AmbiMusic代码分析

源代码：

ebony\_2k14\_msd85x0qv\Supernova\projects\msrv\common\inc\MSrv\_Ambilight.h

ebony\_2k14\_msd85x0qv\Supernova\projects\msrv\common\inc\MSrv\_Ambilight\_tpv.h

ebony\_2k14\_msd85x0qv\Supernova\projects\msrv\common\src\MSrv\_Ambilight.cpp

ebony\_2k14\_msd85x0qv\Supernova\projects\msrv\common\src\MSrv\_Ambilight\_tpv.cpp

Ambi系列功能的工作线程位于MSrv\_Ambilight.cpp文件的MonitorThreadFunc函数，涉及AmbiMusic功能的主要代码注释如下：

void \*MSrv\_Ambilight::**MonitorThreadFunc**(void \*arg)

{

AMBI\_MODE\_T enLastAmbiMode = AMBI\_M\_NONE;

AMBI\_MODE\_T enCurrAmbiMode = AMBI\_M\_NONE;

long lDelayTime = 0;

while(bThreadLoop)

{

if(enPowerOnMode == AMBI\_P\_ALARM)

{

enCurrAmbiMode = AMBI\_M\_MUSIC;

}

else if(bBlackLight)

{

if(enLastAmbiMode != AMBI\_M\_BLACK)

{

bSetAmbilightHardware(0,0,0);

enLastAmbiMode = enCurrAmbiMode = AMBI\_M\_BLACK;

lDelayTime = 0;

}

usleep(1);

continue;

}

else if(bAndroidControl)

{

if(enLastAmbiMode != AMBI\_M\_ANDROID)

{

//bSetAmbilightHardware(0,0,0);

enLastAmbiMode = enCurrAmbiMode = AMBI\_M\_ANDROID;

lDelayTime = 0;

}

usleep(1);

continue;

}

else if(TPV\_GetToggleAmbistart())

{

enCurrAmbiMode = AMBI\_M\_START;

}

else if(TPV\_GetSphereModeOnOff())

{

enCurrAmbiMode = AMBI\_M\_SPHERE;

}

else if(TPV\_GetAmbiphoto())

{

enCurrAmbiMode = AMBI\_M\_PHOTO;

}

else if(TPV\_GetToggleAmbiweather())

{

enCurrAmbiMode = AMBI\_M\_WEATHER;

}

else if(TPV\_GetToggleAmbimusic())

{

enCurrAmbiMode = AMBI\_M\_MUSIC;

}

else if(TPV\_GetToggleAmbilight())

{

enCurrAmbiMode = AMBI\_M\_LIGHT;

}

else

{

if(enLastAmbiMode != AMBI\_M\_NONE)

{

bSetAmbilightHardware(0,0,0);

enLastAmbiMode = enCurrAmbiMode = AMBI\_M\_NONE;

lDelayTime = 0;

}

usleep(1);

continue;

}

if(enCurrAmbiMode==AMBI\_M\_LIGHT)

{

static EN\_AMBI\_LIGHT\_COLOUR\_TYPE enLastAmbiColor = AMBI\_L\_C\_MAX;

static EN\_AMBI\_LIGHT\_COLOUR\_TYPE enCurrAmbiColor = AMBI\_L\_C\_MAX;

enCurrAmbiColor = static\_cast<EN\_AMBI\_LIGHT\_COLOUR\_TYPE>(TPV\_GetColourMode());

if(enLastAmbiMode != enCurrAmbiMode ||

(enLastAmbiColor != enCurrAmbiColor && enCurrAmbiColor==AMBI\_L\_C\_DYNAMIC))

{

// set all window to black

memset(ui4ScreenRGBValue,0,MS\_SCREEN\_WINDOW\*AMBI\_COLOR\_MAX\*sizeof(ui4ScreenRGBValue[0][0]));

}

enLastAmbiColor = enCurrAmbiColor;

}

lDelayTime += 40L\*1000L; // 40 ms

***工作线程的默认周期为40毫秒，也就是每40毫秒改变一次灯光颜色。***

switch(enCurrAmbiMode)

{

case AMBI\_M\_START:

TPV\_ExcuteAmbistart();

break;

case AMBI\_M\_WEATHER:

