life.augmented

AN4996
应用笔记

STM32F76/77xxx和STM32H7x3系列微控制器中的硬件JPEG编解码外设

引言

本应用笔记描述在STM32F76/77xxx和STM32H7x3系列微控制器中如何对JPEG解码/编码应用使用硬件JPEG编解码外设。

STM32F76/77xxx和STM32H7x3系列微控制器嵌入了专用的硬件JPEG编解码外设，它提供了快速又简单的硬件JPEG图像压缩程序和解压程序，具备：

- JPEG文件头的全面管理能力，
- 完全可编程的霍夫曼表（两个AC表和两个DC表），
- 多达四个可编程量化表，
- 完全可编程最小编码单元（MCU）。

硬件JPEG编解码器支持YCbCr或RGB（3个颜色分量）、灰度（1个颜色分量）和CMYK（4个颜色分量）模式的像素输入/输出格式，每个分量的下采样因子完全可编程。

为了充分利用本应用笔记，用户应熟悉：

- STM32的JPEG编解码外设，如STM32F76/77xxx参考手册（RM0410）所述，该手册可从意法半导体网站*www.st.com*上获得。
- STM32的JPEG编解码外设，如STM32H7x3参考手册（RM0433）所述，该手册可从意法半导体网站*www.st.com*上获得。
- JPEG压缩标准（JPEG ISO/IEC 10918-1 ITU-T建议T.81）和JFIF文件格式标准（JPEG文件交换格式）。

参考文档

- *STM32F76xxx和STM32F77xxx基于32位MCU的高级Arm®参考手册*（RM0410），
- STM32H7x3*基于32位MCU*的高级*Arm®参考手册*（RM0433），
- STM32F7系列（STM32CubeF7）和STM32H7系列（STM32CubeH7）的嵌入式软件。

表1. 适用产品

类型	产品线和部件号
微控制器	STM32F777BI、STM32F777II、STM32F777NI、STM32F777VI、STM32F777ZI、STM32F7x8系列、STM32F7x9系列
	STM32H7x3系列

目录

表格索引

图片索引

1 硬件JPEG编解码器概述

硬件JPEG编解码外设符合JPEG标准（JPEG ISO/IEC 10918-1 ITU-T建议T.81）。它可以解码/编码JPEG压缩图像，每个样本8位。

硬件JPEG编解码外设为熵编解码段（ECS）编码和解码提供硬件加速。它支持JPEG文件头生成和解析。硬件JPEG编解码外设还支持JFIF（JPEG文件交换格式），使用事实标准对JPEG图像进行编码。但是，在这些数据流中找到的所有应用特定的标记段均被忽略。JPEG编解码器支持最多四个颜色分量、四个量化表和两组DC和AC霍夫曼表。

硬件JPEG编解码器可以灵活地指定要对每个分量使用的量化表和霍夫曼表。

JPEG标准定义的JPEG编码和解码操作按块执行。JPEG标准将MCU（最小编解码单元）定义为可以编码或解码的最小块数。在硬件JPEG编解码外设中，MCU的构成是可编程的。硬件JPEG编解码器可以定义每个MCU中属于特定颜色分量的块数。每个块是样本的一个8x8数组，每个样本被定义为8位（1个字节）。因此，每个块均是一个64字节数组（每个样本1个字节）。

硬件JPEG编解码器支持YCbCr或RGB（3个颜色分量）、灰度（1个颜色分量）和CMYK（4个颜色分量）模式的像素输入/输出格式，每个分量的下采样因子完全可编程。

使用STM32H7x3系列器件进行JPEG解码操作，当输出颜色格式为YCbCr时，Chrom-Art Accelerator™外设（也称为DMA2D）能够将YCbCr块（JPEG解码器的输出）转换为可直接显示的RGB像素。

使用STM32H7x3系列器件进行编码（所有颜色格式）或非YCbCr（灰度或CMYK颜色格式时）颜色格式的解码时，从/至RGB像素的转换不进行硬件加速且必须通过软件执行。

使用STM32F76/77xxx器件进行解码或编码时，YCbCr至RGB的转换不加速且必须通过软件执行。

STM32CubeF7/H7固件包提供包含必要API的专用JPEG实用工具软件，能够执行JPEG MCU块至/从RGB像素的转换（位于\Firmware\Utilities\JPEG目录下）。

STM32CubeF7/H7为JPEG编解码外设提供专用HAL（硬件抽象层）驱动程序：

- STM32CubeF7：stm32f7xx_hal_jpeg.c/ stm32f7xx_hal_jpeg.h
- STM32CubeH7：stm32h7xx_hal_jpeg.c/ stm32h7xx_hal_jpeg.h

本文档适用于基于Arm®的器件。

arm

2 不同色彩空间的硬件JPEG编解码器设置

2.1 YCbCr色彩空间

2.1.1 YCbCr与RGB之间的相互转换

JPEG文件交换格式（JFIF）标准描述了YCbCr与RGB之间的相互转换和色度采样。JFIF合规文件通常具有如下扩展名：.jpg、.jpeg、.JPG、.JPEG。

JPEG标准（JPEG ISO/IEC 10918-1 ITU-T建议T.81）没有定义要对源行图像使用的色彩空间，而JFIF标准则定义了两个可能的色彩空间：灰度（Y亮度）或彩色（YCbCr亮度和色度）。

JFIF标准使用YCbCr颜色而不是原始的RGB色彩空间。该色彩空间能够从给出像素颜色的2个色度分量Cb和Cr中分离出亮度分量（Y），即像素亮度（本质上是灰度信号）。RGB色彩空间与YCbCr之间相互转换的转换矩阵如下：

图1. YCbCr/RGB颜色转换

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.16874 & -0.33126 & 0.5 \\ 0.5 & -0.41869 & -0.08131 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.77200 & 0 \end{bmatrix} \begin{bmatrix} Y \\ Cb - 128 \\ Cr - 128 \end{bmatrix}$$

已知人眼对亮度变化比对颜色变化更敏感，可使用YCbCr对两个独立的量化表进行定义，分别用于亮度和色度（Cb和Cr）分量，以便进一步量化色度（至少对于低频率）。

2.1.2 YCbCr量化表

*图 2*和*图 3*所示为JPEG标准提供的样本亮度和色度量化表。按照标准中的描述，这些表在每个样本8位的亮度和色度图像上给出很好的结果（对于STM32F7/H7硬件JPEG编解码器）。

该标准还对这些表格做了如下描述：如果将这些量化值除以2，得到的重构图像与源图像的差异通常难以辨别。

图2. YCbCr亮度量化表

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

图3. YCbCr色度量化表

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

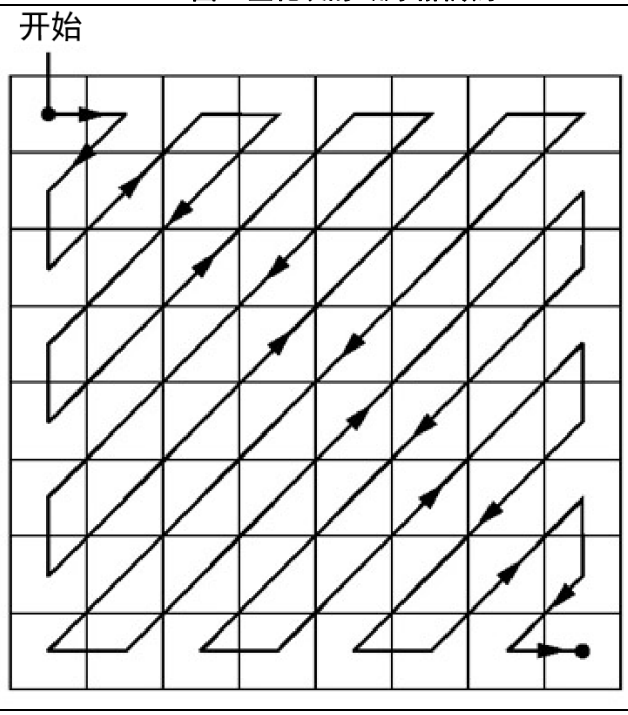
请注意，这些表格采用Z形排序。

*图 4*和*图 5*提供了Z形排序的顺序。

图4. 量化表的Z形排序顺序

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

图5. 量化表的Z形扫描顺序



STM32CubeF7和STM32CubeH7使用这些默认表格，结合用户质量因子进行编码和解码。

- 进行编码时，STM32CubeF7/H7 JPEG HAL驱动程序允许用户定义百分比质量因子（1%至100%）。然后，按如下方式使用质量因子对以上表格进行比例计算：
  - 当质量处于50%至100%范围内时：scaling_factor = 200 - 2 x 质量。
  - 当质量低于50%时：scaling_factor = 5000 / 质量。

