SDK应用_FlexCAN模块配置及应用（二）

1. 前言

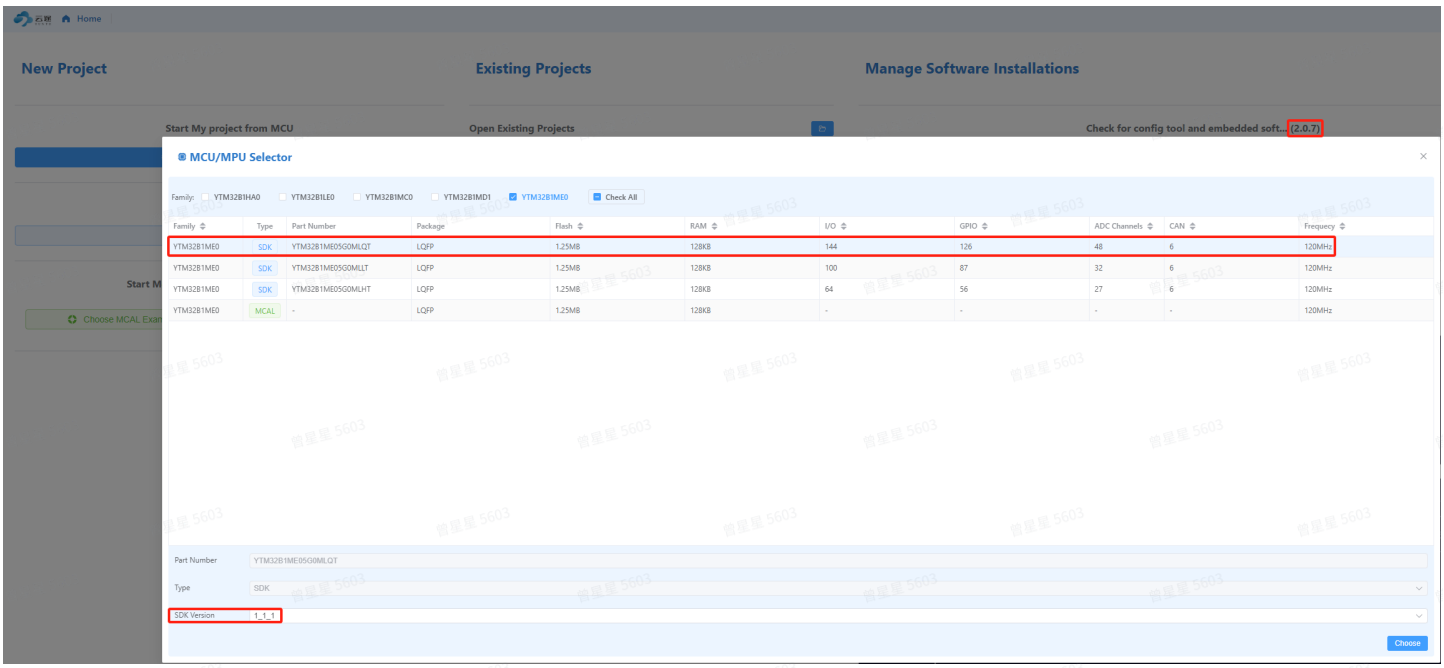
当有以下需求时：

1. Bootloader需要支持队列刷写功能，来提高App程序刷写效率，以满足队列诊断的功能；
2. 需要固定接收一定数量的报文，而报文ID随机，不能通过设置接收掩码实现这些报文的接收；
3. 需要增强CAN通信的安全性，防止数据接收过程中被高优先级的进程打断造成丢帧。

面对这些情况，使用FIFO模式就会更加方便。针对这样的应用场景，本文介绍了基于YT Config Tool的FlexCAN模块legacy FIFO模式的配置步骤和使用方法（L系列和M系列的legacy FIFO配置配方式大致相同，二者之间的区别请参考SDK应用_FlexCAN模块配置及应用（一））。