TPV\_ExcuteAmbiweather();

break;

case AMBI\_M\_PHOTO:

TPV\_ExcuteAmbiphoto();

break;

case AMBI\_M\_SPHERE:

TPV\_SetAmbiSphere();

break;

case AMBI\_M\_MUSIC:

unsigned long ui4BandPeakValue[AMBI\_AUDIO\_MAX][MS\_AUDIO\_BAND];

bool bAudioSignal;

MAPI\_AUDIO\_PROCESSOR\_TYPE targetProcessor;

MSAPI\_AUDIO\_OUTPORT\_SOURCE\_INFO pSourceInfo;

AUDIO\_VOL\_SOURCE\_TYPE volSrcType;

targetProcessor = ((TRUE == MSrv\_Control::GetInstance()->IsFocusOnSubSource() )? AUDIO\_PROCESSOR\_SUB : AUDIO\_PROCESSOR\_MAIN);

mapi\_interface::Get\_mapi\_audio()->GetOutputSourceInfo(&pSourceInfo);

if (pSourceInfo.SpeakerOut == targetProcessor)

volSrcType = VOL\_SOURCE\_SPEAKER\_OUT;

else

volSrcType = VOL\_SOURCE\_HP\_OUT;

if(MSrv\_Control::GetInstance()->GetMuteFlag()==MAPI\_FALSE &&

MSrv\_Control::GetMSrvSSSound()->GetAudioVolume(volSrcType) > 0)

{

bAudioSignal = true;

short i2\_chIdx;

for(i2\_chIdx = 0 ; i2\_chIdx < MS\_AUDIO\_BAND ; ++i2\_chIdx)

{

ui4BandPeakValue[AMBI\_AUDIO\_L][i2\_chIdx] = mapi\_interface::Get\_mapi\_audio()->SND\_GetParam(Sound\_GET\_PARAM\_Ambi\_Lpeak\_, i2\_chIdx);

ui4BandPeakValue[AMBI\_AUDIO\_R][i2\_chIdx] = mapi\_interface::Get\_mapi\_audio()->SND\_GetParam(Sound\_GET\_PARAM\_Ambi\_Rpeak\_, i2\_chIdx);

}

lDelayTime -= 800L; // 0.8 ms

***读取左右两个声道各11个频段的频率还是振幅值，后面会用来计算灯光颜色。估计耗时0.8毫秒，因此将lDelayTime的值减去0.8。***

}

else

{

bAudioSignal = false;

}

TPV\_SetAmbilightAudiotoTPV(bAudioSignal,ui4BandPeakValue);

***计算AmbiMusic颜色，注意这个函数只是将颜色值保存在一个全局变量，还没有真正去改变灯光颜色。***

break;

case AMBI\_M\_LIGHT:

unsigned long ui4ScreenRGBValueLocal[MS\_SCREEN\_WINDOW][AMBI\_COLOR\_MAX];

if(bHasVideo==false)

{

lDelayTime = 0;

usleep(1);

continue;

}

//if(bHasVideo)

{

memcpy(ui4ScreenRGBValueLocal,ui4ScreenRGBValue,MS\_SCREEN\_WINDOW\*AMBI\_COLOR\_MAX\*sizeof(ui4ScreenRGBValueLocal[0][0]));

}

TPV\_SetAmbilightVideotoTPV(true,ui4ScreenRGBValueLocal);

break;

default:

break;

}

unsigned char uiAmbilightRGB[4][AMBI\_COLOR\_MAX];

bool bSetColorNow = TPV\_GetAmbilightValuefromTPV(uiAmbilightRGB);

***从全局变量中读取灯光颜色。***

if(bSetColorNow || enLastAmbiMode != enCurrAmbiMode)

{

bSetAmbilightHardware(uiAmbilightRGB);

if(wAmbiModuleCount==2)

lDelayTime -= 2L \* 1000L; // 2 ms

else

lDelayTime -= 4L \* 1000L; // 4 ms

***硬件设置灯光颜色，估计设置每排灯带耗时1毫秒，不同机种有2排或4排灯带，因此将lDelayTime的值减去2或4。***

}

enLastAmbiMode = enCurrAmbiMode;

if(lDelayTime > 0)

