


TXW80X 视频方案开发指南



注意

由于产品版本升级或者其他原因，本文档会不定期更新。除非另行约定，本文档仅作为使用指导，不做任何担保。

保密等级	A	TXW80X SDK 快速入门手册	文件编号	TX-0000
发行日期	2022-05-15		文件版本	V1.0
修订记录				
日期	版本	描 述	修订人	
2022-05-15	V1.0	初始版本	TX	

保密等级	A	TXW80X SDK 快速入门手册	文件编号	TX-0000
发行日期	2022-05-15		文件版本	V1.0
<div>目录</div> <div><div>1. 概述..... 1</div><div>2. 硬件开发板..... 1</div><div><div>2.1. 音视频开发板..... 2</div><div><div>2.1.1. 音视频开发板接口介绍..... 2</div></div></div><div>3. 视频开发配置流程..... 3</div><div><div>3.1. 视频方案相关配置..... 3</div><div>..... 4</div><div>..... 4</div><div>..... 4</div><div>..... 4</div><div><div>4.1.1. Dvp 摄像头..... 4</div><div>4.1.2. USB 摄像头..... 9</div></div></div></div>				
<div></div>		<div>珠海泰芯半导体有限公司</div> <div>Zhuhai Taixin Semiconductor Co.,Ltd</div>	<div>珠海市高新区港湾一号科技园港 11 栋 3 楼</div>	
<div>版权所有 侵权必究</div> <div>Copyright © 2022 by Tai Xin All rights reserved</div>				

