Hello,

I'm getting "Bit Dominant Error" on every attempted transmission in the "CAN_MODE_SILENT_LOOPBACK" of "CAN1" in STM32F105RBT6. TEC's is then increased. REC is not being increased. I'm wondering if this is even possible, as this mode should route the TX line to RX internally, and the clock must be in sync, since this is the same peripheral? Also, what does "Bit Dominant Error" mean - peripheral tried to send dominant, but received recessive, or the other way around?

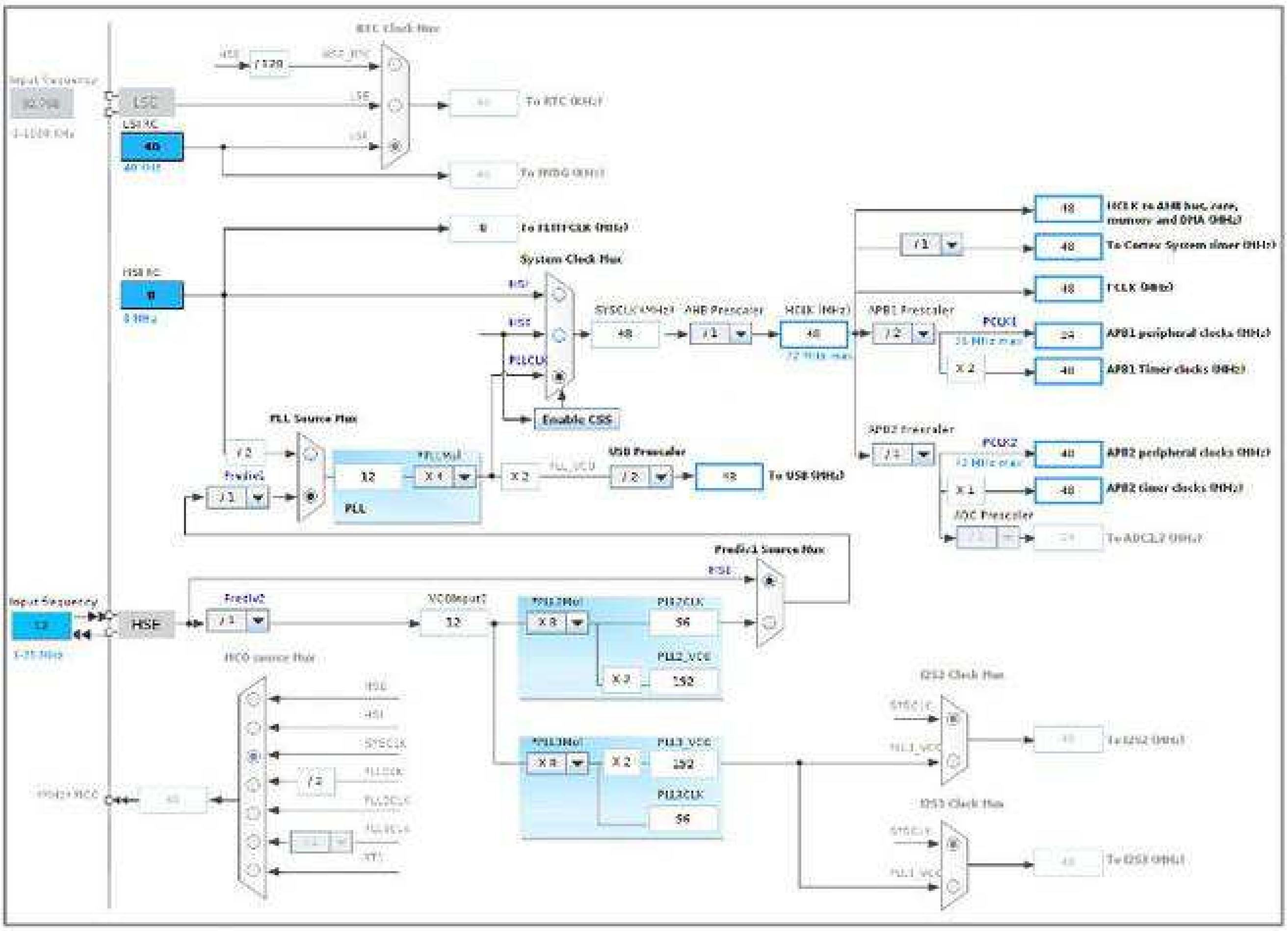
I will admit, that it is possible that I've damaged the TX GPIO, since I connected it to the RX line of my CAN interface IC. However, I've thought, that this should not have consequences for SILENT_LOOPBACK mode.