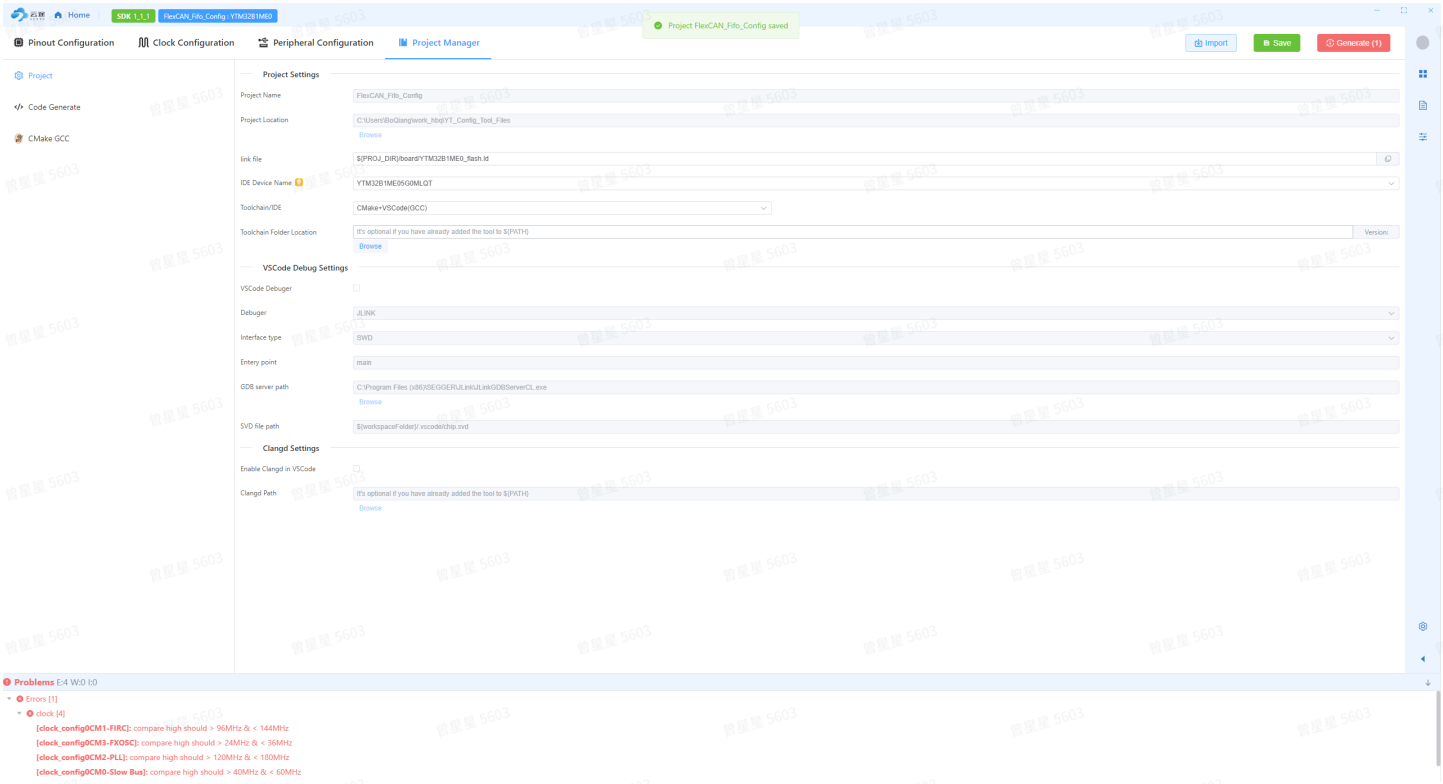
2. 芯片型号及工具版本信息

- MCU型号：YTM32B1ME0MLQT
- 配置工具版本：2.0.7
- SDK版本：1_1_1

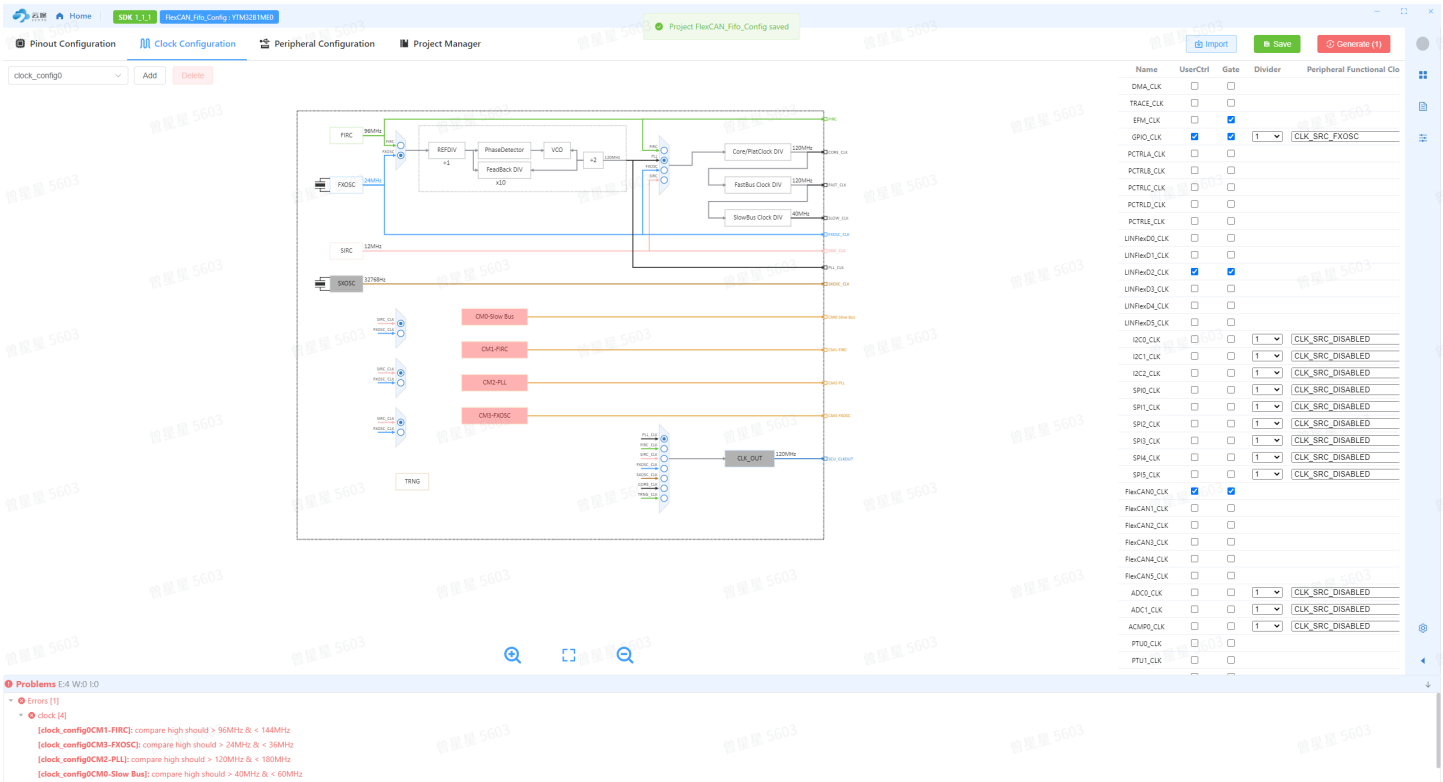


3. 配置步骤

