AN12562

RT 系列上H.264视频解码的开发

Rev. 0 — 28 August, 2019

应用笔记

内容

1 介绍

本应用笔记介绍了如何使用NXP i.MX RT1050处理器开发一个H.264视频解码应用程序。

对于此类应用，i.MX RT1050从microSD卡读取H.264视频源，然后调用FFMPEG库对视频源进行解码并生成YUV数据。通过像素处理管道（PXP）对YUV数据进行缩放和颜色空间转换后，在LCD显示屏上显示。

i.MX RT1050是具有单Arm Cortex-M7内核的处理器，其运行频率高达600 MHz。强大的处理能力和实时功能，以及丰富外围设备集成使i.MX RT1050成为许多应用的理想选择，例如工业计算，电机控制，电源转换，智能消费类产品，高端音频系统，家庭和楼宇自动化。

Section 2 介绍了演示应用程序的硬件和软件平台。Section 3 介绍了基于FFMPEG 库和 i.MX RT1050 Software Development Kit (SDK).使用 i.MX RT1050开发 H.264 视频解码应用程序的过程。

2 硬件和软件平台

本节简要介绍了演示应用程序的硬件和软件平台。

2.1 i.MX RT1050 处理器

NXP i.MX RT1050交叉处理器具有高达600 MHz的Arm Cortex-M7单核。它具有512 KB的片上RAM，可以灵活配置作为内核紧密耦合内存（TCM）或通用RAM。它提供连接各种外部存储器的接口，以及各种串行通信接口，例如USB，以太网，SDIO，CAN，UART，I2C和SPI。它还具有丰富的音频和视频功能，包括LCD显示屏，基本2D图形，相机接口，SPDIF和I2S音频接口。其他显着功能包括用于安防，电机控制，模拟信号处理和电源管理的各种模块。

增强型液晶显示接口（eLCDIF）是RGB接口显示控制器，它支持8/16/18/24位宽的数据端口和高达1366x768的分辨率。 为了传输帧数据以进行显示刷新，eLCDIF充当总线主控器或总线从属器，与SoC集成DMA引擎协同工作。 在这两种情况下，CPU都无需处理帧数据。

像素通道（PXP）模块集成了多种2D图形处理功能，包括缩放，色彩空间转换（CSC）和旋转。

2.2 i.MX RT1050 评估板

i.MX RT1050 评估板 是一个旨在展示i.MX RT1050处理器最常用功能的平台。评估板具有以下功能：

- 内存：256 Mbit SDRAM，64Mbit Quad SPI Flash，512 Mbit Hyper Flash，TF卡插槽
- 通信接口：USB 2.0 OTG连接器，USB 2.0主机连接器，10/100 Mbit / s以太网连接器，CAN总线连接器
- 多媒体接口：CMOS传感器连接器，LCD连接器
- 音频接口：3.5毫米立体声耳机孔，板上麦克风，S / PDIF连接器（默认情况下未安装）
- 调试接口：带有DAP-Link，JTAG 20引脚连接器的板载调试适配器
- Arduino界面
- 用户按钮和LED

Figure 1 展示i.MX RT1050 评估板的照片.

Figure 1. i.MX RT1050 EVK

2.3 RK043FN02H-CT TFT LCD 面板

RK043FN02H-CT TFT LCD 显示屏特征如下：

- 4.3英寸物理尺寸
- 480 * 272像素（RGB888）
- 24位RGB888接口，支持DE或HV模式
- LED 背光灯
- I2C接口电容式触摸接口

LCD显示屏经由通过40针FPC线和6针FPC线分别引出它的显示信号和触摸信号。它们与i.MX RT1050 EVK板上的连接器兼容。

i.MX RT1050 评估板仅利用部分24位显示数据信号，最多支持RGB565。

2.4 像素管道 (PXP)

像素处理通道用于在发送到LCD显示器之前对图像/视频缓冲区执行图像处理。它由几个流水线块组成，这些块执行视频源帧缩放，颜色空间转换，alpha混合/颜色关键算法，辅助CSC，像素校正。

该应用程序使用PXP模块。引入PXP模对图像分辨率进行缩放以 适配LCD的分辨率（480x272）。并将色彩空间从YUV转换为RGB， 然后eLCDIF将调整大小的帧传输到LCD显示屏。