1. 概述

本文主要描述视频开发流程。

本文档主要适用于以下工程师：

- 技术支持工程师
- 方案软件开发工程师

本文档适用的产品范围：

型号	封装	包装
TXW806	QFN56/48/40	

2. 硬件开发板

为了快速入门和方案评估，我们提供各种应用场景的开发板。

2.1. 音视频开发板

2.1.1. 音视频开发板接口介绍

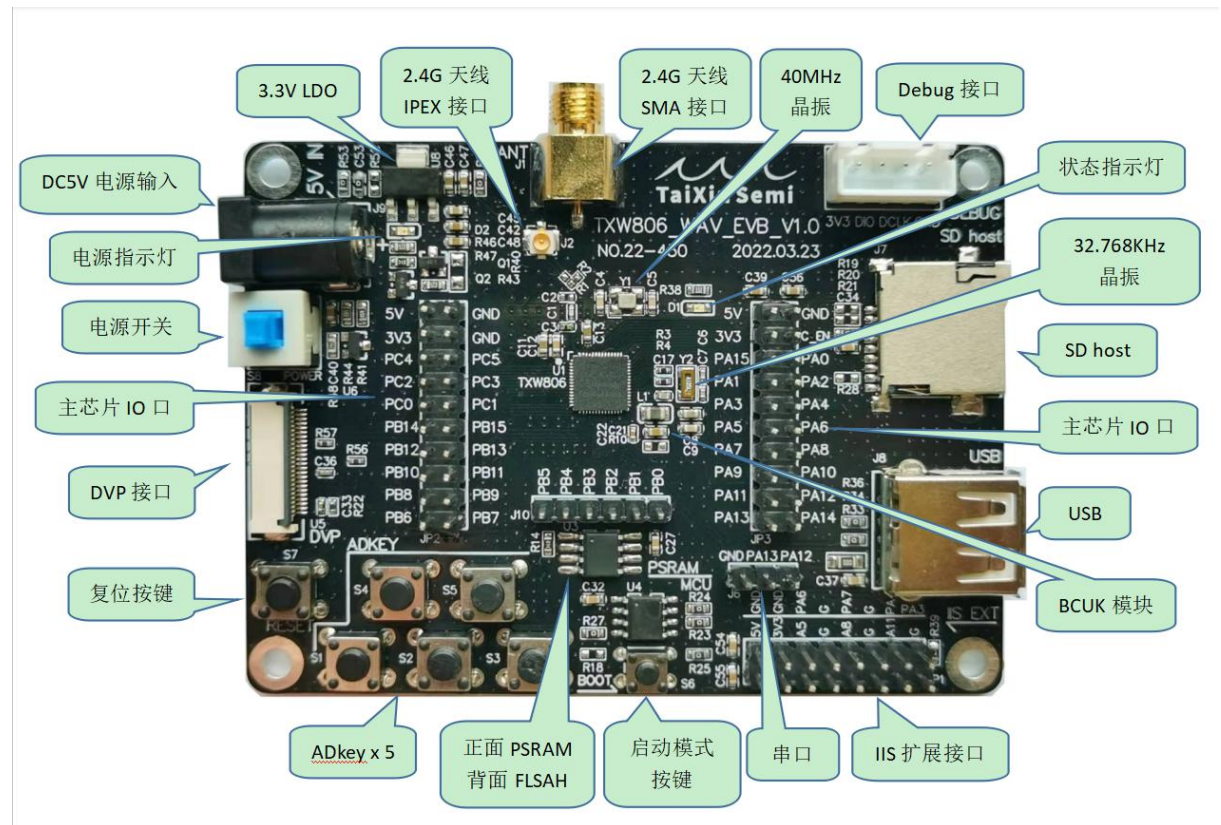


图 3.1.1.1 - 音视频开发板主视图

特殊说明

模式启动按键：此按键可以一键拯救系统，在芯片上电即跑死，cklink 烧写和其他升级都失效的情况下使用。

IIS 扩展接口：此接口注意是为了扩展音频子板。

3. 视频开发配置流程

3.1. 视频方案相关配置

方案配置主要看 project_config.h 配置, 以下配置都是基于 sdk 的图传协议。

```
#define USB_HOST_EN          1
#define MACBUS_USB

#define USB_EN              0
#define DVP_EN              0
#define JPG_EN              (1 && DVP_EN)
#define SDH_EN              1
#define FS_EN               1
#define VCAM_EN             (0 || DVP_EN)

#define FRAME_MANAGEMENT    1
#define OPENDML_EN          1
#define UART_INPUT_EN       1
#define UART_FLY_CTRL_EN    0
#define PWM_EN              0

#define AUDIO_EN             (0 && FRAME_MANAGEMENT)
#define IIS_AUDIO_EN        (0 && FRAME_MANAGEMENT)
#define RTP_SOUND           (1 && (AUDIO_EN || IIS_AUDIO_EN))

#define MJPEG_VIDEO          (1 && FRAME_MANAGEMENT && OPENDML_EN && FS_EN && SDH_EN && JPG_EN) //基于框架的mjpeg录像
#define UVC_VIDEO            (1 && FRAME_MANAGEMENT && OPENDML_EN && FS_EN && SDH_EN && USB_EN) //基于框架的uvc录像
```

- 1、USB_EN 启动 USB 摄像头(usb 与 dvp 暂时不能共存)
- 2、DVP_EN 启动 dvp 摄像头(usb 与 dvp 暂时不能共存)
- 3、SDH_EN、FS_EN 是 sd 卡与文件系统初始化
- 4、FRAME_MANAGEMENT (仅仅支持 sdk 的图传协议) 框架启动, 在外挂 psram 的时候, 需要使能
- 5、OPENDML_EN 在需要录像的时候, 需要启动(仅仅支持 sdk 的录像格式, 其他格式需要自己实现)

