```
    typedef double XFLOAT;
    typedef double OTA_FLOAT;

namespace POLQAV2
{

XFLOAT LpqWeight (CPOLQAData            *POLQAHandle,
                  const int             pPowerSyllabe,
                  const int             pPowerTime,
                  const CDoubleArray    &pDisturbance,
                  const CDoubleArray    &pTimeWeight)
{
    XFLOAT      resultTime = 0;
    XFLOAT      totalTimeWeightTime = 0;

    const int stopFrameIdx = POLQAHandle->statics->stopFrameIdx;

    for (int startFrameOfSyllabe = POLQAHandle->statics->startFrameIdx;
         startFrameOfSyllabe <= stopFrameIdx;
         startFrameOfSyllabe += NUMBER_OF_PSQM_FRAMES_PER_SYLLABE/2)
    {
        XFLOAT  resultSyllabe = 0;
        int     countSyllabe = 0;

        for (int frameIndex = startFrameOfSyllabe;
            frameIndex < startFrameOfSyllabe + NUMBER_OF_PSQM_FRAMES_PER_SYLLABE &&
frameIndex <= stopFrameIdx;
            frameIndex++)
        {
            resultSyllabe +=  pow (pDisturbance.m_pData[frameIndex], pPowerSyllabe);
            countSyllabe++;
        }

        resultSyllabe /= countSyllabe;
        resultSyllabe = pow (resultSyllabe, (XFLOAT) 1.0 / (XFLOAT)pPowerSyllabe);

        resultTime +=  pow (pTimeWeight.m_pData[startFrameOfSyllabe] * resultSyllabe,
pPowerTime);
        totalTimeWeightTime += pow (pTimeWeight.m_pData[startFrameOfSyllabe],
pPowerTime);
    }

    resultTime /= totalTimeWeightTime;
    resultTime = pow (resultTime, (XFLOAT) 1.0 / (XFLOAT)pPowerTime);

    return (XFLOAT) resultTime;
}

XFLOAT LppqqWeight (CPOLQAData          *POLQAHandle,
                    const double            pPowerSyllabe,
                    const double            pPowerTime,
                    const CDoubleArray    &pDisturbance,
                    const CDoubleArray    &pTimeWeight)
{
    XFLOAT      resultTime = 0;
    XFLOAT      totalTimeWeightTime = 0;
    XFLOAT      hulp;

    const int stopFrameIdx = POLQAHandle->statics->stopFrameIdx;

    for (int startFrameOfSyllabe = POLQAHandle->statics->startFrameIdx;
         startFrameOfSyllabe <= stopFrameIdx;
         startFrameOfSyllabe += NUMBER_OF_PSQM_FRAMES_PER_SYLLABE/2)
    {
        XFLOAT  resultSyllabe = 0;
        int     countSyllabe = 0;

        for (int frameIndex = startFrameOfSyllabe;
            frameIndex < startFrameOfSyllabe + NUMBER_OF_PSQM_FRAMES_PER_SYLLABE &&
frameIndex <= stopFrameIdx;
            frameIndex++)
        {
            resultSyllabe +=  pow (pDisturbance.m_pData[frameIndex], pPowerSyllabe);
            countSyllabe++;
```

```
        }

        resultSyllabe /= countSyllabe;
        resultSyllabe = pow (resultSyllabe, (XFLOAT) 1.0 / (XFLOAT)pPowerSyllabe);

        hulp = pow(pTimeWeight.m_pData[startFrameOfSyllabe],20.0);
        resultTime +=  pow (hulp * resultSyllabe, pPowerTime);
        totalTimeWeightTime += pow (hulp, pPowerTime);
    }

    resultTime /= totalTimeWeightTime;
    resultTime = pow (resultTime, (XFLOAT) 1.0 / (XFLOAT)pPowerTime);

    return (XFLOAT) resultTime;
}

BOOL CPairParameters::DisturbanceTimeProcess (POLQA_RESULT_DATA*  pOverviewHolder)
{
    XFLOAT SwitchingLevelDeg = mpBGNSwitchingLevel[1];
    XFLOAT NoiseLevelSpeechDeg = mpNoiseDuringSpeechdB[1];
    XFLOAT NoiseLevelSilenceDeg = mpNoiseDuringSilencedB[1];
    XFLOAT NoiseLevelDifference = 1.0;
    if (NoiseLevelSpeechDeg<0) ;
    if (NoiseLevelSilenceDeg<0) ;
    if (SwitchingLevelDeg>9.0) SwitchingLevelDeg = 9.0;
    if (NoiseLevelDifference<1.0) NoiseLevelDifference = 1.0;
    if (NoiseLevelDifference>4.0) NoiseLevelDifference = 4.0;
    if (NoiseLevelSpeechDeg>0.0 && NoiseLevelSilenceDeg>0.0 &&
NoiseLevelSilenceDeg>NoiseLevelSpeechDeg) NoiseLevelDifference =
pow((NoiseLevelSilenceDeg-NoiseLevelSpeechDeg),0.2);

    XFLOAT disturbanceL [NUMBER_OF_POWERS_OVER_FREQ] [NUMBER_OF_POWERS_OVER_SYL]
[NUMBER_OF_POWERS_OVER_TIME];
    XFLOAT addedDisturbanceL [NUMBER_OF_POWERS_OVER_FREQ] [NUMBER_OF_POWERS_OVER_SYL]
