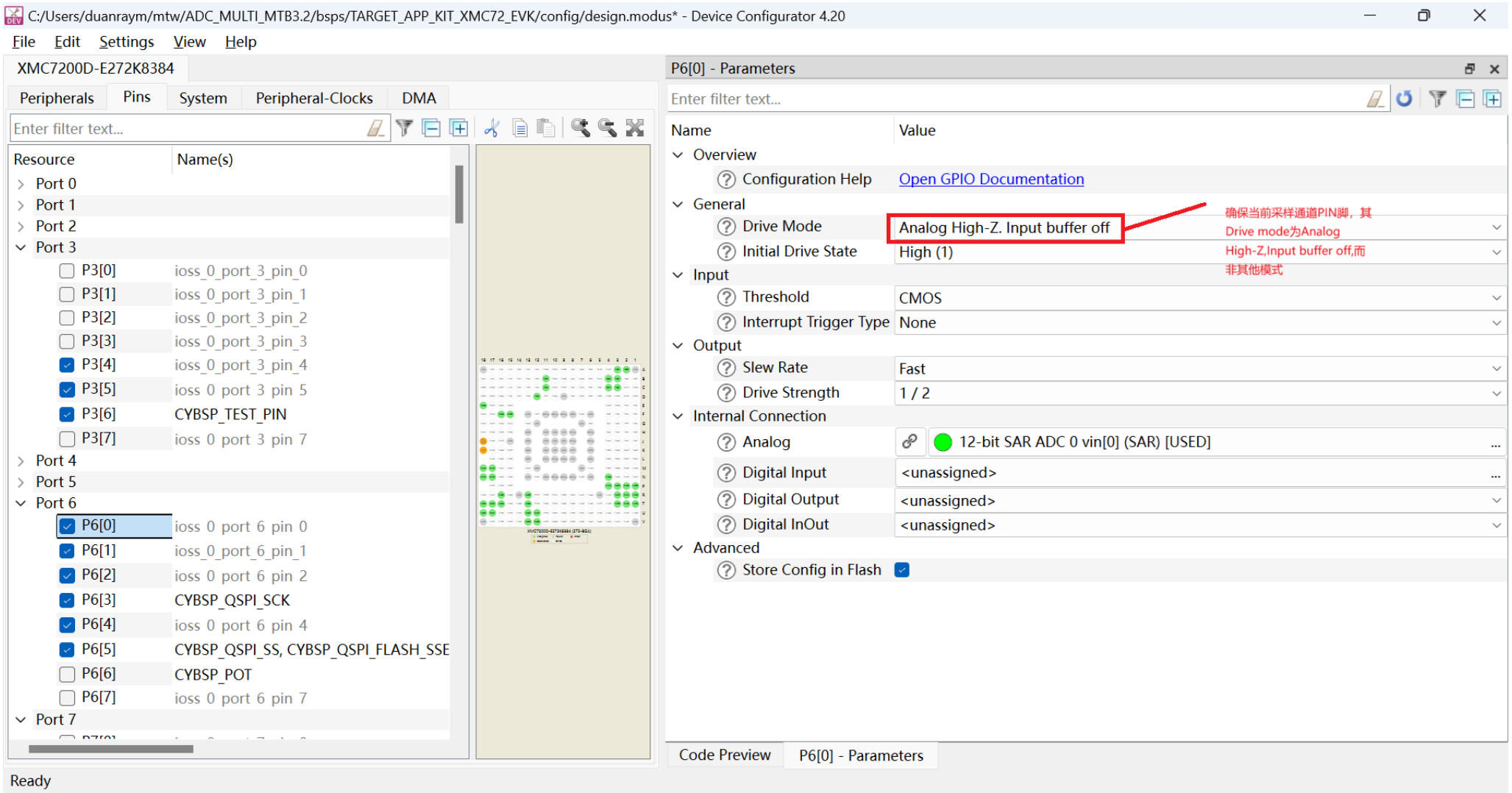


Table of contents

ADC0中断设置以及中断函数实现

```
69 /* SAR0 interrupt configuration structure */
70 const cy_stc_sysint_t SAR_IRQ_cfg = {
71  .intrSrc = ((NvicMux2_IRQn << 16) | SAR_CH5_IRQ),
72  .intrPriority = 7u
73 };
89 static void sar0_interrupt_handler(void)
90 {
91     /* Check if End-Of-Scan trigger has occurred. If yes, set sar0_isr_set flag to true */
92     uint32_t intrSource =
93         Cy_SAR2_Channel_GetInterruptStatusMasked(SAR_HW, 5);
94
95     //if (Cy_SAR2_Channel_GetInterruptStatus(SAR_HW,0u) ==CY_SAR2_INT_GRP_DONE)
96     if(CY_SAR2_INT_GRP_DONE == (intrSource & CY_SAR2_INT_GRP_DONE))
97     {
98         //Cy_GPIO_Inv(CYBSP_TEST_PIN_PORT,CYBSP_TEST_PIN_NUM);
99         sar0_isr_set = true;
100    }
101    /* Clear the interrupts */
102    Cy_SAR2_Channel_ClearInterrupt(SAR_HW, 5u ,CY_SAR2_INT_GRP_DONE);
103
104 }
201     /*Configure and register SAR0 channel interrupts*/
202     Cy_SysInt_Init(&SAR_IRQ_cfg, sar0_interrupt_handler);
203     NVIC_ClearPendingIRQ(NvicMux2_IRQn);
204     NVIC_EnableIRQ((IRQn_Type) NvicMux2_IRQn);
