AN13205

Development of H.264 Video Encode on RT Series

Rev. 0 — March 25, 2021 Application Note

Contents

1 Introduction

The i.MX RT1170 crossover processor sets speed records at 1 GHz. This ground-breaking family combines superior computing power and multiple media capabilities with more usable as well as real-time functionality. The dual-core i.MX RT1170 runs on the Arm® Cortex® -M7 core at 1 GHz and Arm Cortex-M4 at 400 MHz. It provides best-in-class security. The i.MX RT1170 MCU supports a wide range of temperature and is qualified for consumer, industrial and automotive markets.

This application note describes how to develop an H264 video encoder application with NXP i.MX RT1170 processor.

For such applications, the i.MX RT1170 receives YUV format video source from the microSD card then invokes the X264 Library to encode the video source and generates H264 data.

2 Hardware and software platforms

This section presents short introductions of the hardware and software platforms of the demo application, including the below items.

- i.MX RT1170 processor
- i.MX RT1170 EVK board
- SDK for i.MX RT1170 EVK board
- Libx264

2.1 i.MX RT1170 processor

i.MX RT1170 is a new processor family featuring NXP's advanced implementation of the high performance Arm Cortex®-M7 Core and a power efficient Arm Cortex®-M4 Core. It offers high-performance processing optimized for lowest power consumption and best real-time response. The i.MX RT1170 has 2 MB on-chip RAM in total, including a 512 KB RAM which can be flexibly configured as TCM or general-purpose on-chip RAM. The i.MX RT1170 integrates advanced power management module with DCDC and LDO that reduces complexity of external power supply and simplifies power sequencing. The i.MXRT1170 also provides various memory interfaces, including SDRAM, Raw NAND FLASH, NOR FLASH, SD/eMMC, Quad SPI, HyperRAM/HyperFlash and a wide range of other interfaces for connecting peripherals, such as WLAN, Bluetooth®, GPS, displays, and camera sensors. Same as other i.MX processors, i.MX RT1170 also has rich audio and video features, including MIPI CSI/DSI, LCD display, graphics accelerator, camera interface, SPDIF and I2S audio interface.

2.2 i.MX RT1170 EVK board

The i.MX RT1170 EVK board is a platform designed to showcase the most commonly used features of the i.MX RT1170 processor. The EVK board offers the below features:

- 6468 CoreMark with Cortex-M7 @ 1 GHz + Arm Cortex-M4 @ 400 MHz
- 2 MB SRAM with 512 KB of TCM for Cortex-M7 and 256 KB of TCM for Cortex-M4

- Advanced security, including secure boot and crypto engines, and is part of the EdgeLock Assurance Program
- 2 × Gb ENET with AVB and TSN
- 2D GPU
- MIPI® CSI/DSI
- Supported by MCUXpresso suite of software and tools

Figure 1 presents the picture of the i.MX RT1170 EVK.

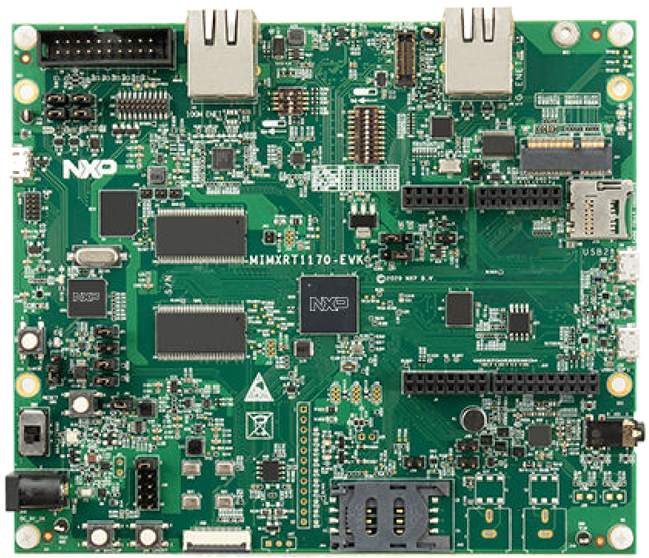

Figure 1. i.MX RT1170 EVK

2.3 SDK for i.MX RT1170 EVK board

SDK provides comprehensive software support for multiple microcontroller families from NXP. The SDK comprises the blow components:

