TFT DRIVER DESIGN

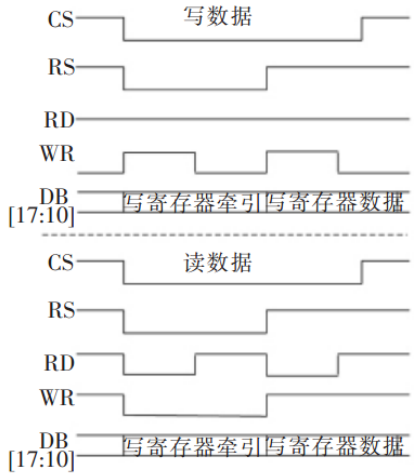
freescale™

第一章 基于 Freescale KL 的 TFT 驱动

一、介绍篇：

随着手机行业的发展，TFT 显示技术经历了从黑白到简易彩屏,再到如今的大屏电容触摸的发展过程，TFT 较之传统的数码管显示、黑白点阵屏等，有着更好的 UI 效果及客户体验 ,在从低分辨率到高分辨率的 TFT ,Freescale 均有相应的驱动方案。

按照液晶屏的尺寸来分，一般低于 3.5 寸的屏上都带有控制器，如常见的 2.8 寸触摸屏内置控制器 ILI9320/ILI9325 ,这种控制器一般的 MCU 都可以驱动，接口为 8080 时序如下图示：



MCU 通过 WR、RD、RS、CS 等控制线以及数据总线即可与之通信，CS 为片选信号，低电平有效，WR、RD 为读写控制信号，RS 为数据、命令选择信号 ,我们选择了 Freescale 的 KL25Z 来做低成本液晶驱动的参考设计。

Kinetis KL2 系列产品资源

- 32-100 引脚
- 最大 48MHz 工作频率
- 32-256KB 闪速

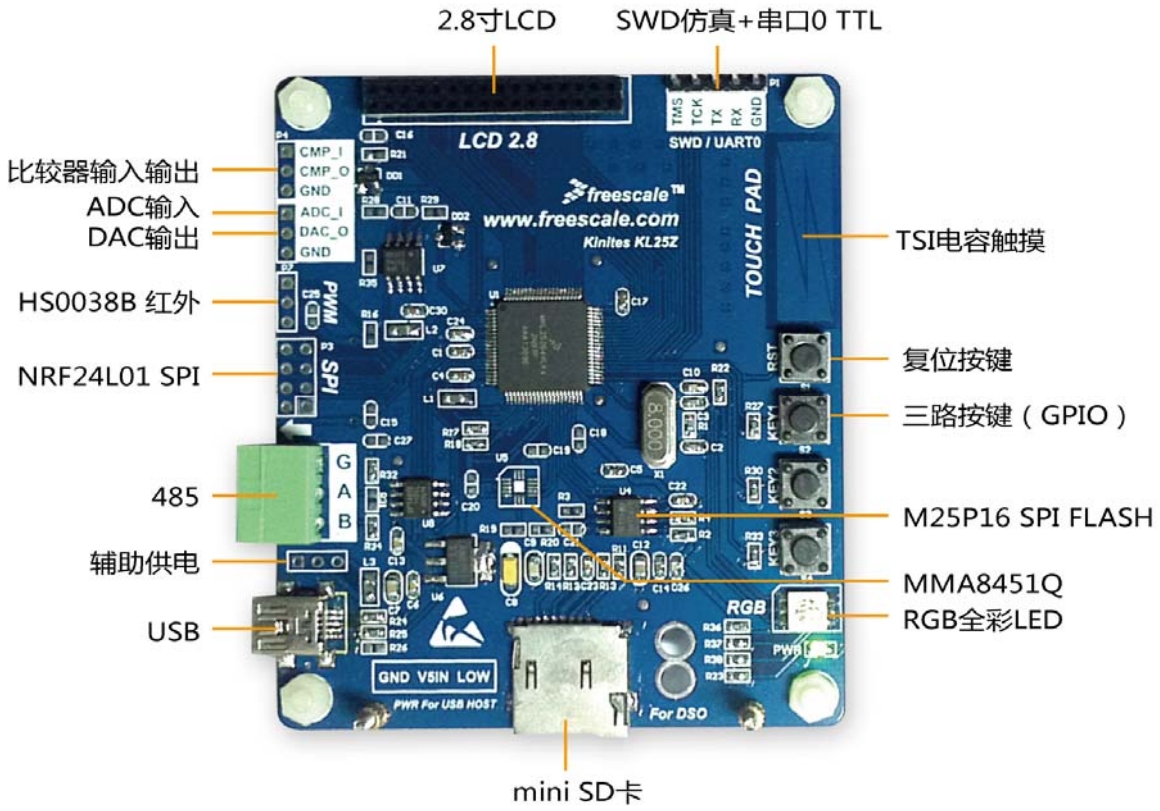
- 8-32KB SRAM
- USB OTG（FS）
- 模拟外设 16 位 A/D 转换器
- 模拟外设 12 位 D/A 转换器
- 高速比较器
- 低功率触摸感应界面（TSI）
- LPUART、UART、SPI、IIC
- 并且拥有多种低功耗工作模式，性价比非常高。

