

计算机原理与嵌入式系统设计实验报告（第六次）



目录

[(一) 实验目的与实验设备 2](#_Toc27930081)

[(二) 实验要求 2](#_Toc27930082)

[(三) 实验原理 2](#_Toc27930083)

[(四) 实验步骤 3](#_Toc27930084)

[(五) 程序分析 4](#_Toc27930085)

[1. 在LCD上显示一张图片 4](#_Toc27930086)

[2. 在LCD上显示多张图片，加入键盘按键功能，制成电子相册，用键盘按键翻页循环显示； 5](#_Toc27930087)

[3. 在LCD上显示一张动态图片（GIF图）。 5](#_Toc27930088)

[(六) 程序代码 7](#_Toc27930089)

[1. 在LCD 上显示一张图片 7](#_Toc27930090)

[2. 在LCD上显示多张图片，加入键盘按键功能，制成电子相册，用键盘按键翻页循环显示； 7](#_Toc27930091)

[3. 在LCD上显示一张动态图片（GIF图）。 9](#_Toc27930092)

[(七) 实验效果（详见视频） 12](#_Toc27930093)

[1. 在LCD上显示一张图片 12](#_Toc27930094)

[2. 在LCD上显示多张图片，加入键盘按键功能，制成电子相册，用键盘按键翻页循环显示； 13](#_Toc27930095)

[3. 在LCD上显示一张动态图片（GIF图）。 15](#_Toc27930096)

[(八) 实验总结 17](#_Toc27930097)

# 实验目的与实验设备

**实验目的：**

1. 掌握LCD 液晶模块的基本原理和功能。
2. 掌握LCD 液晶模块和微控制器的硬件接口及程序设计方法。
3. 熟悉ARM的C语言和汇编语言的编程与调试方法。

**实验设备：**

1. 硬件：W90P710实验系统、PC机、J-LINK仿真器
2. 软件：PC机操作系统、Keil MDK-ARM 4.22

# 实验要求

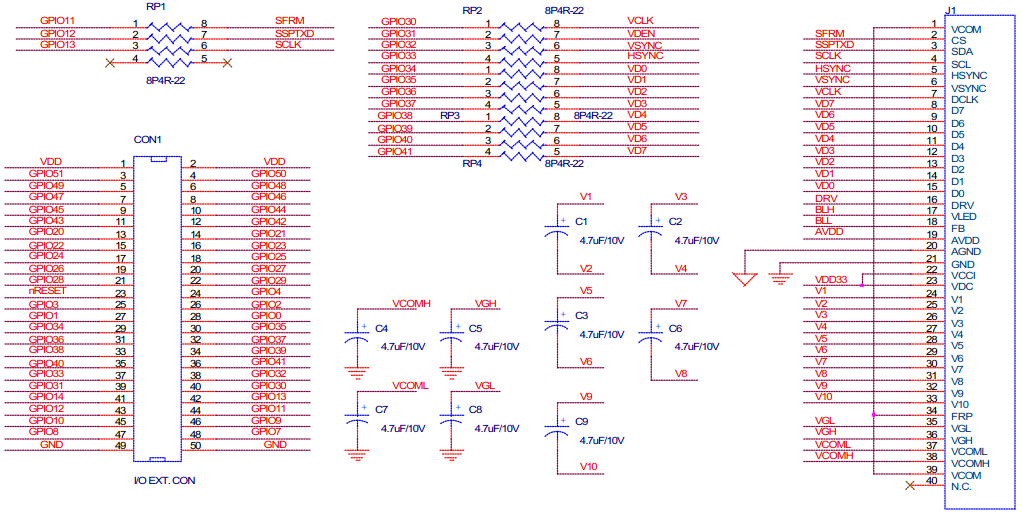
编写LCD 程序：

1. 在LCD上显示一张图片；
2. 在LCD上显示多张图片，加入键盘按键功能，制成电子相册，用键盘按键翻页循环显示；
3. 在LCD上显示一张动态图片（GIF图）。

# 实验原理

W90P710 评估板的LCD 控制器的主要目的是将VIDEO/OSD 的原始图像数据显示到外部液晶屏上。LCD控制器可以支持AUO的TFT屏，LG 的 TFT 屏，CASIO 的TFT 屏，实验中使用的是AUO 3.0" TFT A030DL01（960x240，8位数据总线）数字式TFT液晶屏。此外，LCD控制器还能够连接到STN LCD 和电视编码器上，并且支持VIDEO和OSD 两种显示模式。这两种显示模式的设定由不同的寄存器控制，通过设定相应寄存器的值可以实现不同的模式和功能。

AUO 3.0" TFT A030DL01 液晶屏由单个像素点排列组成，向这些像素点配置不同的值，可以显示不同的颜色。但是，由于AUO TFT液晶屏内部没有缓存，需要在SDRAM中开辟一片缓冲区存储一维数组形式的原始图像数据。然后，将存储原始图像数据的SDRAM 缓冲区的首地址赋值给LCD控制器中的FIFO1SDADR寄存器。接着，原始图像数据在时钟脉冲的作用下依次输入到LCD控制器的FIFO1中，直至FIFO1被填满才将FIFO1中的数据输出到LCD颜色发生器。最后，LCD颜色发生器将图像显示到评估板上的液晶屏中。如果在液晶屏上定位显示图片或其他信息，需要计算出所要显示的内容在液晶屏上的位置。然后，在新建的SDRAM缓冲区中存入需要显示的内容即可。LCD硬件连接图如图33所示。



LCD模块由 40 个引脚与外界电路相连，由于本身没有寄存器，需要由 W90P710中的LCD 控制器提供相应的寄存器，控制选择对应的液晶类型及显示形式、FIFO 使用、颜色显示模式和时序设置。

LCDCON寄存器比较重要，需要根据实际所使用的硬件，对控制寄存器进行配置。本次实验使用16 bpp 模式、真彩型TFT LCD、没有使用查找表功能，分别使能相应的位实现LCDCON寄存器初始化操作。

# 实验步骤

实验步骤与“LED灯闪烁实验”类似，但有两个不同点：在步骤3中不需要新建源文件，直接将文件夹“lcd显示所需文件”中提供的实验文件拷贝到工程新建文件夹中；并将所有文件添加到工程目录中，完成任务a。

然后修改lcd.c主程序，加入键盘按键功能，增加图片，制成电子相册，用键盘按键翻页循环显示完成任务b。步骤如下：

* 准备好几幅图片，320\*240像素，用Image2Lcd工具转换为16位真彩色的\*.c格式文件，并添加到工程中；
* 根据上次实验在主程序中添加按键功能；
* 根据不同键值切换显示图片。