- A flexible set of peripheral drivers.
- A rich set of example applications.
- Various middleware from NXP or incorporated from a third party, such as FreeRTOS, emWin, FatFs, LIBJPEG, LwIP, mbed TLS, USB stack, wolfSSL, and so on.
- The SOC header file, startup files, and linker configuration files for various tool chains.

2.4 Libx264

x264 is a free software library and application for encoding video streams into the H.264/MPEG-4 AVC compression format, and is released under the terms of the GNU GPL.

Features Overview:

- Provides best-in-class performance, compression, and features.
- Achieves dramatic performance, encoding four or more 1080p streams in realtime on a single consumer-level computer.
- Gives the best quality, having the most advanced psychovisual optimizations.

- Support features necessary for many different applications, such as television broadcast, Blu-ray low-latency video applications, and web video.

x264 forms the core of many web video services, such as Youtube, Facebook, Vimeo, and Hulu. It is widely used by television broadcasters and ISPs.

3 Develop H.264 video encoding application

This section describes the procedure to develop H.264 video encoding application based on the hardware and software platforms presented.

3.1 System structure analysis

Figure 2 presents the hardware block diagram of this demo application, which shows the primary components of the system.

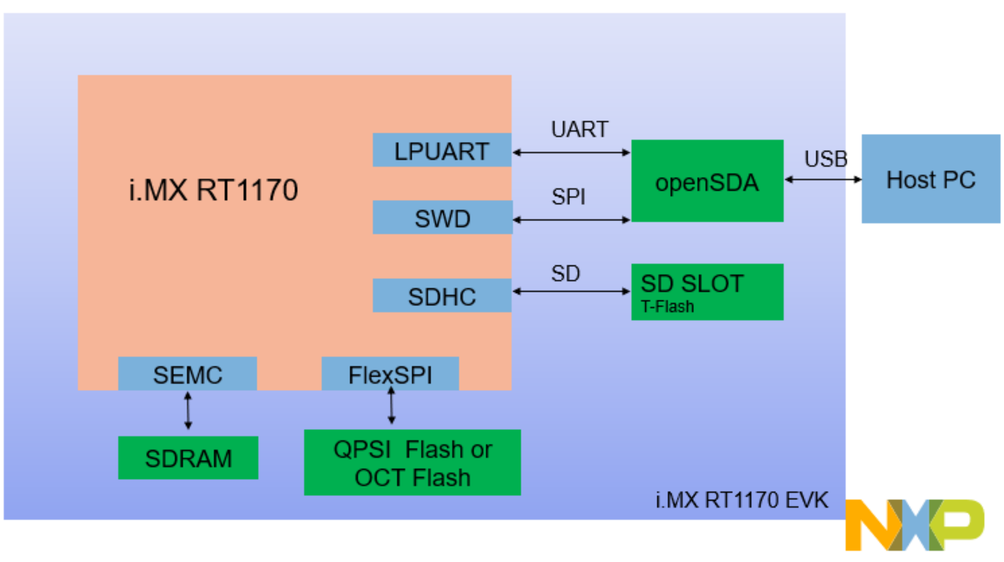

Figure 2. System hardware block diagram

- RT1170 read YUV video source which is saved in microSD by uSDHC module.
- External SDRAM devices provide data space for frame buffer and/or code space. RT1170 accesses SDRAM devices by the Smart External Memory Controller (SEMC) module.
- External QSPI flash or Octal flash provides code space for non-debugging running configuration with XIP capability. The i.MX RT1170 accesses flash devices by the FlexSPI controller.
- The Open-Standard Serial Debug Adapter (OpenSDA) provides SWD debug access, debug UART bridge, and power supply for the board. OpenSDA communicates with the host PC via a USB port，and implements the **CMSIS-DA** debug protocol

Figure 3 shows the frame data flow diagram of this demo application.