因此，当质量设置为100%时，比例因子变为0，然后所有表格条目均变为1（因为为零的条目被系统性地替换为1）。这使得量化损失最小化。

然后，在硬件JPEG编解码器中将量化表编程为专用存储表：QMEM0用于亮度（Y），QMEM1用于色度（Cb和Cr）。

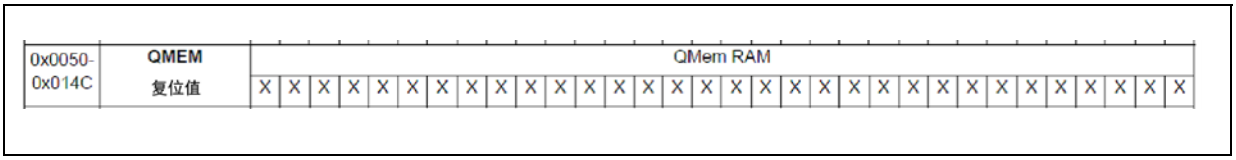
- 在解码时，STM32CubeF7/H7 JPEG HAL驱动程序能够按如下方式检索质量。对于量化表的每个值：
  - 读取量化系数并计算该系数对比参考表中相应值的比例因子（百分比值）。
    比例 = (100 x quantization_coefficient) / reference_table_value)。
  - 如果quantization_coefficient = 1，则质量为100%。
  - 否则，如果比例低于100：质量 = (200 - 比例) /2。
  - 否则，质量 = 5000 / 比例。

按照量化表每个系数（共64个系数）的计算质量平均值计算编码质量。只使用亮度表计算平均质量。

硬件JPEG编解码器的QMEMx内存表用于存储/检索比例量化表（对比参考表）。按Z形顺序访问这些表格。

硬件JPEG编解码器提供RAM“QMEM”区，用于存储/检索最多4个量化表（分别对应于4个颜色分量）。QMEM RAM位于偏移地址0x0050至0x014C。每个表的大小为64字节（即16个32位字）。

图6. 硬件JPEG QMEM RAM



对于YCbCr色彩空间，STM32CubeF7/H7 JPEG HAL驱动程序默认只使用2个量化表：

- QMEM0：用于亮度（Y）分量。64字节位于偏移地址0x0050。
- QMEM1：用于色度（Cb和Cr）分量。64字节位于偏移地址0x0090。

请注意，仅当硬件JPEG编解码器停止时，才能读/写QMEM RAM，即没有正在进行的编码/解码操作（JPEG_CONFR0寄存器的位0“START”设置为0）。

STM32CubeF7/H7 JPEG HAL驱动程序默认对（Y）亮度分量使用*图 2*所示表格，并对Cb和Cr色度分量使用*图 3*所示表格。JPEG HAL驱动程序还为用户按颜色分量定义量化表提供了可能性（本例中为3个量化表）。如需自定义量化表，用户必须提供3个量化表（每个分量一个量化表）。这些表格分别用于（使用质量因子进行比例计算后）硬件JPEG编解码器的QMEM0至QMEM3 RAM表的编程（其中，QMEM2表位于偏移地址0x00D0）。

HAL函数“HAL_JPEG_SetUserQuantTables”是用于自定义用户量化表的API。

2.1.3 YCbCr色度下采样和最小编解码单元（MCU）结构

在YCbCr色彩空间中，可以对色度分量进行下采样（信息减少）而不会显著降低视觉质量。硬件JPEG编解码器允许用户使用JPEG_CONFR4至JPEG_CONFR7寄存器定义水平和垂直采样因子以及每个分量（最多4个分量）的8x8块数。

在文件头解析禁用的情况下进行编码或解码时，这些寄存器用于将每个分量的编码参数通知编解码器。

在文件头解析使能的情况下进行解码时，在JPEG文件头解析完成后由硬件编解码器自动填充这些寄存器，即JPEG_SR寄存器的位6（文件头解析完成标记）变为1。

硬件JPEG编解码器为使用JPEG_CONFR4-7寄存器定义任何采样因子和每个分量的8x8块数提供了可能性，寄存器的描述如下。

JPEG 编解码器配置寄存器 4-7 (JPEG_CONFR4-7)

地址偏移：0x0010 + 0x4 * i，其中“i”为图像分量0至3

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSF[3:0]				VSF[3:0]				NB[3:0]				QT[1:0]		HA	HD
RW				RW				RW				RW		RW	RW

位 31:16 Reserved

位 15:12 **HSF[3:0]**：水平采样因数 (Horizontal Sampling Factor)
分量 i 的水平采样因数。

位 11:8 **VSF[3:0]**：垂直采样因数 (Vertical Sampling Factor)
分量 i 的垂直采样因数。

位 7:4 **NB[3:0]**：块数
属于 MCU 中特定颜色的数据单元数减去 1。

位 3:2 **QT[1:0]**：量化表 (Quantization Table)
选择用于分量i的量化表。

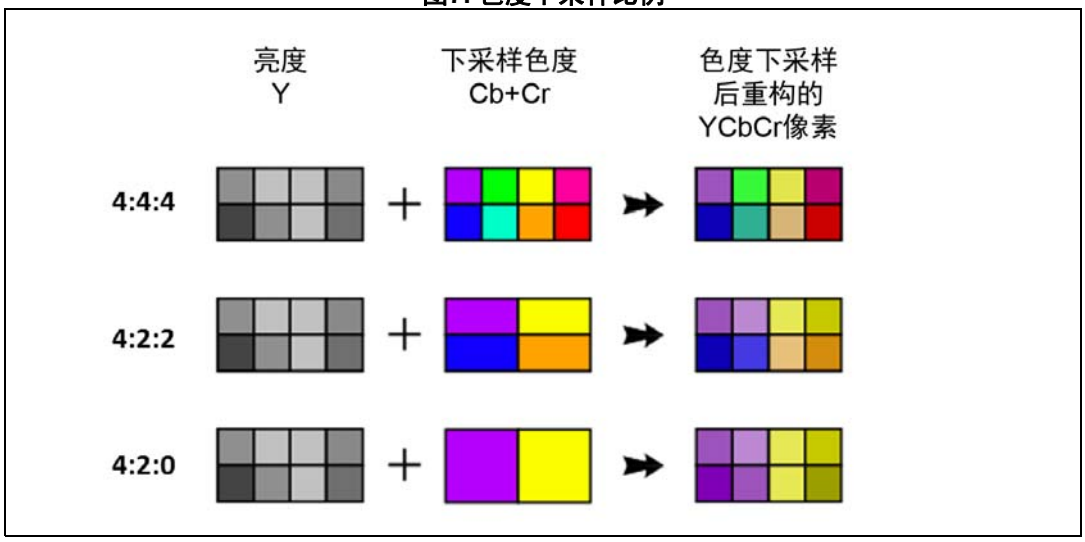
位 1 **HA**：霍夫曼 AC
选择用于编码AC系数的霍夫曼表。

位 0 **HD**：霍夫曼 DC
选择用于编码DC系数的霍夫曼表。

STM32CubeF7/H7 JPEG HAL驱动程序允许用户选择以下色度下采样比例之一：

- 4:4:4 →无色度下采样，保留所有分量Y、Cb和Cr的完整信息。
- 4:2:2 → 以相当于Y分量一半的比例对Cb和Cr进行水平采样（对于两个水平相邻像素，仅保留一个像素的色度信息）。
- 4:2:0 → 以相当于Y分量一半的比例对Cb和Cr进行水平和垂直采样（对于四个相邻像素，仅保留一个像素的色度信息）。

