分支版本

挖机版本b_1.3.8.0_tb

gnss分支 ->base： dev2.0_tb_station

前置指令

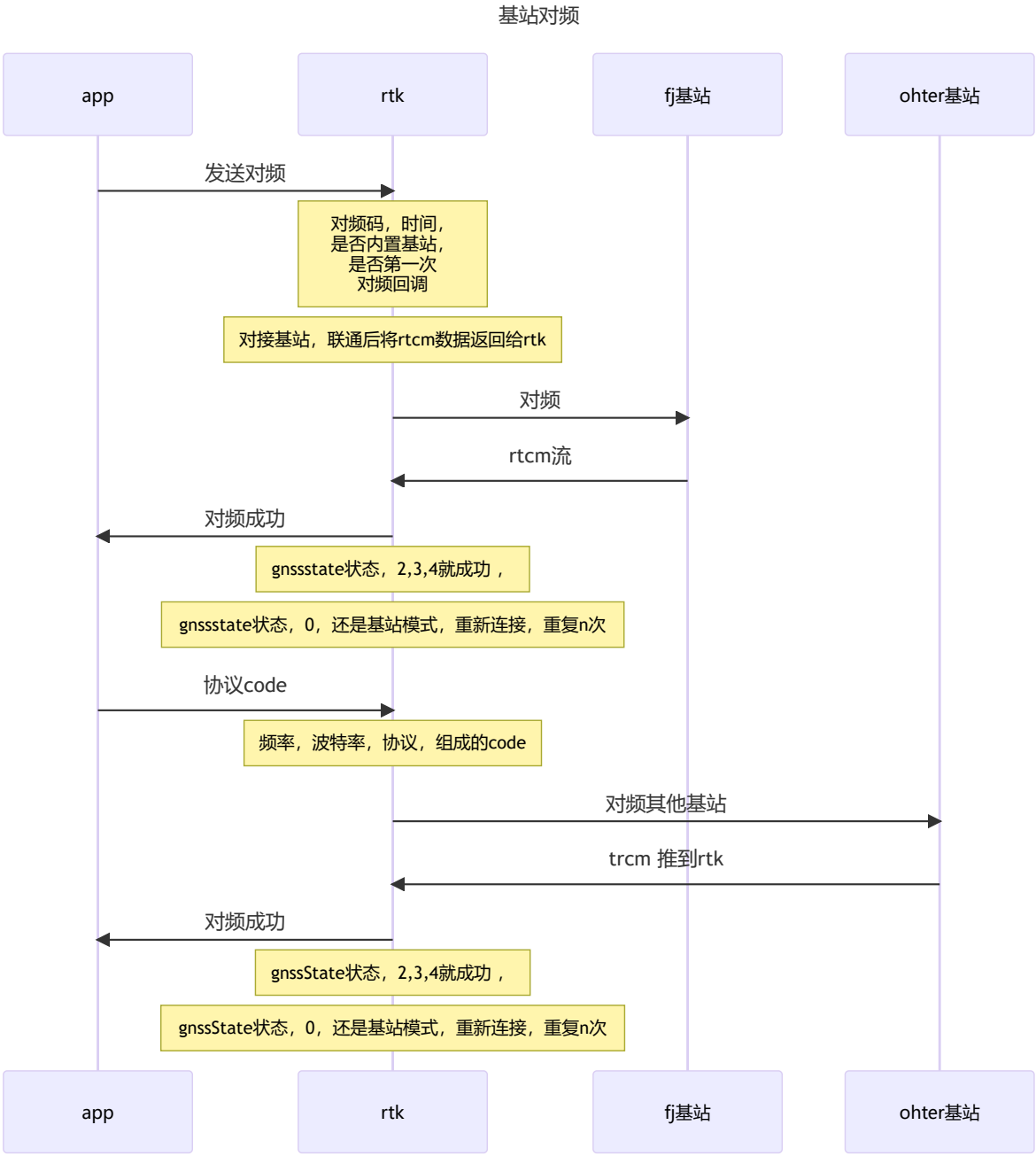
```
//设置 新老基站区分标志位
cc00 新基站：EB900000000000000000000000000010100000001CC00CF0D0A
cc01 老基站：EB900000000000000000000000000020100000001CC01D10D0A
cc02 天宝基站:EB900000000000000000000000000100000001CC02D00D0A
```

参数含义	位	含义	值	例如hexString
hzValue	[0,3]	频次		00064190
rate	[4,4]	波特率枚举	4800:0,8600:1,19200:2	4800:0,8600:1,19200:2
protocol	[5,5]	电台协议枚举	TRIMTALK:0	TRIMTALK：0, TRIMMARK3：1, Transparent-EOT:2, SATEL:3

对频指令

```
*  hzValue         [0~3]    4   uint32_t     频次   所有值*1000
*  rate     [4-5]   2   uint32_ 波特率枚举    4800:0,8600:1,19200:2
*  protocol [6-7]   2   uint32_ 电台协议枚举  TRIMTALK:0
单条内容体
000641900000
指令aa
EB900000000000000000000000000030100000006AA0006419000008B0D0A
```