```

ADC0中断配置参数

ADC0中断函数实现

ADC0中断使能

ADC1中断设置以及中断函数实现

```
75  const cy_stc_sysint_t SAR1_IRQ_cfg = {
76   .intrSrc = ((NvicMux3_IRQn << 16) | SAR1_CH5_IRQ),
77   .intrPriority = 7u
78  };
106 static void sar1_interrupt_handler(void)
107 {
108     /* Check if End-Of-Scan trigger has occurred. If yes, set sar0_isr_set flag to true */
109     uint32_t intrSource =
110         Cy_SAR2_Channel_GetInterruptStatusMasked(SAR1_HW, 5);
111
112     //if (Cy_SAR2_Channel_GetInterruptStatus(SAR_HW,0u) ==CY_SAR2_INT_GRP_DONE)
113     if(CY_SAR2_INT_GRP_DONE == (intrSource & CY_SAR2_INT_GRP_DONE))
114     {
115         //Cy_GPIO_Inv(CYBSP_TEST_PIN_PORT,CYBSP_TEST_PIN_NUM);
116         sar1_isr_set = true;
117     }
118     /* Clear the interrupts */
119     Cy_SAR2_Channel_ClearInterrupt(SAR1_HW, 5u ,CY_SAR2_INT_GRP_DONE);
120
121 }
206      /*Configure and register SAR1 channel interrupts*/
207      Cy_SysInt_Init(&SAR1_IRQ_cfg, sar1_interrupt_handler);
208      NVIC_ClearPendingIRQ(NvicMux3_IRQn);
209      NVIC_EnableIRQ((IRQn_Type) NvicMux3_IRQn);
```