2.5 i.MX RT1050 的SDK

该SDK为恩智浦的多个微控制器系列提供全面的软件支持。 SDK包含以下组件：

- 灵活的外围驱动程序
- 一组丰富的例程
- 来自恩智浦或合作的第三方中间件，例如FreeRTOS，emWin，FatFs，LIBJPEG，LwIP，mbedTLS，USBstack，wolfSSL等。
- SOC头文件，启动文件和适用于各种工具链的链接配置文件

2.6 FFMPEG 库

FFMPEG是一种多媒体框架，能够解码，编码，转码，多路复用，去多路复用，流式传输，过滤和播放人类和机器创造的几乎所有东西。 它支持从最模糊的古老格式，到最前沿的格式。 它还具有高度的可移植性：FFMPEG可在各种构建环境，机器体系结构和配置下，跨Linux，Mac OS X，Microsoft Windows，BSD，Solaris等编译，运行并通过测试基础架构FATE。

它包含可以被应用程序使用的libavcodec，libavutil，libavformat，libavfilter，libavdevice，libswscale和libswresample.

- Libavcodec: 是一个包含用于audio/video codecs的解码器及编码器的库。
- Libavutil: 是一个包含用于简化编程的函数的库，包括随机数生成器，数据结构，数学例程，核心多媒体实用程序等。
- Libavformat: 是一个包含用于多媒体容器格式的解复用器和复用器的库。
- Libavdevice:是一个包含输入和输出设备的库，用于从许多常见的多媒体输入/输出软件框架（包括Video4Linux，Video4Linux2，VfW和ALSA）中获取和呈现。
- libavfilter :是一个包含媒体过滤器的库。
- libswscale :是执行高度优化的图像缩放和颜色空间/像素格式转换操作的库。
- Libswresample: 是执行高度优化音频的重采样，重新矩阵化和样本格式转换操作的库

3 H.264 视频解码应用程序开发

本节介绍了基于Section 2 中介绍的硬件和软件平台开发H.264视频解码应用程序的过程。

3.1 系统结构分析

Figure 2 给出了该演示应用程序的硬件框图，该框图显示了系统的主要组件。

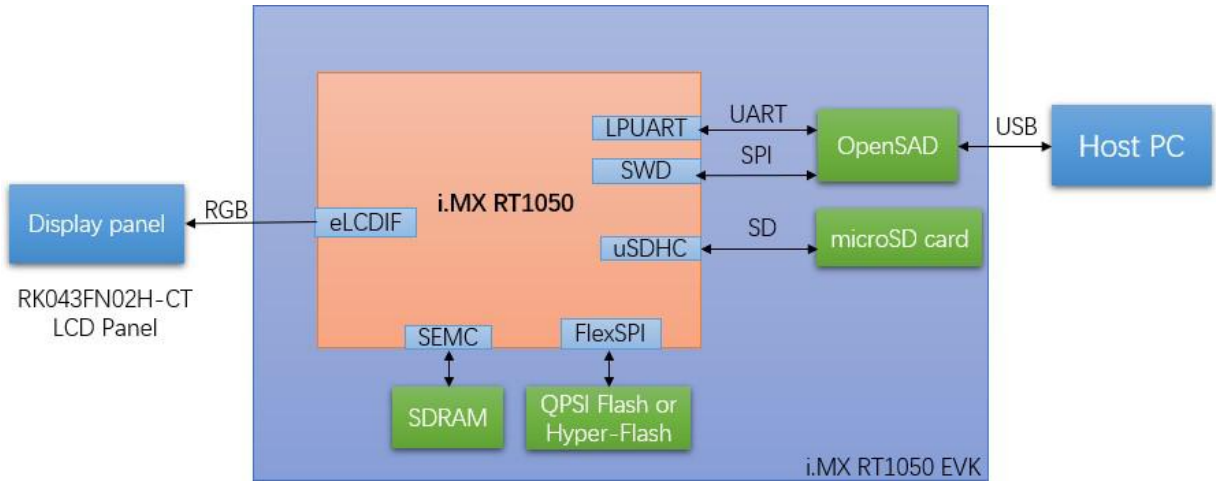

Figure 2. System hardware block diagram

- RT1050通过uSDHC模块读取保存在microSD中的H.264视频源
- 外部SDRAM设备为帧缓冲区和/或代码空间提供了数据空间。 RT1050通过智能外部存储控制器（SEMC）模块访问SDRAM设备。
- 外部QSPI flash及 hyper-flash 为具有XIP功能的非调试运行配置提供了代码空间。 i.MXRT1050通过FlexSPI控制器访问闪存设备。
- 开放标准串行调试适配器（OpenSDA）为板子提供SWD调试访问，调试UART桥和为开发板供电。OpenSDA通过USB端口与主机PC通讯，并实现“ CMSIS-DA”调试协议。
- RT1050通过RGB接口将帧数据传输到LCD显示屏。