最后完成任务c，连续显示一组动态图片。

# 程序分析

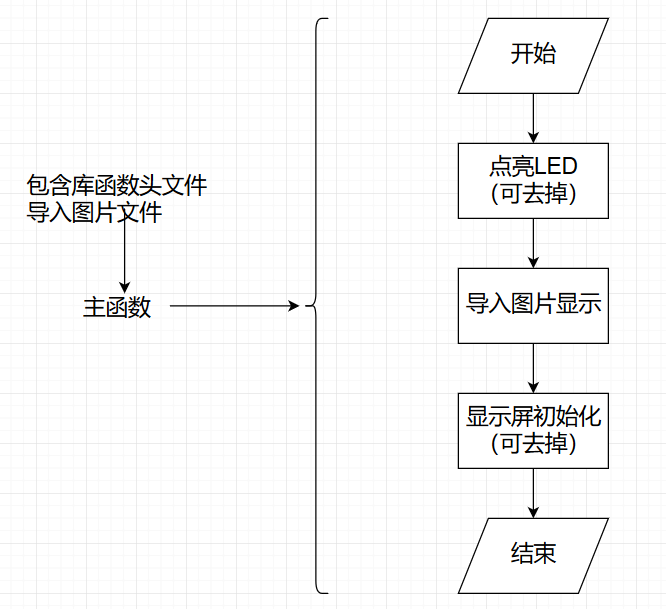
## 在LCD上显示一张图片

根据例程的提示，这是一个非常简单的顺序结构，其关键代码仅为

GLCD\_bitmap (0, 0, 320, 240, Background1);

在引入的Picture.c文件中，将Background1数组改为自己需要替换的图片转换而来的数组即可。

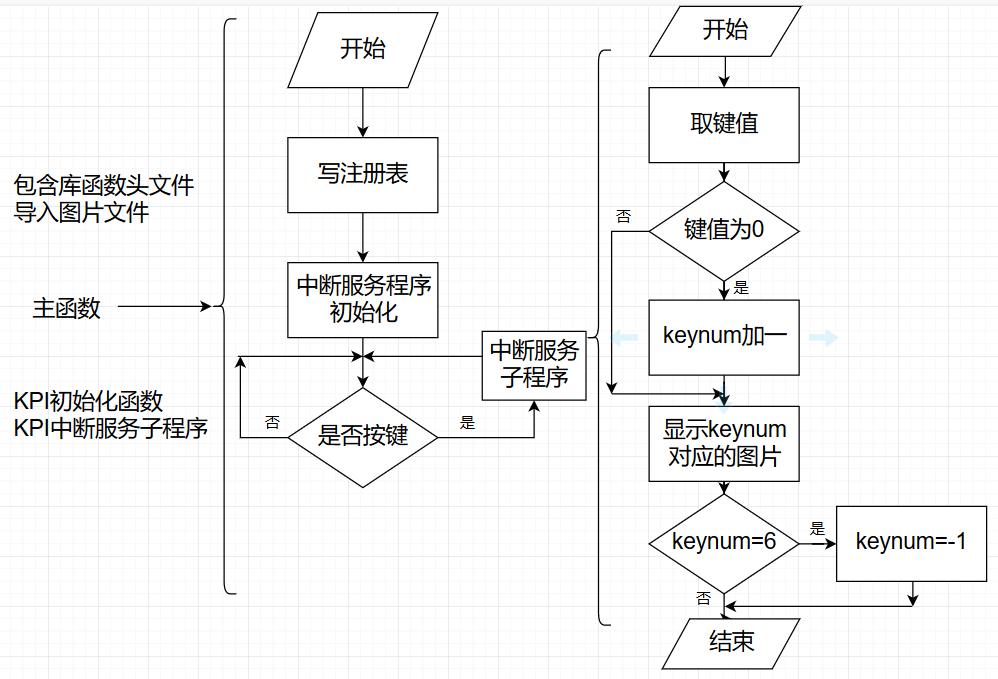
程序流程图如下：



## 在LCD上显示多张图片，加入键盘按键功能，制成电子相册，用键盘按键翻页循环显示；

通过研究键盘扫描阵列实验的例程发现，可以使用KPI中断来实现功能。在上面的实验中，键盘键值的判断采用的是中断模式，需要程序打开键盘控制器中断，并设定优先级。在中断服务程序中，读取按键键值并点亮对应的LED灯。而在本实验中，将点亮LED灯的代码改为要求的功能即可。通过引入一个标志变量keynum来实现电子相册内照片的选择。按下“0”号按键时，进入中断服务子程序，keynum加一，并且通过switch语句选择显示对应的照片，当keynum即将超过照片总数时，将其置-1，下一次按键“0”时值即为0，显示第0张照片。

程序流程图如下：

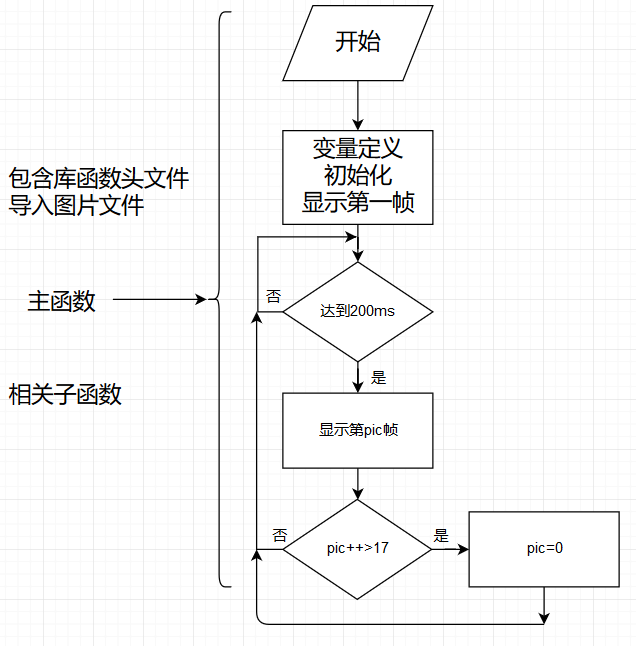


## 在LCD上显示一张动态图片（GIF图）。

由于此任务为播放GIF图片，不需要键盘进行选择，所以不需要在中断中实现，在主函数当中用一个定时和循环即可实现。利用软件将GIF图片分帧，再把每一帧图片比例都调整为320x240，用img2lcd软件转为16位真彩显示数组，将其导入工程即可。

由于人眼的视觉暂留效果，将每一帧的间隔时间定为200ms，并且同任务2一样，引入标志变量，每显示一帧图片，标志加一，在即将超过帧数时置为0。这样即可实现GIF图片的循环播放。实验中尝试了两幅GIF图片，都取得了不错的效果。

程序流程图如下：



# 程序代码

## 在LCD 上显示一张图片

#include <W90P710.H> /\* Winbond W90P710 definitions \*/

#include "GLCD.h"

#define LED\_ALL 0xff

extern unsigned char Background1[];

int main (void) { /\* Main Function \*/

Reg8(0x78000000) = LED\_ALL; /\* Turn off all LEDs \*/

GLCD\_bitmap (0, 0, 320, 240, Background1);

GLCD\_init();

}

## 在LCD上显示多张图片，加入键盘按键功能，制成电子相册，用键盘按键翻页循环显示；

extern unsigned char Background0[];

extern unsigned char Background1[];

extern unsigned char Background2[];

extern unsigned char Background3[];

extern unsigned char Background4[];

extern unsigned char Background5[];

