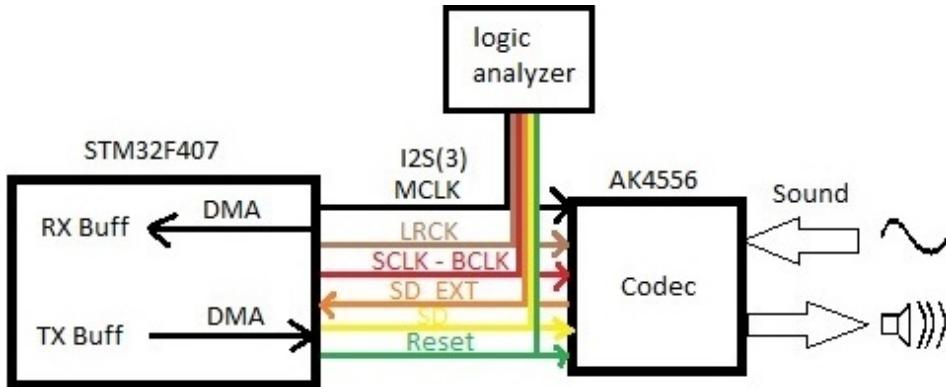


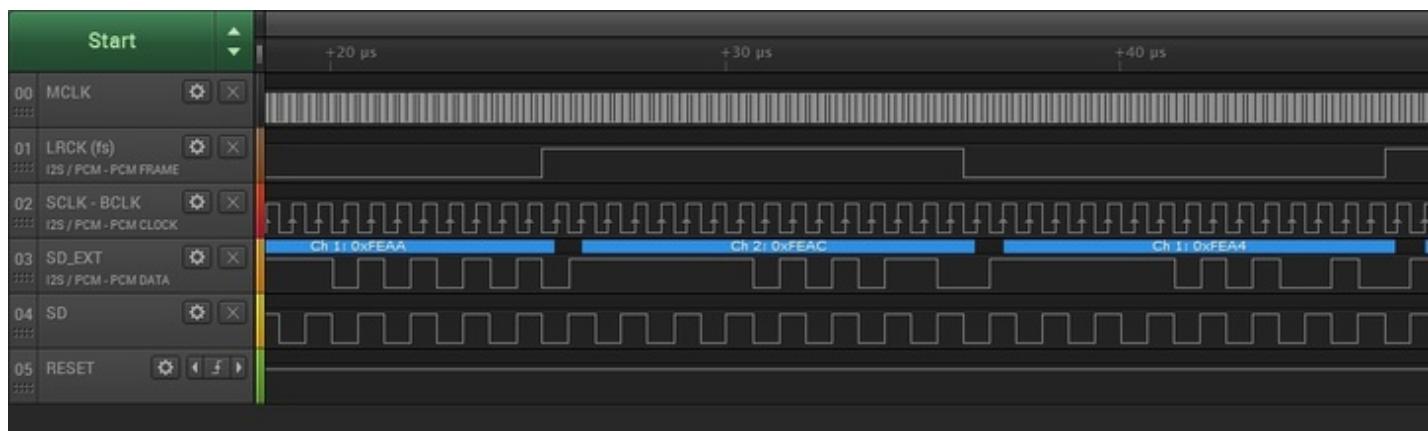
I'm having a problem with my project.

The project is just a simple sound loop?



It is a stm32f4discovery connected to a audio codec. They communicate over I2S3 in Full Duplex mode and the stm32f4 uses DMA to handle data.

My problem is this: I'm able to send data out, but not receive any my Rx buffer is written to zero all the time.



So I can see my data is on the SD_EXT line, but I'm just not able to get it inside the stm32f4.

My Code:

Defines:

```

//I2S
#define CODEC_I2S          SPI3
#define CODEC_I2S_EXT        I2S3ext
#define CODEC_I2S_CLK         RCC_APB1Periph_SPI3

//DMA DEFINES
#define I2S_DMA_CLOCK           RCC_AHB1Periph_DMA1
#define AUDIO_I2S_EXT_DMA_FLAG_TC ((uint32_t)0x18000000)
#define AUDIO_I2S_DMA_FLAG_TC      DMA_FLAG_TCIF7

//DMA_InitStructure
#define AUDIO_I2S_DMA_STREAM      DMA1_Stream7
#define AUDIO_I2S_DMA_CHANNEL     DMA_Channel_0
#define Audio_I2S_IRQHandler DMA1_Stream7_IRQHandler
  