4.1.1. Dvp 摄像头

4.1.1.1. Dvp 摄像头相关参数配置

1、采用 VGA 摄像头的时候, IMAGE_W 与 IMAGE_H 配置为 VGA。对应宏定义#define CMOS_AUTO_LOAD 1 打开(位于 project_config.h), 可以自动选择所有支持的摄像头, 如果不使用自动选择, 只希望支持某一款摄像头, 则将对摄像头宏打开, 例如:#define DEV_SENSOR_GC0308 1(位于 project_config.h), 支持的摄像头以及宏定义在 csi.h, 如图:

```

#ifndef DEV_SENSOR_OV7725
#define DEV_SENSOR_OV7725 (0 || CMOS_AUTO_LOAD)
#endif

#ifndef DEV_SENSOR_OV7670
#define DEV_SENSOR_OV7670 (0 || CMOS_AUTO_LOAD)
#endif

#ifndef DEV_SENSOR_GC0308
#define DEV_SENSOR_GC0308 (0 || CMOS_AUTO_LOAD)
#endif

#ifndef DEV_SENSOR_GC0309
#define DEV_SENSOR_GC0309 (0 || CMOS_AUTO_LOAD)
#endif

#ifndef DEV_SENSOR_GC0328
#define DEV_SENSOR_GC0328 (0 || CMOS_AUTO_LOAD)
#endif

#ifndef DEV_SENSOR_BF3A03
#define DEV_SENSOR_BF3A03 (0 || CMOS_AUTO_LOAD)
#endif

#ifndef DEV_SENSOR_BF3703
#define DEV_SENSOR_BF3703 (0 || CMOS_AUTO_LOAD)
#endif

#ifndef DEV_SENSOR_OV2640
#define DEV_SENSOR_OV2640 0
#endif

#ifndef DEV_SENSOR_BF2013
#define DEV_SENSOR_BF2013 (0 || CMOS_AUTO_LOAD)
#endif

#ifndef DEV_SENSOR_XC7016_H63
#define DEV_SENSOR_XC7016_H63 0
#endif

#ifndef DEV_SENSOR_XC7011_H63
#define DEV_SENSOR_XC7011_H63 0
#endif

#ifndef DEV_SENSOR_XC7011_GC1054
#define DEV_SENSOR_XC7011_GC1054 0
#endif

#ifndef DEV_SENSOR_XCG532
#define DEV_SENSOR_XCG532 0
#endif

#ifndef DEV_SENSOR_GC2145
#define DEV_SENSOR_GC2145 0
#endif

```

2、720p 摄像头处理形式分两种。

- 1) 外接 isp 芯片
- 2) 自带 isp 功能摄像头

如果所需支持摄像头为无 isp 功能摄像头，则需外接 isp 芯片，目前已适合两款 isp 芯片，XC7016 和 XC7011。当然，如果芯片直接支持 isp 功能，能正常输出已处理好的 YUV 数据，则可直接接到本主控上。在配置上 IMAGE_W 与 IMAGE_H 配置为 720P。

4.1.1.2. Dvp 摄像头添加

请参考已有摄像头的配置文件 sensor_XXX.c，先添加摄像头的配置文件，并记录对的数据结构体 _Sensor_Adpt_与 _Sensor_Ident

_Sensor_Ident 包含当前摄像头的读写命令，命令与数据操作长度，ID 地址与 ID 值，在 sensorCheckId 中使用此配置信息去判断，当前插入的摄像头是否吻合此配置。

_Sensor_Adpt_包含的重要信息有 pixelw, pixelh, hsync, vsyn, init, mclk。pixelw 与 pixelh 为当前图像配置的输入分辨率。hsyn 与 vsyn 为当前摄像头的配置中，hsync 与 vsync 的有效电平是高电平还是低电平。mclk 为当前配置摄像头被插入时，系统给此摄像头提供的 mclk 频率。init 为当前摄像头的初始化配置信息，默认配置输出为 yuyv。

配置摄像头的配置文件 sensor_XXX.c 配置好后，请在 csi.h 中加入对应的声明，并在 devSensorInitTable 与 devSensorOPTable 添加对应摄像头的信息

4.1.1.3. Dvp 摄像头 mjpeg 编码参数配置

- 1、mjpeg 的总 size 配置为 JPG_NODE*(JPG_BUF_LEN+JPG_TAIL_RESERVER)，宏定义参考 project_config.h 文件，JPG_BUF_LEN 不能过小，这个是决定 mjpeg 的中断频率，JPG_BUF_LEN 越大(根据剩余内存空间配置)，一张完整图的中断越少，效率以及出错情况越少，JPG_NODE 是节点数，mjpeg 中断会从 JPG_NODE 的节点池获取节点，如果节点不够，那么当前编码的图片会由于内存不足丢掉，JPG_TAIL_RESERVER 是用于自定义参数，如果不需要，设置为 0，JPG_HEAD_RESERVER 是头保留，也是自定义，不需要可以设置为 0，在官方 sdk 中是默认 24，会在 sdk 本身图传协议上使用

```
#if FRAME_MANAGEMENT
uint8 psram_jpg_buf[JPG_NODE*(JPG_BUF_LEN+JPG_TAIL_RESERVER)]__attribute__((section(".psram.src")));
#else
uint8 psram_jpg_buf[JPG_NODE*(JPG_BUF_LEN+JPG_TAIL_RESERVER)];__attribute__((section(".psram.src")));
#endif
```