#define u32 unsigned int

#define REG\_GPIO\_CFG2 0xFFF83020

#define REG\_AIC\_SCR29 0xFFF82074

#define REG\_AIC\_MECR 0xFFF82120

#define REG\_KPICONF 0xFFF88000

#define REG\_KPISTATUS 0xFFF8800C

#define EBILED\_ADDRESS 0x78000000

#define REG\_EXT0CON 0xFFF01018

#define EBI\_EXT0\_VALUE 0xF0078003

#define WriteReg(Address,Value) \*(unsigned int volatile \*)(Address) = Value

#define ReadReg(Address) \*(unsigned int volatile \*)(Address)

int keynum=-1;

void KPIInit(void);

int main (void){

WriteReg(REG\_EXT0CON, EBI\_EXT0\_VALUE);

KPIInit();

while(1);

return 0;

}

void KPIInit(void){

WriteReg(REG\_GPIO\_CFG2, 0x000aaaaa); //将GPIO设置为KPI模式

WriteReg(REG\_AIC\_SCR29, 0x00000045); //将KPI 中断设置为高电平有效，优先级为5

WriteReg(REG\_AIC\_MECR, 0x20000000);

WriteReg(REG\_KPICONF, 0x00142fff);

}

//KPI中断服务子程序

\_\_irq void IRQ\_Handler(void){

//extern int keynum;

//int keynum=0;

u32 KeyValue, temp;

temp = ReadReg(REG\_KPISTATUS);

temp &= 0x0000000f;

if(temp > 0xB){

KeyValue = temp - 0xC;

}

else if(temp < 0x4){

KeyValue = temp + 0xC;

}

else{

KeyValue = temp;

}

if(KeyValue==0) keynum++;

switch (keynum)

{

case 0:

{

GLCD\_bitmap (0, 0, 320, 240, Background0);

GLCD\_init();

break;

}

case 1:

{

GLCD\_bitmap (0, 0, 320, 240, Background1);

GLCD\_init();

break;

}

case 2:

{

GLCD\_bitmap (0, 0, 320, 240, Background2);

GLCD\_init();

break;

}

case 3:

{

GLCD\_bitmap (0, 0, 320, 240, Background3);

GLCD\_init();

break;

}

case 4:

{

GLCD\_bitmap (0, 0, 320, 240, Background4);

GLCD\_init();

break;

}

case 5:

{

GLCD\_bitmap (0, 0, 320, 240, Background5);

GLCD\_init();

break;

}

}

if(keynum==6) keynum=-1;

}

## 在LCD上显示一张动态图片（GIF图）。

int main (void) { /\* Main Function \*/

int num = -1;

int dir = 1;

int off = 0;

int pic = 0;

int clk\_ani = 0;

Reg8(0x78000000) = LED\_ALL; /\* Turn off all LEDs \*/

/\* Setup Timer1 to generate interrupt every 10 ms \*/

REG\_TCR1 = 0xAC000000 + (15-1); /\* Dbg, IE, period mode, rst, prsc=80 \*/

REG\_TCR1 = 0xA8000000 + (15-1); /\* Dbg, IE, period mode, prsc=80 \*/

REG\_TICR1 = (15000000 / (15 \* 100)); /\* Reload value=15MHz/(prescaler\*100) \*/

REG\_TCR1 |= (1 << 30); /\* Start Timer 1 \*/

/\* Setup AIC for Timer1 interrupt \*/

REG\_AIC\_SCR14 = (1 << 6) | 1; /\* Int trig on high level, priority 1 \*/

REG\_AIC\_TEST = 1; /\* Enable Debugging of AIC \*/

REG\_AIC\_MECR = (1 << IRQ\_TIMER1); /\* Enable Timer 1 interrupt \*/

GLCD\_init();

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren001);

for (;;)

{ /\* Loop forever \*/

if (Clock100ms)

{

Clock100ms = 0;

if (clk\_ani++ == 1)

{ /\* Draw animation picture every 200 ms\*/

clk\_ani = 0;

switch(pic)

{

case 0:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren001);

break;

case 1:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren002);

break;

case 2:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren003);

break;

case 3:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren004);

break;

case 4:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren005);

break;

case 5:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren006);

break;

case 6:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren007);

break;

case 7:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren008);

break;

case 8:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren009);

break;

case 9:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren010);

break;

case 10:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren011);

break;

case 11:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren012);

break;

case 12:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren013);

break;

case 13:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren014);

break;

case 14:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren015);

break;

case 15:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren016);

break;

case 16:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren017);

break;

case 17:

GLCD\_bitmap (0, 0, 320, 240, gImage\_ren018);

break;

}

if (pic++ > 17) pic = 0;

}

}

if (Clock1s)

{ /\* Blink LED every 1 second \*/

Clock1s = 0;

if (!off)

{

/\* Calculate 'num': 0, 1, ... , LED\_NUM-1, LED\_NUM-1, ... , 1, 0, 0, .\*/

num += dir;

if (num == LED\_NUM) { dir = -1; num = LED\_NUM-1; }

else if (num < 0) { dir = 1; num = 0; }

Reg8(0x78000000) = ~led\_mask[num]; /\* Turn on LED with index 'num' \*/

} else

{

Reg8(0x78000000) = LED\_ALL; /\* Turn off all LEDs \*/

}

off = !off;