[NUMBER_OF_POWERS_OVER_TIME];
    XFLOAT  log10aPureFrqLoudnessMean, hulpLevel1=0.0, hulpLevel2 ;
    XFLOAT crestFactorCompensation000;

    int             i, j, k;

    for (i = 0; i < NUMBER_OF_POWERS_OVER_FREQ; i++) {
        for (j = 0; j < NUMBER_OF_POWERS_OVER_SYL; j++) {
            for (k = 0; k < NUMBER_OF_POWERS_OVER_TIME; k++) {

                disturbanceL [i][j][k] =       LpqWeight (POLQAHandle, MINIMUM_POWER_SYL
+ STEP_POWER_SYL*j,
                                                          MINIMUM_POWER_TIME +
STEP_POWER_TIME*k,
                                                           aDisturbance [i],
aTimeWeight);

                addedDisturbanceL [i][j][k] =      LpqWeight (POLQAHandle,
MINIMUM_POWER_SYL + STEP_POWER_SYL*j,
                                                             MINIMUM_POWER_TIME +
STEP_POWER_TIME*k,
                                                              aAddedDisturbance [i],
aTimeWeight);

            }
        }
    }

    fractionNoTimeclip = 1.0;
    crestFactorCompensation000 = crestFactor;
    if (crestFactorCompensation000<80.0)    crestFactorCompensation000=80.0;
    if (crestFactorCompensation000>200.0)   crestFactorCompensation000=200.0;
    crestFactorCompensation000 = pow(crestFactorCompensation000,0.09)/1.6;

    XFLOAT n000 = LpqWeight(POLQAHandle, 1, 1, aAddedSilentDisturbance [0],
aTimeWeight);

    XFLOAT n000mosIntellCorrection = LppqqWeight(POLQAHandle, 0.9 , 1.4,
aAddedSilentDisturbance [0], aTimeWeight);
    XFLOAT n000hulp = n000;
    XFLOAT n000hulp2 = n000;
    XFLOAT n011 = fractionNoTimeclip*LpqWeight(POLQAHandle, 3,2, aAddedSilentDisturbance
```

```
[0], aTimeWeight);
    XFLOAT n120 = fractionNoTimeclip*LpqWeight(POLQAHandle, 5,1, aAddedSilentDisturbance
[1], aTimeWeight);
    XFLOAT n161 = fractionNoTimeclip*LpqWeight(POLQAHandle, 13,2,
aAddedSilentDisturbance [1], aTimeWeight);
    XFLOAT noiseDisturbanceCorrection = fractionNoTimeclip*LpqWeight(POLQAHandle, 1, 1,
aAddedSilentDisturbance [1], aTimeWeight);

    XFLOAT predictedMosOverall, predictedMosUnclipped = -1.0,
predictedCvcPureIntelligibility;
    XFLOAT predictedMosPureFrq, predictedMosPureNoise,
predictedMosOverallNearTransparent;
    XFLOAT HulpCvc, testMos1, testMos2, testMos3, testMos4, testMos5, newPolqaPlusMos;

    if (n000mosIntellCorrection<2.5) n000mosIntellCorrection = 2.5;

//POLQAMAIN PART 3
//NOISE in silent intervals compensation for addedDisturbanceL loud noises are reduced
in impact

    if (aListeningCondition==STANDARD_IRS) {
        if (n000<0.2) n000 = 0.2;
        hulpLevel1 = 0.45*pow((n000+1.0),0.015);
    }
    if (aListeningCondition==WIDE_H) {

        if (n000<0.2) n000 = 0.2;
        hulpLevel1 = 1.03*pow((n000+1.0),0.11);

    }
    if (aListeningCondition==NARROW_H) {
        if (n000<0.1) n000 = 0.1;
        hulpLevel1 = 0.45*pow((n000+1.0),0.05);
    }

//REVERB compensation for addedDisturbanceL
    HulpCvc = reverbIndicator;

    if (aListeningCondition==STANDARD_IRS) {
        if (HulpCvc<80.0) HulpCvc=80.0;
        HulpCvc = pow(HulpCvc,0.004);
    }
    if (aListeningCondition==NARROW_H) {
        if (HulpCvc<100.0) HulpCvc=100.0;
        HulpCvc = pow(HulpCvc,0.006);
    }
    if (aListeningCondition==WIDE_H){
        HulpCvc = 1.0;
    }

    for (i = 0; i < NUMBER_OF_POWERS_OVER_FREQ; i++) {
        for (j = 0; j < NUMBER_OF_POWERS_OVER_SYL; j++) {
            for (k = 0; k < NUMBER_OF_POWERS_OVER_TIME; k++) {

                disturbanceL [i][j][k] /= HulpCvc;
                addedDisturbanceL [i][j][k] /= pow(HulpCvc, 2);
                if (aListeningCondition==STANDARD_IRS) {
                    addedDisturbanceL [i][j][k] /= pow((n000+9.0),hulpLevel1);
                }
                if (aListeningCondition==WIDE_H) {

                    addedDisturbanceL [i][j][k] /= pow((n000+6.0),hulpLevel1);

                }
                if (aListeningCondition==NARROW_H) {
                    addedDisturbanceL [i][j][k] /= pow((n000+9.0),hulpLevel1);
