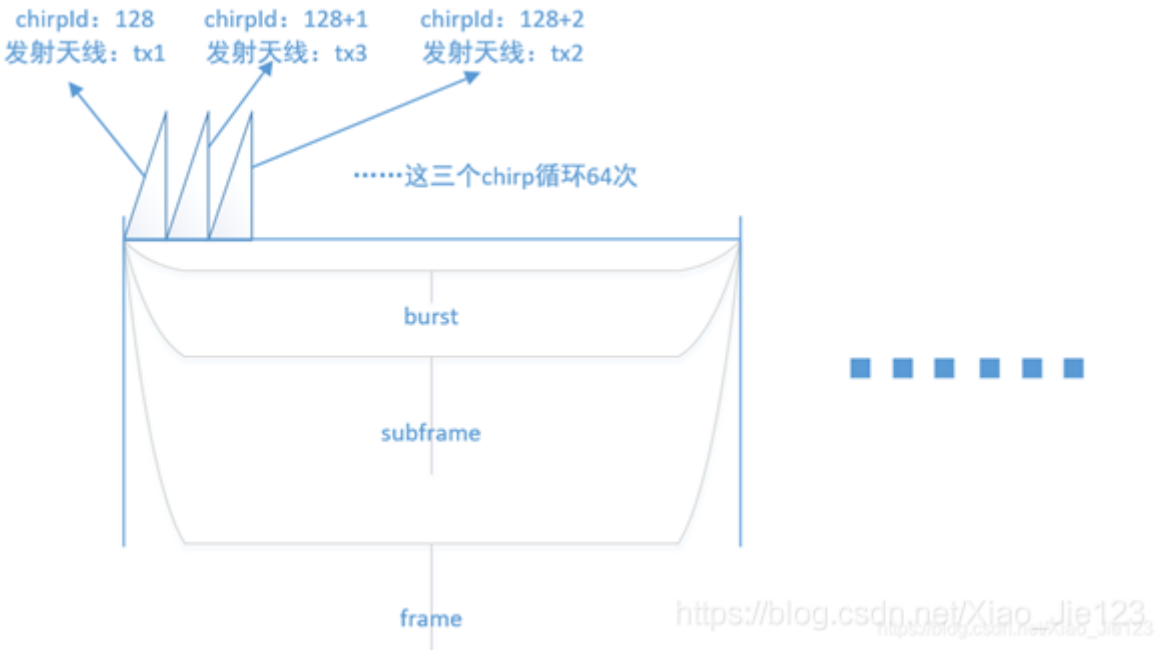
/Falcon_18xx_mss/Source/system/hal/RF/cfg_Ext.h

```
extern void Cfg_AdvFrameCfgInitParams (rlAdvFrameCfg_t* ptrAdvFrameCfg);
extern void Cfg_FrameCfgInitParams (rlFrameCfg_t* ptrFrameCfg);
extern void Cfg_ProfileCfgInitParams (uint8_t profileNum, rlProfileCfg_t*
ptrProfileCfg);
extern void Cfg_ChirpCfgInitParams (uint32_t chirpNum, rlChirpCfg_t* ptrChirpCfg);
extern void Cfg_LowPowerModeInitParams (rlLowPowerModeCfg_t* ptrLowPowerMode);
extern void Cfg_ChannelCfgInitParams (rlChanCfg_t* ptrChannelCfg);
extern void Cfg_ADCOutCfgInitParams (rlAdcOutCfg_t* ptrADCOutCfg);
```

advanced frame概念:

> 为了提供frame中chirp的最大灵活性，advanced frame提供了将frame分解为不同的subframe（最多四个？），每个sub-frame由多个burst组成（多达64次），每个burst最多可以由512个不同的chirp组成，每个chirp要与4个profile中的一个关联