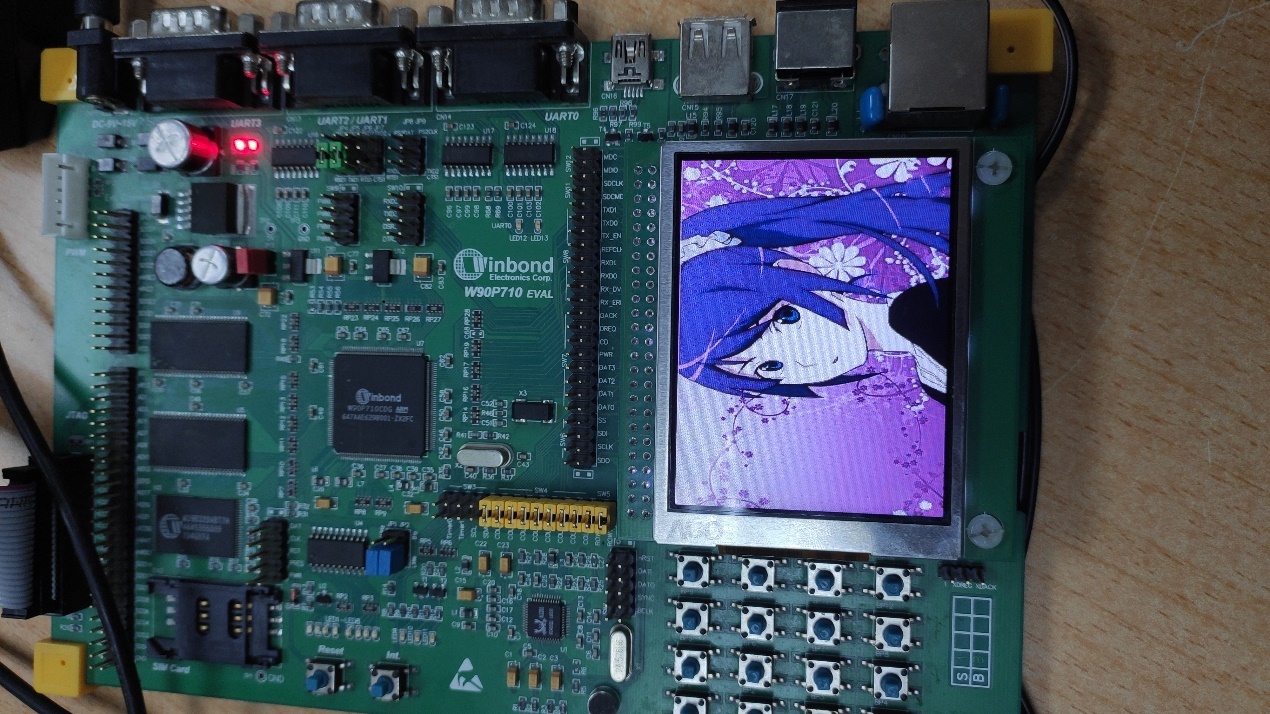
}

}

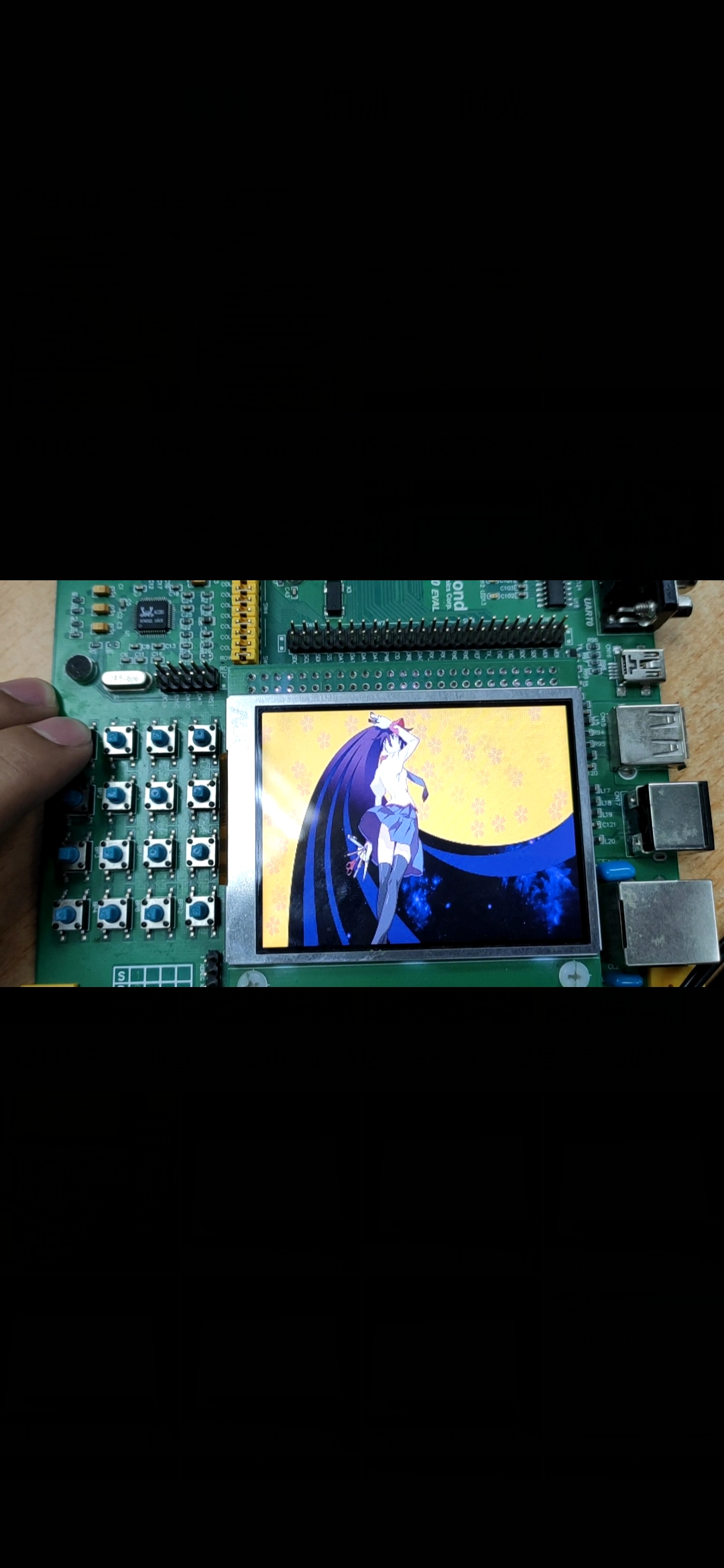
}

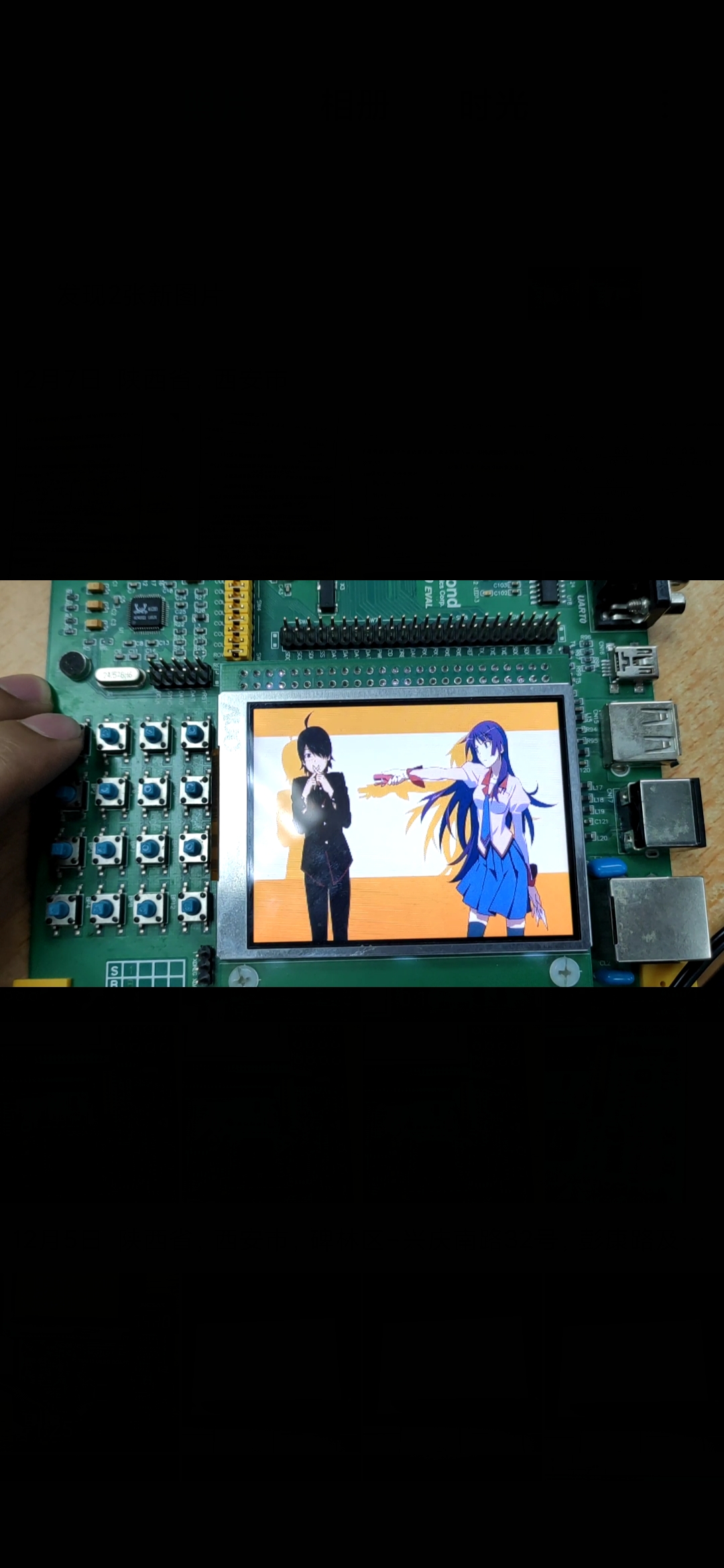
