TJpgDec Module Application Note

TJpgDec 模块应用说明

How to Use

First of all, you should build and run the sample program shown below. This is a typical usage of TJpgDec module and it helps to narrow down the problem on debugging.

首先，您应该构建并运行如下所示的示例程序。这是 TJpgDec 模块的典型用法，它有助于缩小调试问题的范围。

The decompression session is divided in two stages. The first stage is to analyze the JPEG image and the second stage is to decompress it.

解压缩会话分为两个阶段。第一阶段是分析 JPEG 图像，第二阶段是对其进行解压缩。

1. Initialize input stream. (e.g. open a file)
2. Allocate JPEG decompression object and work area.
3. Call **jd_prepare()** to analyze and prepare to decompress the JPEG image.
4. Initialize output device with the image info in the decompression object.
5. Call **jd_decomp()** to decompress the JPEG image.

1. 初始化输入流。（例如打开文件）

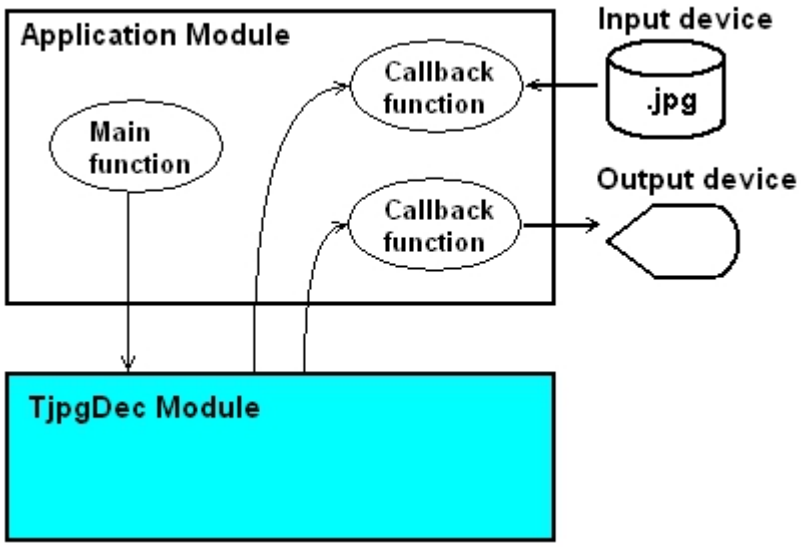
2. 分配 JPEG 解压缩对象和工作区域。

3. 调用 jd_prepare()对 JPEG 图像进行分析并准备解压缩。

4. 使用解压缩对象中的图像信息初始化输出设备。

5. 调用 jd_decomp()对 JPEG 图像进行解压缩。

System Organization



Example

```
/*-----------------------------------------------*/
/* TJpgDec Quick Evaluation Program for PCs
*//*-----------------------------------------------*/
#include <stdio.h>
#include <string.h>
#include "tjpgd.h"/* Bytes per pixel of image output */
#define N_BPP (3 - JD_FORMAT)


/* Session identifier for input/output functions (name, members and usage are as user defined) */typedef
struct {
    FILE *fp;               /* Input stream */
    uint8_t *fbuf;          /* Output frame buffer */
    unsigned int wfbuf;     /* Width of the frame buffer [pix] */
} IODEV;
```

```
/*-----------------------------*//* User defined input funciton  *//*-----------------------------*/


size_t in_func (    /* Returns number of bytes read (zero on error) */
    JDEC* jd,       /* Decompression object */
    uint8_t* buff,  /* Pointer to the read buffer (null to remove data) */
    size_t nbyte    /* Number of bytes to read/remove */
)
{
    IODEV *dev = (IODEV*)jd->device;   /* Session identifier (5th argument of jd_prepare function) */



    if (buff) { /* Raad data from imput stream */
        return fread(buff, 1, nbyte, dev->fp);
    } else {    /* Remove data from input stream */
        return fseek(dev->fp, nbyte, SEEK_CUR) ? 0 : nbyte;
    }
}


/*-----------------------------*//* User defined output funciton *//*-----------------------------*/


int out_func (      /* Returns 1 to continue, 0 to abort */
    JDEC* jd,       /* Decompression object */
    void* bitmap,   /* Bitmap data to be output */
    JRECT* rect     /* Rectangular region of output image */
)
{
    IODEV *dev = (IODEV*)jd->device;   /* Session identifier (5th argument of jd_prepare function) */
    uint8_t *src, *dst;
    uint16_t y, bws;
    unsigned int bwd;



    /* Progress indicator */
    if (rect->left == 0) {