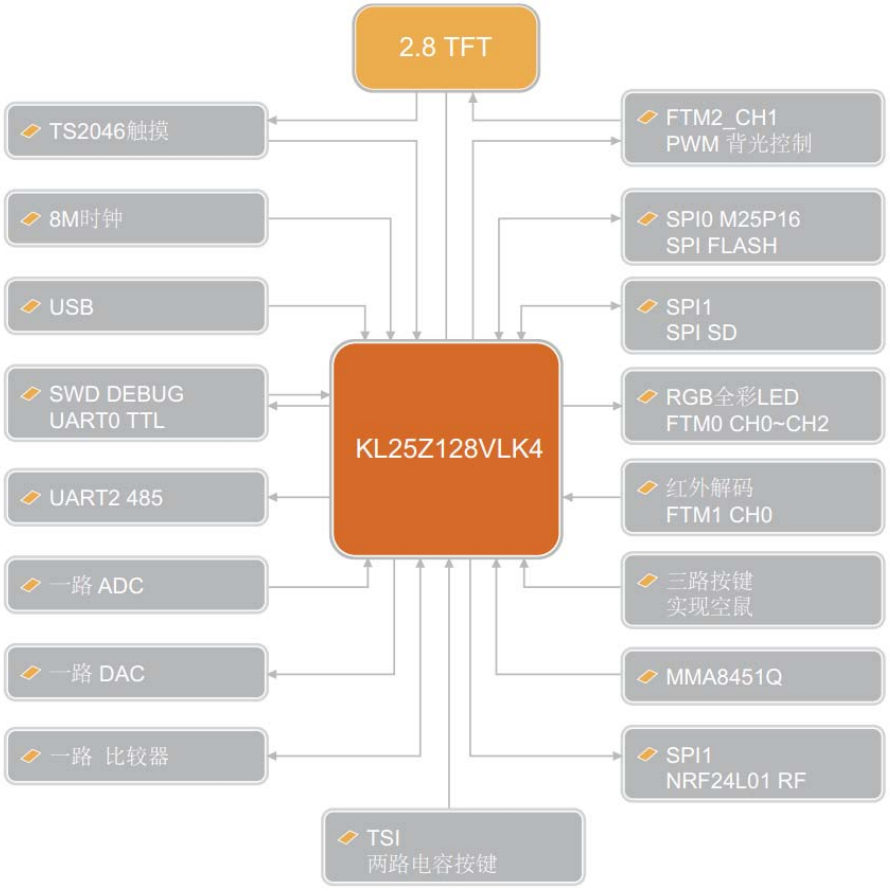
二、硬件篇：

因为 KL25Z 不支持外部扩展总线，我们可以通过 GPIO 以及一组 Port 口模拟 8080 总线时序，得益于 KL25Z 的 48M 主频以及 FastGPIO,经测试 2.8 寸 320*240 分辨率的 TFT 刷屏速度非常快。硬件连接如下：

PTC0/ADC0_SE14/TSI0_CH13/EXTRG_IN/CMP0_OUT	55	LCD_D0
PTC1/LLWU_P6/RTC_CLKIN/ADC0_SE15/TSI0_CH14/I2C1_SCL/FTM0_CH0	56	LCD_D1
PTC2/ADC0_SE11/TSI0_CH15/I2C1_SDA/FTM0_CH1	57	LCD_D2
PTC3/LLWU_P7/UART1_RX/FTM0_CH2/CLKOUT	58	LCD_D3
PTC4/LLWU_P8/SPI0_PCS0/UART1_TX/FTM0_CH3	61	LCD_D4
PTC5/LLWU_P9/SPI0_SCK/LPTMR0_ALT2/CMP0_OUT	62	LCD_D5
PTC6/LLWU_P10/CMP0_IN0/SPI0_MOSI/EXTRG_IN/SPI0_MISO	63	LCD_D6
PTC7/CMP0_IN1/SPI0_MISO/SPI0_MOSI	64	LCD_D7
PTC8/CMP0_IN2/I2C0_SCL/FTM0_CH4	65	LCD_D8
PTC9/CMP0_IN3/I2C0_SDA/FTM0_CH5	66	LCD_D9
PTC10/I2C0_SCL	67	LCD_D10
PTC11/I2C0_SDA	68	LCD_D11
PTC12/FTM_CLKIN0	69	LCD_D12
PTC13/FTM_CLKIN1	70	LCD_D13
PTC16	71	LCD_D14
PTC17	72	LCD_D15