ADC1中断配置参数

ADC1中断函数实现

ADC1中断使能

ADC2中断设置以及中断函数实现

```
80  const cy_stc_sysint_t SAR2_IRQ_cfg = {
81   .intrSrc = ((NvicMux4_IRQn << 16) | SAR2_CH5_IRQ),
82   .intrPriority = 7u
83  };
123 static void sar2_interrupt_handler(void)
124 {
125     /* Check if End-Of-Scan trigger has occurred. If yes, set sar0_isr_set flag to true */
126     uint32_t intrSource =
127         Cy_SAR2_Channel_GetInterruptStatusMasked(SAR2_HW, 5);
128
129     //if (Cy_SAR2_Channel_GetInterruptStatus(SAR_HW,0u) ==CY_SAR2_INT_GRP_DONE)
130     if(CY_SAR2_INT_GRP_DONE == (intrSource & CY_SAR2_INT_GRP_DONE))
131     {
132         //Cy_GPIO_Inv(CYBSP_TEST_PIN_PORT,CYBSP_TEST_PIN_NUM);
133         sar2_isr_set = true;
134     }
135     /* Clear the interrupts */
136     Cy_SAR2_Channel_ClearInterrupt(SAR2_HW, 5u ,CY_SAR2_INT_GRP_DONE);
137
138 }
211      /*Configure and register SAR2 channel interrupts*/
212      Cy_SysInt_Init(&SAR2_IRQ_cfg, sar2_interrupt_handler);
213      NVIC_ClearPendingIRQ(NvicMux4_IRQn);
214      NVIC_EnableIRQ((IRQn_Type) NvicMux4_IRQn);
```

ADC2中断配置参数

ADC2中断函数实现

ADC2中断使能

ADC0初始化

```
144     /* Initialize SAR0 */
145     status = Cy_SAR2_Init(SAR_HW, &SAR_config);
146     if (CY_SAR2_SUCCESS != status)
147     {
148     CY_ASSERT(0);
149     }
165     /* Set ePASS MMIO reference buffer mode for bangap voltage */
166     Cy_SAR2_SetReferenceBufferMode(PASS0_EPASS_MMIO, CY_SAR2_REF_BUF_MODE_OFF);
167
168     /*Initialize SAR0 channel 0~5*/
169     Cy_SAR2_Channel_Init(SAR_HW, 0u, &SAR_channel_0_config);
170     Cy_SAR2_Channel_Init(SAR_HW, 1u, &SAR_channel_1_config);
171     Cy_SAR2_Channel_Init(SAR_HW, 2u, &SAR_channel_2_config);
172     Cy_SAR2_Channel_Init(SAR_HW, 3u, &SAR_channel_3_config);
173     Cy_SAR2_Channel_Init(SAR_HW, 4u, &SAR_channel_4_config);
174     Cy_SAR2_Channel_Init(SAR_HW, 5u, &SAR_channel_5_config);
192     /*Set SAR0 channel 5 interrupt mask*/
193     Cy_SAR2_Channel_SetInterruptMask(SAR_HW, 5u,CY_SAR2_INT_GRP_DONE);
```

ADC1初始化

```
151     /* Initialize SAR1 */
152     status = Cy_SAR2_Init(SAR1_HW, &SAR1_config);
153     if (CY_SAR2_SUCCESS != status)
154     {
155     CY_ASSERT(0);
156     }
176     /*Initialize SAR1 channel 0~5*/
177     Cy_SAR2_Channel_Init(SAR1_HW, 0u, &SAR1_channel_0_config);
178     Cy_SAR2_Channel_Init(SAR1_HW, 1u, &SAR1_channel_1_config);
179     Cy_SAR2_Channel_Init(SAR1_HW, 2u, &SAR1_channel_2_config);
180     Cy_SAR2_Channel_Init(SAR1_HW, 3u, &SAR1_channel_3_config);
181     Cy_SAR2_Channel_Init(SAR1_HW, 4u, &SAR1_channel_4_config);
182     Cy_SAR2_Channel_Init(SAR1_HW, 5u, &SAR1_channel_5_config);
195     /*Set SAR1 channel 5 interrupt mask*/
196     Cy_SAR2_Channel_SetInterruptMask(SAR1_HW, 5u,CY_SAR2_INT_GRP_DONE);
```