Figure 3 显示了此演示应用程序的帧数据流程图。

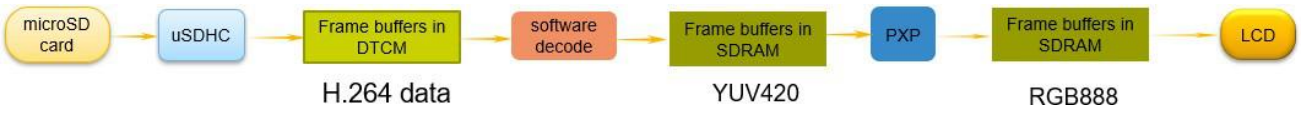

Figure 3. Frame data flow diagram

i.MX RT1050 评估版从microSD卡读取视频源，并将其存储在DTCM中的帧缓冲区中。 软件使用（定制版本的）FFMPEG解码视频数据。 PXP将帧的大小从原始分辨率调整为LCD分辨率，并将色彩空间从YUV格式转换为RGB格式，然后eLCDIF将帧传输到LCD显示屏。

3.2 搭建历程并运行

此AN的代码包是独立的，您按可以如下所示非常简单地构建项目：

3.2.1 从闪存中搭建并运行

1. 使用IAR打开“ <h264deocode.eww>”，使用默认项目配置，该配置为Flash XIP构建项目：

2. 按“ F7”进行搭建，可能需要一分钟

3. 按下“Ctrl-D”进入调试模式或点击 

4. IAR进入调试模式会停在main（）函数处，按“ F5”键运行。由于代码已写入Flash，您可以稍后重置板子以使其再次运行。

3.2.2 从 SDRAM 搭建并运行

如果要将程序下载到sdram，请继续以下操作。

NOTE

In SDRAM configuration, we show how to leverage ITCM & DTCM to further improve performance. We reconfigured ITCM to 384 kB and DTCM to 128 kB, which is not included in Flash configuration. It is because if ITCM & DTCM size is reconfigured, most SDK examples which rely on the default ITCM (128 kB), DTCM(128 kB), and OCRAM(256 kB) size could malfunction. Since code in Flash is persistent among power cycles, it may prevent you from running other examples, so we did not enable ITCM/DTCM for flash configuration.

1. 选择搭建模式

2. 重复3.2.1节中的步骤2, 3, 4

对于此演示，程序两种不同的构建运行模式具有不同的启动函数。 当程序以FLASH模式运行时，启动函数为startup_MIMXRT1052.s。 当程序以SDRAM模式运行时，启动函数为sdram_startup.s。