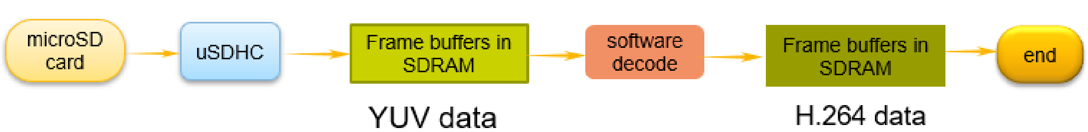

Figure 3. Frame data flow diagram

i.MX RT1170 EVK reads video source from microSD card and stores in the frame buffers located in SDRAM. Software encode video data with X264.

3.2 Build the demo project and run

The code package with this document is self-contained, and you can build the project quite straightforward as below:

3.2.1 Build and run from Flash

1. Use IAR to open **<h264encode.eww>**. Use the default project configuration, which will build the project for flash XIP:

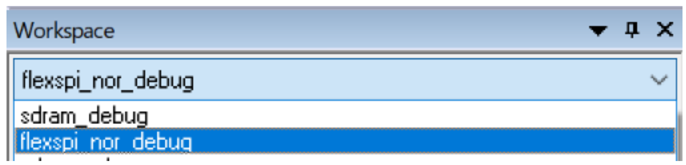

Figure 4. Use default project configuration

2. Press **F7** to build and it may take a minute.
3. After the build completes, press **Ctrl-D** or click  to enter debug session.
4. After IAR enters debug session and stop at `main()` function, press **F5** to run.

Since the code is programmed to Flash, reset the board later to let it run again.

3.2.2 Build and run

To download the program, continue with the following operations.

NOTE

The configurations show how to leverage ITCM & DTCM to further improve performance. We reconfigure ITCM to 448 kB and DTCM to 64 kB, which is also included in Flash configuration.

1. Select build mode.

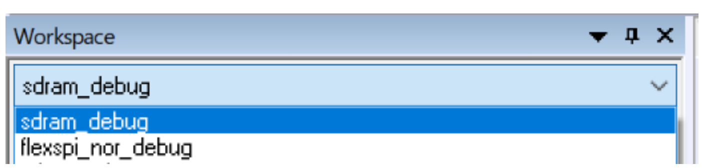

Figure 5. Select build mode

2. Repeat Step 2, Step 3, and Step 4 in Build and run from Flash.

For this demo, the two different build and run modes of the program have the same startup functions. The startup function is `startup_MIMXRT1176_cm7_x264.s`. Compared with `startup_MIMXRT1176_cm7.s` file in the SDK, this file adds the reconfiguration of TCM.

3.3 Memory space allocation

For this demo application, we allocate the memory space with the schemes as shown in Figure 6.