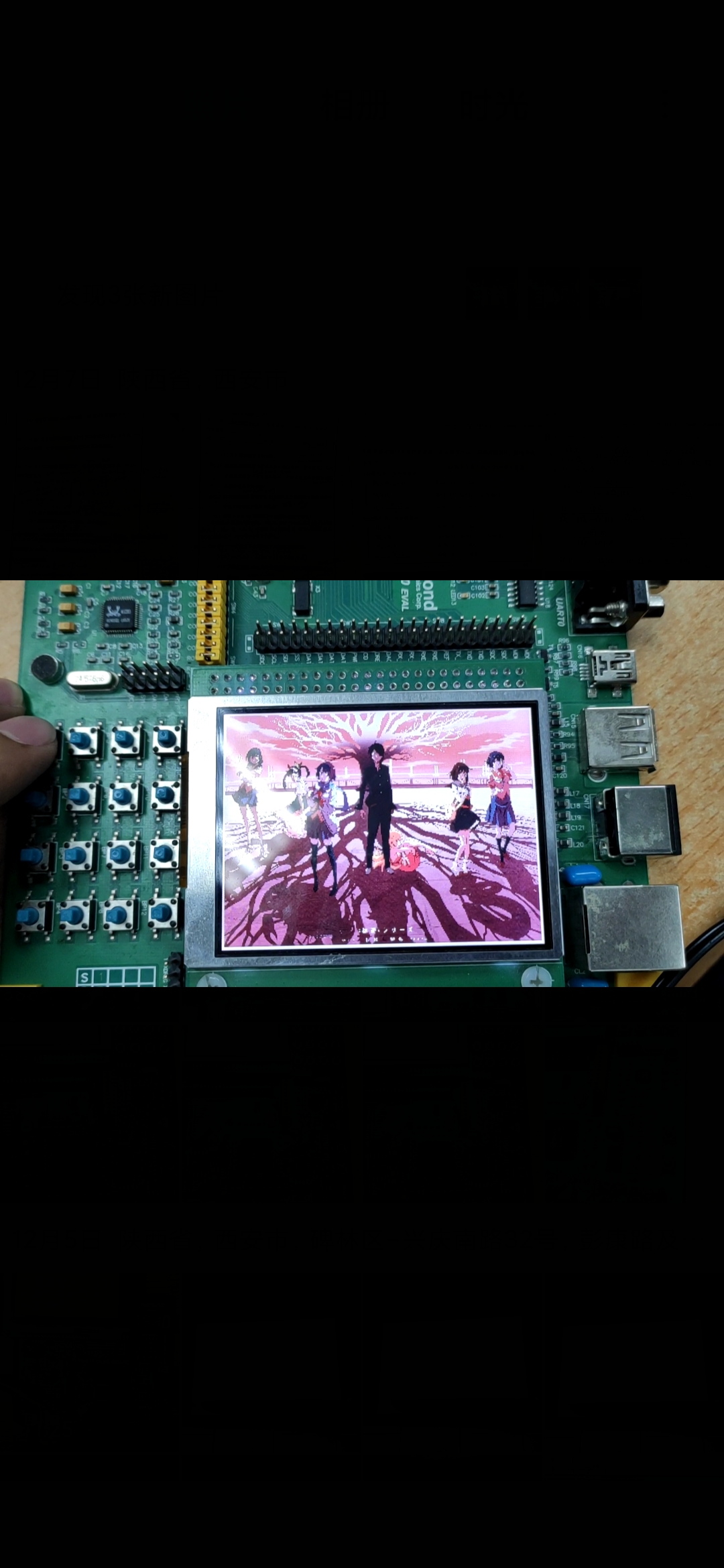
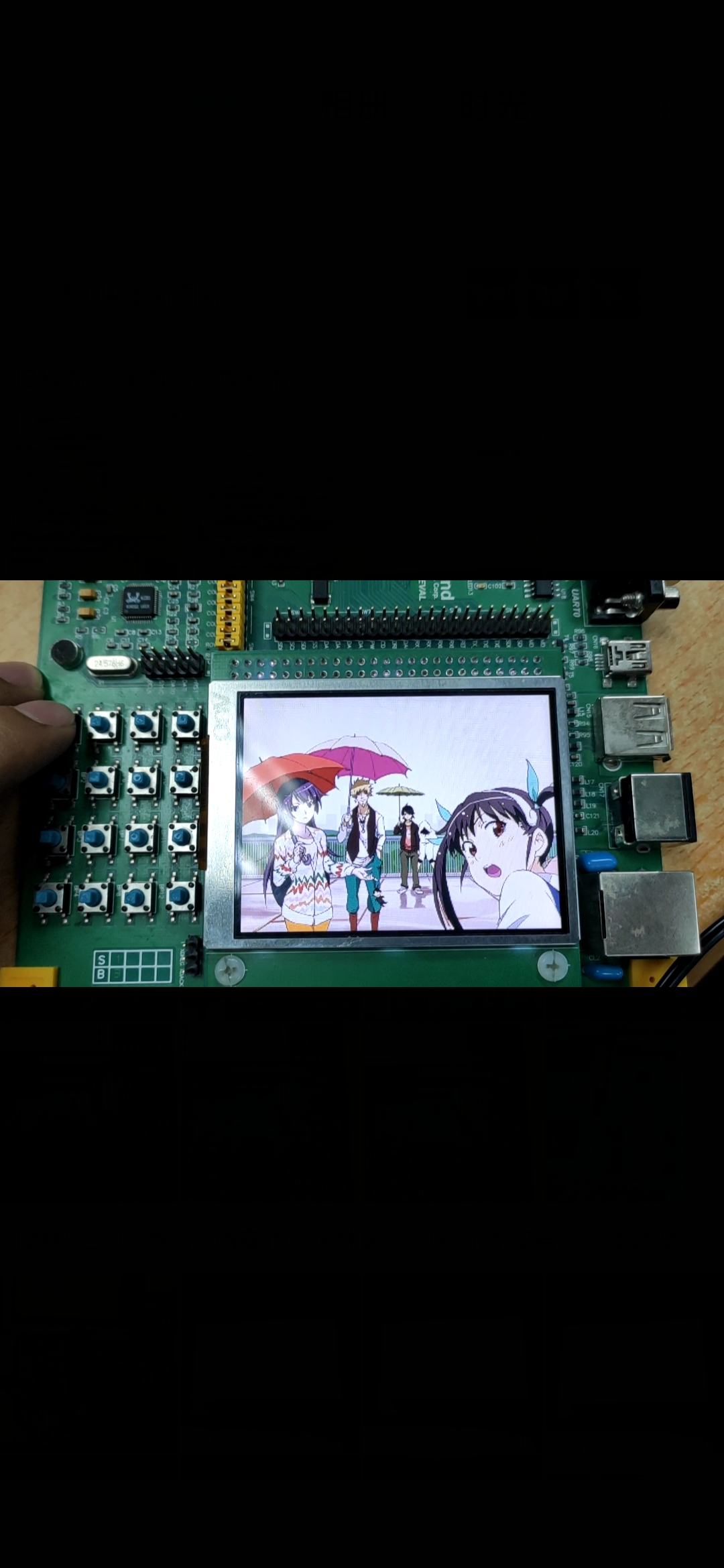
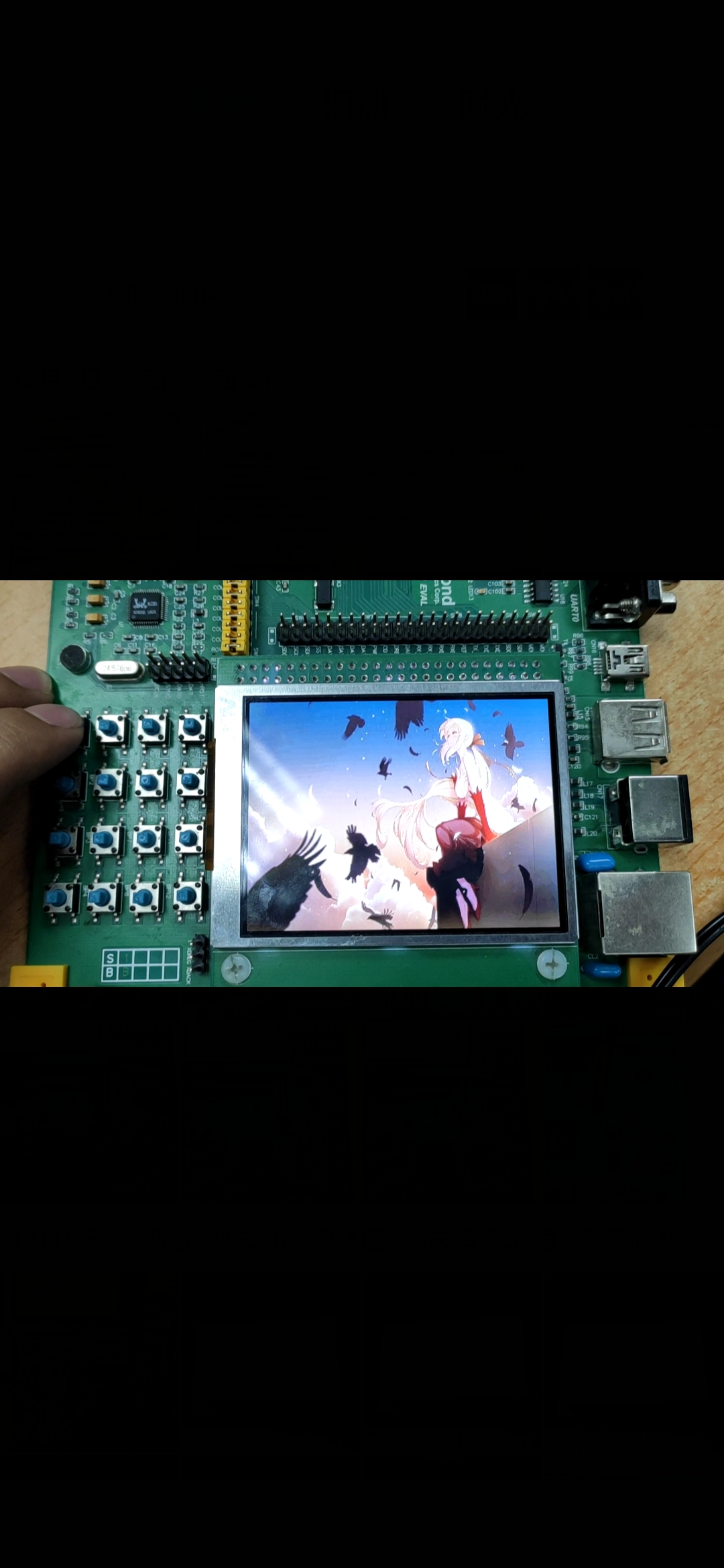
# 实验效果（详见视频）