Figure 4 显示了sdram模式下启动文件的选择方法.。

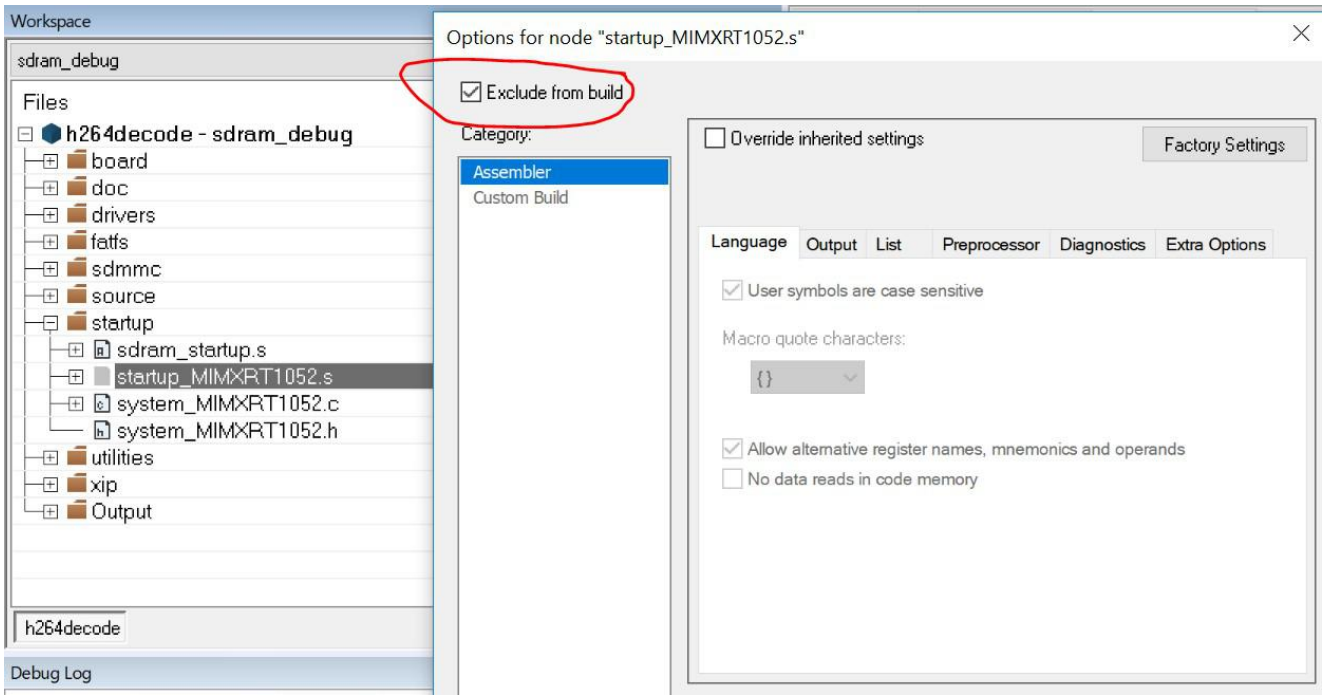

Figure 4. Startup files selection

3.3 内存空间分配

对于此演示应用程序，我们使用 Figure 5 所示的方案分配内存空间。

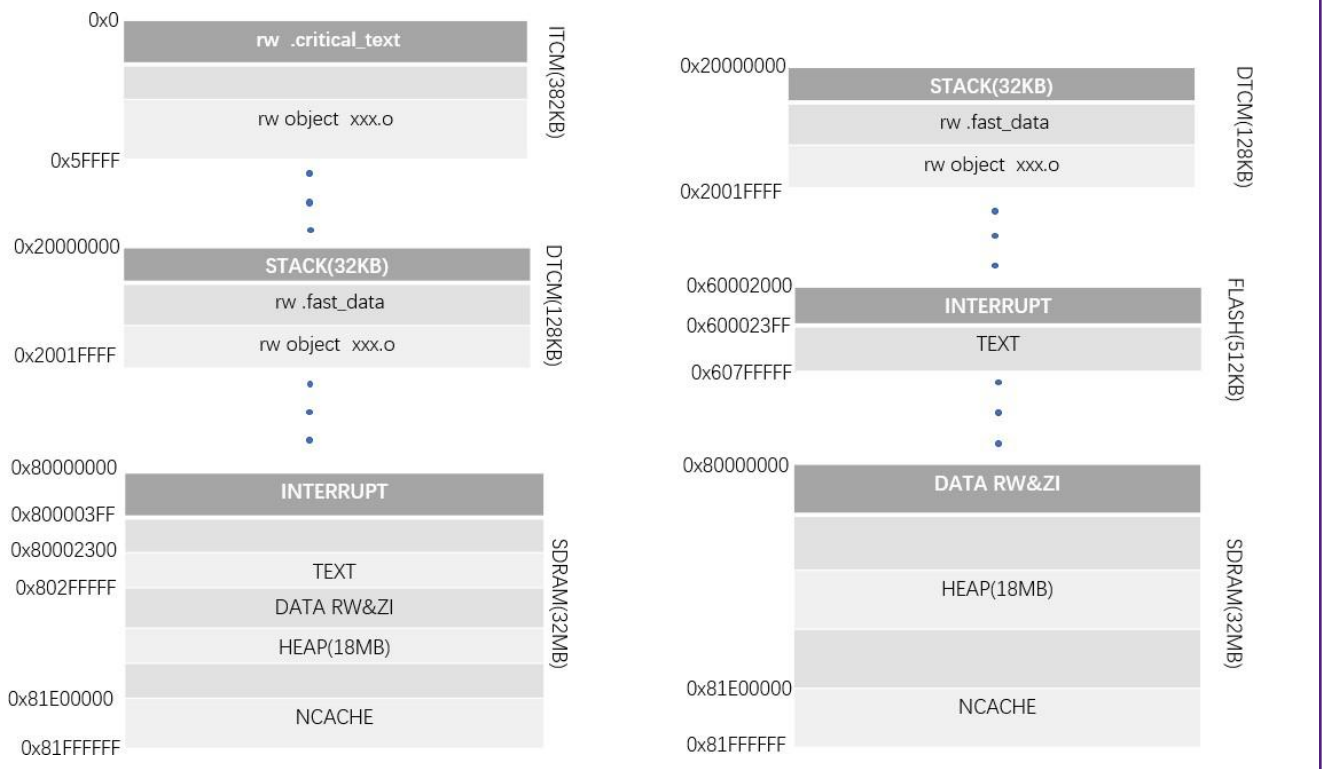

Figure 5. Memory allocation schemes