                }

            }
        }
    }
//FREQUENCY SPURT AND TIME INTEGRATION
    const XFLOAT d0s3t3 = disturbanceL [0][3][3];
    const XFLOAT d0s1t3 = disturbanceL [0][1][3];
    const XFLOAT d0s2t0 = disturbanceL [0][2][0];
    const XFLOAT d0s2t1 = disturbanceL [0][2][1];
```

```
const XFLOAT d0s3t1 = disturbanceL [0][3][1];
const XFLOAT d0s6t6 = disturbanceL [0][6][6];
const XFLOAT d0s7t0 = disturbanceL [0][7][0];
const XFLOAT d0s7t1 = disturbanceL [0][7][1];
const XFLOAT d1s1t1 = disturbanceL [1][1][1];
const XFLOAT d1s1t2 = disturbanceL [1][1][2];
const XFLOAT d1s6t1 = disturbanceL [1][6][1];
const XFLOAT d1s7t1 = disturbanceL [1][7][1];
const XFLOAT d1s7t3 = disturbanceL [1][7][3];
const XFLOAT d2s0t1 = disturbanceL [2][0][1];
const XFLOAT d2s0t2 = disturbanceL [2][0][2];
const XFLOAT d2s0t3 = disturbanceL [2][0][3];
const XFLOAT d2s1t1 = disturbanceL [2][1][1];
const XFLOAT d2s1t2 = disturbanceL [2][1][2];
const XFLOAT d2s2t1 = disturbanceL [2][2][1];
const XFLOAT d2s5t1 = disturbanceL [2][5][1];
const XFLOAT d2s7t3 = disturbanceL [2][7][3];
const XFLOAT d3s0t2 = disturbanceL [3][0][2];
const XFLOAT d3s0t3 = disturbanceL [3][0][3];
const XFLOAT d3s1t2 = disturbanceL [3][1][2];
const XFLOAT d3s1t1 = disturbanceL [3][1][1];
const XFLOAT d3s2t2 = disturbanceL [3][2][2];
const XFLOAT d3s3t2 = disturbanceL [3][3][2];
const XFLOAT d3s4t2 = disturbanceL [3][4][2];
const XFLOAT d4s0t0 = disturbanceL [4][0][0];
const XFLOAT d4s2t0 = disturbanceL [4][2][0];
const XFLOAT d4s5t2 = disturbanceL [4][5][2];
const XFLOAT d4s7t3 = disturbanceL [4][7][3];
const XFLOAT d5s1t2 = disturbanceL [5][1][2];
const XFLOAT d5s0t3 = disturbanceL [5][0][3];
const XFLOAT d5s5t3 = disturbanceL [5][5][3];
const XFLOAT d5s6t1 = disturbanceL [5][6][1];

const XFLOAT a0s0t0 = addedDisturbanceL [0][0][0];
const XFLOAT a0s0t1 = addedDisturbanceL [0][0][1];
const XFLOAT a0s1t0 = addedDisturbanceL [0][1][0];
const XFLOAT a0s1t1 = addedDisturbanceL [0][1][1];
const XFLOAT a0s1t3 = addedDisturbanceL [0][1][3];
const XFLOAT a0s2t0 = addedDisturbanceL [0][2][0];
const XFLOAT a0s2t1 = addedDisturbanceL [0][2][1];
const XFLOAT a0s2t3 = addedDisturbanceL [0][2][3];
const XFLOAT a0s3t1 = addedDisturbanceL [0][3][1];
const XFLOAT a0s4t1 = addedDisturbanceL [0][4][1];
const XFLOAT a0s5t1 = addedDisturbanceL [0][5][1];
const XFLOAT a0s5t2 = addedDisturbanceL [0][5][2];
const XFLOAT a0s6t1 = addedDisturbanceL [0][6][1];
const XFLOAT a0s7t1 = addedDisturbanceL [0][7][1];
const XFLOAT a0s7t2 = addedDisturbanceL [0][7][2];
const XFLOAT a1s2t2 = addedDisturbanceL [1][2][2];
const XFLOAT a1s3t2 = addedDisturbanceL [1][3][2];
const XFLOAT a1s4t1 = addedDisturbanceL [1][4][1];
const XFLOAT a1s5t1 = addedDisturbanceL [1][5][1];
const XFLOAT a1s7t1 = addedDisturbanceL [1][7][1];
const XFLOAT a2s1t2 = addedDisturbanceL [2][1][2];
const XFLOAT a2s6t3 = addedDisturbanceL [2][6][3];
const XFLOAT a3s0t2 = addedDisturbanceL [3][0][2];
const XFLOAT a3s2t2 = addedDisturbanceL [3][2][2];
const XFLOAT a3s4t3 = addedDisturbanceL [3][4][3];
const XFLOAT a4s4t3 = addedDisturbanceL [4][4][3];
const XFLOAT a4s7t3 = addedDisturbanceL [5][7][3];

XFLOAT rawFrqInd = log10(aPureFrqLoudnessMean+0.0001);

aPureFrqLoudnessMean1 = log10(aPureFrqLoudnessMean+0.0001);
if (aPureFrqLoudnessMean1<1.0) aPureFrqLoudnessMean1 = 1.0;

aPureFrqLoudnessMean2 = log10(aPureFrqLoudnessMean+0.0001);

if (aPureFrqLoudnessMean2<3.0) aPureFrqLoudnessMean2 = 3.0;

aPureFrqLoudnessMean3 = log10(aPureFrqLoudnessMean+0.0001);
if (aPureFrqLoudnessMean3<3.0) aPureFrqLoudnessMean3 = 3.0;

log10aPureFrqLoudnessMean = log10(aPureFrqLoudnessMean+0.0001);
if (log10aPureFrqLoudnessMean<1.5) log10aPureFrqLoudnessMean = 1.5;
```

```
    if (aListeningCondition==WIDE_H) {
        predictedMosPureFrq = -0.8453*log10aPureFrqLoudnessMean + 6.0131 +0.7;
    } else {
        predictedMosPureFrq = -0.8066*log10aPureFrqLoudnessMean + 5.7362 +0.4;
    }