## 在LCD上显示一张图片

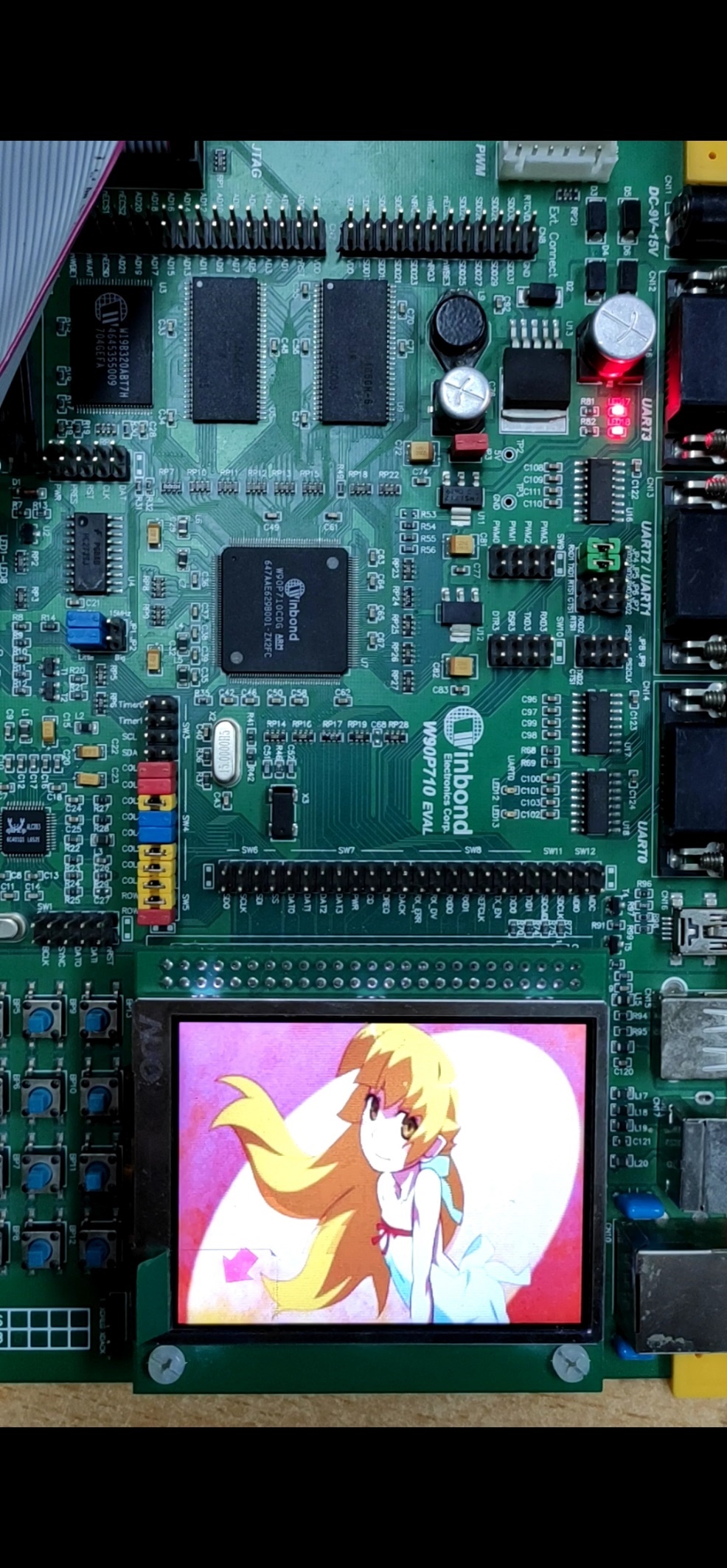
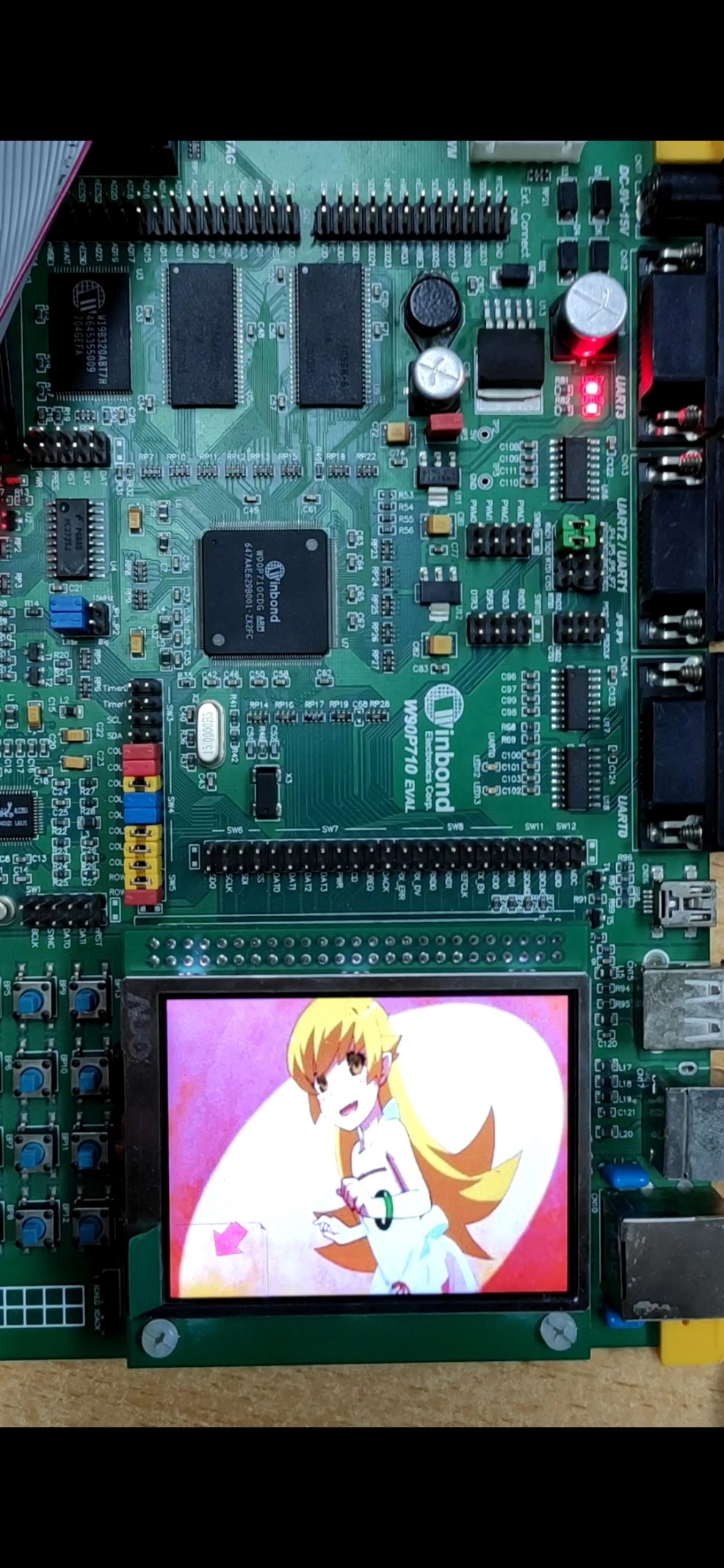