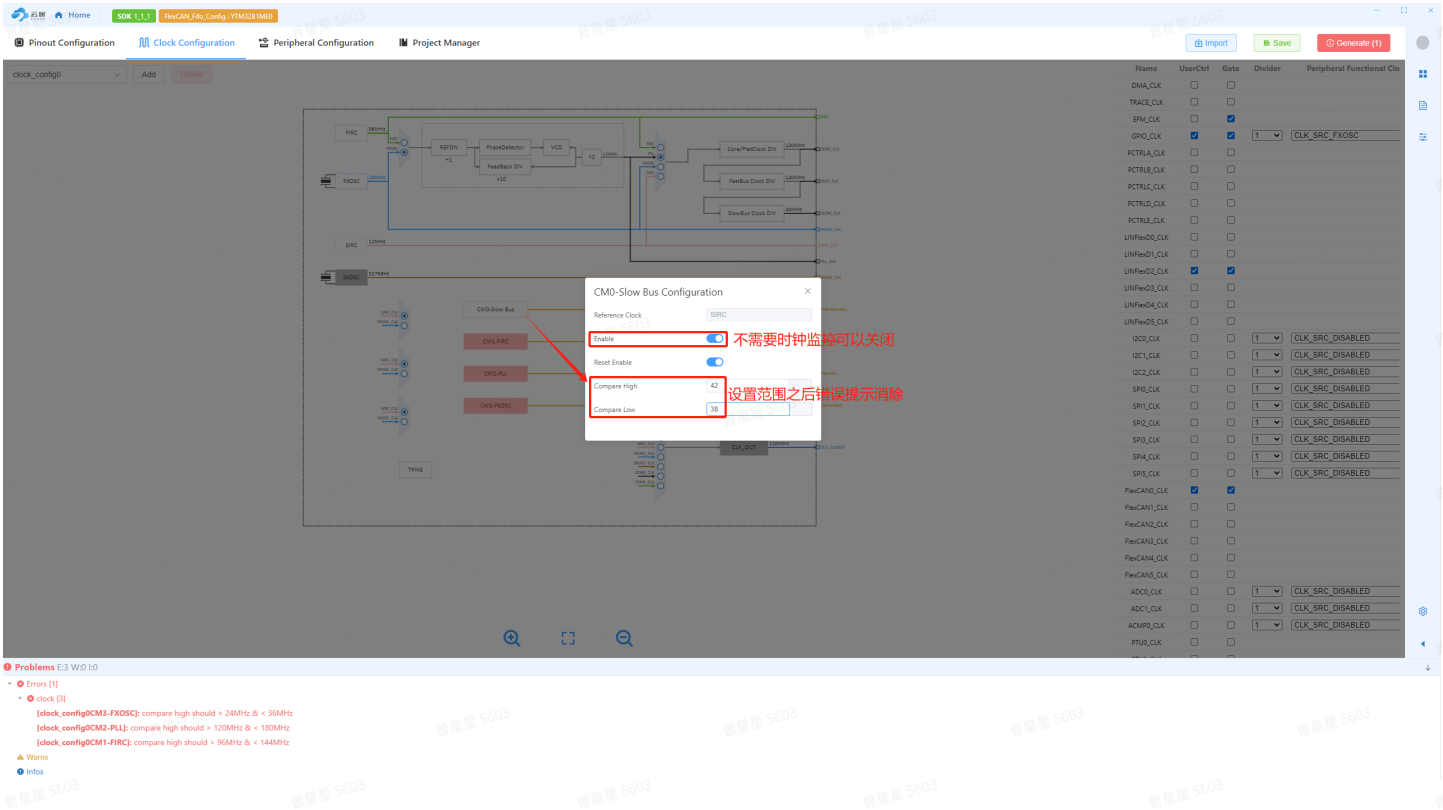
3.1 建立工程



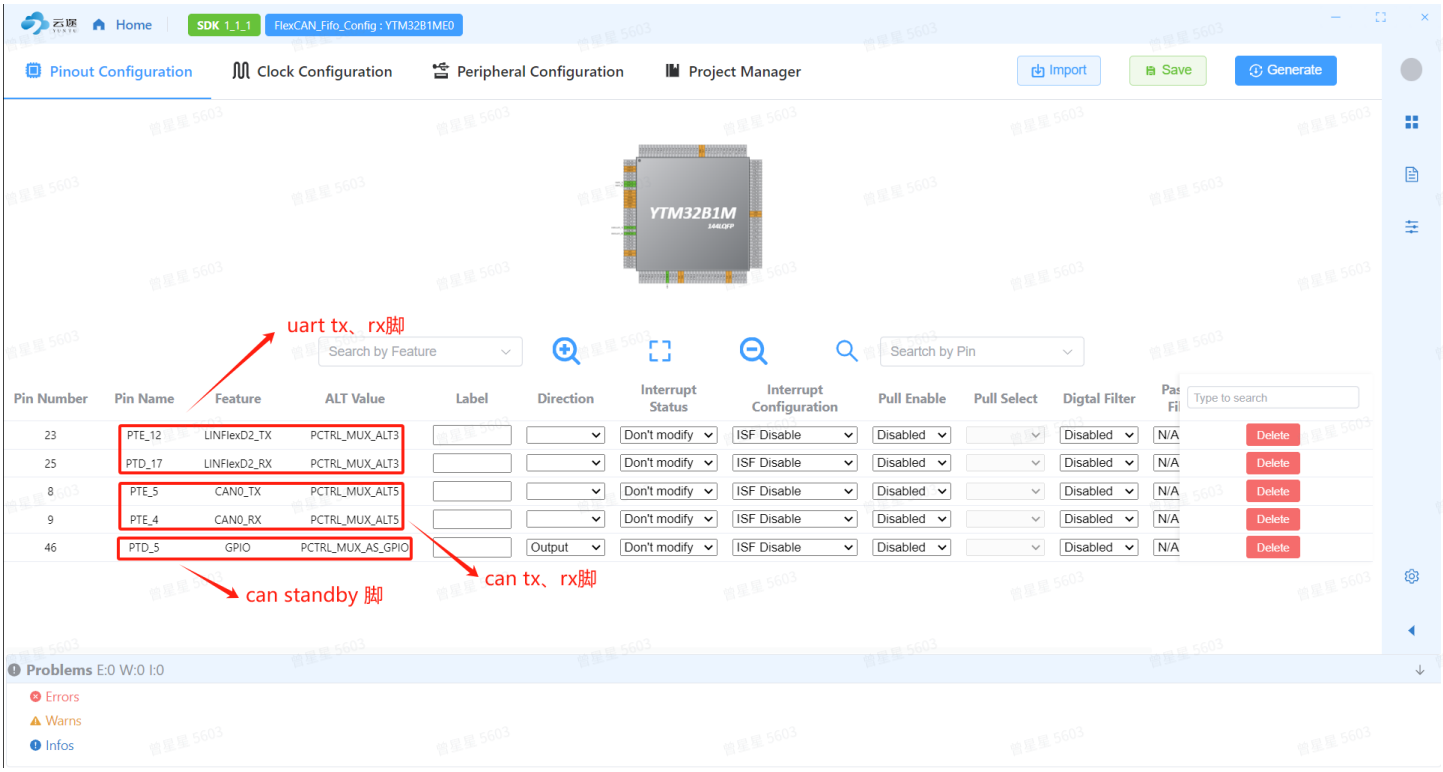
3.2 时钟配置



左下方时钟配置有提示错误，这是因为监控时钟没有设置监控范围，旧版本默认不打开时钟监控，可以忽略。如果不需要时钟监控，可以关闭这项功能；如果使用了这项配置，需要根据实际需要设置时钟监控范围。