{

usleep(lDelayTime);

lDelayTime = 0;

}

***睡眠等待，睡眠时间为40毫秒减去之前读取声音数据和设置灯带颜色的耗时，确保两次设置灯光的间隔在40毫秒。***

}

pthread\_exit(0);

}

计算AmbiMusic颜色位于MSrv\_Ambilight\_tpv.cpp文件的AmbimusicControl函数，主要代码注释如下：

#if CHIP\_PANASONIC

void AmbimusicControl( unsigned long ui4BandPeakValue[2][16] )

#else

void **AmbimusicControl**( unsigned long ui4BandPeakValue[2][BANDNUM] )

#endif

{

static int iCnt = 180;

unsigned char ui\_LeftMaxBandIdx,ui\_RightMaxBandIdx;

unsigned char bLeftSwitchEnable, bRightSwitchEnable;

PSSCHANNELINFO pssAmbiMusicInfo = &ssAmbiMusicInfo;

PSSAMBIMUSICSETTING pssAmbiMusicSetting = &ssAmbiMusicSetting;

PSSBANDINFO pssLeftBand = pssAmbiMusicInfo->ssLeftBand;

PSSBANDINFO pssRightBand = pssAmbiMusicInfo->ssRightBand;

PSSHSBFORMAT pssLeftHSBClr = pssAmbiMusicInfo->ssLeftHSBClr;

PSSHSBFORMAT pssRightHSBClr = pssAmbiMusicInfo->ssRightHSBClr;

SSRGBFORMAT ssLeftLedClr = { 30, 30, 30 }, ssRightLedClr = { 30, 30, 30 };

\*pssLeftBand = pssAmbiMusicInfo->ssDefBand;

\*pssRightBand = pssAmbiMusicInfo->ssDefBand;

#if CHIP\_PANASONIC || CHIP\_MSTAR\_818

ChangeBandvalue(ui4BandPeakValue);

#endif

// Get band peak form audio chip

ScaleBandPeak( pssAmbiMusicInfo, ui4BandPeakValue, iCnt);

// Sorting Band Peak

ui\_LeftMaxBandIdx = SortingBandwithFrequency( pssLeftBand , pssAmbiMusicSetting->bSortingFreqMode );

ui\_RightMaxBandIdx = SortingBandwithFrequency( pssRightBand, pssAmbiMusicSetting->bSortingFreqMode );

if(ui\_LeftMaxBandIdx>0)

--ui\_LeftMaxBandIdx;

if(ui\_RightMaxBandIdx>0)

--ui\_RightMaxBandIdx;

// AdjustHue by left or right side band of max band

AdjustHue(pssLeftBand ,ui4BandPeakValue[0],ui\_LeftMaxBandIdx );

AdjustHue(pssRightBand,ui4BandPeakValue[1],ui\_RightMaxBandIdx);

// Change band peak by threshold (MUSIC\_VOLUME\_MAX and MUSIC\_VOLUME\_MIN)

pssLeftBand ->sPeak = (short)MappingCurveProc(ui4BandPeakValue[0][ui\_LeftMaxBandIdx ]);

pssRightBand->sPeak = (short)MappingCurveProc(ui4BandPeakValue[1][ui\_RightMaxBandIdx]);

//pssLeftBand ->sPeak = (short)MappingCurveProc(ui4BandPeakValue[0][ui\_LeftMaxBandIdx ] , pssAmbiMusicInfo->iThreshold /\*, iGain\*/ );

//pssRightBand->sPeak = (short)MappingCurveProc(ui4BandPeakValue[1][ui\_RightMaxBandIdx] , pssAmbiMusicInfo->iThreshold /\*, iGain\*/ );

// Check Peak is or not over SwitchOff value

bLeftSwitchEnable = (pssLeftBand->sPeak > pssAmbiMusicSetting->bSwitchOff)?0:1;

bRightSwitchEnable = (pssRightBand->sPeak > pssAmbiMusicSetting->bSwitchOff)?0:1;

// Mapping Peak to HSB color domain

MappingPeaktoHSBCLR( pssLeftHSBClr, pssLeftBand, pssAmbiMusicSetting->bCoorAxisMode, bLeftSwitchEnable );