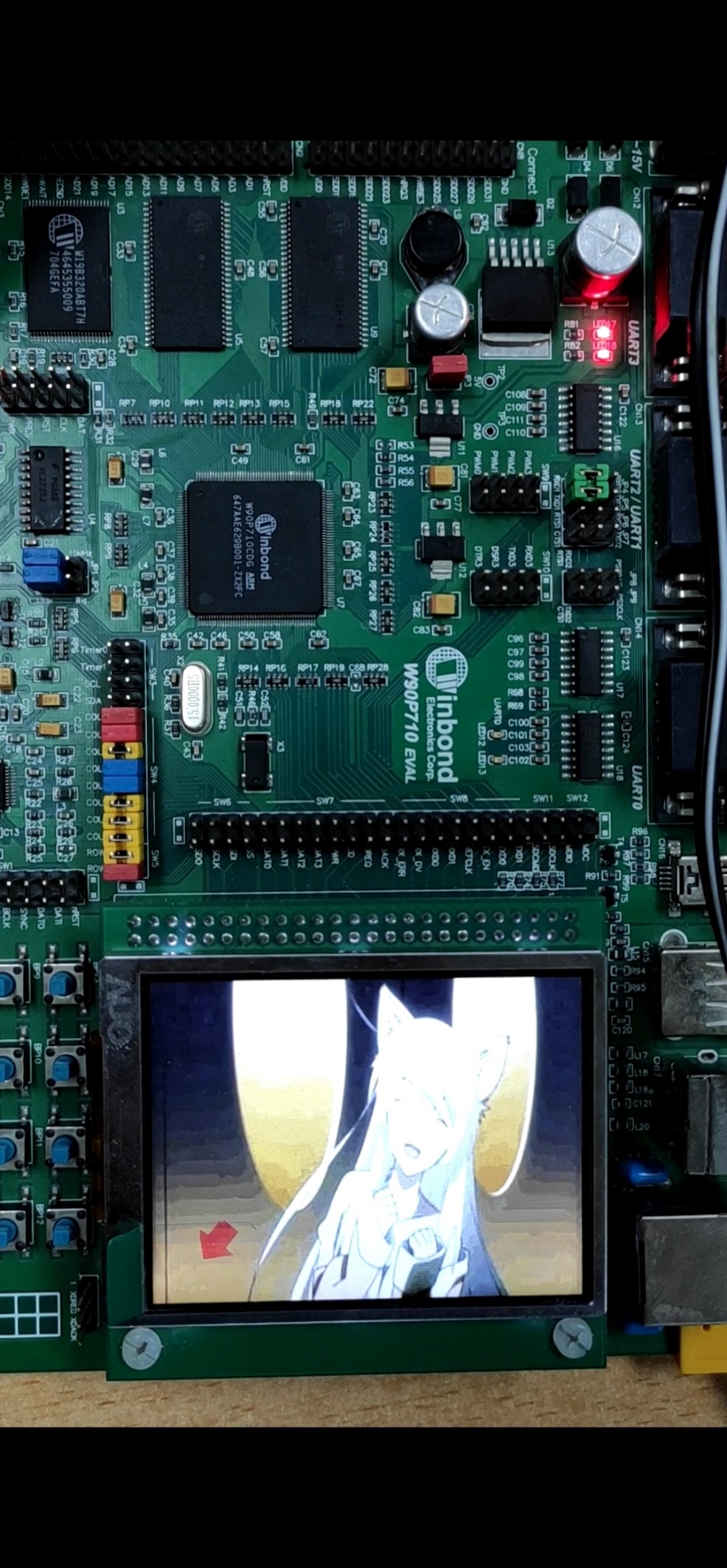
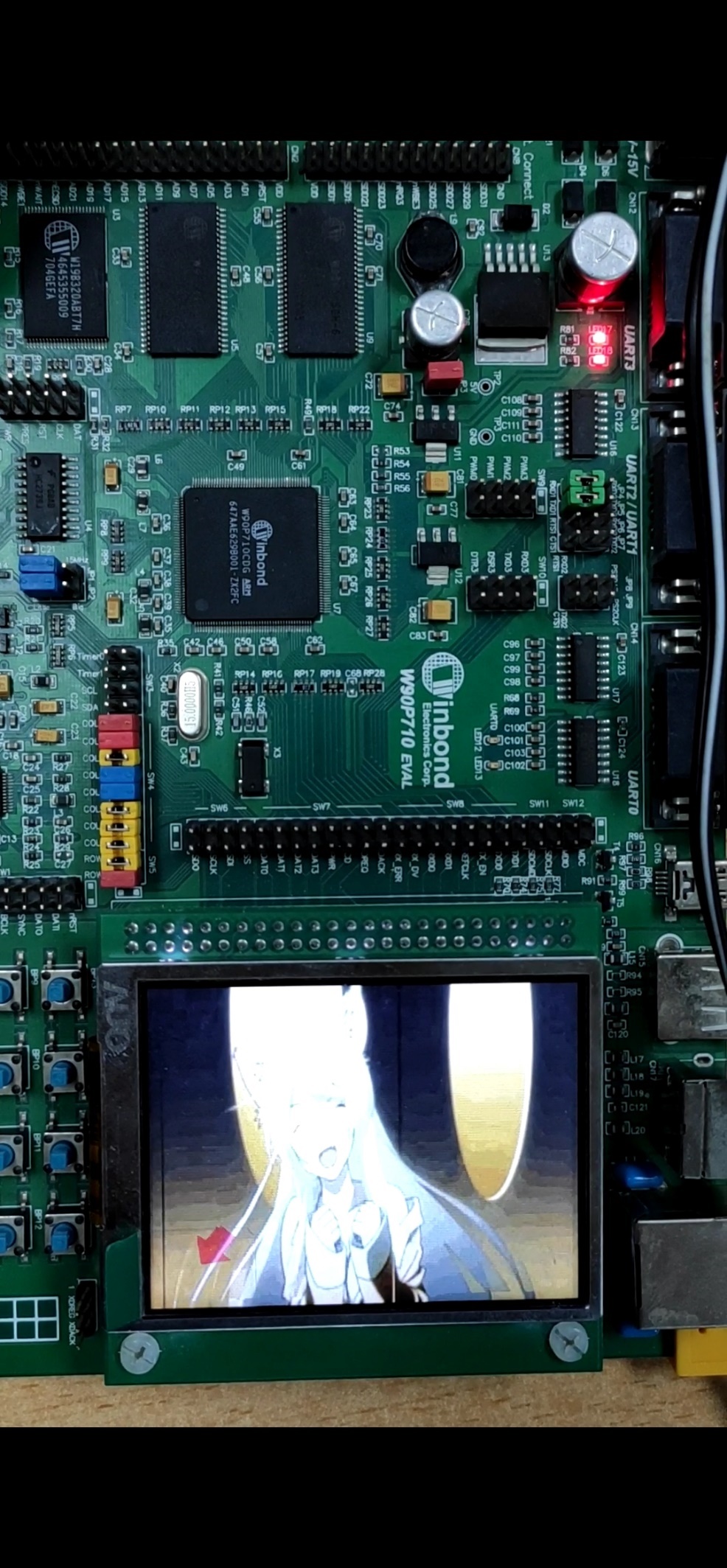