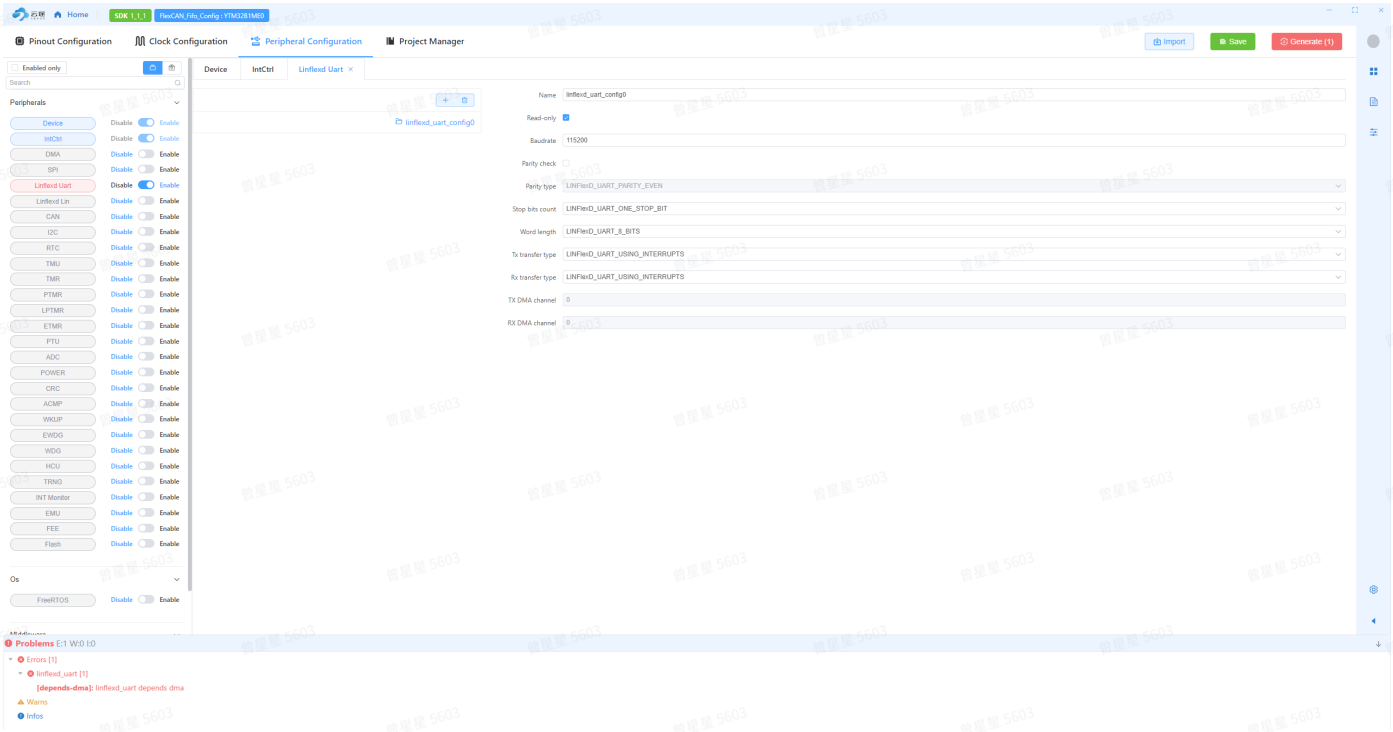


3.3 引脚配置

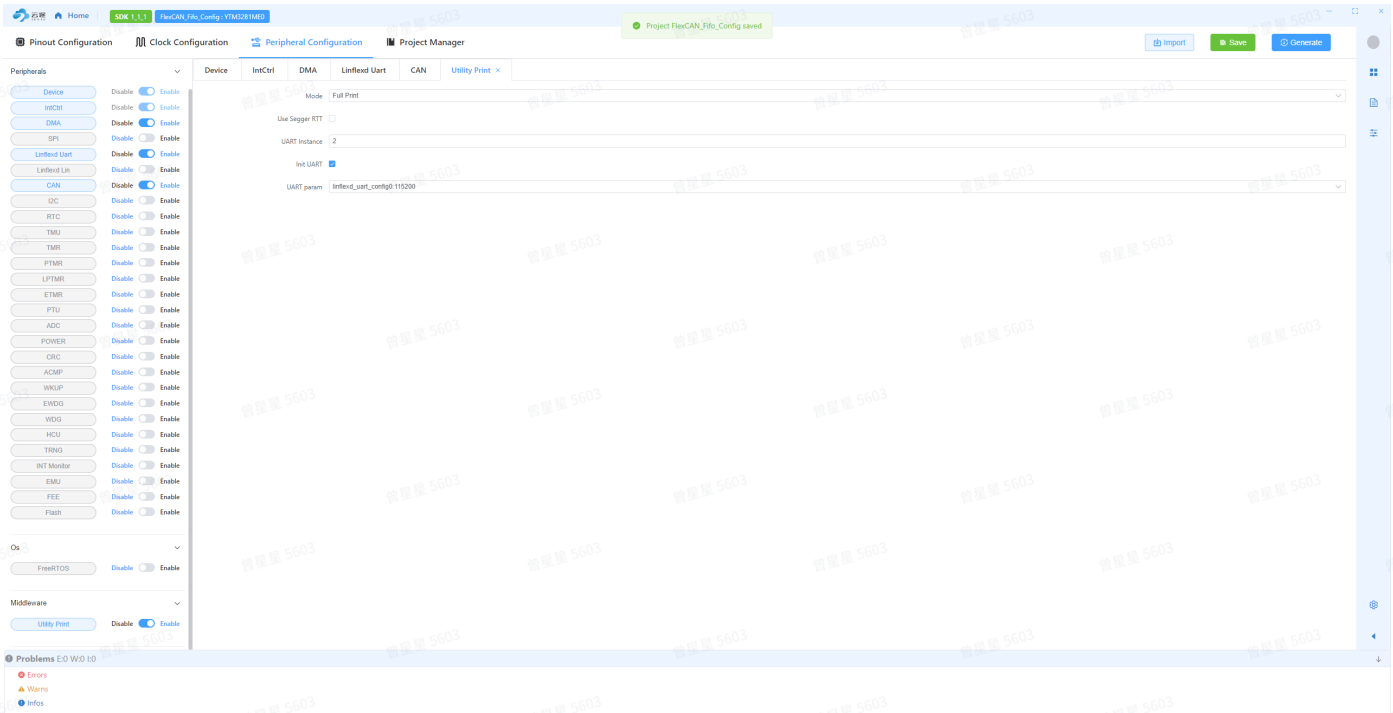


3.4 模块功能配置

3.4.1 UART功能配置

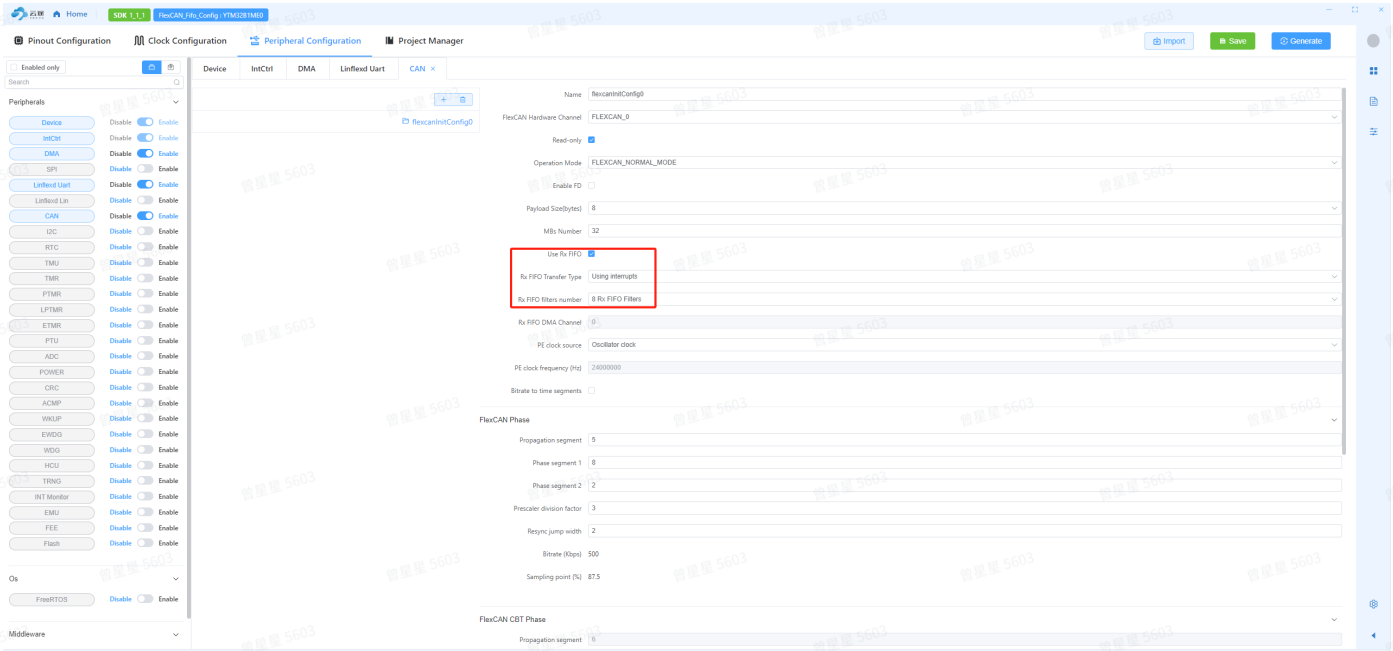