    if (predictedMosPureFrq < 0.8) predictedMosPureFrq = 0.8;
    if (predictedMosPureFrq > 5.0) predictedMosPureFrq = 5.0;

    distortedLoudnessTimbreLow /= 450.0;

    distortedLoudnessTimbreHigh /= 1.0e5;
    distortedLoudnessTimbreHighSilent /= 1.0e5;

    if (distortedLoudnessTimbreLow > 1.0)   distortedLoudnessTimbreLow = 1.0;

    XFLOAT hulp1, hulp2, hulpFrq;
    hulpFrq = aPureFrqLoudnessMean2;
    hulpFrq -= 3.9;
    if (hulpFrq<0.0) hulpFrq = 0.0;

    hulpFrq = -0.2*hulpFrq;

    newPolqaPlusMos = -0.601*d1s1t1 - a0s1t1;

//MAPPING TO INTERMEDIATE MOS SCORE

    if (aListeningCondition==WIDE_H) {

        hulp1 = d0s3t1;
        if (hulp1>45.0) hulp1 = 45.0;
        testMos2 =  - 0.508*aPureFrqLoudnessMean2 - 0.341*hulp1 - a0s0t1;
        if (testMos2<-30.0) testMos2 = -30.0;
        testMos2 = 0.0062*testMos2*testMos2 + 0.3504*testMos2 + 5.7406;
        testMos2 = 1.1857*testMos2 - 0.8029;
        if (testMos2<0.8) testMos2 = 0.8;
        if (testMos2>4.9) testMos2 = 4.9;

        hulp2 = a0s0t1;
        if (hulp2>4.0) hulp2 = 4.0;
        testMos3 =  - 0.508*aPureFrqLoudnessMean2 - 0.341*d0s3t1 - hulp2;
        if (testMos3<-30.0) testMos3 = -30.0;
        testMos3 = 0.0062*testMos3*testMos3 + 0.3504*testMos3 + 5.7406;
        testMos3 = 1.1857*testMos3 - 0.8029;
        if (testMos3>4.9) testMos3 = 4.9;

        hulp2 = a0s0t1;
        if (hulp2>6.0) hulp2 = 6.0;

        testMos4 =  - 0.46*aPureFrqLoudnessMean2 - 0.35*d0s3t1 - hulp2;

        if (testMos4<-30.0) testMos4 = -30.0;
        testMos4 = 0.0062*testMos4*testMos4 + 0.3504*testMos4 + 5.7406;
        testMos4 = 1.1857*testMos4 - 0.8029;
        if (testMos4<0.8) testMos4 = 0.8;
        if (testMos4>4.9) testMos4 = 4.9;

        hulp1 = d0s3t1;
        if (hulp1>50.0) hulp1 = 50.0;
        hulp2 = a0s0t1;
        if (hulp2>6.0) hulp2 = 6.0;
        testMos5 =  - 0.508*aPureFrqLoudnessMean2 - 0.341*hulp1 - hulp2;
        if (testMos5<-30.0) testMos5 = -30.0;
        testMos5 = 0.0062*testMos5*testMos5 + 0.3504*testMos5 + 5.7406;
        testMos5 = 1.1857*testMos5 - 0.8029;
        if (testMos5<0.8) testMos5 = 0.8;
        if (testMos5>4.9) testMos5 = 4.9;

        predictedMosOverall = testMos4;
if (!(predictedMosOverall==predictedMosOverall)) DebugBreak();

        predictedMosOverall = 0.9*predictedMosOverall;

        predictedMosOverall = 0.89*predictedMosOverall + 0.75;

        if (predictedMosOverall<0.8) predictedMosOverall = 0.8;
```

```
        if (predictedMosOverall>4.9) predictedMosOverall = 4.9;

        testMos1 = predictedMosOverall;
        testMos1 = testMos1 + 0.024*d2s0t2;
        testMos1 = 1.1913*testMos1 - 1.1039;
        if (testMos1<0.8) testMos1 = 0.8;
        if (testMos1>4.9) testMos1 = 4.9;

        testMos1 = predictedMosOverall;

        predictedMosOverall = 1.1913*predictedMosOverall - 1.1039;
        if (predictedMosOverall<0.8) predictedMosOverall = 0.8;
        if (predictedMosOverall>4.9) predictedMosOverall = 4.9;

    } else {
        hulp1 = d1s1t2;
        if (hulp1>45.0) hulp1 = 45.0;
        hulp2 = a0s2t1;
        if (hulp2>15.0) hulp2 = 15.0;
        predictedMosOverall = -distortedLoudnessTimbreLow  - 0.059*hulp1 - 0.073*hulp2 +
hulpFrq;

if(!(predictedMosOverall==predictedMosOverall)) DebugBreak();

        if (predictedMosOverall<-5.0) predictedMosOverall = -5.0;
        predictedMosOverall = 0.1441*predictedMosOverall*predictedMosOverall +
1.666*predictedMosOverall + 5.2762;
        if (predictedMosOverall<0.8) predictedMosOverall = 0.8;
        if (predictedMosOverall>4.7) predictedMosOverall = 4.7;

        testMos1 = -distortedLoudnessTimbreLow  - 0.059*d1s1t2 - 0.073*a0s2t1;
        if (testMos1<-5.0) testMos1 = -5.0;
        testMos1 = 0.1441*testMos1*testMos1 + 1.666*testMos1 + 5.2762;
        if (testMos1<0.8) testMos1 = 0.8;
        if (testMos1>4.7) testMos1 = 4.7;

        hulp1 = d1s1t2;
        if (hulp1>40.0) hulp1 = 40.0;
        hulp2 = a0s2t1;
        if (hulp2>15.0) hulp2 = 15.0;