为了能够更好的发挥 KL25 的功能，扩展了其他丰富的外设，可以作为 Kinetis KL 系列的评估板使用。


2.8寸LCD
SWD仿真+串口0 TTL
比较器输入输出
ADC输入
DAC输出
HS0038B 红外
NRF24L01 SPI
485
辅助供电
USB
TSI电容触摸
复位按键
三路按键（GPIO）
M25P16 SPI FLASH
MMA8451Q
RGB全彩LED
mini SD卡
LCD 2.8
SWD / UART0
TOUCH PAD
www.freescale.com
Kinites KL25Z
RGB
GND V5IN LOW
PWR For USB HOST
For DSO


2.8 TFT
TS2046触摸
8M时钟
USB
SWD DEBUG
UART0 TTL
UART2 485
一路 ADC
一路 DAC
一路 比较器
TSI
两路电容按键
FTM2_CH1
PWM 背光控制
SPI0 M25P16
SPI FLASH
SPI1
SPI SD
RGB全彩LED
FTM0 CH0~CH2
红外解码
FTM1 CH0
三路按键
实现空鼠
MMA8451Q
SPI1
NRF24L01 RF
KL25Z128VLK4

三、软件篇：

我们使用PTC口作为LCD的数据总线接口，因为选用的KL25Z128VLK4 PTC口没有PTC14和PTC15，但是有PTC16和PTC17，所以对应于LCD_D14和LCD_D15，程序中对需做下变换即可：dat = dat | ((dat>>13)<<15);其他控制线使用GPIO口模拟控制。

```
void lcd_data_port(u32 dat) {
        dat = dat | ((dat>>13)<<15);
        FGPIOC_PDOR = dat;
}
```

LCD初始化代码详见void lcd_cfg_init()函数。通过GPIO输出高低电平作为控制信号，实现如下函数：

```
void LCD_WriteCmd(u16 val){
      LCD_CS_L();
       LCD_RS_L();
       LCD_RD_H();
      lcd_data_port(val);
       LCD_WR_L();
       LCD_WR_H();
       LCD_CS_H();
}
void LCD_WriteData(u16 val){
  LCD_CS_L();
       LCD_RS_H();
       LCD_RD_H();
        lcd_data_port(val);
       LCD_WR_L();
       LCD_WR_H();
       LCD_CS_H();
```

```
}
void LCD_WriteReg(u16 index,u16 val){
        LCD_WriteCmd(index);
        LCD_WriteData(val);
}
```

然后实现基本的画点函数：

```
void LCD_Point(u16 xpos, u16 ypos,u16 color) {
    LCD_Cursor(xpos,ypos);
    LCD_WriteCmd(0x0022);
    LCD_WriteData(color);
}
```

根据画点函数依次实现画线、画圆、矩形填充、清屏、以及字符显示、数字显示、BMP图像显示等函数：

```
//*********************************************************************************//
void LCD_Fill(u16 x1, u16 y1, u16 x2, u16 y2, u16 color);          //矩形填充
void LCD_Point(u16 xpos, u16 ypos,u16 color);                      //画点
void LCD_Clear(u16 Color);                                         //清屏
//*********************************************************************************//
void LCD_Char(u16 Xpos, u16 Ypos, u8 Ascii, u16 color,int bk_clr);  //显示字符
void LCD_String(u16 xpos, u16 ypos, u8 *ptr, u16 color,int bk_clr); //显示字符串
void LCD_Num(u16 x,u16 y,u32 num,u8 len,u16 color,int bk_clr);      //显示数字
void LCD_Bmp16bit(u16 x,u16 y,u16 b,u16 h,const u8 *addr);          //显示BMP
//*********************************************************************************//
```