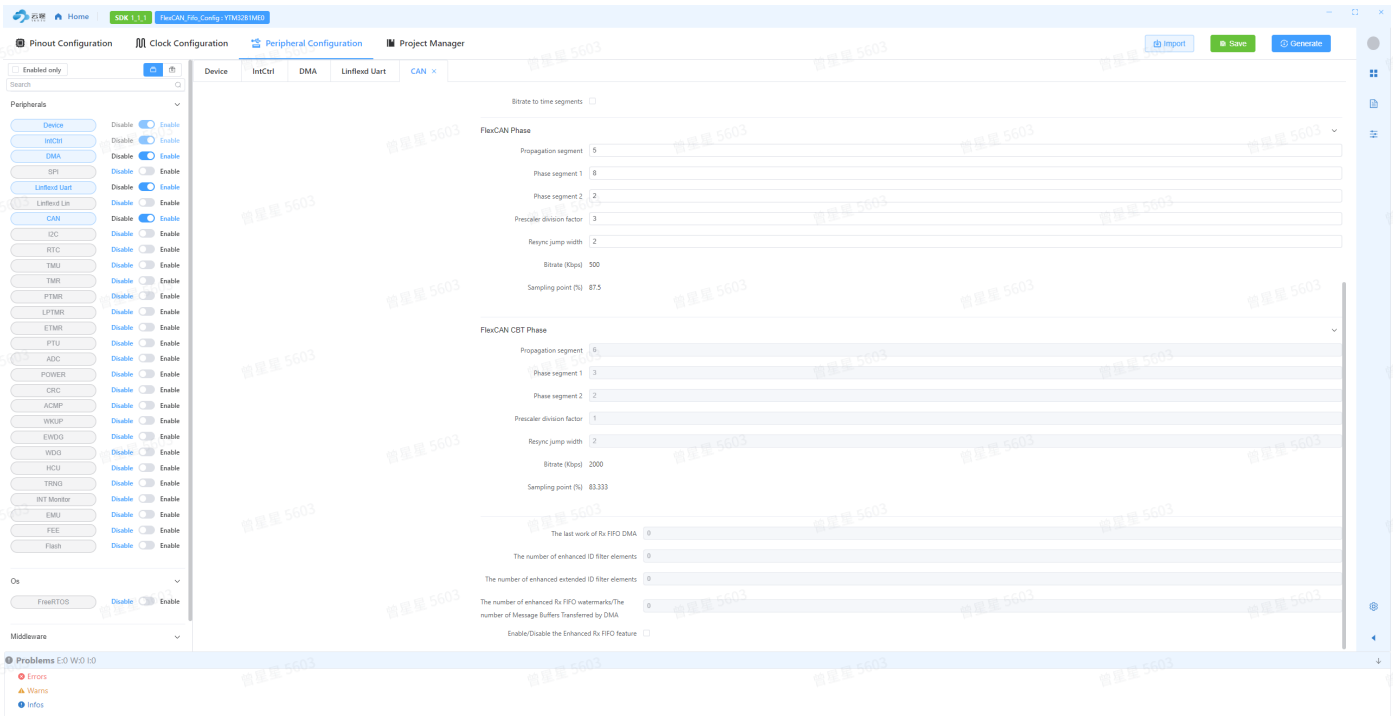
3.4.2 添加打印函数



3.4.3 CAN功能配置

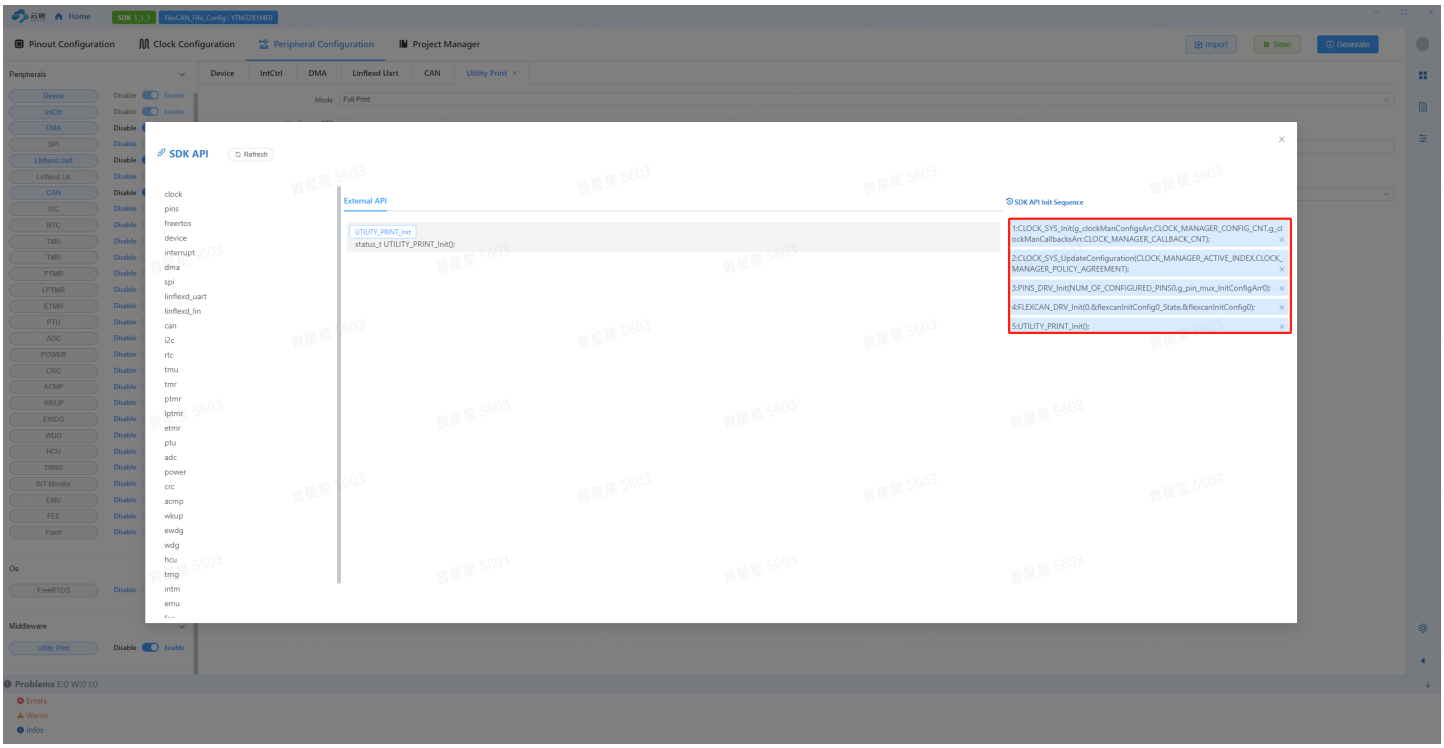
使能FIFO模式，FIFO传输类型设置为中断方式，滤波器数量为8，其他参数保持默认配置。