        testMos2 = -distortedLoudnessTimbreLow  - 0.059*hulp1 - 0.073*hulp2;
        if (testMos2<-5.0) testMos2 = -5.0;
        testMos2 = 0.1441*testMos2*testMos2 + 1.666*testMos2 + 5.2762;
        if (testMos2<0.8) testMos2 = 0.8;
        if (testMos2>4.7) testMos2 = 4.7;

        hulp1 = d1s1t2;
        if (hulp1>45.0) hulp1 = 45.0;
        hulp2 = a0s2t1;
        if (hulp2>10.0) hulp2 = 10.0;
        testMos3 = -distortedLoudnessTimbreLow  - 0.059*hulp1 - 0.073*hulp2;
        if (testMos3<-5.0) testMos3 = -5.0;
        testMos3 = 0.1441*testMos3*testMos3 + 1.666*testMos3 + 5.2762;
        if (testMos3<0.8) testMos3 = 0.8;
        if (testMos3>4.7) testMos3 = 4.7;

        hulp1 = d1s1t2;
        if (hulp1>40.0) hulp1 = 40.0;
        hulp2 = a0s2t1;
        if (hulp2>10.0) hulp2 = 10.0;
        testMos4 = -distortedLoudnessTimbreLow  - 0.059*hulp1 - 0.073*hulp2;
        if (testMos4<-5.0) testMos4 = -5.0;
        testMos4 = 0.1441*testMos4*testMos4 + 1.666*testMos4 + 5.2762;
        if (testMos4<0.8) testMos4 = 0.8;
        if (testMos4>4.7) testMos4 = 4.7;

        hulp1 = d1s1t2;
        if (hulp1>50.0) hulp1 = 50.0;
        hulp2 = a0s2t1;
        if (hulp2>20.0) hulp2 = 20.0;
        testMos5 = -distortedLoudnessTimbreLow  - 0.059*hulp1 - 0.073*hulp2;
        if (testMos5<-5.0) testMos5 = -5.0;
        testMos5 = 0.1441*testMos5*testMos5 + 1.666*testMos5 + 5.2762;
        if (testMos5<0.8) testMos5 = 0.8;
        if (testMos5>4.7) testMos5 = 4.7;
```

```
    }

    XFLOAT hulpMos;
    hulpMos = predictedMosOverall;
    hulpLevel1 = predictedMosOverall;
    hulpLevel2 = predictedMosOverall;
    if (hulpLevel1<2.0) hulpLevel1 = 2.0;
    if (hulpLevel2<1.0) hulpLevel2 = 1.0;

    if (predictedMosOverall>4.0) predictedMosOverall -=
8.0*(globalScaleDistortedToFixedlevelHulp-1.0);

    if (aListeningCondition==STANDARD_IRS) {

            if (frameFlatnessDisturbanceAvgCompensation000silent > 0.98)
frameFlatnessDisturbanceAvgCompensation000silent = 0.98;

        predictedMosOverall -= correlationOriginalWithDisturbanceCompensation000ForMNRU;
        predictedMosOverall =
predictedMosOverall*frameFlatnessDisturbanceAvgCompensation000silent/frameFlatne
ssDisturbanceAvgCompensation000active;
        predictedMosOverall += 0.017*a4s7t3;
        predictedMosOverall /= CVCratioSNRlevelRangecompensation0001;
        predictedMosOverall -= envelopeContinuityCompensation000/6.0;
        predictedMosOverall += 0.01;
        if (predictedMosOverall>4.50) predictedMosOverall = 4.50;

    }

    if (aListeningCondition==NARROW_H) {
        predictedMosOverall += 0.016*d4s7t3;
        predictedMosOverall = 1.1348*predictedMosOverall - 0.9262;
        predictedMosOverall *= CVCratioSNRlevelRangecompensation0001;
    }

    if (aListeningCondition==WIDE_H) {

        predictedMosOverall -= correlationOriginalWithDisturbanceCompensation000ForMNRU;
        predictedMosOverall =
predictedMosOverall*frameFlatnessDisturbanceAvgCompensation000silent*CVCratioSNR
levelRangecompensation0001 - distortedLoudnessTimbreHighPerFrameAvgActive000 +
distortedLoudnessTimbrePerFrameLoudAvg000 -
frameFlatnessDistortedAvgCompensationSilent000 -
noiseIndicatorHighBandsCompensation000;
        predictedMosOverall /= frameCorrelationTimeDisturbanceAvgCompensation000;
        reverbIndicator -= 70.0;
        if (reverbIndicator<0.0) reverbIndicator = 0.0;
        if (reverbIndicator>60.0) reverbIndicator = 60.0;
        reverbIndicator /=
(correlationOriginalWithDisturbanceCompensation000ForReverb+0.5);
        predictedMosOverall += reverbIndicator/260.0;

        if (predictedMosOverall>4.75) predictedMosOverall = 4.75;
    }

    predictedMosOverall /= pow(aGlobalCompensation3,0.02);

    testMos3 /= pow(aGlobalCompensation3,0.2);

//Global MAPPING TO MOS-LQO

    predictedMosUnclipped = predictedMosOverall;

    if (aListeningCondition==STANDARD_IRS)
    {
        predictedMosOverall = -0.4 +
             0.8   *     predictedMosOverall +