内置电台,外置电台 一致


基站对频
app
rtk
fj基站
ohter基站
发送对频
对频码，时间，
是否内置基站，
是否第一次
对频回调
对接基站，联通后将rtcm数据返回给rtk
对频
rtcm流
对频成功
gnssstate状态，2,3,4就成功，
gnssstate状态，0，还是基站模式，重新连接，重复n次
协议code
频率，波特率，协议，组成的code
对频其他基站
trcm 推到rtk
对频成功
gnssState状态，2,3,4就成功，
gnssState状态，0，还是基站模式，重新连接，重复n次
app
rtk
fj基站
ohter基站

组合发送

```
前置cc指令
天宝站:EB900000000000000000000000000000100000001CC02D00D0A
 新基站:EB900000000000000000000000000010100000001CC00CF0D0A
 老基站:EB900000000000000000000000000020100000001CC01D10D0A
 对频指令
发送的数据:hex 000641900000
 指令: aa
发送到底层的包: EB900000000000000000000000000030100000006AA0006419000008B0D0A
```

新增Api IBaseStationManager

```
    /**
     * @param hexCode  7=>
     * hzValue                [0~3] 4   uint32_t    频次   所有值*1000
     * rate     [4-5]   2   uint32_ 波特率枚举    4800:0,8600:1,19200:2
     * protocol [6-7]   2   uint32_ 电台协议枚举   TRIMTALK:0，TRIMTALK3:1，
Tansparent-EDT:2,SATEL:3
     */
    void setOtherPairingCode(String hexCode);

        /**
     * 设置其他基站对拼码 区别于新老封疆基站
     * @param hexCode
     */
    void setPairingOtherCode(String hexCode);


      /**
     *
     * @param hexCode 对频码
     *  hzValue         [0~3]   4   uint32_t    频次   所有值*1000
     *  rate     [4-5]  2   uint32_ 波特率枚举    4800:0,8600:1,19200:2
     *  protocol [6-7]  2   uint32_ 电台协议枚举   TRIMTALK:0
     * @param time  次数
     * @param hasInternalRadio 是否内置电台
     * @param first 是否第一次
     * @param listener 回调
     *                 hzValue
     *                 00066A94 01 01
     */
    void pairBaseOtherStation(String hexCode, int time, boolean hasInternalRadio,
boolean first, OnFrequencyPairingListener listener);
```

IGnssManager

```
    /**
     * 初始化其他基站
     * @param hexCode
     */
    void initBsOtherManager(String hexCode);
```

自动对频做的处理 根据对频码进行判断是那个基站自动连接

```
/**
 * 自动对频
 *
 * @param gpsState
 */
@Override
public synchronized void autoPairing(int gpsState) {
    if (gpsState <= 1 && !GnssManager.getInstance().getNetRtcmManager().isLock()
&&
            mBaseStationState != BaseStationState.PAIRING && mBaseStationState !=
BaseStationState.PAIR_SUCCESS) {
        FJXLog.INSTANCE.d(TAG, "autoPairing: 基站自动对频");
        //判断对频码是否存在
        if (!TextUtils.isEmpty(mPairingCode)){
            // Howard.Zhang on 2021/6/29 重试次数从1次增加到10次，避免偶现的自动对频失败
            pairBaseStation(mPairingCode, 10,
GnssManager.getInstance().isHasInternalRadio(), true, new
OnFrequencyPairingListener() {
                @Override
                public void onPairingResult(boolean isSuccess) {
                    FJXLog.INSTANCE.d(TAG, "autoPairing: 自动对频 = " +
isSuccess);
                }

                @Override
                public void onReceivePairingCmd() {
                    FJXLog.INSTANCE.d(TAG, "autoPairing: 对频码设置成功");
                }
            });
        }
        //验证是否是其他对频模式
        if
(BaseStateionValidataUtils.baseOhterPairCodeValidate(mOtherPairingCode)){
            pairBaseOtherStation(mOtherPairingCode, 10,
GnssManager.getInstance().isHasInternalRadio(), true, new
OnFrequencyPairingListener() {
                @Override
                public void onPairingResult(boolean isSuccess) {
                    FJXLog.INSTANCE.d(TAG, "autoPairing: 自动对频 = " +
isSuccess);
```

```
                }

                @Override
                public void onReceivePairingCmd() {
                    FJXLog.INSTANCE.d(TAG, "autoPairing: 对频码设置成功");
                }
            });
        }
    }
}
```

###

注意 当前只满足记录一个对频信息，在对频一个设备的时候，会将其他对频数据清除掉

```
@Override
public void savePairingCode(String code) {
    // 保存对频码到本地缓存
    ACache.system().put(ACacheKey.BS_PAIRING_CODE, code);
    ACache.system().put(ACacheKey.BS_TB_PAIRING_CODE, "");
}

@Override
public void saveOtherPairingCode(String hexcode) {
    //保存天宝的对频码
    ACache.system().put(ACacheKey.BS_TB_PAIRING_CODE, hexcode);
    ACache.system().put(ACacheKey.BS_PAIRING_CODE, "");
}
```