- 2、图片的压缩情况，请参考函数 jpg_quality_tidy，此函数在中断中调用，用于通过当前图片的大小来控制图像质量的调整方向，可理解为此函数为 jpg 图片大小的控制范围。但此函数只是期望值，并不是一定能控制到距离实际太大的范围，如将 720P 的图片压缩大小控制在 20K 以内，这种是无法压缩出来这种大小的，压缩的大小与当前使用的量化表与微调值相关，量化表请参考 quality_tab，目前提供了 6 张量化表，客户有需要，可自行修改。微调是指在此量化表外，还能进行依据当前使用量

化表进行数据 16 个等级的微调，等级 8 为量化表不变，1~7 质量向上优化，9~F 质量向下调整。

4.1.1.4. Dvp 图片的获取流程

dvp 图片获取接口如图: (jpgdef.h)

```
0:  #ifndef JPG_EN == 1
7:      extern int get_jpg_node_len();
8:      #define GET_NODE_LEN(f)      get_jpg_node_len(f)
9:      #define GET_JPG_LEN(f)      get_jpeg_len(f)
0:      #define GET_JPG_buf(f)      get_jpeg_first_buf(f)
1:      #define GET_JPG_FRAME()      get_frame()
2:      #define FREE_JPG_NODE(f)      del_jpeg_first_node(f)
3:      #define DEL_JPG_FRAME(f)      del_jpeg_frame(f)
4:
```

- 1、首先获取图片帧 $f = \text{GET_JPG_FRAME}()$ ，如果获取到代表已经获取到整一张图片，图片的组成是以节点形式连接，只要通过循环读取就能获取到整一张图
- 2、 $\text{GET_JPG_LEN}(f)$ 获取一张图片的 size
- 3、 $\text{GET_JPG_buf}(f)$ 获取图片帧的首个节点，
- 4、 $\text{FREE_JPG_NODE}(f)$ 释放图片帧首个节点
- 5、 $\text{GET_NODE_LEN}(f)$ 获取一个节点的长度(图片帧内所有节点长度都是一样的，所以获取到图片帧最后一个节点的时候，实际内容 $\leq \text{GET_NODE_LEN}(f)$ ，所以处理最后一帧的时候，应该根据图片 size 计算最后节点的实际内容的大小)
- 6、 $\text{DEL_JPG_FRAME}(f)$ 删除图片帧，同时归还所有节点，在图片帧使用完毕后，需要调用该接口去释放，否则导致图片帧丢失，后续获取不到图片帧

伪代码:

```

int read_size = 0;
int photo_size_tmp, photo_size, node_len;
uint8 *buf;
//获取图片帧
void *f = GET_JPG_FRAME()
if(f)
{
    //获取节点长度
    node_len = GET_NODE_LEN(f);
    //获取图片的size
    photo_size_tmp = photo_size = GET_JPG_LEN(f);

    //循环读取图片帧的节点
    while(photo_size_tmp)
    {
        buf = GET_JPG_buf(f);
        //异常,不能出现这种情况
        if(!buf)
        {
            printf("get buf err\n");
            while(1);
        }
        if(photo_size_tmp >= node_len)
        {
            read_size = node_len;
        }
        else
        {
            read_size = photo_size_tmp;
        }
        //保存图片的buf,自定义实现,buf read_size

        FREE_JPG_NODE(f);
        photo_size_tmp -= read_size;
    }

    //图片帧使用完毕
    DEL_JPG_FRAME(f)
}

```

4.1.1.5. Dvp 数据流说明

Dvp➡JPG 数据流的生成步骤与基本思想如下:

- 1: JPG 数据 buf 以节点的形式分配好
- 2: DVP 采集足够的行数据
- 3: JPG 提取压缩
- 4: 当 JPG 压缩出来数据量到达所配置的数据 buf 长度或者图片获取结束, 生成 out_buf_full 或者 done 中断 (详情请看 jpg 模块 api 函数说明)
- 5: 中断里记录好节点的情况并分配下次 jpg 数据所存放的节点位置
- 6: 若 done 中断完成, 应用可根据 4.1.1.3 说明提取对应的链表数据, 重组 jpg 图片

其中, jpg 数据 buf 节点分配, 请查看 jpg_room_init 函数, 这里特说明, frame 根节点 2 个, 即 jpg_frame[2], 分别被 jpg_p (中断) 与 usr_p (应用) 指针指向, 用于表示对应 frame 的使用是在中断还是在应用。除了 jpg_frame 的两个根节点外, 还有 free_tab 一个根节点, 用于管理由 free_table_init 中生成的众多空间子节点。每个空间子节点都会绑定自己专属的空间, 图片数据存放则存放到此众多空间中, 整帧图片数据并不连续, 但能根据 jpg_frame 的应用节点查找到图片数据排布并进行数据重组。