## 在LCD上显示多张图片，加入键盘按键功能，制成电子相册，用键盘按键翻页循环显示；

## 在LCD上显示一张动态图片（GIF图）。

# 实验总结

这是嵌入式系统的最后一次实验，我们也第一次接触到了课堂上讲的ARM开发板。我被ARM丰富的功能所吸引，相较前几次的实验板，它的显示屏更加清晰而且支持16位真彩显示。进行了初步的调试、简单功能测试的验证性实验后，我们开始进行自主设计的实验。实验任务一比较简单，通过把例程当中的图片修改为自己想要的图片即可完成。而实验任务二则比较复杂，在前面keyboard键盘扫描实验中得到启示，要想实现按键切换，需要引入中断。而刚开始我对于实验要求有所误解，以为是按下不同的键显示不同的照片，而老师的要求是按下同一个键实现照片的切换。因为按键只有十六个，照片多的话就不能实现功能了，而且还有其他功能可能会需要键盘。而在修改了程序以后发现第一张照片不显示，经过调试发现，原来是因为置位是给标志变量置了0，而下一次运行时直接加一就跳过了第一张图片，将置位改为-1即可解决这一问题。在GIF显示任务中，刚开始使用中断来实现播放GIF，但是每一帧之间会有一个黑屏，而把它放到主函数当中就不会出现这种情况。经过向老师请教，需要在子函数中关掉一个指令，这样才能避免每次刷新屏幕时的黑屏。由于人眼的视觉暂留效应，将每帧的间隔定位200ms时就能显得比较流畅，之前刷新的时间太长，就会导致GIF图片播放卡顿。而且原始GIF图的质量也会影响流畅度，若是原始图片本身帧数就低，那么播放出来的效果就也不好，所以要从这两个方面来优化图片显示效果。

这学期的嵌入式实验最大的收获就是能够把自己编写的代码在真正的硬件上跑起来，实现一个个简单的功能。这也是我们专业最重要的能力之一。我学会了嵌入式系统的简单运行和调试方法，也进一步锻炼了自己的代码能力。在半个学期的实验中虽然也会碰到各种各样的困难，但是在大家的讨论与思考当中都得到了解决。虽然实验结束了，但是我的学习之旅不会结束，在后面，我也会自己进行单片机与嵌入式系统的练习，并且利用寒假的时间学习一下51单片机的内容，预习一下下学期要进行的树莓派实验的内容，做好承前启后的工作。