             0.232 * pow(predictedMosOverall, 2) +
            -0.038 * pow(predictedMosOverall, 3);

        predictedMosOverall = (predictedMosOverall-1.0)*1.018849 + 1.0;

        predictedMosUnclipped = min(predictedMosOverall, 4.50);

        predictedMosOverall += 0.06;
        if (predictedMosOverall>4.50) predictedMosOverall = 4.50;
```

```
        if (predictedMosOverall<1.00) predictedMosOverall = 1.00;
    }
    if (aListeningCondition==WIDE_H)
    {
        predictedMosOverall =  0.40 +
             0.40   *     predictedMosOverall +
             0.29  * pow(predictedMosOverall, 2) +
            -0.038 * pow(predictedMosOverall, 3);

        predictedMosOverall = (predictedMosOverall-1.0)*0.994538 + 1.0;

        predictedMosUnclipped = min(predictedMosOverall, 4.75);

        predictedMosOverall += 0.03;
        if (predictedMosOverall > 4.75) predictedMosOverall = 4.75;
        if (predictedMosOverall < 1.00) predictedMosOverall = 1.00;
    }

    if (aListeningCondition == STANDARD_IRS)
    {

        predictedMosOverall = 0.79 + 0.0036*predictedMosOverall +
0.2117*predictedMosOverall*predictedMosOverall -
0.0065*pow(predictedMosOverall,3);

        if (predictedMosOverall < 1.00) predictedMosOverall = 1.00;
        if (predictedMosOverall > 4.50) predictedMosOverall = 4.50;

        predictedMosOverall =-0.25723 + 1.438911*predictedMosOverall -
0.215535*predictedMosOverall*predictedMosOverall +
0.029045*pow(predictedMosOverall,3);
        if (predictedMosOverall < 1.00) predictedMosOverall = 1.00;
        if (predictedMosOverall > 4.5) predictedMosOverall = 4.5;

    }

    if (aListeningCondition == WIDE_H)
    {
        predictedMosOverall = 0.276 + 0.7203*predictedMosOverall -
0.00756*predictedMosOverall*predictedMosOverall +
0.01141*pow(predictedMosOverall,3);

        if (predictedMosOverall < 1.00) predictedMosOverall = 1.00;

        predictedMosOverall += 0.04;

        if (predictedMosOverall > 4.75) predictedMosOverall = 4.75;

        predictedMosOverall = 1.193116 - 0.6471599*predictedMosOverall +
0.49272175*predictedMosOverall*predictedMosOverall -
0.0382774*pow(predictedMosOverall,3);

            predictedMosOverall = -0.11096 + 1.16833181*predictedMosOverall +
-0.0687258*predictedMosOverall*predictedMosOverall +
0.00826943*pow(predictedMosOverall,3);

        if (predictedMosOverall < 1.00) predictedMosOverall = 1.00;
        if (predictedMosOverall > 4.75) predictedMosOverall = 4.75;

    }

    XFLOAT predictedMosOverall_WIDE_H = predictedMosOverall;

    if (aListeningCondition==WIDE_H) {
        if (n120>37.0) n120 = 37.0;
        if (n120<19)
            predictedMosPureNoise = -0.0009*pow(n120, 3) + 0.0352*pow(n120, 2) -
0.494*n120 + 4.75;
        else
            predictedMosPureNoise = 0.0019*pow(n120, 2) - 0.1677*n120 + 4.52;
        predictedMosPureNoise *= pow(aGlobalCompensation3,0.6);
        if (predictedMosPureNoise < 1.0) predictedMosPureNoise = 1.0;
        if (predictedMosPureNoise > 4.75) predictedMosPureNoise = 4.75;
    } else {
        if (n120>40.0) n120 = 40.0;
        predictedMosPureNoise = 0.0016*pow(n120, 2) - 0.153*n120 + 4.50;
```

```
        predictedMosPureNoise *= pow(aGlobalCompensation3,0.6);
        if (predictedMosPureNoise < 1.0) predictedMosPureNoise = 1.0;
        if (predictedMosPureNoise > 3.50) predictedMosPureNoise =
1.25*predictedMosPureNoise - 0.875;
        if (predictedMosPureNoise > 4.50) predictedMosPureNoise = 4.50;
    }

        predictedMosOverallNearTransparent = predictedMosOverall -
0.574*aPureFrqLoudnessMean2;

    pOverviewHolder-> m_PredictedMos = predictedMosOverall;
    pOverviewHolder-> m_PredictedMosUnclipped = predictedMosUnclipped;
    pOverviewHolder-> m_PredictedMosPureFrq = predictedMosPureFrq;
    pOverviewHolder-> m_PredictedMosPureNoise = predictedMosPureNoise;

    aDistortedLoudnessMeanIndicator2 = aDistortedLoudnessMeanIndicator1;
    aDistortedLoudnessMeanIndicator3 = aDistortedLoudnessMeanIndicator1;