(a) SDRAM 配置 (b) FlexSPI-NOR 配置

3.4 软件解码

解码过程从microSD卡读取视频源，并使用剪辑FFMPEG库（版本3.0.11）对视频数据进行解码。 FFMPEG视频解压缩的过程如 Figure 6 所示:

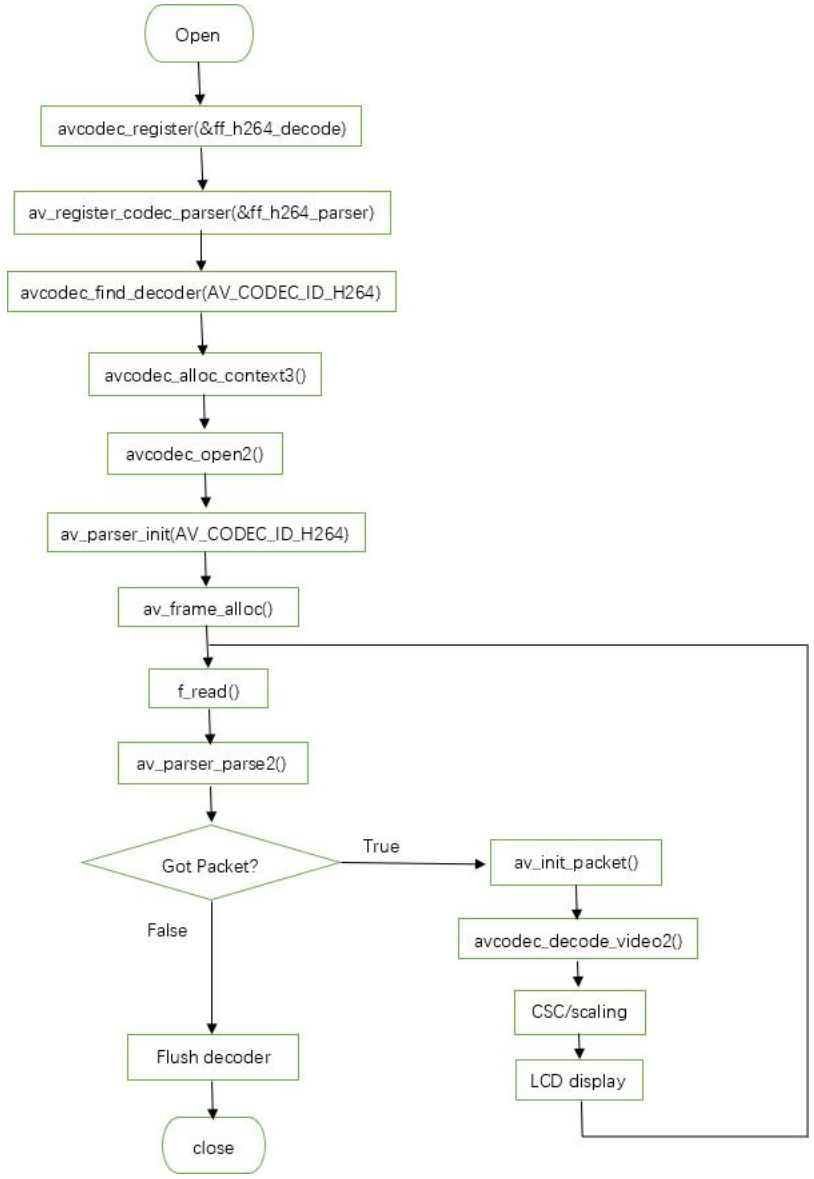

Figure 6. video decoding

该功能解释如下:

1. avcodec_register(&ff_h264_decoder): h264解码器的注册
2. av_register_codec_parser(&ff_h264_parser): h264解码器的注册
3. avcodec_find_decoder(AV_CODEC_ID_H264): 找到 h264 解码器
4. avcodec_alloc_context3(): 分配解码器上下文以及与之关联的所有内容
5. avcodec_open2():打开解码器
6. av_parser_init(AV_CODEC_ID_H264): 选择并初始化解析器
7. av_frame_alloc():将内存分配给AVFrame结构。AVFrame中的数据缓冲区必须在其他位置进行管理
8. av_parser_parse2(): 解析以获取数据包

9. av_init_packet(): 初始化数据包的值
10. avcodec_decode_video2(): 解码一帧数据

After the FFMPEG decoding, the original H.264 video source is decompressed into the YUV data format YUV data. To display on LCD panel, it still needs color space conversion(CSC) and scale scaling.

3.5 缩放及色彩空间转化

LCD的显示分辨率与视频源的分辨率不同，LCD只能显示RGB格式的数据，PXP模块负责缓冲区间缩放和CSC。

由于用户视频源的分辨率不同，需要修改相应的参数。

```
#define APP_PS_WIDTH 480/* 720,352,image resolution*/
```

APP_InitPxp（）函数中的数据格式配置，参数配置如 Figure 7 所示：

```
static void APP_InitPxp(void)
{
    PXP_Init(APP_PXP);

    /* PS configure. */
    psBufferConfig.pixelFormat = kPXP_PsPixelFormatYVU420;
    psBufferConfig.swapByte = false;
    psBufferConfig.bufferAddr = 0U;
    psBufferConfig.bufferAddrU = 0U;
    psBufferConfig.bufferAddrV = 0U;
    psBufferConfig.pitchBytes = APP_PS_WIDTH;

    PXP_SetProcessSurfaceBackGroundColor(APP_PXP, 0U);

    PXP_SetProcessSurfaceBufferConfig(APP_PXP, &psBufferConfig);

    /* Disable AS. */
    PXP_SetAlphaSurfacePosition(APP_PXP, 0xFFFFU, 0xFFFFU, 0U, 0U);

    /* Output config. */
    outputBufferConfig.pixelFormat = kPXP_OutputPixelFormatRGB888;
    outputBufferConfig.interlacedMode = kPXP_OutputProgressive;
    outputBufferConfig.buffer0Addr = (uint32_t)s_psBufferLcd[0];
    outputBufferConfig.buffer1Addr = 0U;
    outputBufferConfig.pitchBytes = APP_IMG_WIDTH * APP_BPP;
    outputBufferConfig.width = APP_IMG_WIDTH;/*lcd_width   480   scale*/
    outputBufferConfig.height = APP_IMG_HEIGHT;/*lcd_heigth  272*/

    PXP_SetOutputBufferConfig(APP_PXP, &outputBufferConfig);

    /* Disable CSC1, it is enabled by default. */
    PXP_SetCsc1Mode(APP_PXP, kPXP_Csc1YCbCr2RGB);
    PXP_EnableCsc1(APP_PXP, true);
}
static void APP_InitLcdif(void)
```

Figure 7. PXP module initialization configuration

输入视频格式:

Processed Surface（PS）configue：psBufferConfig.pixelFormat = kPXP_PsPixelFormatYVU420

输出视频格式:

输出配置: outputBufferConfig.pixelFormat = kPXP_OutputPixelFormatRGB888

CSC:

只有当CSC1被使能时，CSC1模块才能从缩放引擎接收被缩放的YUV / YCbCr444像素，并将像素进行RGB888颜色空间转换

使能 CSC1:

PXP_SetCsc1Mode(APP_PXP, kPXP_Csc1YCbCr2RGB)

PXP_EnableCsc1(APP_PXP, true)

FFMPEG解码后生成的Y，U，V数据地址对应于PS配置的Y，U，V地址。

引入了两个LCD帧缓冲区和两个缓冲区指针。 PXP通过两个指针填充两个缓冲区，而eLCDIF也通过两个指针耗尽两个缓冲区。 类似地，使用两个指针还可以使缓冲区的填充和消耗同步连续进行。

Figure 8 illustrates the LCD frame buffer accessing scheme.