/Falcon_18xx_mss/Source/system/hal/RF/cfg.c

```
/**
 *  @b Description
 *  @n
 *      The function initializes the frame configuration with the default
 *      parameters. 该函数使用默认参数初始化advanced frame帧配置。
 *
```

```
 *  @param[out] ptrAdvFrameCfg
 *      Pointer to the adavance frame configuration
 *
 *  @retval
 *      Not applicable
 */
void Cfg_AdvFrameCfgInitParams (rlAdvFrameCfg_t* ptrAdvFrameCfg)
{
  #if  SINGLE_TRIGGER//single  trigger
    ptrAdvFrameCfg->frameSeq.numFrames     = 1u;
    ptrAdvFrameCfg->frameSeq.subFrameTrigger=1u;//为每个sub-frame提供触发器
  #else //continuous trigger
    ptrAdvFrameCfg->frameSeq.numFrames     = 0u;
    ptrAdvFrameCfg->frameSeq.subFrameTrigger=0u;//应为每个burst提供触发器。
}
```

该函数可以定义高级帧AdvFrame的属性，如子帧中的burst的数量、burst中的chirps和 loops 的数、要发送的sub-frame序列、要发送的frames的数量，帧的周期性和触发方法。

```
/**
 *  @b Description
 *  @n
 *      The function initializes the frame configuration with the default
 *      parameters.
 *
 *  @param[out] ptrFrameCfg
 *      Pointer to the frame configuration
 *
 *  @retval
 *      Not applicable
 */
void Cfg_FrameCfgInitParams (rlFrameCfg_t* ptrFrameCfg)
{

}
```

这个函数定义如何对这些chirps进行排序。同一个chirp可以简单的循环以创建一个大的FMCW帧，或者可以对多个独特的chirp进行排序来创建帧

Chirp Start和end Index定义了如何在一个帧中对它们进行排序。

该API还允许配置要传输的帧数，帧的周期性和触发器方法。触发方法可以是基于SW-API的触发器或基于HW-SYNC-IN的触发器。这个API内部调用两个API，一个调用RadarSS进行传感器配置，另一个调用MasterSS进行数据路径配置

```
/**
 *  @b Description
 *  @n
 *      The function initializes the profile configuration with the default
 *      parameters.
 *
 *  @param[in] profileNum
 *      Profile number to be initialized,Profiles的数量
 *  @param[out] ptrProfileCfg
 *      Pointer to the profile configuration
 *
 *  @retval
 *      Not applicable
 */
void Cfg_ProfileCfgInitParams (uint8_t profileNum, rlProfileCfg_t* ptrProfileCfg)
{
  if (profileNum == 0U)
    {
        /* Populate the default configuration for profile 0  */
        ptrProfileCfg->profileId             = PROFILE0_MRR_PROFILE_ID;
        ptrProfileCfg->startFreqConst        = PROFILE0_MRR_START_FREQ_VAL; //开始频率
        ptrProfileCfg->idleTimeConst         = PROFILE0_MRR_IDLE_TIME_VAL; //空闲时间
        ptrProfileCfg->adcStartTimeConst     = PROFILE0_MRR_ADC_START_TIME_VAL;// adc采
样开始时间
        ptrProfileCfg->rampEndTime           = PROFILE0_MRR_RAMP_END_TIME_VAL; // 斜率
结束时间 3000u   1 LSB = 10 ns\n  30us
        ptrProfileCfg->txOutPowerBackoffCode = PROFILE0_MRR_TXOUT_POWER_BACKOFF; ////0u
        ptrProfileCfg->txPhaseShifter        = PROFILE0_MRR_TXPHASESHIFTER_VAL;//0u
        ptrProfileCfg->freqSlopeConst        = PROFILE0_MRR_FREQ_SLOPE_VAL;  //波形斜率s
        ptrProfileCfg->txStartTime           = PROFILE0_MRR_TX_START_TIME_VAL; //tx开始
时间
        ptrProfileCfg->numAdcSamples         = PROFILE0_MRR_ADC_SAMPLE_VAL;    //ADC采
样时间
        ptrProfileCfg->digOutSampleRate      = PROFILE0_MRR_DIGOUT_SAMPLERATE_VAL;//采
样率
        ptrProfileCfg->hpfCornerFreq1        = PROFILE0_MRR_HPFCORNER_FREQ1_VAL;// HPF
1 corner frequency.
        ptrProfileCfg->hpfCornerFreq2        =
PROFILE0_MRR_HPFCORNER_FREQ2_VAL;//Supported HPF-2 corner frequencies
        ptrProfileCfg->rxGain                = PROFILE0_MRR_RX_GAIN_VAL;//Rx gain is
kept at the maximum
        /*ptrProfileCfg->txCalibEnCfg =0x80;*/
    }
    else if(profileNum == 1U){}
    else if(profileNum == 2U){}
    else if(profileNum == 3U){}
}
```