图7. 色度下采样比例

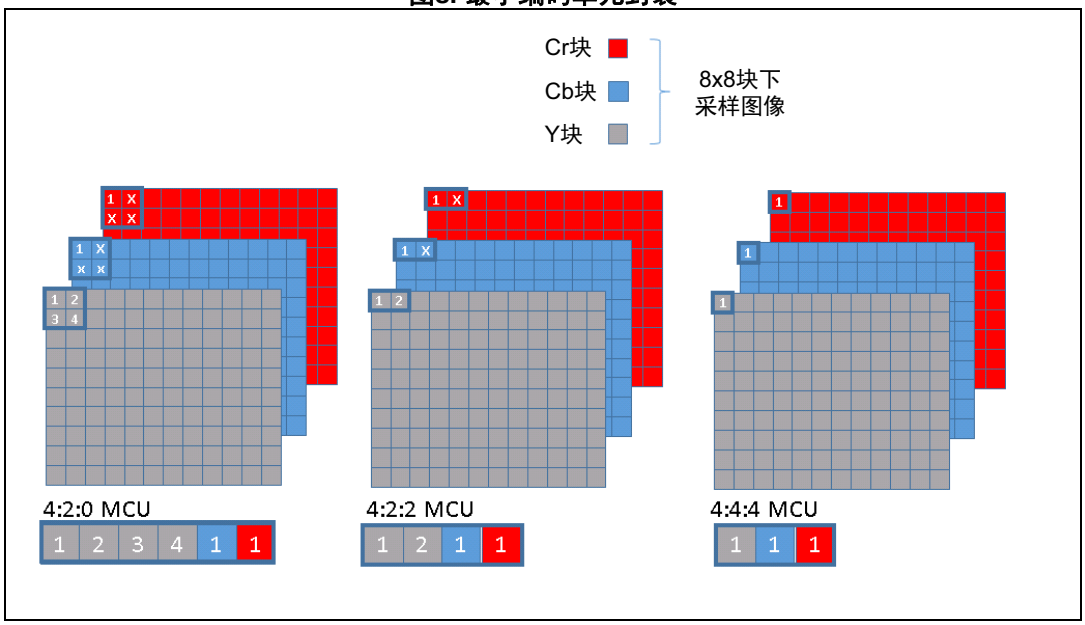


然后，将下采样YCbCr像素封装到名为MCU（最小编码单元）的8x8块中。

每个MCU包含：

- 4:4:4 → 1个8x8 Y块 + 1个8x8 Cb块 + 1个8x8 Cr块 → 共192个字节。
- 4:2:2 → 2个8x8 Y块 + 1个8x8 Cb块 + 1个8x8 Cr块 → 共256个字节。
- 4:2:0 → 4个8x8 Y块 + 1个8x8 Cb块 + 1个8x8 Cr块 → 共384个字节。

图8. 最小编码单元封装



以下是YCbCr色彩空间中的JPEG编解码器寄存器设置，具体取决于色度采样。

2.1.4 CONFR1寄存器设置

- 位31:16 **YSIZE[15:0]**：该字段代表编码/解码图像中的行数。进行解码时，由硬件JPEG编解码器使用JPEG文件头自动填充该寄存器。
- 位8 **HDR**：文件头处理：这是选填字段，仅用于解码。用户可以将该位设置为零以禁用JPEG文件头处理。这种情况下，必须由用户对其他配置寄存器和量化/霍夫曼表进行编程。

如果上一次解码时使能了文件头解析，则还可以通过硬件进行这些寄存器和表格的编程，前提是下一幅图像（将在文件头解析禁用时解码）具有相同的量化表和霍夫曼表以及相同的尺寸、色彩空间和色度下采样。

- 位7:6 **NS[1:0]**：在文件头标记段中，该字段代表分量的数量减1。因此，对于YCbCr色彩空间，将其设置为“2”。在使能了文件头解析的情况下进行解码时，该字段由硬件填充。
- 位5:4 **COLORSPACE[1:0]**：该字段代表量化表的数量减1。因此，对于YCbCr色彩空间，将其设置为1，因为YCbCr（一个用于Y亮度，一个用于Cb和Cr色度）需要两个量化表。在使能了文件头解析的情况下进行解码时，该字段由硬件填充。

请注意，如果用户选择自定义量化表（为每个分量提供个性化表格），则将该字段设置为3 – 1 = 2。

- 位3**DE**：解码时设置为1，编码时设置为0。该字段必须由用户设置，以便选择编码或解码。
- 位1:0**NF[1:0]**：该字段代表颜色分量的数量减1。因此，对于YCbCr色彩空间，将其设置为2。在使能了文件头解析的情况下进行解码时，该字段由硬件填充。

2.1.5 CONFR2寄存器设置

在CONFR2寄存器中，仅位25:0有用，即**NMCU[25:0]**字段。在JPEG图像中，它代表MCU的数量减1。

在文件头解析禁用的情况下进行编码和解码时，对于YCbCr色彩空间，必须将该字段设置如下：

NMCU = (hMCU * vMCU) - 1

其中：hMCU和vMCU分别是每个水平行和垂直列的MCU数。

- 4:4:4 色度采样：

  hMCU = scaled_Image_width / 8

  vMCU = scaled_Image_ Height / 8

其中：scaled_Image_width代表舍入到8的下一个倍数的图像宽度，scaled_Image_Height代表舍入到8的下一个倍数的图像高度。

- 4:2:2 采样：

  hMCU = scaled_Image_width / 16

  vMCU = scaled_Image_ Height / 8

其中：scaled_Image_width代表舍入到16的下一个倍数的图像宽度，scaled_Image_Height代表舍入到8的下一个倍数的图像高度。

- 4:2:0 采样：

  hMCU = scaled_Image_width / 16

  vMCU = scaled_Image_ Height / 16

其中：scaled_Image_width代表舍入到16的下一个倍数的图像宽度，scaled_Image_Height代表舍入到16的下一个倍数的图像高度。

当在文件头解析使能的情况下进行解码时，由硬件填充该字段。但是，它给出的是完整MCU的数量，即它不考虑行和列结尾处的不完整MCU。必须使用以上公式获取MCU的准确数量。

2.1.6 CONFR3寄存器设置

在该寄存器中，仅位15:0有用，即**XSIZE [15:0]**字段。它代表每行的像素数。

2.1.7 CONFR4-7寄存器设置

在YCbCr色彩空间中，使用了三个颜色分量：Y表示亮度，Cb表示蓝色色度，Cr表色红色色度。因此，只使用CONFR4至CONFR6这三个寄存器。

CONFR4寄存器用于亮度（Y）分量。CONFR5和CONFR6寄存器分别用于Cb和Cr色度分量。

在文件头解析禁用的情况下进行编码或解码时，这些寄存器的所有字段均由用户设置，而在文件头解析使能的情况下进行解码时，由硬件设置。

- 位15:12**HSF[3:0]**：该字段代表每个分量的水平采样因子。它是8x8水平块的数量。其设置如下：
  - 对于CONFR5和CONFR6，HSF[3:0]字段总是设置为1，因为Cb和Cr分量总是细分为8x8块：每个MCU 1个。
  - 如下所示，CONFR4 HSF[3:0]字段的设置取决于色度采样：

    4:4:4：设置为1，因为每个MCU有1个8x8（Y）块。

    4:2:2：设置为2，因为每个MCU有2个水平相邻的8x8（Y）块。

    4:2:0：设置为2，因为本例中每个MCU有2个水平相邻的8x8（Y）块（和2个垂直相邻的块，因此VSF字段也设置为2）。
- 位11:8 **VSF[3:0]**：该字段代表每个分量的垂直采样因子。它是8x8垂直块的数量。其设置如下：
  - 对于CONFR5和CONFR6，VSF[3:0]字段总是设置为1，因为Cb和Cr分量总是细分为8x8块：每个MCU 1个。
  - 如下所示，CONFR4 VSF[3:0]字段的设置取决于色度采样：