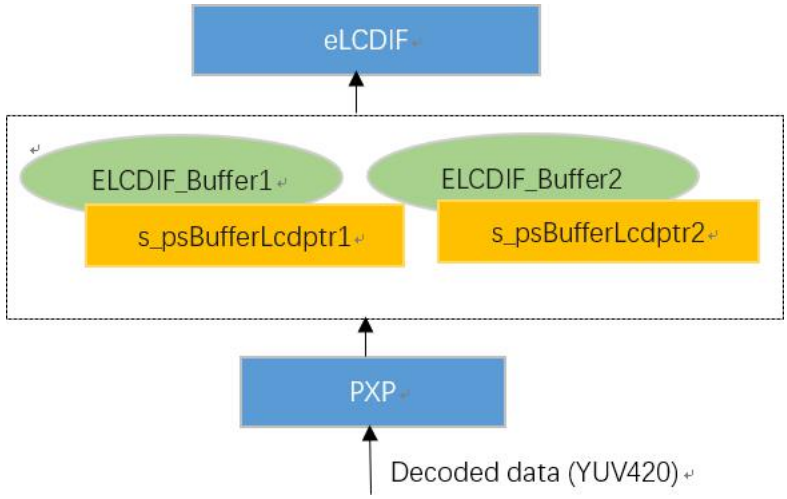

Figure 8. LCD Frame buffers accessing scheme

3.6 运行例程

该软件包以及本文档提供了例程的所有源码和项目文件。 应用运行步骤:

- 用微型USB电缆连接在主机PC和EVK-MIMXRT1050板上的OpenSDA USB J28端口。
- 打开具有115200波特率，8个数据位，无奇偶校验位和1个停止位的设置的串行终端工具以显示调试日志（可选）。
- 设置启动模式，将SW7-1，SW7-2，SW7-3，SW7-4设置为off，on，on，off.
- 启动调试会话或将二进制文件下载到处理器。
- 在IDE中启动调试器，或按下重置按钮SW4开始运行例程。

4 性能分析

在此应用中，CPU时钟，IPG时钟，SDRAM工作频率，LCD刷新率和SDRAM工作数据长度分别是600 MHz，150 MHz，164 MHz，60 Hz和16位。

选择三个不同的视频源进行解码，对此例程使用的不同搭建和运行方法来显示和测试每秒帧数（fps）。

Table 1 显示了不同视频源的测试结果。

Table 1. Test results(fps)

Video source	Resolution	Total frame	SDRAM(fps)	FLASH(fps)
clown_720x576.h264	720x576	250	21.9	18.5
bigbuckbunny_480x272.h264	480x272	250	34.2	25.4
formen_352x288.h264	352x288	240	41.2	31.7

Table 1 包含了视频读取，解码以及pxp模块处理和LCD播放的总时间。

对于读取文件名和文件格式的限制，我们可以配置fatfs以支持长文件名，只需在ffconf.h中将FF_USE_LFN设置为1，对于测试不同的视频的客户更加友好。

PXP转换时间与src和dest缓冲区的存储内存类型有关。 在以前的测试中，将src和dest缓冲区都放入SDRAM。 现在，对src和dest缓冲区的不同内存分配进行了更多测试（通过修改.icf文件）。

Table 2 和 Table 3 显示了不同视频的测试时间

Table 2. Both src and dest video resolutions are 288x180

\Dest buffer Src buffer\	SDRAM	DTCM	ITCM	OCRAM
SDRAM	2.129 ms	1.059 ms	1.059 ms	1.059 ms
DTCM	1.067 ms	0.471 ms	0.471 ms	0.384 ms
ITCM	1.067 ms	0.471 ms	0.470 ms	0.384 ms
OCRAM	1.067 ms	0.384 ms	0.384 ms	0.384 ms

#unique_20 src 和dest 的视频分辨率都是 480x272

Table 3. Both src and dest video resolutions are 480x272

\Dest buffer Src buffer\	SDRAM	DTCM	ITCM	OCRAM
SDRAM	6.172 ms	2.939 ms	2.939 ms	2.939 ms
DTCM	3.394 ms	/	/	0.932 ms
ITCM	3.394 ms	/	/	0.932 ms
OCRAM	3.394 ms	0.932 ms	0.932 ms	0.932 ms

src 缓冲区: 用于存储在解码后生成的YUV数据。

dest 缓冲区: 用于存储pxp模块处理后生成的RGB数据。

5 总结

本篇应用笔记介绍了基于i.MX RT1050 评估板的SDK使用i.MX RT1050处理器开发H.264视频解码应用程序的步骤，从构建项目到完成应用程序。 SDK提供的外围设备驱动程序和各种中间件使整个开发过程变得容易。

随本应用笔记一起提供了例程的源代码。 在此基础上，您可以开发自己的定制H.264视频解码应用程序。

NOTE

When users use the sdk version of 2.5.0 and above to build project tree, delete the red part of Figure 9, otherwise it overrides the stack and heap configurations/allocations in ".icf" linker files, and the demo will fail to run.