函数功能：

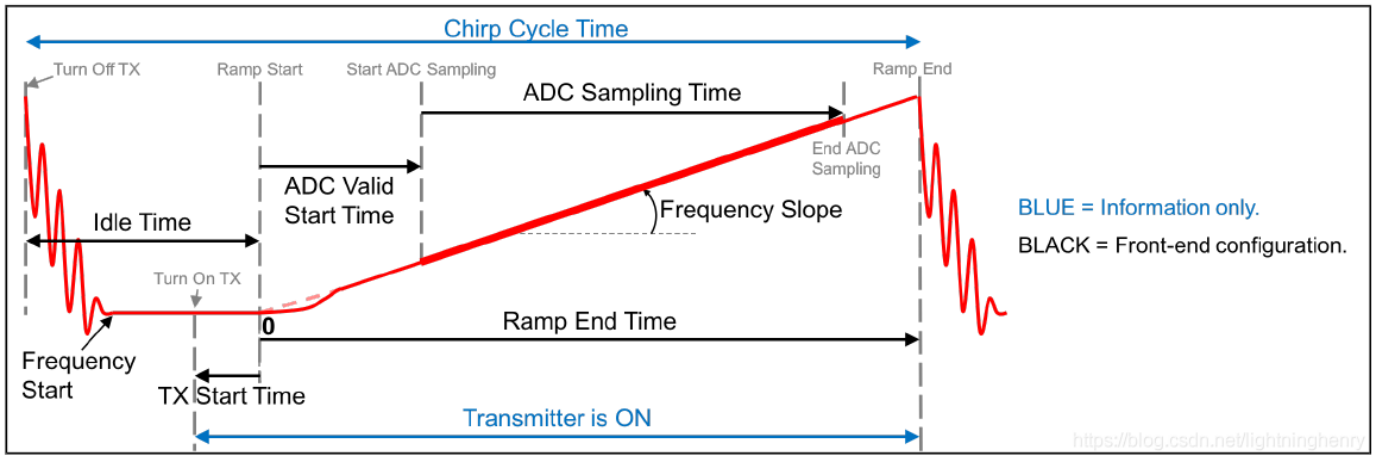
这个函数用于设置mmWave Front end的chirp profile，一个profile就像一个模板，其中包含有关FMCW信号的粗略信息，如起始频率、chirp斜率、chirp持续时间、发生功能等。

注意：

1.每个chirp要与4个profile中的一个关联，最多可以设置多达4个profiles。每个profile包含粗略信息。可以使用chirp配置API来添加精细的抖动