    aDistortedLoudnessMeanIndicator4 = aDistortedLoudnessMeanIndicator1;
    aDistortedLoudnessMeanIndicator5 = aDistortedLoudnessMeanIndicator1;
    aDistortedLoudnessMeanIndicator6 = aDistortedLoudnessMeanIndicator1;
    aDistortedLoudnessMeanIndicator7 = aDistortedLoudnessMeanIndicator1;
    aDistortedLoudnessMeanIndicator8 = aDistortedLoudnessMeanIndicator1;
    aDistortedLoudnessMeanIndicator9 = aDistortedLoudnessMeanIndicator1;
    if (aDistortedLoudnessMeanIndicator1<6.0) {
        aDistortedLoudnessMeanIndicator1 = pow( (7.0-aDistortedLoudnessMeanIndicator1),
2);
    } else {
        aDistortedLoudnessMeanIndicator1 = aDistortedLoudnessMeanIndicator1 - 5.0;
    }
    if (aDistortedLoudnessMeanIndicator2<6.0) {
        aDistortedLoudnessMeanIndicator2 = pow( (7.0-aDistortedLoudnessMeanIndicator2),
5);
    } else {
        aDistortedLoudnessMeanIndicator2 = pow( (aDistortedLoudnessMeanIndicator2-5.0),
2);
    }
    if (aDistortedLoudnessMeanIndicator3<6.0) {
        aDistortedLoudnessMeanIndicator3 = pow( (7.0-aDistortedLoudnessMeanIndicator3),
10.0);
    } else {
        aDistortedLoudnessMeanIndicator3 = pow( (aDistortedLoudnessMeanIndicator3-5.0),
5);
    }

    if (aDistortedLoudnessMeanIndicator4<4.0) {
        aDistortedLoudnessMeanIndicator4 = pow( (5.0-aDistortedLoudnessMeanIndicator4),
2);
    } else {
        aDistortedLoudnessMeanIndicator4 = aDistortedLoudnessMeanIndicator4 - 3.0;
    }
    if (aDistortedLoudnessMeanIndicator5<4.0) {
        aDistortedLoudnessMeanIndicator5 = pow( (5.0-aDistortedLoudnessMeanIndicator5),
5);
    } else {
        aDistortedLoudnessMeanIndicator5 = pow( (aDistortedLoudnessMeanIndicator5-3.0),
2);
    }
    if (aDistortedLoudnessMeanIndicator6<4.0) {
        aDistortedLoudnessMeanIndicator6 = pow( (5.0-aDistortedLoudnessMeanIndicator6),
10.0);
    } else {
        aDistortedLoudnessMeanIndicator6 = pow( (aDistortedLoudnessMeanIndicator6-3.0),
5);
    }

    if (aDistortedLoudnessMeanIndicator7<2.0) {
        aDistortedLoudnessMeanIndicator7 = pow( (3.0-aDistortedLoudnessMeanIndicator7),
2);
    } else {
        aDistortedLoudnessMeanIndicator7 = aDistortedLoudnessMeanIndicator7 - 2.0;
    }

    if (aDistortedLoudnessMeanIndicator8<2.0) {
        aDistortedLoudnessMeanIndicator8 = pow( (3.0-aDistortedLoudnessMeanIndicator8),
5);
    } else {
```

```
        aDistortedLoudnessMeanIndicator8 = pow( (aDistortedLoudnessMeanIndicator8-2.0),
2);
    }
    if (aDistortedLoudnessMeanIndicator9<2.0) {
        aDistortedLoudnessMeanIndicator9 = pow( (3.0-aDistortedLoudnessMeanIndicator9),
10.0);
    } else {
        aDistortedLoudnessMeanIndicator9 = pow( (aDistortedLoudnessMeanIndicator9-2.0),
5);
    }
    aDistortedLoudnessMeanIndicator3 = aGlobalCompensation1;
    aDistortedLoudnessMeanIndicator4 = aGlobalCompensation1;
    aDistortedLoudnessMeanIndicator5 = aGlobalCompensation1;
    aDistortedLoudnessMeanIndicator6 = aGlobalCompensation1;
    aDistortedLoudnessMeanIndicator7 = aGlobalCompensation1-1.5;
    aDistortedLoudnessMeanIndicator8 = aGlobalCompensation1-1.5;
    aDistortedLoudnessMeanIndicator9 = aGlobalCompensation1-1.5;
    if (aDistortedLoudnessMeanIndicator3<1.5) aDistortedLoudnessMeanIndicator3 = 1.5;
    if (aDistortedLoudnessMeanIndicator4<2.0) aDistortedLoudnessMeanIndicator4 = 2.0;
    aDistortedLoudnessMeanIndicator5 =
aDistortedLoudnessMeanIndicator3*aDistortedLoudnessMeanIndicator3;
    aDistortedLoudnessMeanIndicator6 =
aDistortedLoudnessMeanIndicator4*aDistortedLoudnessMeanIndicator4;
    if (aDistortedLoudnessMeanIndicator7<0.0) aDistortedLoudnessMeanIndicator7 = 0.0;
    if (aDistortedLoudnessMeanIndicator8<0.5) aDistortedLoudnessMeanIndicator8 = 0.5;
    constantDelayIndicator *= 1.04f;

    if (hulpKPNmobile == 0) {
      pOverviewHolder->aIndicator[0] = (XFLOAT) predictedMosOverall;
      pOverviewHolder->aIndicator[1] = (XFLOAT) pOverviewHolder->m_GlobalDelay;
      pOverviewHolder->aIndicator[2] = (XFLOAT) predictedMosOverall_WIDE_H;