        printf("\r%lu%%", (rect->top << jd->scale) * 100UL / jd->height);
    }
```

```
    /* Copy the output image rectangle to the frame buffer */
    src = (uint8_t*)bitmap;                             /* Output bitmap */
    dst = dev->fbuf + N_BPP * (rect->top * dev->wfbuf + rect->left);  /* Left-top of rectangle in the
frame buffer */
    bws = N_BPP * (rect->right - rect->left + 1);       /* Width of the rectangle [byte] */
    bwd = N_BPP * dev->wfbuf;                           /* Width of the frame buffer [byte] */
    for (y = rect->top; y <= rect->bottom; y++) {
        memcpy(dst, src, bws);   /* Copy a line */
        src += bws; dst += bwd;  /* Next line */
    }

    return 1;    /* Continue to decompress */
}


/*-----------------------------*//* Program Main
*//*-----------------------------*/


int main (int argc, char* argv[])
{
    JRESULT res;        /* Result code of TJpgDec API */
    JDEC jdec;          /* Decompression object */
    void *work;         /* Pointer to the work area */
    size_t sz_work = 3500; /* Size of work area */
    IODEV devid;        /* Session identifier */



    /* Initialize input stream */
    if (argc < 2) return -1;
    devid.fp = fopen(argv[1], "rb");
    if (!devid.fp) return -1;


    /* Prepare to decompress */
    work = (void*)malloc(sz_work);
    res = jd_prepare(&jdec, in_func, work, sz_work, &devid);
    if (res == JDR_OK) {
        /* It is ready to dcompress and image info is available here */
```

```
        printf("Image size is %u x %u.\n%u bytes of work ares is used.\n", jdec.width, jdec.height, sz_work
- jdec.sz_pool);


        /* Initialize output device */
        devid.fbuf = (uint8_t*)malloc(N_BPP * jdec.width * jdec.height); /* Create frame buffer for output
image */
        devid.wfbuf = jdec.width;


        res = jd_decomp(&jdec, out_func, 0);   /* Start to decompress with 1/1 scaling */
        if (res == JDR_OK) {
            /* Decompression succeeded. You have the decompressed image in the frame buffer here. */
            printf("\rDecompression succeeded.\n");


        } else {
            printf("jd_decomp() failed (rc=%d)\n", res);
        }


        free(devid.fbuf);   /* Discard frame buffer */


    } else {
        printf("jd_prepare() failed (rc=%d)\n", res);
    }


    free(work);             /* Discard work area */


    fclose(devid.fp);       /* Close the JPEG file */


    return res;
}
```

Limits

JPEG standard: Baseline only. Progressive and Lossless JPEG format are not supported.

JPEG 标准：仅限基线。不支持渐进式和无损 JPEG 格式。

Image size: Upto 65520 x 65520 pixels.

Colorspace: Y-Cb-Cr (color) and Grayscale (monochrome).

Sampling factor: 4:4:4, 4:2:2, 4:2:2(V) or 4:2:0 for color image.

Memory Usage

These are the memory usage of some platforms at default configuration. Each compilations are optimized in code size.