    4:4:4：设置为1，因为每个MCU有1个8x8（Y）块。

    4:2:2：设置为1，因为本例中每个MCU只有1个垂直相邻的8x8（Y）块。

    4:2:0：设置为2，因为每个MCU有4个8x8（Y）块（2个水平相邻和2个垂直相邻）。
- 位7:4 **NB[3:0]**：该字段代表每个分量的8x8块数量减1。因此，在YCbCr色彩空间中，对于CONFR5和CONFR6寄存器，将其设置为1，因为Cb和Cr分量总是细分为8x8块：每个MCU 1个。
  - 如下所示，CONFR4 **NB[3:0]**字段的设置取决于色度采样：

    4:4:4：设置为0，因为每个MCU有1个8x8（Y）块。

    4:2:2：设置为1，因为每个MCU有2个8x8（Y）块。

    4:2:0：设置为3，因为每个MCU有4个8x8（Y）块。

- 位3:2 **QT[1:0]**：该字段代表与给定分量相关的量化表。
  - 对于CONFR4寄存器，将其设置为0，因为（Y）分量使用QMEM0。对于CONFR5和CONFR6寄存器，将其设置为1，因为Cb和Cr分量使用相同的QMEM1表。
  - 请注意，如果用户选择自定义量化表（为每个分量提供一个表），则将CONFR6寄存器的QT[1:0]字段设置为“2”，因此Cr分量使用QMEM2量化表。
- 位1 **HA[1]**和位[2] **HD[1]**均设置为：
  - 0（对于CONFR4寄存器），因为（Y）分量使用AC霍夫曼表0和DC霍夫曼表0。
  - 1（对于CONFR5和CONFR6寄存器），因为Cb和Cr分量均使用AC霍夫曼表1和DC霍夫曼表1。

2.2 灰度色彩空间

灰度色彩空间只使用一个代表亮度（Y）的颜色分量。因此，这种情况下色度下采样不适用，MCU总是包含一个8x8（Y）块。

2.2.1 RGB至灰度的转换

可使用以下公式实现从RGB色彩空间至灰度的转换：

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

请注意，该公式是*图 1*中（Y）分量的子集。

2.2.2 灰度量化表

由于使用一个分量（即（Y）亮度）代表灰度色彩空间，因此只需要一个量化表。*图 3*所示表格的使用方式与质量因子和Z形扫描的相同。然后，使用获得的比例量化表（根据质量因子）填充硬件JPEG编解码器QMEM0表。

以下是灰度色彩空间中的JPEG编解码器寄存器设置。

2.2.3 CONFR1寄存器设置

- 位7:6 **NS[1:0]**：在JPEG文件头中，该字段代表用于扫描的分量数量减1。因此，对于灰度色彩空间，将其设置为“0”。在使能了文件头解析的情况下进行解码时，该字段由硬件填充。
- 位5:4 **COLORSPACE[1:0]**：该字段代表量化表的数量减1。因此，对于灰度色彩空间，将其设置为“0”（需要1个量化表）。
- 位1:0 **NF[1:0]**：该字段代表颜色分量的数量减1。因此，对于灰度色彩空间，将其设置为“0”。在使能了文件头解析的情况下进行解码时，该字段由硬件填充。

对于该寄存器的所有其他字段，YCbCr部分所述规则同样适用（YSIZE、HDR和DE字段）。

2.2.4 CONFR2寄存器设置

由于在灰度色彩空间中MCU总是包含8x8（Y）块，因此将NMCU字段设置如下：

NMCU = (hMCU * vMCU) - 1

其中：hMCU和vMCU分别是每个水平行和垂直列的MCU数。

hMCU = scaled_Image_width / 8

vMCU = scaled_Image_ Height / 8

其中：scaled_Image_width代表舍入到8的下一个倍数的图像宽度，scaled_Image_Height代表舍入到8的下一个倍数的图像高度。

2.2.5 CONFR3寄存器设置

在该寄存器中，仅位15:0有用，即XSIZE [15:0]字段。它代表每行的像素数。

2.2.6 CONFR4-7寄存器设置

在灰度色彩空间中，仅CONFR4寄存器有用，它代表唯一的（Y）分量的设置。在灰度色彩空间中，该寄存器的设置与YCbCr 4:4:4的情况类似。即：

- HSF [3:0]和VSF [3:0]：均设置为1。
- NB [3:0]：设置为0。
- QT[1:0]：设置为0。
- HA[1]和HD[1]均设置为0。

2.3 CMYK色彩空间

CMYK是用于打印应用的色彩空间。通过3种颜色（青色、品红色和黄色）和一种定位套版色（即墨水量）表示CMYK图像。因此，对于JPEG编码/解码，CMYK对应于4个颜色分量。对于YCbCr，不需要进行分量下采样。可对全部4个分量使用同一量化表。

对于JPEG编码/解码，CMYK MCU包含4个8x8块，顺序如下：一个8x8青色块、一个8x8品红色块、一个8x8黄色块和最后一个8x8定位套版色块。因此，MCU的总大小为8x8 x 4 = 256字节。

2.3.1 CMYK量化表

可对全部4个分量使用同一量化表。虽然如此，硬件JPEG编解码器为定义每个分量的单独表格提供了可能性。STM32CubeF7/H7 JPEG HAL驱动程序默认对所有CMYK分量使用*图 3*所示表格。JPEG HAL驱动程序还为用户按颜色分量定义量化表提供了可能性（本例中为4个量化表）。

如需自定义量化表，用户必须提供4个量化表（每个分量一个量化表）。这些表格分别用于对（使用质量因子进行比例计算后）硬件JPEG编解码器的QMEM0至QMEM3 RAM表进行编程（其中，QMEM2表位于偏移地址0x00D0，QMEM3表位于偏移地址0x0110）。

HAL函数“HAL_JPEG_SetUserQuantTables”是用于自定义用户量化表的API。

质量因子和Z形扫描的方式同样适用。然后，使用获得的比例量化表（根据质量因子）填充硬件JPEG编解码器QMEM0表（或QMEM0至QMEM3）。

以下是CMYK色彩空间中的JPEG编解码器寄存器设置。

2.3.2 CONFR1寄存器设置

- 位7:6 **NS[1:0]**：在JPEG文件头中，该字段代表用于扫描的分量数量减1。因此，对于CMYK色彩空间，将其设置为“3”。在使能了文件头解析的情况下进行解码时，该字段由硬件填充。
- 位5:4 **COLORSPACE[1:0]**：该字段代表量化表的数量减1。因此，对于CMYK色彩空间，将其设置为默认值“0”（对全部4个分量使用1个量化表）。

如果用户选择自定义量化表（为每个分量提供一个表格），则将该字段设置为3（使用4个量化表）。

- 位1:0 **NF[1:0]**：该字段代表颜色分量的数量减1。对于CMYK色彩空间，将其设置为“3”。在使能了文件头解析的情况下进行解码时，该字段由硬件填充。