MappingPeaktoHSBCLR( pssRightHSBClr, pssRightBand, pssAmbiMusicSetting->bCoorAxisMode, bRightSwitchEnable );

// Limiting step of Color domain

AdjustingColorStep( pssLeftHSBClr, pssAmbiMusicSetting->iUpdateHueThre, pssAmbiMusicSetting->iUpdateBriThre, pssAmbiMusicSetting->iUpdateSatThre );

AdjustingColorStep( pssRightHSBClr, pssAmbiMusicSetting->iUpdateHueThre, pssAmbiMusicSetting->iUpdateBriThre, pssAmbiMusicSetting->iUpdateSatThre );

// Mapping HSB color domain to RGB domain

MappingHSBtoRGB( &ssLeftLedClr, (pssLeftHSBClr+1) );

MappingHSBtoRGB( &ssRightLedClr, (pssRightHSBClr+1) );

// Adjust Brightness

AmbimusicAdjustBrighness( &ssLeftLedClr, (pssLeftHSBClr+1) );

AmbimusicAdjustBrighness( &ssRightLedClr, (pssRightHSBClr+1) );

// Update AmbiLed

UpdateAmbiLed( &ssLeftLedClr, &ssRightLedClr);

***这是关键部分，将AmbiMusic颜色计算结果拷贝到一个全局变量，如果希望得到AmbiMusic颜色也是从这个全局变量中读取。***

***前面计算颜色的代码没有仔细看，后面是打印调试信息并不重要。***

if ( bOpenDebugMsg && !iCnt )

{

iCnt = 180;

AmbiPrint("Max Idx,Peak L:%d,%d R:%d,%d\n",

ui\_LeftMaxBandIdx,pssLeftBand->sPeak,ui\_RightMaxBandIdx,pssRightBand->sPeak);

AmbiPrint( "HSB L:%03d,%03d,%03d R:%03d,%03d,%03d\n",

pssLeftBand->ssColor.sHue, pssLeftBand->ssColor.bSaturation, pssLeftBand->ssColor.bBrightness,

pssRightBand->ssColor.sHue, pssRightBand->ssColor.bSaturation, pssRightBand->ssColor.bBrightness );

AmbiPrint( "HSB L:%03d,%03d,%03d R:%03d,%03d,%03d\n",

(pssLeftHSBClr+1)->sHue, (pssLeftHSBClr+1)->bSaturation, (pssLeftHSBClr+1)->bBrightness,

(pssRightHSBClr+1)->sHue, (pssRightHSBClr+1)->bSaturation, (pssRightHSBClr+1)->bBrightness );

AmbiPrint( "RGB L:%03d,%03d,%03d R:%03d,%03d,%03d\n",

ssLeftLedClr.bRed ,ssLeftLedClr.bGreen ,ssLeftLedClr.bBlue,

ssRightLedClr.bRed ,ssRightLedClr.bGreen ,ssRightLedClr.bBlue );

}

iCnt--;

}

#endif

UpdateAmbiLed函数注释如下：

void **UpdateAmbiLed**( PSSRGBFORMAT pssLeftLedClr, PSSRGBFORMAT pssRightLedClr)

{

unsigned char ui\_Window[AMBILIGHT\_MAX\_MODULE][AMBILIGHT\_ALL\_COLOR];

unsigned char ui\_idx;

#if (AMBILIGHT\_MAX\_MODULE!=16)

unsigned char ui\_ModuleMidIdx = uiAmbilightModuleCount/2;

#endif

#if (AMBILIGHT\_MAX\_MODULE == 16)

unsigned short ui2\_Red = pssLeftLedClr->bRed;

unsigned short ui2\_Green = pssLeftLedClr->bGreen;

unsigned short ui2\_Blue = pssLeftLedClr->bBlue;

ui2\_Red = (ui2\_Red + pssRightLedClr->bRed)/2;

ui2\_Green = (ui2\_Green + pssRightLedClr->bGreen)/2;

ui2\_Blue = (ui2\_Blue + pssRightLedClr->bBlue)/2;

for(ui\_idx=0;ui\_idx<4;++ui\_idx)