这些是一些平台在默认配置下的内存使用情况。每个编译都在代码大小上进行了优化。

	AVR	PIC24	CM0	IA-32
Compiler	GCC	C30	GCC	MSC
text+const	6.1k	5.1k	3.1k	3.7k

TJpgDec requires a work area upto 3100 bytes for most JPEG images. It exactly depends on what parameter has been used to create the JPEG image to be decompressed. The 3100 bytes is the maximum memory requirement in default configuration and it varies depends on JD_SZBUF and JD_FASTDECODE.

对于大多数 JPEG 图像，TJpgDec 需要高达 3100 字节的工作区域。这完全取决于用于创建要解压缩的 JPEG 图像的参数。3100 字节是默认配置中的最大内存需求，它因 JD_SZBUF 和 JD_FASTDECODE 而异。

Options

TJpgDec has some configuration options on output format, performance and memory usage. These options are in tjpgdcnf.h.

TJpgDec 在输出格式、性能和内存使用方面有一些配置选项。这些选项在 tjpgdcnf.h 中。

JD_SZBUF

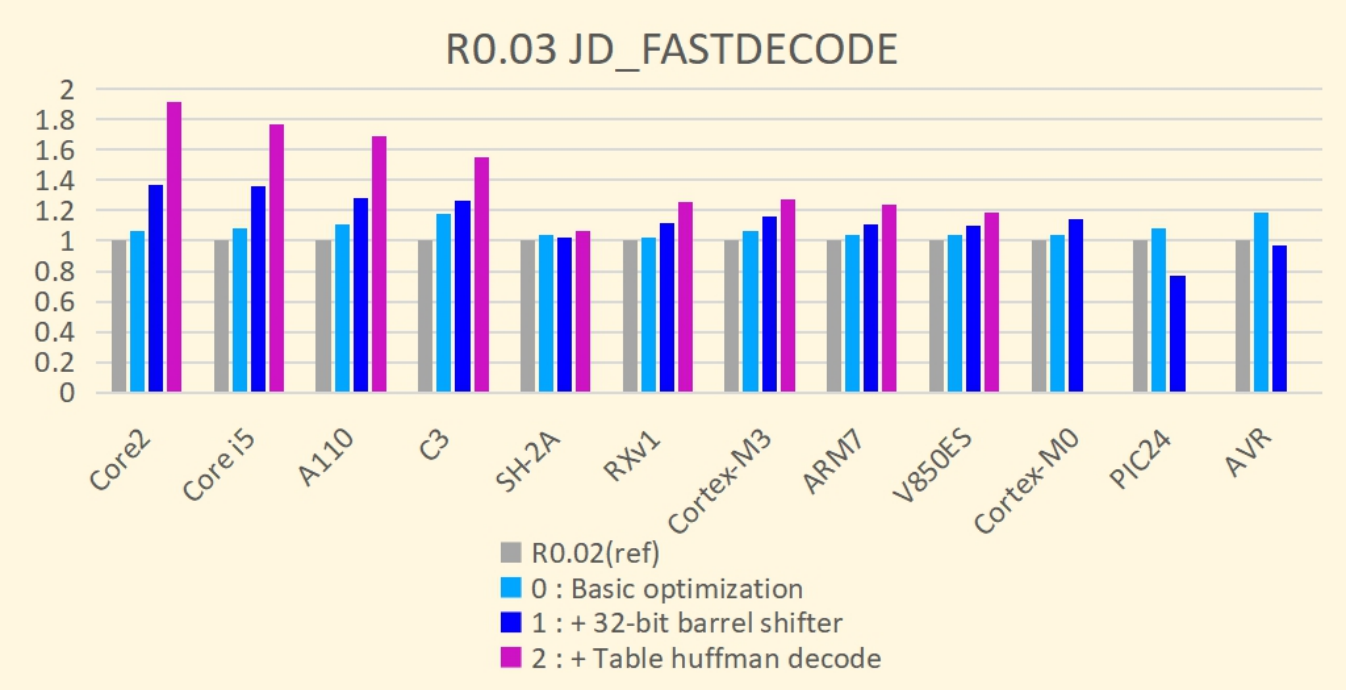
This option speficies how many bytes read from input stream at a time. TJpgDec alignes each read request to the buffer size, so that 512, 1024, 2048... byte is ideal to read data from the storage device.
此选项指定一次从输入流读取的字节数。TJpgDec 将每个读取请求与缓冲区大小对齐，因此 512、1024、2048…字节是从存储设备读取数据的理想选择。