对于该寄存器的所有其他字段，YCbCr部分所述规则同样适用（YSIZE、HDR和DE字段）。

2.3.3 CONFR2寄存器设置

在CMYK色彩空间中，NMCU字段设置如下：

NMCU = (hMCU * vMCU) - 1

其中：hMCU和vMCU分别是每个水平行和垂直列的MCU数

hMCU = scaled_Image_width / 8

vMCU = scaled_Image_Height / 8

其中：scaled_Image_width代表舍入到8的下一个倍数的图像宽度，scaled_Image_Height代表舍入到8的下一个倍数的图像高度。

2.3.4 CONFR3寄存器设置

在该寄存器中，仅位15:0有用，即XSIZE [15:0]字段。它代表每行的像素数。

2.3.5 CONFR4-7寄存器设置

在CMYK色彩空间中，使用了4个颜色分量。因此，设置寄存器CONFR4至CONFR7（每个分量一个寄存器）。

在文件头解析禁用的情况下进行编码或解码时，这些寄存器的所有字段均由用户设置。而在文件头解析使能的情况下进行解码时，则由硬件设置。

每个CMYK MCU包含4个8x8块：每个分量1个块：

- HSF [3:0]和VSF [3:0]：均设置为1（在CMYK中）。
- NB [3:0]：设置为0。

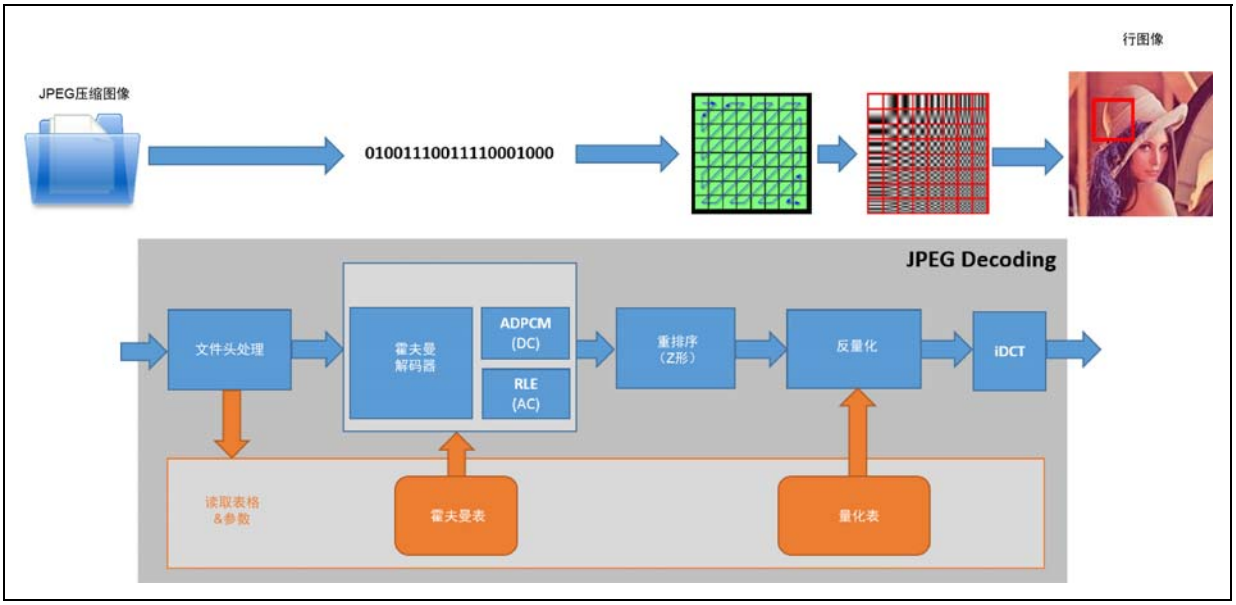
QT[1:0]字段默认设置为0，因此全部4个分量均使用同一量化QMEM0表。当用户选择每个分量一个表时，分别为寄存器CONFR4至CONFR7将该字段设置为0、1、2和3。（因此，每个分量将分别使用QMEM0至QMEM3量化表）。

HA[1]和HD[1]均设置为0：所有分量均使用相同霍夫曼AC和DC表。

3 JPEG 解码

硬件JPEG编解码器能够按照JPEG标准（*ISO/IEC 10918-1*）中的规定对JPEG压缩图像进行解码。它可以解析JPEG文件头并更新编解码器寄存器（CONFR1至CONFR7寄存器）、量化表（QMEM）和霍夫曼表。

图9. JPEG解码流



进行解码时，在MCU块中组织JPEG编解码器输出数据。MCU包含（图像的）许多8x8块，具体取决于色彩空间和色度采样，详情见前文。

然后，应用必须重新组织这些块，移除色度采样并将颜色转换为RGB，以便显示解码图像。

简而言之，必须按以下方式组织MCU：

表2. JPEG MCU组织

色彩空间	色度采样	MCU组织	MCU大小（以字节计）
YCbCr	4:4:4	一个Y 8x8块 + 一个Cb 8x8块 + 一个Cr 8x8块	192
	4:2:2	两个Y 8x8块 + 一个Cb 8x8块 + 一个Cr 8x8块	256
	4:2:0	四个Y 8x8块 + 一个Cb 8x8块 + 一个Cr 8x8块	384
灰度	N.A	一个Y 8x8块	64
CMYK	N.A	一个青色8x8块 + 一个品红色8x8块 + 一个黄色8x8块 + 一个8x8定位套版色块	256

应用可以等待硬件JPEG编解码器结束解码操作并输出所有MCU，然后将这些块转换为RGB像素。或者，它可以在某些MCU可用后开始MCU至RGB的转换。

在STM32CubeF7/H7中，JPEG HAL驱动程序能够从硬件JPEG编解码器按块（具有用户指定大小）获取输出数据。当应用需要在输出MCU可用后立即转换MCU时，以及应用必须处理不同色彩空间和色度采样时，建议将输出块大小设置为768字节的倍数。通过从JPEG编解码器输出按块（包含768字节的倍数）获取数据，可使块中包含完整MCU：

根据色彩空间和色度采样，768字节相当于：

- YCbCr 4:4:4 → 4个MCU
- YCbCr 4:2:2 → MCU
- YCbCr 4:2:0 → 2个MCU
- 灰度 → 12个MCU
- CMYK：→ 3个MCU

请注意，硬件JPEG编解码器总是输出完整的MCU。如果原始图像宽度和/或高度不是8或16的倍数（取决于色彩空间和色度采样），则行和/或列末尾的MCU将以空数据结尾（通常为之前像素数据的重复）。在将输出MCU转换为RGB像素时，必须删除这些额外数据。

3.1 MCU重排序和转换

3.1.1 在STM32H7x3系列器件上

Chrom-Art Accelerator™外设也称为DMA2D，在STM32H7x3系列器件上实现，提供能够将YCbCr MCU（作为硬件JPEG编解码器的输出）转换为RGB像素并重排序的新特性，全部进行色度上采样。DMA2D支持的色度采样因子为：4:4:4、4:2:2和4:2:0。

DMA2D能够处理最多2个图层，即前景和背景。只有在STM32H7x3系列器件上，前景图层才能将YCbCr MCU块转换为RGB像素。

为了配置将YCbCr MCU转换为RGB像素的DMA2D，需要以下寄存器设置：

FGPFCCR 寄存器：

- 位19:18 **CSS[1:0]**：色度下采样选择
  - 00：4:4:4（无色度下采样）
  - 01: 4:2:2
  - 10: 4:2:0
- 位3:0 **CM[3:0]**：输入颜色模式选择
  1011：YCbCr

FGOR 寄存器：

该寄存器用于选择DMA2D前景输入行偏移地址。必须按如下方式编程：

- 色度采样4:4:4
  - 如果图像宽度为8个像素的倍数，将FGOR设置为0
  - 否则，FGOR = scaled_Image_width - Image_width
    scaled_Image_width是舍入到8的下一个倍数的图像宽度（以像素计）。
- 色度采样4:2:2或4:2:0
  - 如果图像宽度为16个像素的倍数，将FGOR设置为0
    否则，FGOR = scaled_Image_width - Image_width
    scaled_Image_width是舍入到16的下一个倍数的图像宽度（以像素计）。
    当图像尺寸不是8或16的倍数时，FGOR寄存器的设置能够删除覆盖行末尾区域的MCU中的额外数据。

为了进行像素格式转换，通常必须完成其他DMA2D寄存器配置。

必须通过软件处理其他颜色空间（灰度和CMYK）。下一段描述如何使用STM32CubeF7/H7中提供的JPEG实用工具来执行MCU至RGB的转换。它适用于STM32F76/77xxx器件（所有色彩空间）和STM32H7x3系列器件（灰度和CMYK）。

STM32CubeH7中提供的多个JPEG解码示例显示了如何使用硬件JPEG编解码外设：

- 使用硬件JPEG编解码外设解码和显示JPEG压缩文件。Chrom-Art Accelerator™（DMA2D）用于YCbCr至RGB的转换。
- 解码并显示MJPEG视频文件：使用硬件JPEG编解码外设和用于YCbCr至RGB转换的Chrom-Art Accelerator™（DMA2D）。