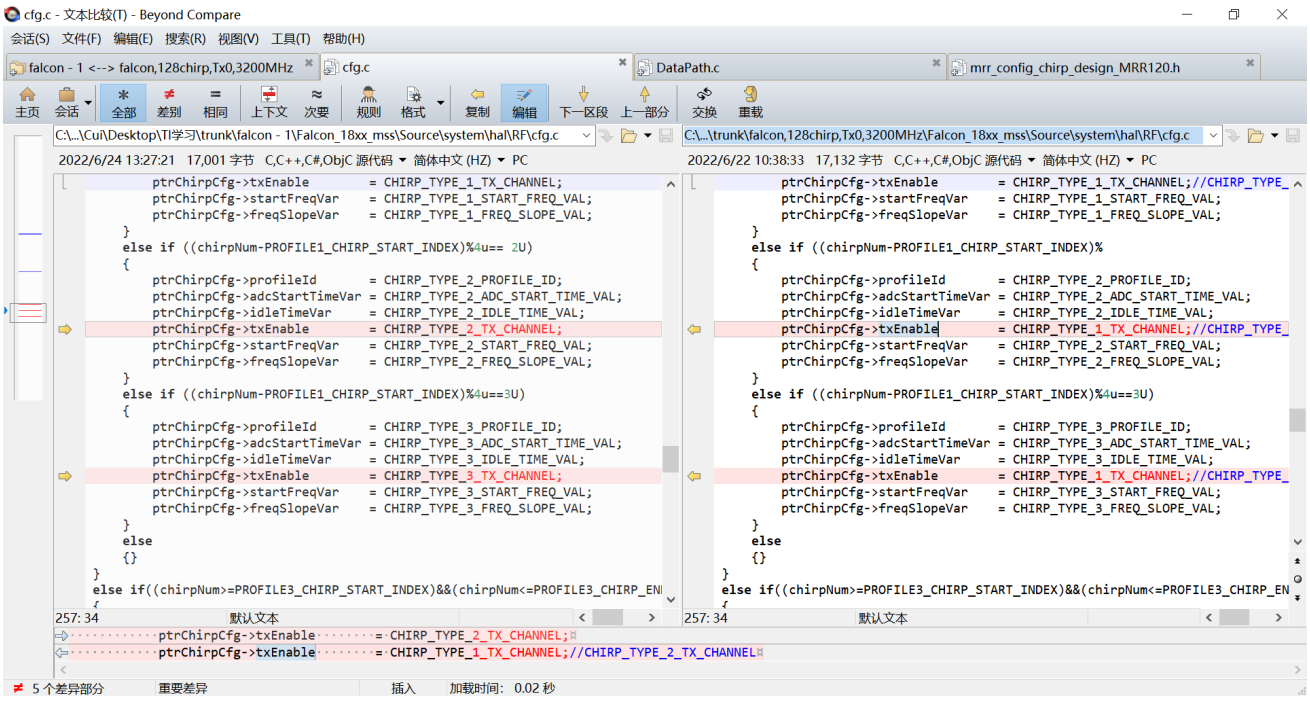
2.可以动态地调用这个API来更改profile地参数。少数的参数不能被修改，为：？？

3.最大的TX输出功率仅支持20dB



本程序中Tx通道为3个，chirp数量为512.