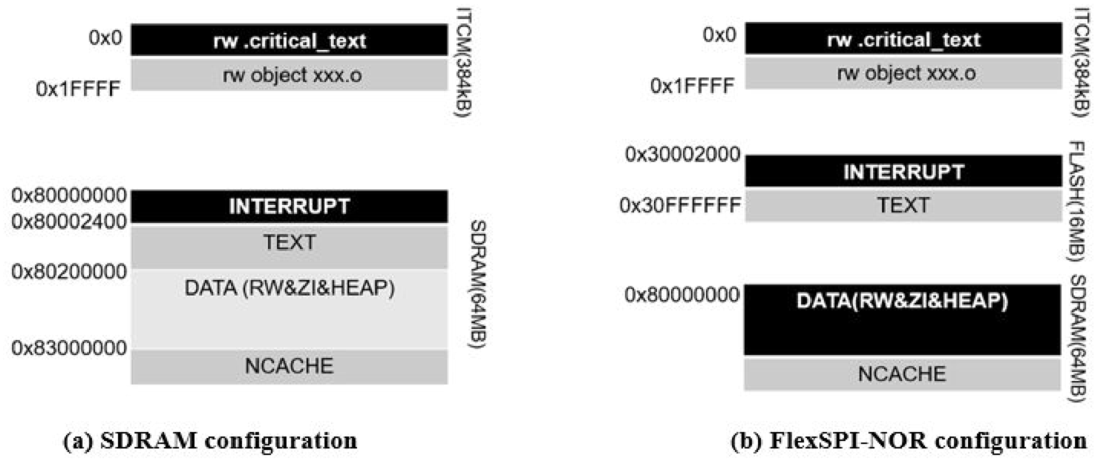

Figure 6. Memory allocation schemes

3.4 Software encode

The encoding process reads the video source from the microSD card and encode the video data using the `libx264`. Figure 7 shows the process of `libx264` video decompression.

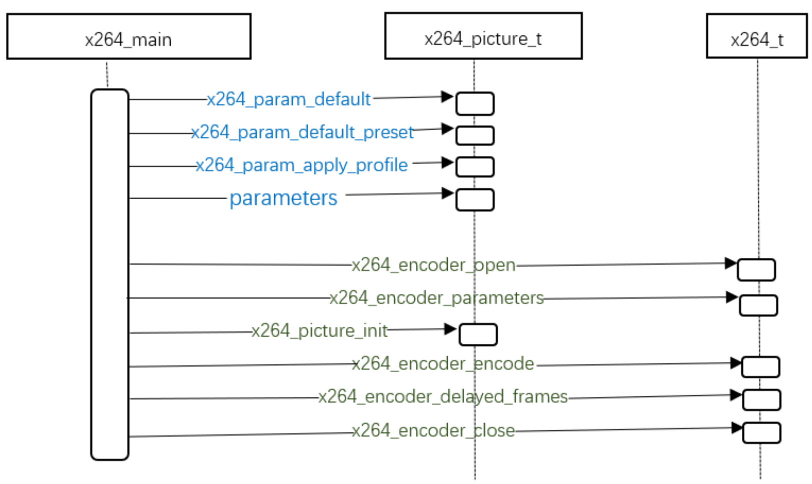

Figure 7. H.264 video encoding

As shown in Figure 7, the `libx264` encoder can be divided into: initialize the encoder, encode, and close the encoder.

The dotted arrow indicates that it will be called under certain circumstances.

Structures and functions starting with `x264` are defined in `libx264`.

Initialize the encoder:

1. **x264_ param_ Default()**: Set the default parameters.
2. **x264_ param_ default_ Preset ()**: Set the default **preset** when the **preset** or **tune** is not **NULL**. This demo sets **char * preset = ultra fast**.
3. **x264_ param_ apply_ Profile ()**: Set **profile**. If **profile** is **NULL**. This demo sets **char * profile = baseline**.
4. **Various parameters to be set**: You can modify **param->i_frame_total**. The default input and output files are *test.yuv, test.h264*. Due to the limitation of reading file name and file format, it is better for users to change the name of video

source to **test.yuv** when testing their own video source. To modify the parameters, press **Ctrl+Shift+f** to modify, as show in Figure 8.