*表 3*汇总了STM32CubeH7固件包中提供的JPEG解码示例：

表3. STM32CubeH7固件包中JPEG解码示例的列表

示例	说明
JPEG_DecodingFromFLASH_DMA	为了解码并显示保存在内部闪存中的JPEG压缩图像，可使用硬件JPEG解码器（DMA模式）和用于YCbCr至RGB转换的DMA2D。
JPEG_DecodingUsingFs_DMA	为了解码并显示保存在SD卡中的JPEG压缩图像， 可使用硬件JPEG解码器（DMA模式）和用于YCbCr至RGB转换的DMA2D。
JPEG_DecodingUsingFs_Interrupt	为了解码并显示保存在SD卡中的JPEG压缩图像， 可使用硬件JPEG解码器（中断模式）和用于YCbCr至RGB转换的DMA2D。
JPEG_DecodingUsingFs_Polling	为了解码并显示保存在SD卡中的JPEG压缩图像， 可使用硬件JPEG解码器（轮询模式）和用于YCbCr至RGB转换的DMA2D。
JPEG_MJPEG_VideoDecoding	为了解码并显示保存在SD卡中的MJPEG视频文件，可使用硬件JPEG解码器。使用DMA2D执行YCbCr至RGB的转换。
JPEG_MJPEG_VideoDecodingFromQSPI	为了解码并显示保存在外部Quad-SPI闪存中的MJPEG视频文件，可使用硬件JPEG解码器。使用DMA2D执行YCbCr至RGB的转换。

3.1.2 在STM32F76/77xxx器件上

使用专用软件层将MCU块转换为可在显示器上显示的RGB像素。MCU至RGB像素的转换包括色度上采样和YCbCr至RGB的颜色转换。该软件层可在STM32CubeF7/H7中的\Utilities\JPEG目录下找到。

STM32CubeF7中提供的多个JPEG解码示例显示了如何使用硬件JPEG外设：

- 使用硬件JPEG编解码外设解码和显示JPEG压缩文件。通过JPEG实用工具软件执行YCbCr块至RGB像素的转换。
- 解码和显示MJPEG视频文件：使用JPEG解码器外设。通过JPEG实用工具软件执行YCbCr块至RGB像素的转换。

*表 4*汇总了STM32CubeF7固件包中提供的JPEG解码示例：

表4. STM32CubeF7固件包中JPEG解码示例的列表

示例	说明
JPEG_DecodingFromFLASH_DMA	为了解码并显示保存在内部闪存中的JPEG压缩图像，可使用硬件JPEG解码器（DMA模式）和用于YCbCr至RGB转换的JPEG实用工具软件
JPEG_DecodingUsingFs_DMA	为了解码并显示保存在SD卡中的JPEG压缩图像，可使用硬件JPEG解码器（DMA模式）和用于YCbCr至RGB转换的JPEG实用工具软件。
JPEG_DecodingUsingFs_Interrupt	为了解码并显示保存在SD卡中的JPEG压缩图像，可使用硬件JPEG解码器（中断模式）和用于YCbCr至RGB转换的JPEG实用工具软件。
JPEG_DecodingUsingFs_Polling	为了解码并显示保存在SD卡中的JPEG压缩图像，可使用硬件JPEG解码器（轮询模式）和用于YCbCr至RGB转换的JPEG实用工具软件。
JPEG_MJPEG_VideoDecoding	为了解码并显示保存在SD卡中的MJPEG视频文件，可使用硬件JPEG解码器。通过JPEG实用工具软件执行YCbCr至RGB的转换。

使用该实用工具解码需要执行下述步骤。

- 复制用户应用文件夹中的jpeg_utils_conf_template.h文件并进行如下修改：
  - 将其重命名为“jpeg_utils_conf.h”。
  - 取消注释include行（#include "stm32fXxx_hal.h" 和 #include "stm32fXxx_hal_jpeg.h）并分别修改为（#include "stm32f7xx_hal.h" 和 #include "stm32f7xx_hal_jpeg.h）。
  - 使用#define JPEG_RGB_FORMAT选择输出RGB格式ARGB8888、RGB888或RBG565。
  - 或者，可使用#define JPEG_SWAP_RB选择红蓝调换（设置为1以调换像素的红蓝顺序）
- 在应用中调用函数JPEG_InitColorTables，以便初始化红色、绿色和蓝色查找表。该函数能够初始化4个查找表（CR_RED_LUT、CB_BLUE_LUT、CR_GREEN_LUT和CB_GREEN_LUT），用于避免YCbCr至RGB颜色转换过程中的乘法和浮点计算（按照*图 1*所示公式）。应用中只能执行一次该步骤，即使必须执行多次YCbCr至RGB的转换和/或必须转换多幅JPEG图像。
- 下一步是按照色彩空间和色度采样，通过调用函数JPEG_GetDecodeColorConvertFunc选择YCbCr转换函数。该函数还根据图像设置（尺寸、色彩空间和色度采样）对必要的内部变量进行初始化。该函数的参数如下：
  - JPEG_ConfTypeDef *pJpegInfo：指向包含JPEG图像信息（色彩空间、色度下采样、图像高度和宽度）的JPEG_ConfTypeDef结构的指针。在通过硬件JPEG编解码器完成JPEG文件头解析后，可在HAL驱动程序回调函数HAL_JPEG_InfoReadyCallbackHAL_JPEG_InfoReadyCallback下找到这些信息。还可以使用函数HAL_JPEG_GetInfo检索这些信息（在文件头解析后或在JPEG解码操作结束时）。
  - JPEG_YCbCrToRGB_Convert_Function *pFunction：该参数返回指向函数的指针，该函数用于将JPEG编解码器输出MCU转换为目标图像帧缓冲区中的RGB像素。
  - uint32_t *ImageNbMCUs：该参数用于根据图像尺寸、色彩空间和色度采样向用户返回MCU总数。
- 然后，可以调用转换函数将YCbCr MCU转换为目标RGB帧缓冲区中的RGB像素。转换函数的参数如下：
  - uint8_t *pInBuffer：包含许多完整MCU的缓冲区（硬件JPEG编解码器的输出）
  - uint8_t *pOutBuffer：保存RGB图像的RGB目标缓冲区。
  - uint32_t BlockIndex：当前输入缓冲区（pInBuffer）中第一个MCU相对于MCU总数的索引。
  - uint32_t DataCount：输入缓冲区（pInBuffer）大小，以字节为单位。

– uint32_t *ConvertedDataCount：留作将来使用（用于返回从输入缓冲区转换的字节数）。

如果按块转换（不是一次性），转换函数将返回从输入缓冲区转换到输出RGB缓冲区的MCU数量，用于下一次调用该函数时的参数BlockIndex。

*表 5*提供了每个色彩空间的转换函数作为参考。在“jpeg_utils.c”源文件中，这些函数以静态实现。应用无需直接调用这些函数，但需要调用“JPEG_GetDecodeColorConvertFunc()”，以便检索对应于给定图像色彩空间和色度采样的函数指针。

表5. MCU至RGB内部转换函数列表

色彩空间	色度采样	MCU至RGB的转换函数
YCbCr	4:4:4	JPEG_MCU_YCbCr444_ARGB_ConvertBlocks()
	4:2:2	JPEG_MCU_YCbCr422_ARGB_ConvertBlocks()
	4:2:0	JPEG_MCU_YCbCr420_ARGB_ConvertBlocks()
灰度	N.A	JPEG_MCU_Gray_ARGB_ConvertBlocks()
CMYK	N.A	JPEG_MCU_YCCK_ARGB_ConvertBlocks()

MCU块至RGB的转换函数作用于整个MCU并假设图像宽度和高度为8或16的倍数（取决于色彩空间和色度采样）。与此同时，硬件JPEG编解码器总是输出完整MCU，并在转换为RGB像素时使图像尺寸（高度和宽度）为8或16的倍数。

在解码尺寸（宽度和高度）不是8或16的倍数的图像时，为了使用JPEG实用工具层，可使用以下技巧：

- 在调用“JPEG_GetDecodeColorConvertFunc()”之前，根据色彩空间和色度采样，按照如下方式更新结构pJpegInfo的ImageWidth和ImageHeight：
  - YCbCr 4:4:4，CMYK的灰度：将ImageWidth和ImageHeight均舍入到8的下一个倍数。
  - YCbCr 4:2:2：将ImageWidth舍入到16的下一个倍数，并将ImageHeight舍入到8的下一个倍数。
  - YCbCr 4:2:0，CMYK的灰度：将ImageWidth和ImageHeight均舍入到16的下一个倍数。
- 先进行MCU转换。如上文所述，输出RGB图像的高度和宽度将扩展到8或16的下一个倍数。