ADC2初始化

```
158     /* Initialize SAR2 */
159     status = Cy_SAR2_Init(SAR2_HW, &SAR2_config);
160     if (CY_SAR2_SUCCESS != status)
161     {
162     CY_ASSERT(0);
163     }
184     /*Initialize SAR2 channel 0~5*/
185     Cy_SAR2_Channel_Init(SAR2_HW, 0u, &SAR2_channel_0_config);
186     Cy_SAR2_Channel_Init(SAR2_HW, 1u, &SAR2_channel_1_config);
187     Cy_SAR2_Channel_Init(SAR2_HW, 2u, &SAR2_channel_2_config);
188     Cy_SAR2_Channel_Init(SAR2_HW, 3u, &SAR2_channel_3_config);
189     Cy_SAR2_Channel_Init(SAR2_HW, 4u, &SAR2_channel_4_config);
190     Cy_SAR2_Channel_Init(SAR2_HW, 5u, &SAR2_channel_5_config);
198     /*Set SAR2 channel 5 interrupt mask*/
199     Cy_SAR2_Channel_SetInterruptMask(SAR2_HW, 5u,CY_SAR2_INT_GRP_DONE);
```

PWM，PWM1,PWM2初始化

```
293     /* Initialize PWM using the config structure generated using device configurator*/
294     if (CY_TCPWM_SUCCESS != Cy_TCPWM_PWM_Init(PWM_HW, PWM_NUM, &PWM_config))
295     {
296         CY_ASSERT(0);
297     }
298     /* Enable the initialized PWM */
299     Cy_TCPWM_PWM_Enable(PWM_HW, PWM_NUM);
303     /* Initialize PWM1 using the config structure generated using device configurator*/
304     if (CY_TCPWM_SUCCESS != Cy_TCPWM_PWM_Init(PWM1_HW, PWM1_NUM, &PWM1_config))
305     {
306         CY_ASSERT(0);
307     }
308     /* Enable the initialized PWM */
309     Cy TCPWM PWM Enable(PWM1 HW, PWM1 NUM);
313     /* Initialize PWM2 using the config structure generated using device configurator*/
314     if (CY_TCPWM_SUCCESS != Cy_TCPWM_PWM_Init(PWM2_HW, PWM2_NUM, &PWM2_config))
315     {
316         CY_ASSERT(0);
317     }
318     /* Enable the initialized PWM */
319     Cy_TCPWM_PWM_Enable(PWM2_HW, PWM2_NUM);
```

PWM_S初始化

```
324     /* Initialize PWM1 using the config structure generated using device configurator*/
325     if (CY_TCPWM_SUCCESS != Cy_TCPWM_PWM_Init(PWM_S_HW, PWM_S_NUM, &PWM_S_config))
326     {
327         CY_ASSERT(0);
328     }
329     /* Enable the initialized PWM */
330     Cy_TCPWM_PWM_Enable(PWM_S_HW, PWM_S_NUM);
331     /* Then start the TCPWM_S to synchronous PWM_U, PWM_V and PWM_W */
332     Cy_TCPWM_TriggerReloadOrIndex_Single(PWM_S_HW, PWM_S_NUM);
```

ADC0 6通道采样数据读取

```
337         if(sar0_isr_set)
338          {
339             sar0Result0 = Cy_SAR2_Channel_GetResult(SAR_HW, 0u, NULL);
340             printf("sar0Result0 = %d \r\n",sar0Result0 );
341             sar0Result1 = Cy_SAR2_Channel_GetResult(SAR_HW, 1u, NULL);
342             printf("sar0Result1 = %d \r\n",sar0Result1 );
343             sar0Result2 = Cy_SAR2_Channel_GetResult(SAR_HW, 2u, NULL);
344             printf("sar0Result2 = %d \r\n",sar0Result2 );
345             sar0Result3 = Cy_SAR2_Channel_GetResult(SAR_HW, 3u, NULL);
346             printf("sar0Result3 = %d \r\n",sar0Result3 );
347             sar0Result4 = Cy_SAR2_Channel_GetResult(SAR_HW, 4u, NULL);
348             printf("sar0Result4 = %d \r\n",sar0Result4 );
349             sar0Result5 = Cy_SAR2_Channel_GetResult(SAR_HW, 5u, NULL);
350             printf("sar0Result5 = %d \r\n",sar0Result5 );
351             sar0_isr_set = false;
352          }
```