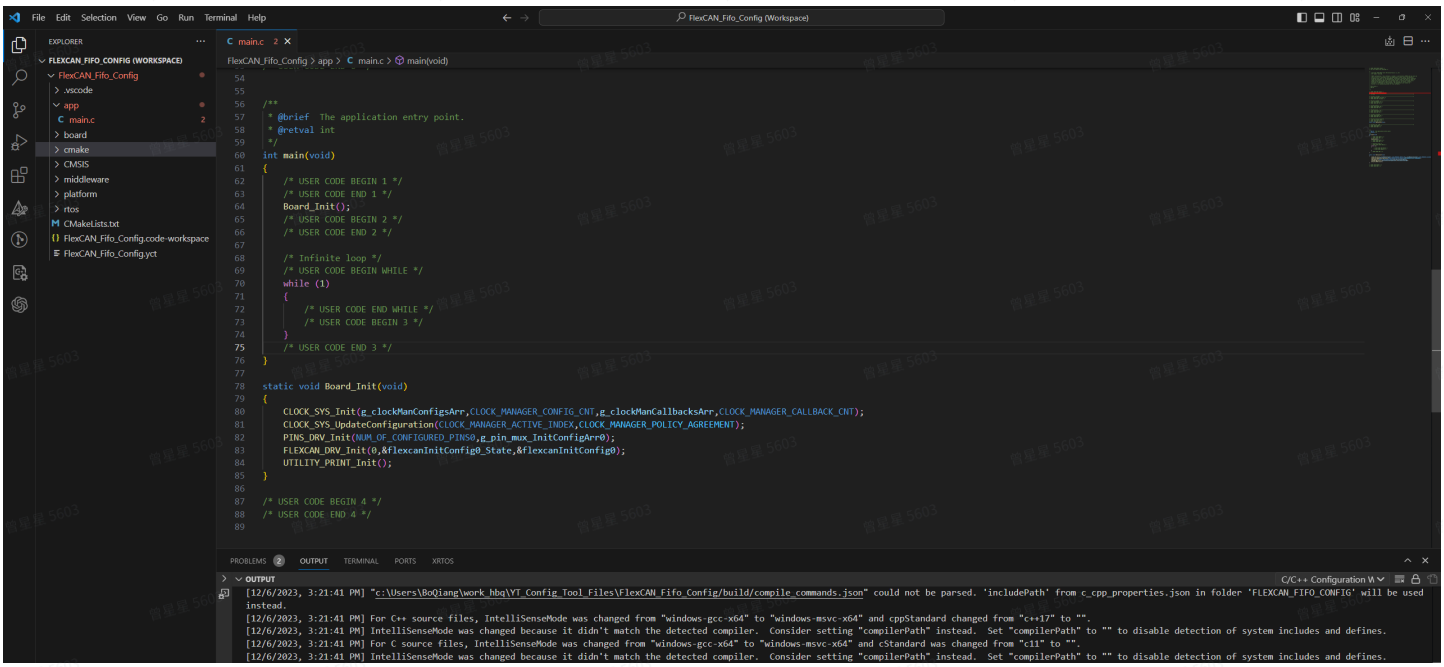


其他基础配置项的介绍请参考“SDK应用_FlexCAN模块配置及应用（一）”。

3.5 添加API



4. 生成工程展示



5. 应用实例

本例使用了FlexCAN的legacy FIFO功能，配置了8个滤波ID，包括4个标准帧ID，4个扩展帧ID。主程序中利用发送邮箱（编号为36）发送ID为0x310的标准帧，并设置FIFO区域的11位标准帧掩码为0x7F0，目标接收标准帧报文ID为0x110~0x11F，扩展帧报文ID为0x10FF1230~0x10FF1233（标准帧掩码对扩展帧无影响）。

5.1 通用配置

YT Config Tool不支持直接配置CAN FIFO模式的过滤器，需要用户手动添加，添加方式如下：

a. 设置FIFO滤波器数组。由于前面功能配置中设置了filter数量为8，因此这里需要定义一个长度为8的滤波器数组。

```
flexcan_id_table_t can_filter_table[8] =
{
    {
        .isRemoteFrame = false,
        .isExtendedFrame = false,
        .id = 0x110,
    },
    {
        .isRemoteFrame = false,
        .isExtendedFrame = false,
        .id = 0x111,
    },
    {
        .isRemoteFrame = false,
        .isExtendedFrame = false,
        .id = 0x112,
    },
    {
        .isRemoteFrame = false,
        .isExtendedFrame = false,
        .id = 0x113,
    },
    {
        .isRemoteFrame = false,
        .isExtendedFrame = true,
        .id = 0x10FF1230,
    },
    {
        .isRemoteFrame = false,
        .isExtendedFrame = true,
        .id = 0x10FF1231,
    },
    {
        .isRemoteFrame = false,
        .isExtendedFrame = true,
        .id = 0x10FF1232,
    },
    {
        .isRemoteFrame = false,
        .isExtendedFrame = true,
        .id = 0x10FF1233,
    },
};
```

若需要接收扩展帧，则上图中的.isExtendedFrame参数应设置为true。

b. 配置FIFO滤波器以及开启数据接收。这里与CAN pal的驱动有所区别，pal层驱动中的初始化函数将legacy FIFO和enhance FIFO滤波表的寄存器配置集成了进去，而FlexCAN的驱动中的初始化函数不包括legacy FIFO和enhance FIFO的滤波表寄存器配置，所以需要手动添加。

```
int main(void)
{
    /* USER CODE BEGIN 1 */
    /* USER CODE END 1 */
    Board_Init();
    /* USER CODE BEGIN 2 */
    FLEXCAN_DRV_ConfigRxFifo(0, FLEXCAN_RX_FIFO_ID_FORMAT_A, can_filter_table);
    FLEXCAN_DRV_RxFifo(0, &rxFifoMsg);
    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        /* USER CODE END WHILE */
        /* USER CODE BEGIN 3 */
    }
    /* USER CODE END 3 */
}

static void Board_Init(void)
{
    CLOCK_SYS_Init(g_clockManConfigsArr,CLOCK_MANAGER_CONFIG_CNT,g_clockManCallbacksArr,CLOCK_MANAGER_CALLBACK_CNT);
    CLOCK_SYS_UpdateConfiguration(CLOCK_MANAGER_ACTIVE_INDEX,CLOCK_MANAGER_POLICY_AGREEMENT);
    PINS_DRV_Init(NUM_OF_CONFIGURED_PINS0,g_pin_mux_InitConfigArr0);
    FLEXCAN_DRV_Init(0,&flexcanInitConfig0_State,&flexcanInitConfig0);
    UTILITY_PRINT_Init();
}
```

5.2 配置收发邮箱

除了使用FIFO进行接收外，有时候还需要配置额外的接收邮箱。由于FIFO要占用一部分邮箱所在的区域（从RAM0开始），不管是配置接收邮箱还是发送邮箱，都需要注意使用的邮箱编号。以下是滤波器数量与剩余邮箱数的对应关系表。

CTRL2[RFFN]	滤波器接收码数量	首个消息邮箱起始偏移地址	剩余最大邮箱数量（最大 32 个邮箱）	剩余最大邮箱数量（最大 64 个邮箱）
0	8	0x100	24	56
1	16	0x120	22	54
2	24	0x140	20	52
3	32	0x160	18	50
4	40	0x180	16	48
5	48	0x1A0	14	46
6	56	0x1C0	12	44
7	64	0x1E0	10	42
8	72	0x200	8	40

CTRL2[RFFN]	滤波器接收码数量	首个消息邮箱起始偏移地址	剩余最大邮箱数量（最大 32 个邮箱）	剩余最大邮箱数量（最大 64 个邮箱）
9	80	0x220	6	38
10	88	0x240	4	36
11	96	0x260	2	34
12	104	0x280	0	32
13	112	0x2A0	0	32
14	120	0x2C0	0	32
15	128	0x2E0	0	32

例如，设置8个滤波器，即CTRL2[RFFN]=0，此时FIFO占用了8个邮箱，因此FIFO使能的话邮箱0~邮箱7不能配置为收发邮箱，只能使用邮箱编号为8及以后的邮箱。