{

ui\_Window[ui\_idx][COLOR\_IDX\_R] = pssLeftLedClr->bRed;

ui\_Window[ui\_idx][COLOR\_IDX\_G] = pssLeftLedClr->bGreen;

ui\_Window[ui\_idx][COLOR\_IDX\_B] = pssLeftLedClr->bBlue;

}

for(ui\_idx=4;ui\_idx<12;++ui\_idx)

{

ui\_Window[ui\_idx][COLOR\_IDX\_R] = ui2\_Red;

ui\_Window[ui\_idx][COLOR\_IDX\_G] = ui2\_Green ;

ui\_Window[ui\_idx][COLOR\_IDX\_B] = ui2\_Blue;

}

for(ui\_idx=12;ui\_idx<AMBILIGHT\_MAX\_MODULE;++ui\_idx)

{

ui\_Window[ui\_idx][COLOR\_IDX\_R] = pssRightLedClr->bRed;

ui\_Window[ui\_idx][COLOR\_IDX\_G] = pssRightLedClr->bGreen;

ui\_Window[ui\_idx][COLOR\_IDX\_B] = pssRightLedClr->bBlue;

}

#else

if(uiAmbilightModuleCount==4)

{

unsigned short ui2\_Red = pssLeftLedClr->bRed;

unsigned short ui2\_Green = pssLeftLedClr->bGreen;

unsigned short ui2\_Blue = pssLeftLedClr->bBlue;

ui2\_Red = (ui2\_Red + pssRightLedClr->bRed)/2;

ui2\_Green = (ui2\_Green + pssRightLedClr->bGreen)/2;

ui2\_Blue = (ui2\_Blue + pssRightLedClr->bBlue)/2;

ui\_Window[0][COLOR\_IDX\_R] = pssLeftLedClr->bRed;

ui\_Window[0][COLOR\_IDX\_G] = pssLeftLedClr->bGreen;

ui\_Window[0][COLOR\_IDX\_B] = pssLeftLedClr->bBlue;

ui\_Window[1][COLOR\_IDX\_R] = ui2\_Red;

ui\_Window[1][COLOR\_IDX\_G] = ui2\_Green;

ui\_Window[1][COLOR\_IDX\_B] = ui2\_Blue;

ui\_Window[2][COLOR\_IDX\_R] = ui2\_Red;

ui\_Window[2][COLOR\_IDX\_G] = ui2\_Green;

ui\_Window[2][COLOR\_IDX\_B] = ui2\_Blue;

ui\_Window[3][COLOR\_IDX\_R] = pssRightLedClr->bRed;

ui\_Window[3][COLOR\_IDX\_G] = pssRightLedClr->bGreen;

ui\_Window[3][COLOR\_IDX\_B] = pssRightLedClr->bBlue;

}

else

{

for(ui\_idx=0;ui\_idx<ui\_ModuleMidIdx;++ui\_idx)

{

ui\_Window[ui\_idx][COLOR\_IDX\_R] = pssLeftLedClr->bRed;

ui\_Window[ui\_idx][COLOR\_IDX\_G] = pssLeftLedClr->bGreen;

ui\_Window[ui\_idx][COLOR\_IDX\_B] = pssLeftLedClr->bBlue;

}

for(ui\_idx=ui\_ModuleMidIdx;ui\_idx<uiAmbilightModuleCount;++ui\_idx)

{

ui\_Window[ui\_idx][COLOR\_IDX\_R] = pssRightLedClr->bRed;

ui\_Window[ui\_idx][COLOR\_IDX\_G] = pssRightLedClr->bGreen;

ui\_Window[ui\_idx][COLOR\_IDX\_B] = pssRightLedClr->bBlue;

}

}

#endif

AmbilightSetByRGB(ui\_Window,false);

***前面的代码是设置AmbiMusic灯带颜色，简单说就是左侧灯带用左声道的颜色，右侧灯带用右声道的颜色，上方的灯带用两个声道颜色的平均值。***

***这个函数将再次调整AmbiMusic颜色值，并拷贝到全局变量。***

}

AmbilightSetByRGB函数注释如下：

void **AmbilightSetByRGB**(unsigned char ui\_Window[AMBILIGHT\_MAX\_MODULE][AMBILIGHT\_ALL\_COLOR],bool bSlowlyChange)