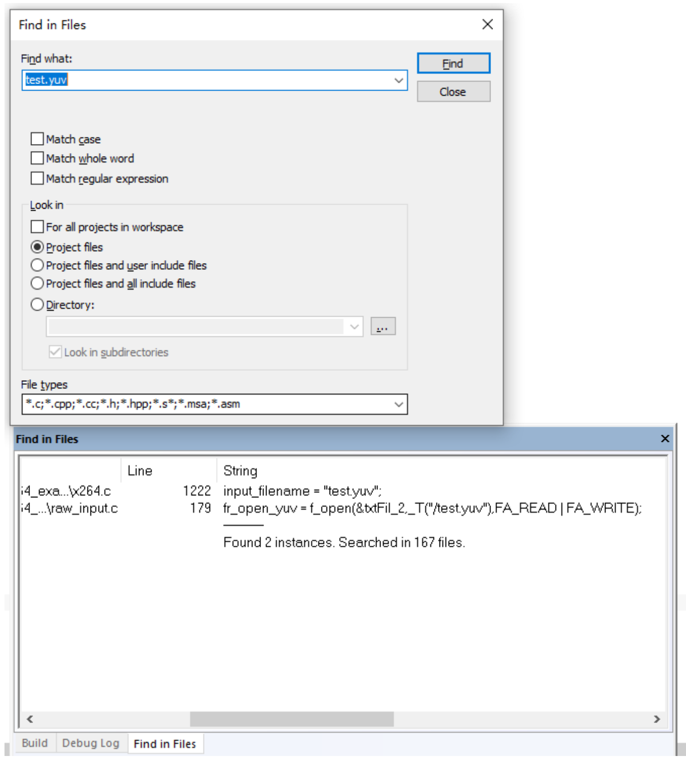

Figure 8. Parameter search

Encode:

1. **x264_encoder_open()**: Turn on the encoder and return an encoder handle `x264_t *`.
2. **x264_encoder_encode**: The function to perform encoding. The input data to be encoded is in `x264_picture_t`, and the output coded data is in the `x264_nal_t` array. At the input, the buffer address of the data to be encoded is assigned to the corresponding variable in `x264_picture_t`.
3. **x264_encoder_delayed_frames**: Returns the number of currently delayed (buffered) frames. The call of `x264_encoder_encode` will loop until the value returned by `x264_encoder_delayed_frames` is **0**.

3.5 Run the demo application

This application note offers the whole source and project files of the demo application. To run the demo:

- Connect a micro USB cable between the host PC and the `OpenSDA` USB port J11 on the EVK-MIMXRT1170 board.
- Optionally, open a serial terminal tool with settings of 115200 baud rate, 8 data bits, no parity bits, and 1 stop bit to display debug logs.
- Set boot mode. Set SW1-1, SW1-2, SW1-3, SW1-4 to **OFF**, **OFF**, **ON**, **OFF**.
- Start the debug session or download the binary to the processor.
- Launch the debugger in the IDE or press the reset button, **SW3**, to begin running the demo.

4 Performance analysis

For this demo, two different video sources are selected to encode and test frame per second (fps) with different build and run methods.

Table 1 shows the test results for different video sources , contains the total time of video reading, encoding.

Table 1. Test results_1(fps)

Video source	Resolution	Total frame	SDRAM (fps)	FLASH (fps)
`bigbuckbunny_480x272.yuv`	480×272	200	23.5	10
`formen_352x288.yuv`	352×288	200	28.8	12

Table 2 contains the total time of video reading, encoding and storing.

Table 2. Test results_2(fps)

Video source	Resolution	Total frame	SDRAM (fps)	FLASH (fps)
`bigbuckbunny_480x272.yuv`	480×272	200	18	8.5
`formen_352x288.yuv`	352×288	200	21	10

5 Conclusion

This application note describes the steps of how to develop H.264 video encode application with the i.MX RT1170 proessor based on the SDK of i.MX RT1170 EVK board, from building project to completing the application. The peripheral drivers and the various middleware offered by the SDK make it easy for the whole development process.

To develop your own customized H.264 video encode applications, see the source code of the demo application provided along with this application note.

6 References

Following documents may offer further reference.

- *i.MX RT1170 Processor Reference Manual* (document IMXRT1170RM)

7 Revision history

Revision number	Date	Substantive changes
0	March 24, 2021	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: March 25, 2021
Document identifier: AN13205