    } else {
      pOverviewHolder->aIndicator[0] = (XFLOAT) predictedMosOverall;
      pOverviewHolder->aIndicator[1] = (XFLOAT) predictedMosPureFrq;
      pOverviewHolder->aIndicator[2] = (XFLOAT) predictedMosPureNoise;
    }
    pOverviewHolder->aIndicator[3] = (XFLOAT) aGlobalCompensation3;
    pOverviewHolder->aIndicator[4] = (XFLOAT) aPureFrqLoudnessMean2;
    pOverviewHolder->aIndicator[5] = (XFLOAT) envelopeContinuityCompensation000;
    pOverviewHolder->aIndicator[6] = (XFLOAT) predictedMosOverallNearTransparent;
    pOverviewHolder->aIndicator[7] = (XFLOAT) predictedMosPureFrq;
    pOverviewHolder->aIndicator[8] = (XFLOAT) reverbIndicator;

    pOverviewHolder->aIndicator[9]  = (XFLOAT) n000;
    pOverviewHolder->aIndicator[10] = (XFLOAT) n011;
    pOverviewHolder->aIndicator[11] = (XFLOAT) n000mosIntellCorrection;
    pOverviewHolder->aIndicator[12] = (XFLOAT) d0s3t1;
    pOverviewHolder->aIndicator[13] = (XFLOAT) predictedMosPureNoise;
    pOverviewHolder->aIndicator[14] = (XFLOAT) pow(constantDelayIndicator, 0.25f);
    pOverviewHolder->aIndicator[15] = (XFLOAT) constantDelayIndicator;
    pOverviewHolder->aIndicator[16] = (XFLOAT)
frameFlatnessDisturbanceAvgCompensation000silent;
    pOverviewHolder->aIndicator[17] = (XFLOAT)
distortedLoudnessTimbreHighPerFrameAvgActive;
    pOverviewHolder->aIndicator[18] = (XFLOAT)
distortedLoudnessTimbreHighPerFrameAvgActive000;
    pOverviewHolder->aIndicator[19] = (XFLOAT)
frameFlatnessDistortedAvgCompensationSilent000;
    pOverviewHolder->aIndicator[20] = (XFLOAT) envelopeContinuityCompensation000;
    pOverviewHolder->aIndicator[21] = (XFLOAT) CVCratioSNRlevelRangecompensation0001;
    pOverviewHolder->aIndicator[22] = (XFLOAT) SNRloudnessRatioOKratio;
    pOverviewHolder->aIndicator[23] = (XFLOAT) crestFactor;

    pOverviewHolder->aIndicator[24] = (XFLOAT) avgPitchLoudFramesRise000;
    pOverviewHolder->aIndicator[25] = (XFLOAT) avgPitchLoudFramesDrop000;
    pOverviewHolder->aIndicator[26] = (XFLOAT) avgPitchLoudFrames000;
    pOverviewHolder->aIndicator[27] = (XFLOAT)
frameCorrelationTimeDisturbanceAvgCompensation000;
    pOverviewHolder->aIndicator[28] = (XFLOAT)
frameCorrelationTimeDisturbanceAvgCompensation000silent;
    pOverviewHolder->aIndicator[29] = (XFLOAT)
frameFlatnessDisturbanceAvgCompensation000active;

    pOverviewHolder->aIndicator[30] = (XFLOAT) fraction_of_sections_inserted;
    pOverviewHolder->aIndicator[31] = (XFLOAT) fraction_of_sections_critical;
    pOverviewHolder->aIndicator[32] = (XFLOAT) fraction_of_sections_invalid;
```

```
    pOverviewHolder->aIndicator[33] = (XFLOAT) globalScaleDistortedToFixedlevelHulp;

    ASSERT (NUMBER_OF_POWER_INDICATORS == 4);
    pOverviewHolder->aIndicator[NUMBER_OF_PREDICTION_INDICATORS + 0] = (XFLOAT) (10 *
log10 (1e-8 + aAvgDistortedPower/1e7));
    pOverviewHolder->aIndicator[NUMBER_OF_PREDICTION_INDICATORS + 1] = (XFLOAT) (10 *
log10 (1e-8 + aAvgOriginalPower/1e7));
    pOverviewHolder->aIndicator[NUMBER_OF_PREDICTION_INDICATORS + 2] = (XFLOAT) (10 *
log10 (1e-8 + aAvgActiveDistortedPower/1e7));
    pOverviewHolder->aIndicator[NUMBER_OF_PREDICTION_INDICATORS + 3] = (XFLOAT) (10 *
log10 (1e-8 + aAvgActiveOriginalPower/1e7));

    for (i = 0; i < NUMBER_OF_POWERS_OVER_FREQ; i++) {
        for (j = 0; j < NUMBER_OF_POWERS_OVER_SYL; j++) {
            for (k = 0; k < NUMBER_OF_POWERS_OVER_TIME; k++) {
                pOverviewHolder->aIndicator[NUMBER_OF_POWER_INDICATORS +
NUMBER_OF_PREDICTION_INDICATORS + (i*NUMBER_OF_POWERS_OVER_SYL +
j)*NUMBER_OF_POWERS_OVER_TIME + k]
                                            = disturbanceL [i][j][k];
                pOverviewHolder->aIndicator[NUMBER_OF_POWER_INDICATORS +
NUMBER_OF_PREDICTION_INDICATORS + ((NUMBER_OF_POWERS_OVER_FREQ +
i)*NUMBER_OF_POWERS_OVER_SYL + j)*NUMBER_OF_POWERS_OVER_TIME + k]
                                            = addedDisturbanceL [i][j][k];

            }
        }
    }

    return TRUE;
}

}
```