4.1.1.6. Dvp 重要打印说明

- 1、打印信息 “sip reset DVP”, DVP 采集出错, 请检查摄像头是否插好, 或者配置表

中信息是否与 IMAGE_W、IMAGE_H 一致，如果 IMAGE_W 与 IMAGE_H 配置为 720P，配置表配置为 VGA，则会出现此打印，因为采集数据上出问题。还有一种情况，配置表中的 hsync 和 vsync 的有效电平配置出错，即配置表中 hsync 和 vsync 的有效电平与 _Semsor_Adpt_ 里面的配置不匹配，这样也会造成采样出错的情况

- 2、打印信息“fh”和“fv”，DVP 处理速度跟不上采集速度，“fh”表示在数据没处理完的情况下，上次采集的 buf 已改写了一半。“fv”表示在数据没处理完的情况下，上次采集的 buf 已被改写。出现这种问题有可能是 DVP 的 CLK 分配过高，CPU 处理不过来，或者 DVP 中断中处理的事情过多，影响到处理效率
- 3、打印“jpg done len err”，这种情况有两种可能性，一是 CPU 处理不过来，即 JPG 的压缩率不高，图像太大，CPU 在多次中断才响应一次。第二种情况是，软件 buf 空间不足，节点不够（参考数据流说明）。针对第一种情况，处理方式为提高代码的运行速率（xip 的运行速度, 请查看《TXW80X TXProgrammer 工具使用文档》）、提升节点的空间大小或者降低图像的质量。第二种情况处理方式为提升代码运行速率，降低图像质量或者在空间允许的情况下，增加节点的数量。
- 4、打印“？”，这种情况为 JPG 出现异常了，代码会复位 DVP 和 JPG，要是 JPG 输出一一直打印这个问题，那请检查一下，FRAME_MANAGEMENT 是否在置 1 的情况下，烧写的代码不正确。在 FRAME_MANAGEMENT 置 1 的情况下，请烧写 project_merge.bin，否则 psram 是不会初始化的，jpeg 数据传输到 psram 是无效的，从而导致 jpg 异常。

4.1.2. USB 摄像头

4.1.2.1. USB 摄像头参数配置

1、宏定义:UVC_BLANK_LEN 是节点的大小，UVC_BLANK_NUM 节点数量, 在无 psram 的情况下，UVC_BLANK_NUM 需要足够大，UVC_BLANK_NUM*UVC_BLANK_LEN 起码需求 1.5 张图片的 size 大小，否则图片会被丢弃，在由 psram 的情况下，UVC_BLANK_NUM 可以小一点，最后所有图片数据都会放在 psram 中，UVC_BLANK_LEN 不宜过小，默认 12*1024 byte，UVC_FRAME_NUM 代表图片的帧，默认值为 2

```
#if USB_EN
extern uint8_t uvc_dat[1024];
#define UVC_BLANK_LEN      12*1024
#if UVC_PSRAM == 1
#define UVC_BLANK_NUM      3
#else
#define UVC_BLANK_NUM      7
#endif
#define UVC_FRAME_NUM      2
UVC_BLANK uvc_blank[UVC_BLANK_NUM];
uint8 blank_space[UVC_BLANK_NUM][UVC_BLANK_LEN+UVC_HEAD_RESERVER]__attribute__((aligned(4)));
```

在分辨率选择上，默认分辨率请修改 uvc_default_dpi 里的返回值，当前默认有两种 UVC_VGA 和 UVC_720P。在传输中途若想修改摄像头的分辨率，请使用 usb 的 ioctl 功能，例如：

```

24 void uvc_ioctl_index(uint8 uvc_idx){
25     uint8 msgbuf[1];
26     uint8* pt;
27
28     struct usb_device *dev;
29     dev = (struct usb_device *)dev_get(HG_USBDEV_DEVID);
30     pt = msgbuf;
31     pt[0] = uvc_idx;
32     usb_host_uvc_ioctl(dev,USB_HOST_IO_CMD_SET_IDX,msgbuf,0);
33 }