JD_FORMAT

This option speficies the output pixel format. 0:RGB888, 1:RGB565 or 2:Grayscale.
此选项指定输出像素格式。0:RGB888、1:RGB565 或 2:灰度。

JD_USE_SCALE

This option switches output scaling feature. When enabled (1), the output image can be descaled on decompression. The descaling ratio is specified in `jd_decomp` function.
此选项用于切换输出缩放功能。启用时（1），可以在解压缩时对输出图像进行降噪。降噪系数在 jd_decomp 函数中指定。



JD_FASTDECODE

This option switches the decompression algorithm. How this option affects the performance is highly depends on the processor architectre as shown in right image. The Lv.0 is the basic optimization in minimal memory usage suitable for 8/16-bit processors. The Lv.1 depends on 32-bit barrel shifter and is suitable for 32-bit processors. It requires additional 320 bytes of work area. The Lv.2 enables the table conversion for huffman decoding. It requires additional `6 << HUFF_BITS`, 6144 by default, bytes of work area.
此选项用于切换解压缩算法。此选项如何影响性能在很大程度上取决于处理器架构，如右图所示。Lv.0 是适用于 8/16 位处理器的最小内存使用量的基本优化。Lv.1 依赖于 32 位桶形移位器，适用于 32 位处理器。它需要额外的 320 字节的工作区域。Lv.2 实现了用于哈夫曼解码的表转换。它需要额外的 6 个<<HUFF_BITS，默认情况下为 6144 个字节的工作区域。

JD_TBLCLIP

This option switches to use table conversion for saturation arithmetics. It requires 1024 bytes of code size.
此选项切换为使用表转换进行饱和运算。它需要 1024 字节的代码大小。

About TJpgDec License

This is a copy of the TJpgDec license document that included in the source codes.

```
/*----------------------------------------------------------------------------/
/ TJpgDec - Tiny JPEG Decompressor R0.xx                       (C)ChaN, 20xx
/----------------------------------------------------------------------------/
/ The TJpgDec is a generic JPEG decompressor module for tiny embedded systems.
/ This is a free software that opened for education, research and commercial
/  developments under license policy of following terms.
/  Copyright (C) 20xx, ChaN, all right reserved.
/ * The TJpgDec module is a free software and there is NO WARRANTY.
/ * No restriction on use. You can use, modify and redistribute it for
/   personal, non-profit or commercial products UNDER YOUR RESPONSIBILITY.
/ * Redistributions of source code must retain the above copyright notice.
/----------------------------------------------------------------------------*/
```

Therefore TJpgDec license is one of the BSD-style license but there is a significant difference. Because TJpgDec is for embedded projects, so that the conditions for redistributions in binary form, such as embedded code, hex file and binary library, are not specified in order to maximize its usability. The documentation of the distributions may or may not include about TJpgDec and its license document. Of course TJpgDec is compatible with the projects under GNU GPL. When redistribute TJpgDec with any modification, the license can also be changed to GNU GPL or any BSD-style license.

因此，TJpgDec 许可证是 BSD 风格的许可证之一，但有很大的区别。因为 TJpgDec 是针对嵌入式项目的，所以没有指定以二进制形式重新分发的条件，如嵌入式代码、十六进制文件和二进制库,以最大限度地提高其可用性。发行版的文件可能包括也可能不包括关于 TJpgDec 及其许可证文件。当然,TJpgDec 与 GNU GPL 下的项目是兼容的。在进行任何修改后重新分发 TJpgDec 时，许可证也可以更改为 GNU GPL 或任何 BSD 风格的许可证。

Return

（itkw_com @ 2023.04.02 译）