- 使用DMA2D将获得的图像修剪为原始尺寸：按照STM32H743xx转换实例进行DMA2D输入行偏移地址（FGOR寄存器）编程。即：

  FGOR寄存器：用于选择DMA2D前景输入行偏移地址。必须按如下方式编程：

  – YCbCr 4:4:4，CMYK的灰度：
  - FGOR = scaled_Image_width - Image_width

  scaled_Image_width是舍入到8的下一个倍数的图像宽度（以像素计）。

  – 色度采样4:2:2或4:2:0：
  - FGOR = scaled_Image_width - Image_width

  scaled_Image_width是舍入到16的下一个倍数的图像宽度（以像素计）。

DMA2D FGOR寄存器的设置能够删除由尺寸（高度和宽度）舍入产生的额外像素。可将DMA2D配置为存储器至存储器或像素的格式转换（以便更改输出图像颜色格式）。

3.2 JPEG解码性能

下表提供了产品的解码性能：

- STM32H7x3系列：使用硬件JPEG外设和用于YCbCr至RGB转换的DMA2D。
- STM32F76/77xxx：使用硬件JPEG外设和用于YCbCr至RGB转换的软件实用工具。

请注意，这些性能测量值通过位于外部SDRAM上的JPEG缓冲区（RGB和YCbCr）给出。

表6. STM32H7x3 JPEG解码性能

产品	图像分辨率	解码（ms）		
		硬件解码	DMA2D YCbCr至RGB的转换	总时间
STM32H743I	VGA：640 x 480	4	6	**10**（100 fps）
	QVGA：320x240	1	1.5	**2.5**（400 fps）

表7. STM32F76/77xxx JPEG解码性能

产品	图像分辨率	解码（ms）		
		硬件解码	软件 YCbCr至RGB的转换	总时间
STM32F769I	VGA：640 x 480	4	22	**26**（38 fps）
	QVGA：320x240	1	5	**6**（166 fps）

以上测量在*表 8*中给出的条件下执行。

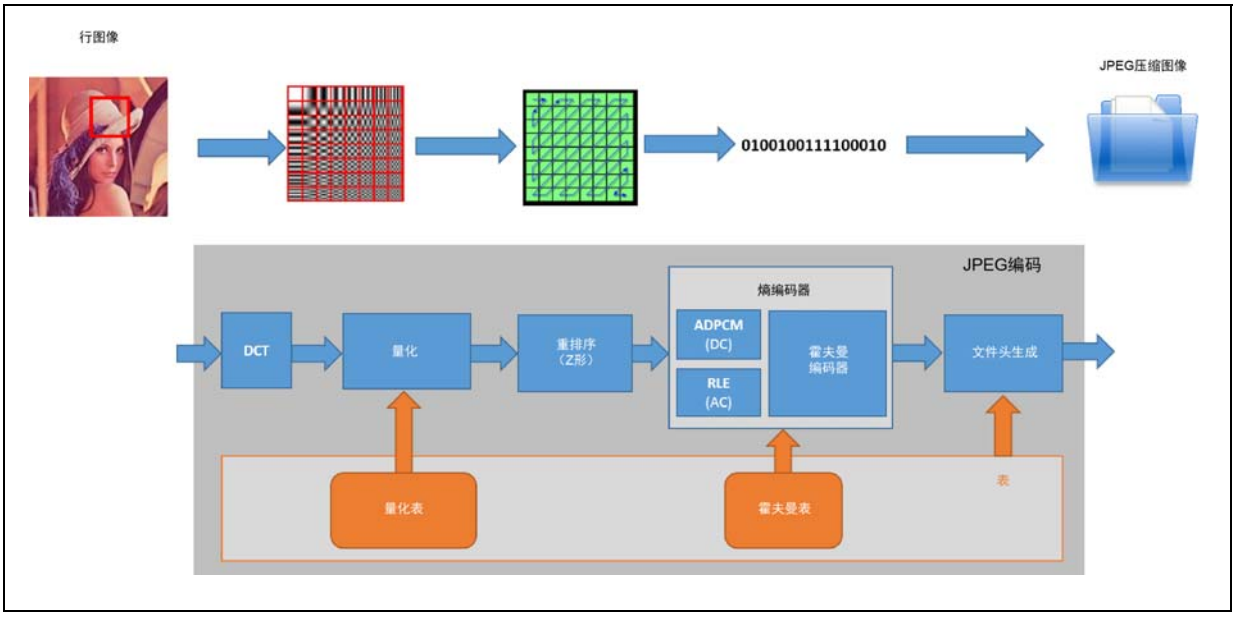
表8. JPEG解码性能测量条件

产品	STM32F76/77xxx	STM32H7x3
板	STM32F769I_EVAL rev.B	STM32H743I_EVAL rev.B
CPU 频率	200 MHz	400 MHz
硬件JPEG编解码器频率	200 MHz	200 MHz
IDE/编译器	面向Arm（7.80版本）的IAR嵌入式workbench	面向Arm（7.80版本）的IAR嵌入式workbench
编译器优化	高速	高速
SDRAM外部存储器	参考：IS42S32800G-6BLI 时钟频率：100 MHz 存取：ROW存取（非文件系统存取）	参考：IS42S32800G-6BLI 时钟频率：200 MHz 存取：ROW存取（非文件系统存取）
JPEG图像	分辨率：640 x 480 颜色格式：YCbCr 色度采样：4:2:0	分辨率：640 x 480 颜色格式：YCbCr 色度采样：4:2:0

4 JPEG 编码

硬件JPEG编解码器能够将图像压缩成符合JPEG文件交换格式（JFIF）的JPEG文件，包含必要的文件头和片段。

图10. JPEG编码流



STM32CubeF7/H7中的JPEG HAL驱动程序提供必要的函数用于执行编码操作，包括使用默认的霍夫曼表初始化编解码器。

在编码模式下，预期根据*表 2*中描述的色彩空间和色度采样在MCU块中组织JPEG编解码器输入数据。

应用必须重新组织并将输入RGB像素转换为MCU块。对于YCbCr色彩空间，还必须进行色度下采样。硬件JPEG编解码器需要完整的MCU。如果RGB图像尺寸（高度和宽度）不是8或16的倍数，则必须在行和列的末尾添加额外像素，以便生成由8x8块组成的完整MCU。然而，在硬件JPEG编解码器寄存器CONFR1和CONFR3中，必须设置原始图像尺寸（在YSIZE和XSIZE字段中）。

随STM32CubeF7/H7提供的软件实用工具可用于执行从输入RGB像素至MCU块的必要转换，MCU块可馈给硬件JPEG编解码器。STM32CubeF7/H7提供的示例显示了如何将RGB图像编码为JPEG压缩文件（使用该软件实用工具进行MCU生成）。

示例位于以下目录下：

– STM32CubeF7: \Firmware\Projects\STM32F769I_EVAL\Examples\JPEG
– STM32CubeH7: \Firmware\Projects\STM32H743I_EVAL\Examples\JPEG

*表 9*汇总了可用的编码示例：

表9. STM32CubeF7/H7固件包中JPEG编码示例的列表

示例	说明
JPEG_EncodingFromFLASH_DMA	为了对保存在内部闪存中的RGB图像进行编码，使用硬件JPEG编解码器（DMA模式）并将得到的JPEG压缩文件保存到SD卡中。 通过JPEG实用工具软件执行RGB至YCbCr的转换（必须在编码之前）。
JPEG_EncodingUsingFs_DMA	为了对保存在SD中的bmp图像进行编码，使用硬件JPEG编解码器（DMA模式）并将得到的JPEG压缩文件保存到SD卡中。 通过JPEG实用工具软件执行RGB至YCbCr的转换（必须在编码之前）。