```

如果摄像头有接入 hub, 也同样可用 IOCTL 配置, USB_HOST_IO_CMD_INSERT_HUB 功能。

4. 1. 2. 2. USB 摄像头获取过程

(实际宏定义与 dvp 摄像头的一样, 调用函数名称改成 uvc 的实现)

uvc 图片获取接口如图: (jpgdef.h)

```

extern int get_node_uvc_len();
#define UVC_HEAD_RESERVER 24 //每段JPEG BUF前面预留用于填充其他数据的数据量, 无需预留填0
#define UVC_PSRAM 0
#define GET_NODE_LEN(f) get_node_uvc_len(f)
#define GET_JPG_LEN(f) get_uvc_frame_len(f)
#define GET_JPG_buf(f) get_uvc_buf(f)
#define GET_JPG_FRAME() get_uvc_message_gloal()
#define FREE_JPG_NODE(f) free_uvc_blank_nopsram(f)
#define DEL_JPG_FRAME(f) uvc_sema_up()

```

- 1、首先获取图片帧 $f = \text{GET_JPG_FRAME}()$, 如果获取到代表已经获取到整一张图片, 图片的组成是以节点形式连接, 只要通过循环读取就能获取到整一张图
- 2、 $\text{GET_JPG_LEN}(f)$ 获取一张图片的 size
- 3、 $\text{GET_JPG_buf}(f)$ 获取图片帧的首个节点,
- 4、 $\text{FREE_JPG_NODE}(f)$ 释放图片帧首个节点
- 5、 $\text{GET_NODE_LEN}(f)$ 获取一个节点的长度(图片帧内所有节点长度都是一样的, 所以获取到图片帧最后一个节点的时候, 实际内容 $\leq \text{GET_NODE_LEN}(f)$, 所以处理最后一帧的时候, 应该根据图片 size 计算最后节点的实际内容的大小)
- 6、 $\text{DEL_JPG_FRAME}(f)$ 删除图片帧, 同时归还所有节点, 在图片帧使用完毕后, 需要调用该接口去释放, 否则导致图片帧丢失, 后续获取不到图片帧

伪代码:

```

int read_size = 0;
int photo_size_tmp, photo_size, node_len;
uint8 *buf;
//获取图片帧
void *f = GET_JPG_FRAME()
if(f)
{
    //获取节点长度
    node_len = GET_NODE_LEN(f);
    //获取图片的size
    photo_size_tmp = photo_size = GET_JPG_LEN(f);

    //循环读取图片帧的节点
    while(photo_size_tmp)
    {
        buf = GET_JPG_buf(f);
        //异常,不能出现这种情况
        if(!buf)
        {
            printf("get buf err\n");
            while(1);
        }
        if(photo_size_tmp >= node_len)
        {
            read_size = node_len;
        }
        else
        {
            read_size = photo_size_tmp;
        }
        //保存图片的buf,自定义实现,buf read_size

        FREE_JPG_NODE(f);
        photo_size_tmp -= read_size;
    }

    //图片帧使用完毕
    DEL_JPG_FRAME(f)
}

```

4.1.2.3. USB 重要打印说明

- 1、打印“babble error”，此打印说明信号线接触不良，系统会重新 reset usb,要是持续打印 babble error，则需要确定 usb 线是否正常。
- 2、打印“disconnect.....”，此打印为设备掉线打印，这种掉线有接触不良掉线或者主动排插掉线，系统会重新 reset usb。
- 3、打印“XXX Err”和“Enum ERR”，此类打印为设备在枚举过程中枚举失败，系统会重新 reset usb。
- 4、打印“uvc can not set resolution :720p”，此类打印为在不支持 720P 的摄像头配置了 720P 输出，会默认改为 VGA 输出。
- 5、打印“_D3_”，此类打印为摄像头的 UVC 图像出错，如果这部分打印过多，请确定摄像头是否正常。
- 6、打印“_D1_”和“_D2_”时，表明当前的数据缓存空间不足，如果是跑无 psram 的方案下，持续打印这个的情况下，请确认图像是否大小超过默认 size，7*12K，如果是，请修改摄像头的输出大小。如果此打印非持续出现，但也高频率出现，表明 cpu 处理不过来，请提高代码的运行效率（XIP 的运行速率，请查看《TXW80X TXProgrammer 工具使用文档》）