5.3 设置掩码

首先调用下图中函数1设置掩码类型为独立掩码，再调用函数2设置独立掩码为0x7F0。

```c
int main(void)
{
    /* USER CODE BEGIN 1 */
    /* USER CODE END 1 */
    Board_Init();
    /* USER CODE BEGIN 2 */
    FLEXCAN_DRV_ConfigRxFifo(0, FLEXCAN_RX_FIFO_ID_FORMAT_A, can_filter_table);
    FLEXCAN_DRV_SetRxMaskType(0, FLEXCAN_RX_MASK_INDIVIDUAL);
    FLEXCAN_DRV_SetRxIndividualMask(0, FLEXCAN_MSG_ID_STD, 0, 0x7F0);
    FLEXCAN_DRV_RxFifo(0, &rxFifoMsg);
    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        if (FLEXCAN_DRV_GetTransferStatus(0, FIFO_REC) != STATUS_BUSY)
        {
            FLEXCAN_DRV_RxFifo(0, &rxFifoMsg);
        }
        if (FLEXCAN_DRV_GetTransferStatus(0, TX_MAILBOX) != STATUS_BUSY)
        {
            FLEXCAN_DRV_Send(0, TX_MAILBOX, &txinfo, TX_MSG_ID, txMsg.data);
        }
        /* Delay for 50 ms */
        OSIF_TimeDelay(50);
        /* USER CODE END WHILE */
        /* USER CODE BEGIN 3 */
    }
    /* USER CODE END 3 */
}

static void Board_Init(void)
{
    CLOCK_SYS_Init(g_clockManConfigsArr,CLOCK_MANAGER_CONFIG_CNT,g_clockManCallbacksArr,CLOCK_MANAGER_CALLBACK_CNT);
    CLOCK_SYS_UpdateConfiguration(CLOCK_MANAGER_ACTIVE_INDEX,CLOCK_MANAGER_POLICY_AGREEMENT);
    PINS_DRV_Init(NUM_OF_CONFIGURED_PINS0,g_pin_mux_InitConfigArr0);
    FLEXCAN_DRV_Init(0,&flexcanInitConfig0_State,&flexcanInitConfig0);
    UTILITY_PRINT_Init();
}
```

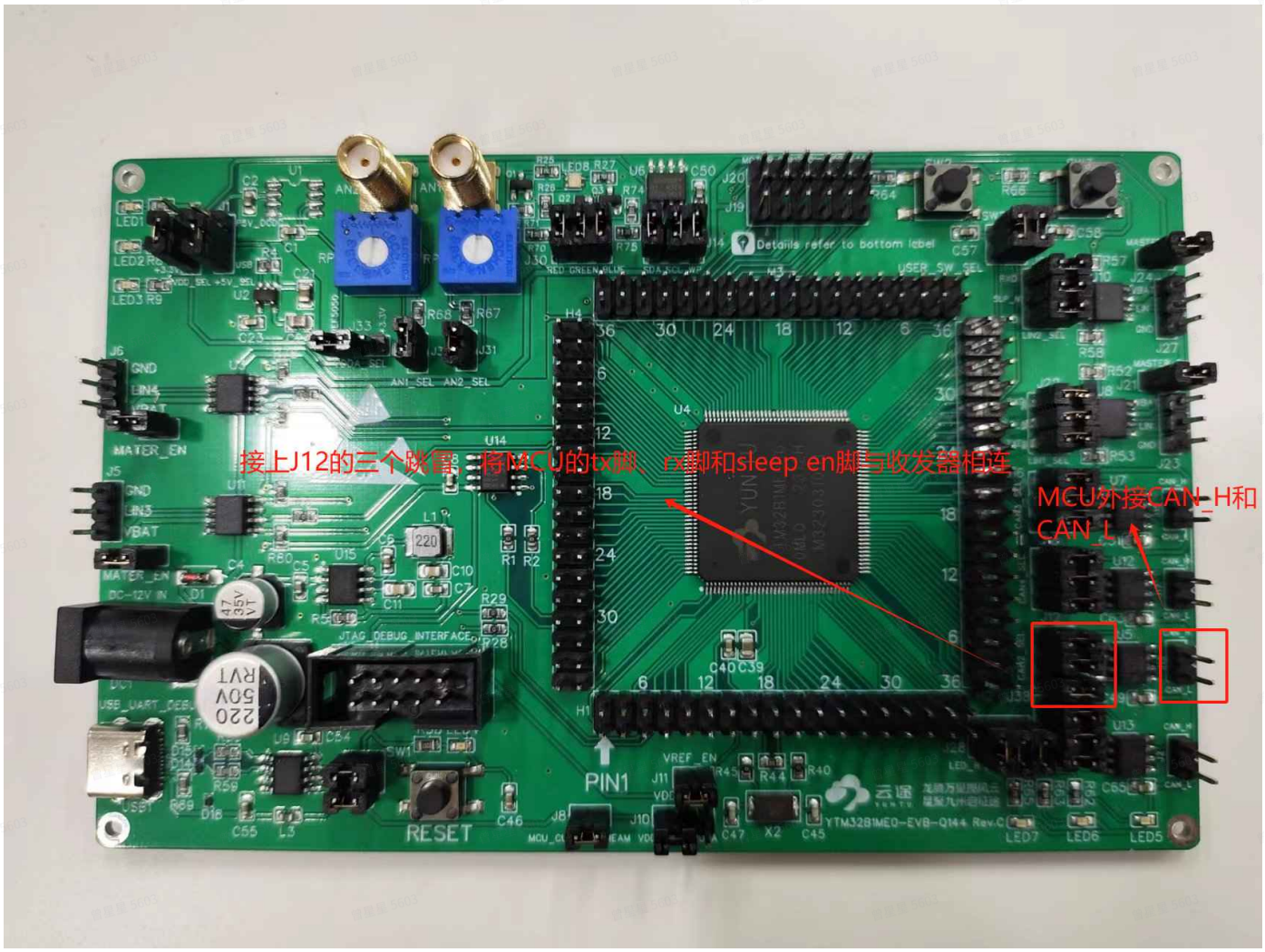
Table 18.33: CAN RXIMR Register Description

Field	Function
31 - 0 MI	**Individual Mask Bits** Each Individual Mask Bit masks the corresponding bit in both the mailbox filter and Legacy Rx FIFO ID Filter Table element in distinct ways. For mailbox filters, see the RXMGMASK register description. For Legacy Rx FIFO ID Filter Table elements, see the RXFGMASK register description. 0b - The corresponding bit in the filter is “don’t care”. 1b - The corresponding bit in the filter is checked.

由上图可知，掩码位为0代表相应的数据比特位不会被过滤，掩码位为1代表相应的数据比特位会与过滤器ID做比较，不匹配则不会接收，因此掩码为0x7F0代表CAN控制器会忽略数据ID的后四位。

5.4 通信结果展示

a. 以ME0的demo板YTM32B1M-EVB-Q144为例进行演示，CAN分析仪使用TSMaster，TSMaster的CAN_H、CAN_L分别连接到demo板上J13的CAN_H和CAN_L，如下图中箭头所指位置，上面为CAN_H，下面为CAN_L：