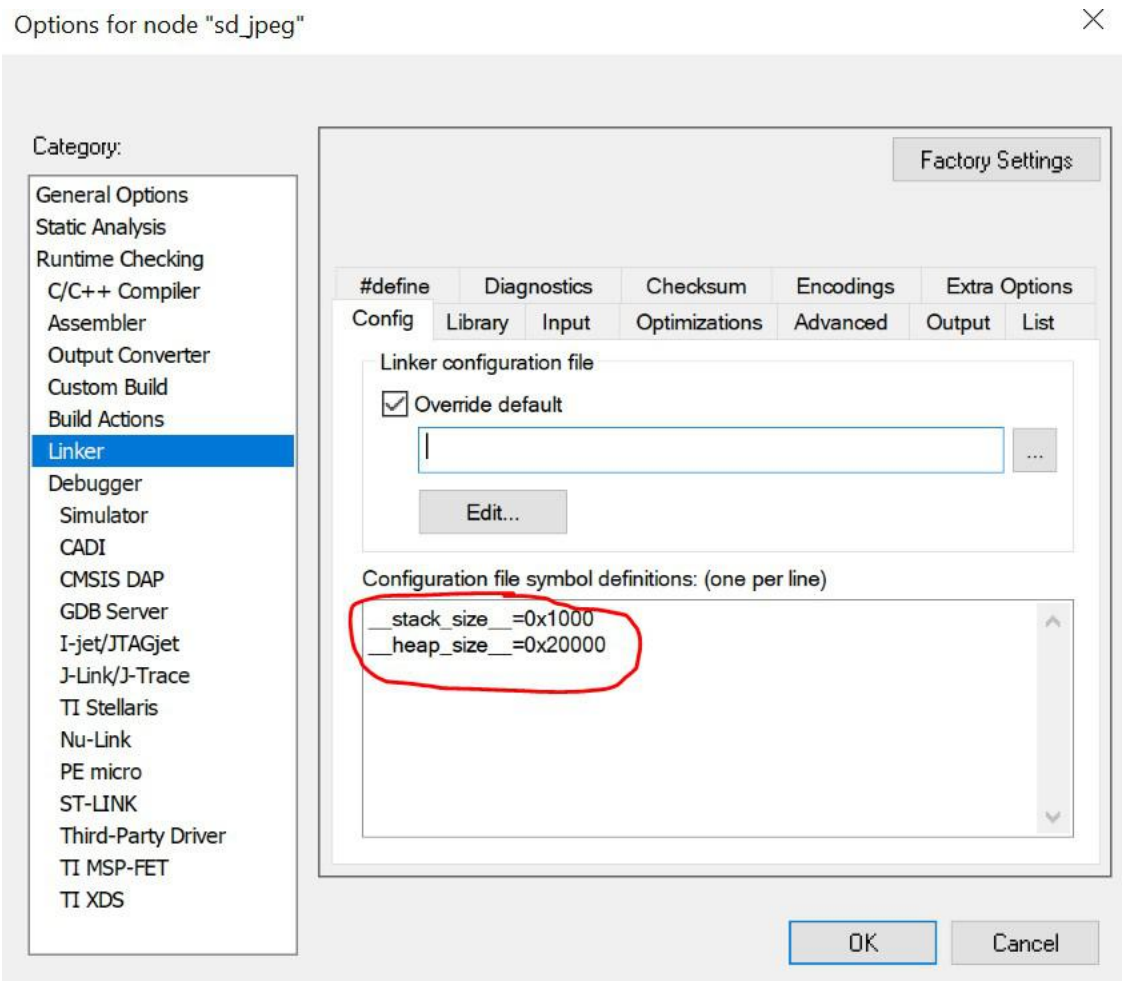

Figure 9. sdk version of 2.5.0 IDE (delete them in red block)

6 参考文献

以下文件可能会提供进一步的参考。

- i.MX RT1050 Processor Reference Manual, Rev. 0
- FFMPEG video decoding

7 修订历史

Table 4. Revision history

Revision number	Date	Substantive changes
0	09/2019	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 28 August, 2019

Document identifier: AN12562

arm