ADC1 6通道采样数据读取

```
354         if(sar1_isr_set)
355          {
356             sar1Result0 = Cy_SAR2_Channel_GetResult(SAR1_HW, 0u, NULL);
357             printf("sar1Result0 = %d \r\n",sar1Result0 );
358             sar1Result1 = Cy_SAR2_Channel_GetResult(SAR1_HW, 1u, NULL);
359             printf("sar1Result1 = %d \r\n",sar1Result1 );
360             sar1Result2 = Cy_SAR2_Channel_GetResult(SAR1_HW, 2u, NULL);
361             printf("sar1Result2 = %d \r\n",sar1Result2 );
362             sar1Result3 = Cy_SAR2_Channel_GetResult(SAR1_HW, 3u, NULL);
363             printf("sar1Result3 = %d \r\n",sar1Result3 );
364             sar1Result4 = Cy_SAR2_Channel_GetResult(SAR1_HW, 4u, NULL);
365             printf("sar1Result4 = %d \r\n",sar1Result4 );
366             sar1Result5 = Cy_SAR2_Channel_GetResult(SAR1_HW, 5u, NULL);
367             printf("sar1Result5 = %d \r\n",sar1Result5 );
368             sar1_isr_set = false;
369          }
```

ADC2 6通道采样数据读取

```
371         if(sar2_isr_set)
372          {
373             sar2Result0 = Cy_SAR2_Channel_GetResult(SAR2_HW, 0u, NULL);
374             printf("sar2Result0 = %d \r\n",sar2Result0 );
375             sar2Result1 = Cy_SAR2_Channel_GetResult(SAR2_HW, 1u, NULL);
376             printf("sar2Result1 = %d \r\n",sar2Result1 );
377             sar2Result2 = Cy_SAR2_Channel_GetResult(SAR2_HW, 2u, NULL);
378             printf("sar2Result2 = %d \r\n",sar2Result2 );
379             sar2Result3 = Cy_SAR2_Channel_GetResult(SAR2_HW, 3u, NULL);
380             printf("sar2Result3 = %d \r\n",sar2Result3 );
381             sar2Result4 = Cy_SAR2_Channel_GetResult(SAR2_HW, 4u, NULL);
382             printf("sar2Result4 = %d \r\n",sar2Result4 );
383             sar2Result5 = Cy_SAR2_Channel_GetResult(SAR2_HW, 5u, NULL);
384             printf("sar2Result5 = %d \r\n",sar2Result5 );
385             sar2_isr_set = false;
386          }
```

Table of contents

问题总结

- 三组SAR ADC unit需要分别用对应的TCPWM来进行定时触发：
  - TCPWM[1] Group[1] Counter0用来触发ADC SAR0
  - TCPWM[1] Group[1] Counter2用来触发ADC SAR1
  - TCPWM[1] Group[1] Counter2用来触发ADC SAR2

- 在采样过程中，为了避免同一采样通道上模拟信号的变化(比如从3.3V突然变成0V)，而造成采样结果的突变（由于ADC内部采样电容的存在，会导致当前采样到的数值不会马上变为0，而是持续一段时间后才归零），这时可以调整SAR中的precondition Time，并Precondition mode为Discharge to VREFL。这样作的目的是，在下次采样前，会先对ADC内部电容进行放电处理，从而避免了采样数据偏差的问题。

- 若每个ADC unit存在多个通道的使用，在设置完最后一个通道后，需要在Device configuration上勾选“Group End”

- 在对应的TCPWM中，通过设置Trigger Outputs来完成TCPWM信号与ADC SAR间的信号关联，以及触发条件。

- 若需要实现多路TCPWM同时启动，可以使用TCPWM中的reload事件，通过设置另一路one shot模式TCPWM作为触发信号，来映射到多路TCPWM的reload事件上。

- 需要确保采样通道对应的IO管脚属性为Analog High-Z,Input buffer off,否则会出现采样结果异常的情况。


Infineon