使用JPEG实用工具编码需要执行下述步骤。

- 复制用户应用文件夹中的jpeg_utils_conf_template.h文件并进行如下修改：
  - 将其重命名为“jpeg_utils_conf.h”。
  - 取消注释include行：#include "stm32fXxx_hal.h" 和 #include "stm32fXxx_hal_jpeg.h并分别修改为：
    - 使用STM32CubeF7：#include "stm32f7xx_hal.h" 和 #include "stm32f7xx_hal_jpeg.h。
    - 使用STM32CubeH7：#include "stm32h7xx_hal.h" 和 #include "stm32h7xx_hal_jpeg.h。
  - 使用#define JPEG_RGB_FORMAT选择输出RGB格式ARGB8888、RGB888或RBG565。
  - 或者，可使用#define JPEG_SWAP_RB选择红蓝调换（设置为1以调换像素的红蓝顺序）
- 在用户应用中调用函数JPEG_InitColorTables，以便初始化红色、绿色和蓝色查找表。该函数能够初始化不同的查找表，用于避免颜色转换过程中的乘法和浮点计算（按照*图 1*所示公式）。应用中只能执行一次该步骤，即使必须编码多幅图像。
- 下一步是按照色彩空间和色度采样选择RGB至YCbCr的转换函数。这通过调用函数JPEG_GetEncodeColorConvertFunc来完成。该函数还根据图像设置（尺寸、色彩空间和色度采样）对RGB至YCbCr MCU转换的必要内部变量进行初始化。该函数的参数如下：
  - JPEG_ConfTypeDef *pJpegInfo：指向包含图像信息（色彩空间、色度下采样、图像高度和宽度）的JPEG_ConfTypeDef结构的指针。用户必须填充这些信息用于编码。
  - PEG_RGBToYCbCr_Convert_FunctiopFunction：该参数返回指向函数的指针，该函数用于将RGB像素转换为MCU。
  - uint32_t *ImageNbMCUs：该参数用于根据图像尺寸、色彩空间和色度采样向用户返回MCU总数。

- 然后，可以调用转换函数将输入图像RGB像素转换为YCbCr MCU。转换函数的参数如下：
  - uint8_t *pInBuffer：包含要转换为MCU的RGB像素的缓冲区。由于MCU对应于原始图像的8x8块，因此输入缓冲区必须对应于：

    -输入RGB图像的8的倍数行（对于YCBCR 4:4:4、YCbCr 4:2:2、灰度或CMYK）。

    - 输入RGB图像的16的倍数行（对于YCbCr 4:2:0）。
  - uint8_t *pOutBuffer：MCU目标缓冲区。然后，使用该缓冲区馈给硬件JPEG编解码器。
  - uint32_t BlockIndex：当前输入缓冲区（pInBuffer）中第一个MCU相对于MCU总数的索引。
  - uint32_t DataCount：输入缓冲区（pInBuffer）大小，以字节为单位。
  - uint32_t *ConvertedDataCount：返回从输入缓冲区转换的字节数。

    如果按块转换（不是一次性），转换函数将返回从输入缓冲区转换到输出MCU缓冲区的MCU数量，用于下一次调用该函数时的参数BlockIndex。

*表 10*提供了每个色彩空间的转换函数作为参考。在“jpeg_utils.c”源文件中，这些函数以静态实现。应用无需直接调用这些函数，但需要调用“JPEG_GetEncodeColorConvert Func ()”，以便检索对应于给定图像色彩空间和色度采样的函数指针。

表10. RGB至MCU内部转换函数列表

色彩空间	色度采样	RGB至MCU转换函数
YCbCr	4:4:4	JPEG_ARGB_MCU_YCbCr444_ConvertBlocks ()
	4:2:2	JPEG_ARGB_MCU_YCbCr422_ConvertBlocks ()
	4:2:0	JPEG_ARGB_MCU_YCbCr420_ConvertBlocks ()
灰度	N.A	JPEG_ARGB_MCU_Gray_ConvertBlocks ()
CMYK	N.A	JPEG_ARGB_MCU_YCCK_ConvertBlocks ()

在使用3种可用模式中的一种开始编码操作之前，必须调用HAL驱动程序函数“HAL_JPEG_ConfigEncoding”，以便使用要编码的图像的参数填充硬件JPEG编解码器寄存器：

- 轮询模式：使用HAL驱动程序函数HAL_JPEG_Encode
- 中断模式：使用HAL驱动程序函数HAL_JPEG_Encode_IT
- DMA模式：使用HAL驱动程序函数HAL_JPEG_Encode_DMA

然后，必须将使用转换实用工具函数检索到的MCU用作以上HAL转换函数的输入。

4.1 JPEG编码性能

下面的表格提供了STM32H7x3系列和STM32F76/F77xxx的编码性能：使用硬件JPEG外设和用于RGB至YCbCr转换的软件实用工具。

请注意，这些性能测量值通过位于外部SDRAM上的JPEG缓冲区（RGB和YCbCr）给出。

表11. STM32H7x3 JPEG编码性能

产品	图像分辨率	编码（ms）		
		软件RGB到YCbCr的转换	硬件编码	总时间
STM32H743I	VGA：640 x 480	58	4	**62**（16 fps）
	QVGA：320x240	14	1	**15**（66 fps）

表12. STM32F76/77xxx JPEG编码性能

产品	图像分辨率	编码（ms）		
		软件RGB到YCbCr的转换	硬件编码	总时间
STM32F769I	VGA：640 x 480	103	4	**107**（9 fps）
	QVGA：320x240	27	1	**28**（35 fps）

以上测量在*表 13*中给出的条件下执行。

表13. JPEG编码性能测量条件

产品	STM32F76/77xxx	STM32H7x3
板	STM32F769I_EVAL rev.B	STM32H743I_EVAL rev.B
CPU 频率	200 MHz	400 MHz
硬件JPEG编解码器频率	200 MHz	200 MHz
IDE/编译器	面向Arm（7.80版本）的IAR嵌入式workbench	面向Arm（7.80版本）的IAR嵌入式workbench
编译器优化	高速	高速
SDRAM外部存储器	参考：IS42S32800G-6BLI 时钟频率：100 MHz 存取：ROW存取（非文件系统存取）	参考：IS42S32800G-6BLI 时钟频率：200 MHz 存取：ROW存取（非文件系统存取）
JPEG图像	分辨率：640 x 480 颜色格式：YCbCr 色度采样：4:2:0	分辨率：640 x 480 颜色格式：YCbCr 色度采样：4:2:0

5 结论

STM32F7/H7硬件JPEG编解码外设为JPEG编码/解码操作提供硬件加速，从而显著提高性能。相比于使用软件进行JPEG编码/解码（libjpeg示例），它还能减少基于JPEG的应用的固件内存占用量（RAM和ROM）。

硬件JPEG编解码器符合JPEG标准（JPEG ISO/IEC 10918-1 ITU-T建议T.81）。随STM32CubeF7/H7固件包提供软件处理，用于处理YCbCr MCU块与RGB像素之间的相互转换，以便符合JPEG文件交换格式（JFIF）。

使用STM32H7x3系列器件，当在YCbCr色彩空间中解码图像时，可使用DMA2D外设为MCU至RGB的转换加速。

STM32CubeF7/H7中提供的多个编码/解码示例显示了如何结合JPEG软件实用工具或DMA2D外设使用JPEG HAL驱动程序。

本应用笔记描述了具有不同图像参数时硬件JPEG编解码器的不同寄存器设置（JPEG HAL驱动程序包括寄存器设置）。它提供了关于如何使用JPEG软件实用工具执行RGB像素与MCU块（供硬件JPEG编解码器使用）之间所需的相互转换的指南。本应用笔记还提供了在解码YCbCr JPEG压缩图像时，使用该外设将硬件JPEG编解码器输出MCU转换为RGB像素所必需的DMA2D设置。只有在STM32H7x3系列器件上才能使用DMA2D的这一特性。

6 版本历史

表14. 文档版本历史

日期	版本	变更
2017年11月14日	1	初始版本。

表15. 中文文档版本历史

日期	版本	变更
2018年8月10日	1	中文初始版本。

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。本文档的中文版本为英文版本的翻译件，仅供参考之用；若中文版本与英文版本有任何冲突或不一致，则以英文版本为准。