I'm using custom board with STM32F105RBT6. With the multimeter I'm measuring 3.3V on both RX and TX pins. I cannot try LOOPBACK or NORMAL modes with oscilloscope till tommorow.

Below I'm pasting my code for CAN init and message sending. My USB-CDC in device mode is working correctly, so I think that clocks are set up fine. Please, help me find my mistake, or confirm that my hardware is most probably damaged.

I've had no success in receiving any correct frame, only miscellaneous errors (form error, stuff error etc.) when connected to real CAN bus.

I'm attaching my clock tree as well.



```
void MX_CAN1_Init(CanSupportedBaudrate timingSet)
{
    hcan1.pRxMsg = &canRxMsg;
    hcan1.pTxMsg = &canTxMsg;
    hcan1.Instance = CAN1;
    hcan1.Init.Prescaler = supportedTimingSets[timingSet].Prescaler; // value: 3, clock: 24MHz
(APB1, 48MHz/2)
    hcan1.Init.Mode = CAN_MODE_SILENT_LOOPBACK;
    hcan1.Init.SJW = CAN_SJW_4TQ;
    hcan1.Init.BS1 = supportedTimingSets[timingSet].BS1;         // value: 10
    hcan1.Init.BS2 = supportedTimingSets[timingSet].BS2;         // value: 5
    hcan1.Init.TTCM = DISABLE;
    hcan1.Init.ABOM = ENABLE;
    hcan1.Init.AWUM = DISABLE;
    hcan1.Init.NART = ENABLE;
    hcan1.Init.RFLM = DISABLE;
    hcan1.Init.TXFP = DISABLE;
    if (HAL_CAN_Init(&hcan1) != HAL_OK) Error_Handler(ERR_SOURCE_CAN);

        if(HAL_CAN_Receive_IT(&hcan1, CAN_FIFO0)) Error_Handler(ERR_SOURCE_CAN);
}

void HAL_CAN_MspInit(CAN_HandleTypeDef* canHandle)
{
    GPIO_InitTypeDef GPIO_InitStruct;
    if(canHandle->Instance==CAN1)
    {
        __HAL_RCC_CAN1_CLK_ENABLE();
        /**CAN1 GPIO Configuration
        PB8     ------> CAN1_RX
        PB9     ------> CAN1_TX
         */
        GPIO_InitStruct.Pin = GPIO_PIN_8;
        GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
        GPIO_InitStruct.Pull = GPIO_NOPULL;
        HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

        GPIO_InitStruct.Pin = GPIO_PIN_9;
        GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
        GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
        HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

        __HAL_AFIO_REMAP_CAN1_2();

        HAL_NVIC_ClearPendingIRQ(CAN1_RX0_IRQn);
        HAL_NVIC_SetPriority(CAN1_RX0_IRQn, 0, 0);
        HAL_NVIC_EnableIRQ(CAN1_RX0_IRQn);
        HAL_NVIC_ClearPendingIRQ(CAN1_RX1_IRQn);
        HAL_NVIC_SetPriority(CAN1_RX1_IRQn, 0, 0);
        HAL_NVIC_EnableIRQ(CAN1_RX1_IRQn);
        HAL_NVIC_SetPriority(CAN1_SCE_IRQn, 0, 0);
        HAL_NVIC_EnableIRQ(CAN1_SCE_IRQn);
    }
}

CanErr canTransmitDataFrame(bool extendedId, uint32_t idValue, uint8_t dataLength, uint8_t *
data)
{
    if(dataLength > 8) return CAN_ERR_WRONG_PARAM;
    canTxMsg.IDE = extendedId ? CAN_ID_STD : CAN_ID_EXT;
    if(extendedId) canTxMsg.ExtId = idValue & 0x1FFFFFFF;
    else canTxMsg.StdId = idValue & 0x7FF;
    canTxMsg.RTR = CAN_RTR_DATA;
    canTxMsg.DLC = dataLength;
    for(int c = 0; c < dataLength; c++) {
        canTxMsg.Data[c] = data[c];
    }

    if(HAL_CAN_Transmit_IT(&hcan1)) return CAN_ERR_NOTREADY;
    else return CAN_ERR_NONE;
}
```