```

```
//DMA_InitStructure2
#define AUDIO_I2S_EXT_DMA_STREAM DMA1_Stream0
#define AUDIO_I2S_EXT_DMA_CHANNEL DMA_Channel_3
#define Audio_I2S_EXT_IRQHandler DMA1_Stream0_IRQHandler

//PLAY
#define AUDIO_I2S_DMA_IRQ DMA1_Stream7_IRQHandler
#define AUDIO_I2S_EXT_DMA_IRQ DMA1_Stream0_IRQHandler
```

Init Configuration:

```
void AudioInit(void)
{
    GPIO_InitTypeDef PinInitStruct;
    GPIO_StructInit(&PinInitStruct);
    DMA_InitTypeDef DMA_InitStructure;
    DMA_InitTypeDef DMA_InitStructure2;
    I2S_InitTypeDef I2S_InitType;

    /*****
     * PIN & PORT settings
     *****/
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA | RCC_AHB1Periph_GPIOB | RCC_AHB1Periph_GPIOC |
    RCC_AHB1Periph_GPIOD , ENABLE);

    //Reset pin as GPIO
    PinInitStruct.GPIO_Pin = I2S_RESET_PIN;
    PinInitStruct.GPIO_Mode = GPIO_Mode_OUT;
    PinInitStruct.GPIO_PuPd = GPIO_PuPd_DOWN;
    PinInitStruct.GPIO_OType = GPIO_OType_PP;
    PinInitStruct.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(I2S_RESET_Port, &PinInitStruct);

    //I2S pins
    PinInitStruct.GPIO_Mode = GPIO_Mode_AF;
    PinInitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;

    PinInitStruct.GPIO_Pin = I2S_MCLK_PIN;
    GPIO_Init(I2S_MCLK_Port, &PinInitStruct);

    PinInitStruct.GPIO_Pin = I2S_SCLK_PIN;
    GPIO_Init(I2S_SCLK_Port, &PinInitStruct);

    PinInitStruct.GPIO_Pin = I2S_WS_PIN;
    GPIO_Init(I2S_WS_Port, &PinInitStruct);

    PinInitStruct.GPIO_Pin = I2S_SD_PIN ;
    GPIO_Init(I2S_SD_Port, &PinInitStruct);

    PinInitStruct.GPIO_Pin = I2S_SD_ext_PIN ;
    GPIO_Init(I2S_SD_ext_Port, &PinInitStruct);

    //prepare output ports for alternate function
    GPIO_PinAFConfig(I2S_WS_Port, I2S_WS_PinSource, SPI);
    GPIO_PinAFConfig(I2S_MCLK_Port, I2S_MCLK_PinSource, SPI);
    GPIO_PinAFConfig(I2S_SCLK_Port, I2S_SCLK_PinSource, SPI);
    GPIO_PinAFConfig(I2S_SD_Port, I2S_SD_PinSource, SPI);
    GPIO_PinAFConfig(I2S_SD_ext_Port, I2S_SD_ext_PinSource, SPI);

    /*****
     * I2S settings
     *****/
}
```

```

    //enable I2S clock
    RCC_APB1PeriphClockCmd(CODEC_I2S_CLK, ENABLE);
    RCC_PLLI2SCmd(ENABLE);

    // configure I2S port
    SPI_I2S_DeInit(CODEC_I2S);
    I2S_InitType.I2S_AudioFreq = I2S_AudioFreq_44k; // SET the Freq
    I2S_InitType.I2S_MCLKOutput = I2S_MCLKOutput_Enable;
    I2S_InitType.I2S_DataFormat = I2S_DataFormat_16b;
    I2S_InitType.I2S_Mode = I2S_Mode_MasterTx;
    I2S_InitType.I2S_Standard = I2S_Standard_Phillips;
    I2S_InitType.I2S_CPOL = I2S_CPOL_Low;

    /* Initialize the I2S main channel for TX */
    I2S_Init(CODEC_I2S, &I2S_InitType);

    /* Initialize the I2S extended channel for RX */
    I2S_FullDuplexConfig(CODEC_I2S_EXT, &I2S_InitType);

    /**************************************************************************
     * DMA settings
     **************************************************************************/
    /* Enable the DMA clock */
    RCC_AHB1PeriphClockCmd(I2S_DMA_CLOCK, ENABLE);

    /* Configure the DMA Stream */
    DMA_Cmd(AUDIO_I2S_DMA_STREAM, DISABLE);
    DMA_Cmd(AUDIO_I2S_EXT_DMA_STREAM, DISABLE);
    DMA_DeInit(AUDIO_I2S_DMA_STREAM);
    DMA_DeInit(AUDIO_I2S_EXT_DMA_STREAM);

    /* Set the parameters to be configured */
    DMA_InitStructure.DMA_Channel = AUDIO_I2S_DMA_CHANNEL;
    DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&SPI3->DR;
    DMA_InitStructure.DMA_Memory0BaseAddr = (uint32_t)txdata;
    DMA_InitStructure.DMA_DIR = DMA_DIR_MemoryToPeripheral;
    DMA_InitStructure.DMA_BufferSize = I2S_BufferSize;
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;//DMA_Mode_Normal; //
    DMA_InitStructure.DMA_Priority = DMA_Priority_High;
    DMA_InitStructure.DMA_FIFOMode = DMA_FIFOMode_Disable;
    DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_1QuarterFull;
    DMA_InitStructure.DMA_MemoryBurst = DMA_MemoryBurst_Single;
    DMA_InitStructure.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;
    DMA_Init(AUDIO_I2S_DMA_STREAM, &DMA_InitStructure);

    /* Set the parameters to be configured */
    DMA_InitStructure2.DMA_Channel = AUDIO_I2S_EXT_DMA_CHANNEL;
    DMA_InitStructure2.DMA_PeripheralBaseAddr = (uint32_t)&I2S3ext->DR;
    DMA_InitStructure2.DMA_Memory0BaseAddr = (uint32_t)&rxdata;
    DMA_InitStructure2.DMA_DIR = DMA_DIR_PeripheralToMemory;
    DMA_InitStructure2.DMA_BufferSize = I2S_BufferSize;
    DMA_InitStructure2.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    DMA_InitStructure2.DMA_MemoryInc = DMA_MemoryInc_Enable;
    DMA_InitStructure2.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
    DMA_InitStructure2.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
    DMA_InitStructure2.DMA_Mode = DMA_Mode_Circular;//DMA_Mode_Normal; //
    DMA_InitStructure2.DMA_Priority = DMA_Priority_Low;
    DMA_InitStructure2.DMA_FIFOMode = DMA_FIFOMode_Disable;
    DMA_InitStructure2.DMA_FIFOThreshold = DMA_FIFOThreshold_1QuarterFull;
    DMA_InitStructure2.DMA_MemoryBurst = DMA_MemoryBurst_Single;
    DMA_InitStructure2.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;
    DMA_Init(AUDIO_I2S_EXT_DMA_STREAM, &DMA_InitStructure2);

    I2S_Cmd(CODEC_I2S, ENABLE); //Enables or disables the
    specified SPI peripheral (in I2S mode).