b. 上位机收到ID为0x310的报文，周期50ms。

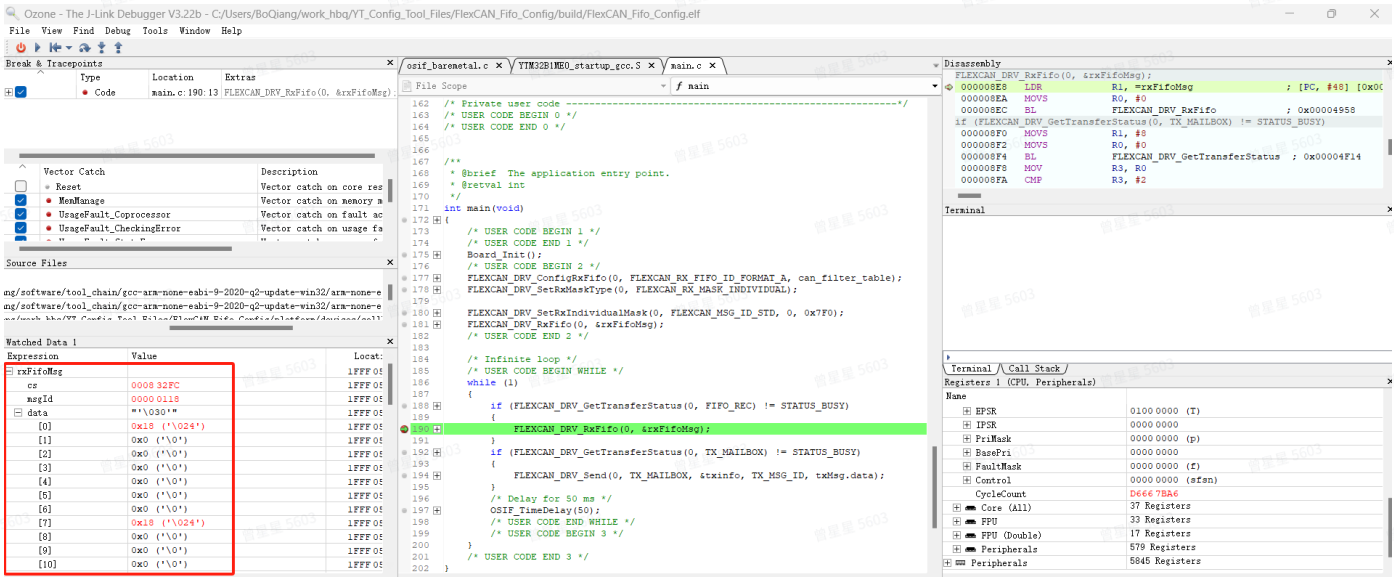
绝对时间	计数	...	标识符	帧率	报文名称	类型	...	DLC	数据长度	BRS	ESI	00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 1
0.049998	108911		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108912		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108913		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049998	108914		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049998	108915		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049998	108916		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108917		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108918		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.050000	108919		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108920		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108921		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108922		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049998	108923		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049999	108924		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049998	108925		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108926		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108927		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108928		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.050000	108929		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108930		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108931		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108932		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049998	108933		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049999	108934		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108935		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108936		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049998	108937		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00
0.049997	108938		310	19		数据帧	...	8	8	-	-	01 02 03 04 05 06 07 00

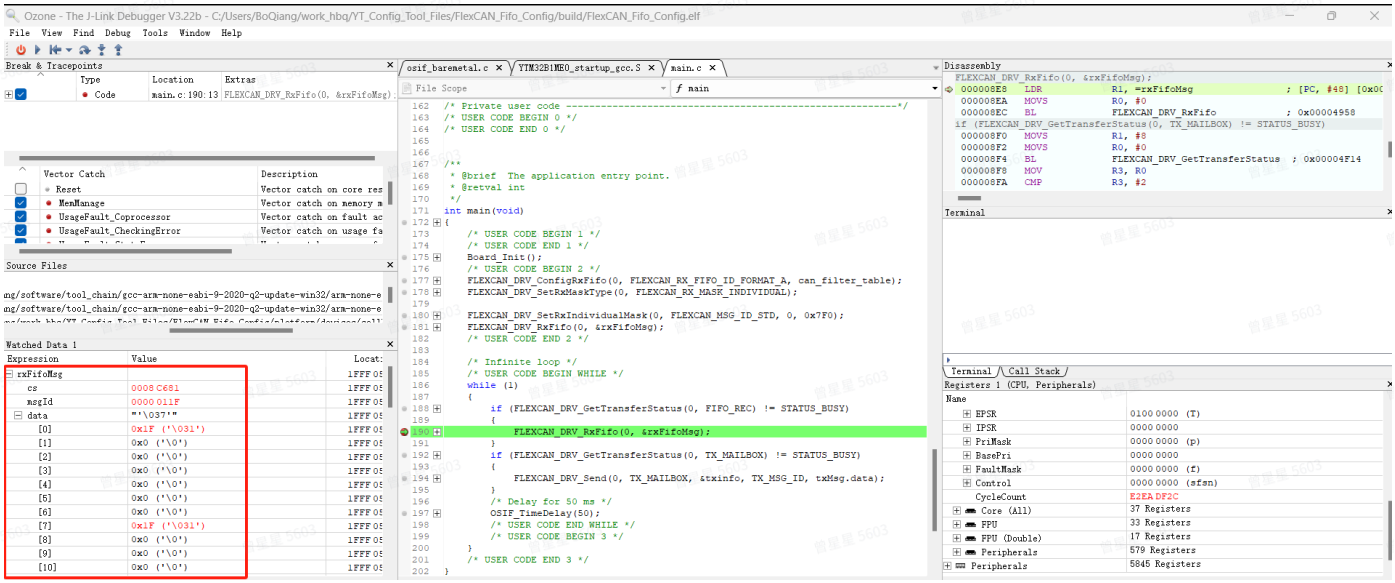
c. demo板能成功接收到滤波表中的4个扩展帧，以及ID为0x11x的标准帧。

行	发送	触发	报文名称	标识符	通道	类型	DLC	BRS	D0	D1	D2	D3	D4	D5	D6	D7	注释
1	▶	手动	NewMsg	110	1	标准数据帧	8	☐	10	00	00	00	00	00	00	10	
2	▶	手动	NewMsg	111	1	标准数据帧	8	☐	11	00	00	00	00	00	00	11	
3	▶	手动	NewMsg	112	1	标准数据帧	8	☐	12	00	00	00	00	00	00	12	
4	▶	手动	NewMsg	113	1	标准数据帧	8	☐	13	00	00	00	00	00	00	13	
5	▶	手动	NewMsg	118	1	标准数据帧	8	☐	18	00	00	00	00	00	00	18	
6	▶	手动	NewMsg	11A	1	标准数据帧	8	☐	1A	00	00	00	00	00	00	1A	
7	▶	手动	NewMsg	11F	1	标准数据帧	8	☐	1F	00	00	00	00	00	00	1F	
8	▶	手动	NewMsg	10FF1230	1	扩展数据帧	8	☐	30	00	00	00	00	00	00	30	
9	▶	手动	NewMsg	10FF1231	1	扩展数据帧	8	☐	31	00	00	00	00	00	00	31	
10	▶	手动	NewMsg	10FF1232	1	扩展数据帧	8	☐	32	00	00	00	00	00	00	32	
11	▶	手动	NewMsg	10FF1233	1	扩展数据帧	8	☐	33	00	00	00	00	00	00	33	

信号 数据字节

字节索引	字节 0	字节 1	字节 2	字节 3	字节 4	字节 5	字节 6	字节 7
0	1F	00	00	00	00	00	00	1F





6. 主要API函数介绍

1. 配置FIFO滤波表：

```
1 void FLEXCAN_DRV_ConfigRxFifo(uint8_t instance,
  flexcan_rx_fifo_id_element_format_t id_format, const flexcan_id_table_t
  *id_filter_table)
```

id_format：滤波器形式，FIFO模式下可选择A（单滤波器）、B（双滤波器）、C（四滤波器）三种形式；

id_filter_table：滤波器配置数组。

2. FIFO数据接收函数：

```
1 status_t FLEXCAN_DRV_RxFifo(uint8_t instance, flexcan_msgbuff_t *data)
```

data：装填接收数据的缓冲区。

该函数用于开启legacy FIFO的数据接收，与邮箱接收数据一样，调用一次只能接收一次，需要不断调用该函数实现连续通信。

其他API介绍请参考"SDK应用_FlexCAN模块配置及应用（一）"。

---

7. 文档历史

版本号	日期	修订记录
V1.0	2024.01.04	初始版本