> 在3200M修改带宽中，1的idleTimeVar置0
>
> Tx通道改为1个：全用1channel，下面的channel不改因为只用到256个chirp




```
、/**
 *  @b Description
 *  @n
 *      The function initializes the chirp configuration with the default
 *      parameters.
 *
 *  @param[out] chirpNum
 *      Chirp Number to be configured，配置Chirp的数量
 *  @param[out] ptrChirpCfg
 *      Pointer to the chirp configuration
 *
 *  @retval
 *      Not applicable
 */
void Cfg_ChirpCfgInitParams (uint32_t chirpNum, rlChirpCfg_t* ptrChirpCfg)
{
   if((chirpNum>=PROFILE1_CHIRP_START_INDEX)&&(chirpNum<=PROFILE1_CHIRP_END_INDEX)) //
(0,255)
    {
        if ((chirpNum-PROFILE1_CHIRP_START_INDEX)%4u== 0U)
        {
            ptrChirpCfg->profileId       = CHIRP_TYPE_0_PROFILE_ID;//4个profiles,案例中
只用到256个chirp（前四个）这8个采用0，1，2，3，3，3，3，3？
            ptrChirpCfg->adcStartTimeVar = CHIRP_TYPE_0_ADC_START_TIME_VAL;//adc采样开始
时间
            ptrChirpCfg->idleTimeVar     = CHIRP_TYPE_0_IDLE_TIME_VAL;//除了1为1000全是
0？后来修改回来
            ptrChirpCfg->txEnable        = CHIRP_TYPE_0_TX_CHANNEL;//规律：发送通道
1,1,2,3使能,因为只有三通道
            ptrChirpCfg->startFreqVar    = CHIRP_TYPE_0_START_FREQ_VAL;//0
            ptrChirpCfg->freqSlopeVar    = CHIRP_TYPE_0_FREQ_SLOPE_VAL;//0
```

```
        }
        else if ((chirpNum-PROFILE1_CHIRP_START_INDEX)%4u== 1U)
        {
            ptrChirpCfg->profileId       = CHIRP_TYPE_1_PROFILE_ID;
            ptrChirpCfg->adcStartTimeVar = CHIRP_TYPE_1_ADC_START_TIME_VAL;
            ptrChirpCfg->idleTimeVar     = CHIRP_TYPE_1_IDLE_TIME_VAL;
            ptrChirpCfg->txEnable        = CHIRP_TYPE_1_TX_CHANNEL;
            ptrChirpCfg->startFreqVar    = CHIRP_TYPE_1_START_FREQ_VAL;
            ptrChirpCfg->freqSlopeVar    = CHIRP_TYPE_1_FREQ_SLOPE_VAL;
        }
       else if ((chirpNum-PROFILE1_CHIRP_START_INDEX)%4u== 2U){}
       else if ((chirpNum-PROFILE1_CHIRP_START_INDEX)%4u== 3U){}
       else{}
    }

//(256,511) ?mss_mrr_cli.c第455行
  else if((chirpNum>=PROFILE3_CHIRP_START_INDEX)&&(chirpNum<=PROFILE3_CHIRP_END_INDEX))
    {
        if ((chirpNum-PROFILE3_CHIRP_START_INDEX)%4u==0U)
        {
            ptrChirpCfg->profileId       = CHIRP_TYPE_4_PROFILE_ID;
            ptrChirpCfg->adcStartTimeVar = CHIRP_TYPE_4_ADC_START_TIME_VAL;
            ptrChirpCfg->idleTimeVar     = CHIRP_TYPE_4_IDLE_TIME_VAL;
            ptrChirpCfg->txEnable        = CHIRP_TYPE_4_TX_CHANNEL;
            ptrChirpCfg->startFreqVar    = CHIRP_TYPE_4_START_FREQ_VAL;
            ptrChirpCfg->freqSlopeVar    = CHIRP_TYPE_4_FREQ_SLOPE_VAL;
        }
        if ((chirpNum-PROFILE3_CHIRP_START_INDEX)%4u==1U){}
        if ((chirpNum-PROFILE3_CHIRP_START_INDEX)%4u==2U){}
        if ((chirpNum-PROFILE3_CHIRP_START_INDEX)%4u==3U){}
        else{}
    }
}
```

函数功能：

此函数用于在chirp profile的顶部设置chirp到chirp的变化，

用于首先应该使用 `Cfg_ProfileCfgInitParams` 定义一个profile。然后，这个函数通过将chirp与 `Cfg_ProfileCfgInitParams` API中定义的特定的profile配置关联起来配置chirp。

此外，用户能使用此API定义profile中的参数的精细抖动。此配置使用的抖动只是 `Cfg_ProfileCfgInitParams` 中编程参数的附加值。此API允许配置1个或512个chirp（每个burst最多可以由512个不同的chirp组成，能被存储在毫米波前端的专用存储器中。因此，用户不需要再运行时对chirp进行编程）。它还允许配置每个chirp使用哪些发射通道（根据雷达）。

```
/**
 *  @b Description
 *  @n
 *      The function initializes the low power configuration with the default
 *      parameters.
 *
 *  @param[out] ptrLowPowerMode
 *      Pointer to the low power mode configuration
 *
 *  @retval
 *      Not applicable
 */
void Cfg_LowPowerModeInitParams (rlLowPowerModeCfg_t* ptrLowPowerMode)
{
}
```

```
/**
 *  @b Description
 *  @n
 *      The function initializes the channel configuration with the default
 *      parameters.
 *
 *  @param[out] ptrChannelCfg
 *      Pointer to the channel configuration
 *
 *  @retval
 *      Not applicable
 */
void Cfg_ChannelCfgInitParams (rlChanCfg_t* ptrChannelCfg)
{
}
```

```
/**
 *  @b Description
 *  @n
 *      The function initializes the ADCOut configuration with the default
 *      parameters.
 *
 *  @param[out] ptrADCOutCfg
 *      Pointer to the ADCOutput configuration
 *
 *  @retval
 *      Not applicable
 */
void Cfg_ADCOutCfgInitParams (rlAdcOutCfg_t* ptrADCOutCfg)
{
```

```
}
```