```

```

I2S_Cmd(CODEC_I2S_EXT, ENABLE); //Enables or disables the
specified SPI peripheral (in I2S mode).

/* Enable the I2S DMA request */
SPI_I2S_DMACmd(CODEC_I2S, SPI_I2S_DMAReq_Tx, ENABLE); //Enables or disables the
SPIx/I2Sx DMA interface.
SPI_I2S_DMACmd(CODEC_I2S_EXT, SPI_I2S_DMAReq_Rx, ENABLE); //Enables or disables the
SPIx/I2Sx DMA interface.

/* Enable the selected DMA interrupts (selected in "stm32f4_discovery_eval_audio_codec.h"
defines) */
//DMA_ITConfig(AUDIO_I2S_DMA_STREAM, DMA_IT_TC, ENABLE); //Enables or disables the
specified DMAy Streamx interrupts.
DMA_ITConfig(AUDIO_I2S_EXT_DMA_STREAM, DMA_IT_TC, ENABLE); //Enables or disables the
specified DMAy Streamx interrupts.

/* I2S DMA IRQ Channel configuration */
//NVIC_EnableIRQ(AUDIO_I2S_DMA_IRQHandler); //This function enables a
device specific interrupt in the NVIC interrupt controller.
NVIC_EnableIRQ(AUDIO_I2S_EXT_DMA_IRQHandler); //This function enables a
device specific interrupt in the NVIC interrupt controller.

/*****************
Start
*****************/

```

//Codec Activate
GPIO_SetBits(I2S_RESET_Port, I2S_RESET_PIN);

/* Enable the I2S DMA Streams */
DMA_Cmd(AUDIO_I2S_DMA_STREAM, ENABLE); //Enables or disables the
specified DMAy Streamx.
DMA_Cmd(AUDIO_I2S_EXT_DMA_STREAM, ENABLE); //Enables or disables the
specified DMAy Streamx.

}

The ISR DMA Stream of the I2S3_EXT_RX:

```

void Audio_I2S_EXT_IRQHandler(void)
{
    STM_EVAL_LEDToggle(LED5);
    if(DMA_GetITStatus(AUDIO_I2S_EXT_DMA_STREAM, DMA_IT_TCIF0))
    {
        DMA_ClearITPendingBit(AUDIO_I2S_EXT_DMA_STREAM, DMA_IT_TCIF0);
    }
}

```

My Main is just a Init and While(1) loop blinking a led:

Thanks in advance