{

unsigned char ui\_idx;

unsigned char ui\_idx2;

if (bSlowlyChange==true && bSetColorImmediately==false)

{

for (ui\_idx=0; ui\_idx<uiAmbilightModuleCount; ++ui\_idx)

{

AmbilightChangeSlowly(ui\_Window[ui\_idx],ui\_LastAmbilightRGB[ui\_idx]);

***这个函数会再次计算AmbiMusc颜色，保证颜色过渡平滑，即计算这次和上次颜色每个分量的差值，如果某个颜色分量差值超过AMBILIGHT\_CHANGE\_MAX\_STEP则调整此分量使之更接近上次的颜色。***

}

}

bSetColorImmediately = false;

#if 1

for (ui\_idx2=0 ; ui\_idx2<uiAmbilightModuleCount ; ++ui\_idx2)

{

for (ui\_idx=0 ; ui\_idx<AMBILIGHT\_ALL\_COLOR ; ++ui\_idx)

{

short i2\_Offset;

i2\_Offset = stAmbiSetting.uiOffset\_LED[ui\_idx2][ui\_idx];

i2\_Offset += stAmbiSetting.iSubBrightness;

if (i2\_Offset > 255)

i2\_Offset = 255;

else if (i2\_Offset < 0)

i2\_Offset = 0;

ui\_Window[ui\_idx2][ui\_idx] = ChangeValueScale\_TPV(255,ui\_Window[ui\_idx2][ui\_idx],0,i2\_Offset,0);

***再次调整颜色值，因为前面计算出的颜色值范围是0-255，但电视未必能支持如此大的范围，因此将颜色从0-255范围线性映射到0-i2\_Offset范围。从字面上看这个步骤应该是跟工厂菜单有关。***

}

}

#endif

#if SIMULATE\_ON\_WINDOWS

#if AMBILIGHT\_MAX\_MODULE>=16

if (uiAmbilightModuleCount==16)

{

AmbiPrint("SetByRGB %d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\n",

ui\_Window[0][0],ui\_Window[0][1],ui\_Window[0][2],

ui\_Window[1][0],ui\_Window[1][1],ui\_Window[1][2],

ui\_Window[2][0],ui\_Window[2][1],ui\_Window[2][2],

ui\_Window[3][0],ui\_Window[3][1],ui\_Window[3][2]);

AmbiPrint("%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\n",

ui\_Window[4][0],ui\_Window[4][1],ui\_Window[4][2],

ui\_Window[5][0],ui\_Window[5][1],ui\_Window[5][2],

ui\_Window[6][0],ui\_Window[6][1],ui\_Window[6][2],

ui\_Window[7][0],ui\_Window[7][1],ui\_Window[7][2]);

AmbiPrint("%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\n",

ui\_Window[8][0],ui\_Window[8][1],ui\_Window[8][2],

ui\_Window[9][0],ui\_Window[9][1],ui\_Window[9][2],

ui\_Window[10][0],ui\_Window[10][1],ui\_Window[10][2],

ui\_Window[11][0],ui\_Window[11][1],ui\_Window[11][2]);

AmbiPrint("%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\n",

ui\_Window[12][0],ui\_Window[12][1],ui\_Window[12][2],

ui\_Window[13][0],ui\_Window[13][1],ui\_Window[13][2],

ui\_Window[14][0],ui\_Window[14][1],ui\_Window[14][2],

ui\_Window[15][0],ui\_Window[15][1],ui\_Window[15][2]);

}

else

#endif

#if AMBILIGHT\_MAX\_MODULE>=2

if (uiAmbilightModuleCount==2)

{

AmbiPrint("SetByRGB %d,%d,%d,%d,%d,%d\n",

ui\_Window[0][0],ui\_Window[0][1],ui\_Window[0][2],

ui\_Window[1][0],ui\_Window[1][1],ui\_Window[1][2]);

}

#endif

#endif

memcpy(ui\_CurrAmbilightRGB ,ui\_Window ,uiAmbilightModuleCount\*AMBILIGHT\_ALL\_COLOR\*sizeof(ui\_Window[0][0]));

***最终将AmbiMusic颜色值拷贝到ui\_CurrAmbilightRGB全局变量中保存，等到需要真正设置灯光颜色时再从此全局变量中读取颜色。***

}