这些是 GUI 的基础，基于画点、画线函数才可以实现更为复杂的 UI 效果。在此不得不提的是 LCD 取模软件：Image2LCD

经过Image2LCD取出图片的数据，保存成图像.h文件，编译到程序中，通过调用LCD_Bmp16bit(u16 x,u16 y,u16 b,u16 h,const u8 *addr) 函数即可在指定的位置显示相应的图片。效果如下图：

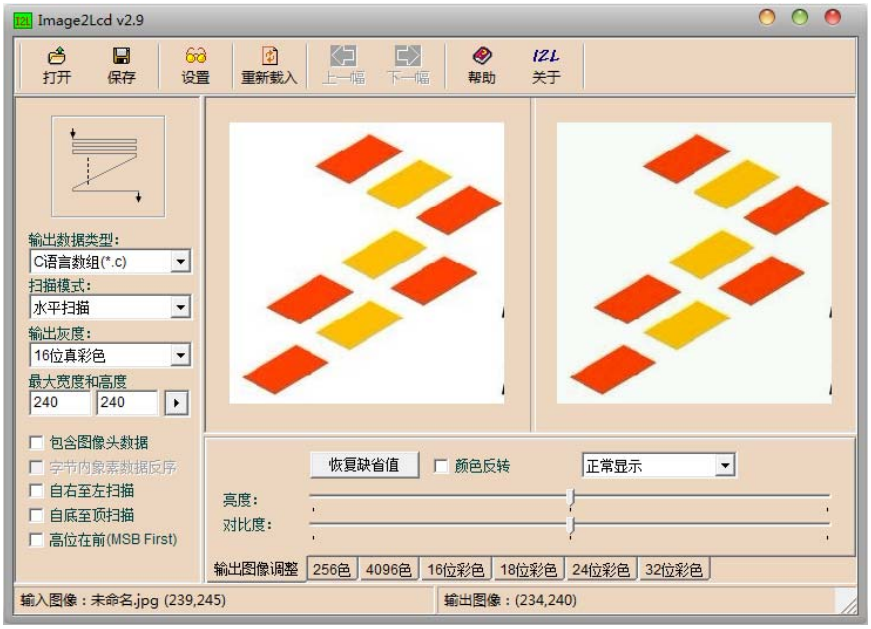


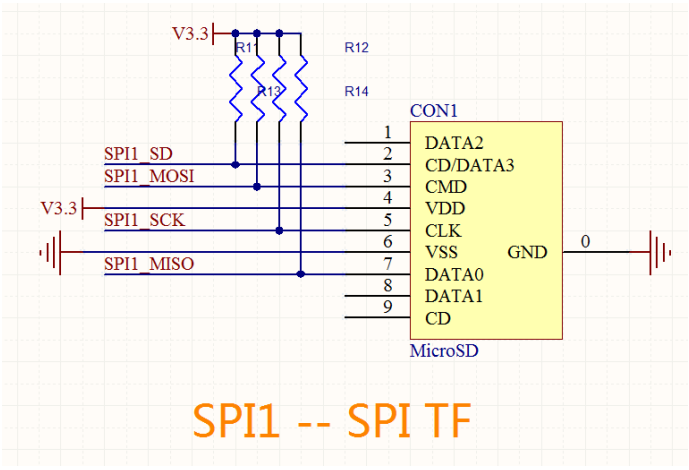
Image2Lcd v2.9
打开
保存
设置
重新载入
上一幅
下一幅
帮助
关于
输出数据类型:
C语言数组(*.c)
扫描模式:
水平扫描
输出灰度:
16位真彩色
最大宽度和高度
240
240
包含图像头数据
字节内象素数据反序
自右至左扫描
自底至顶扫描
高位在前(MSB First)
恢复缺省值
颜色反转
正常显示
亮度:
对比度:
输出图像调整
256色
4096色
16位彩色
18位彩色
24位彩色
32位彩色
输入图像：未命名.jpg (239,245)
输出图像：(234,240)


freescale
Kinetis
www.freescale.com
CMP_
CMP_
GND
ADC_
DAC_
GND
PWM & SPI
RST
KEY1
KEY2
KEY3
PWR
AA002AH

四、应用扩展篇——基于此平台实现简易数码相册功能

4-1 SPI接口 SD卡驱动及FatFS文件系统移植

2.8寸TFT屏分辨率为240*320，一个像素为16位565格式，所以一张满屏显示的图片需要占用240*320*2/1024 = 150K，这么大的图像数据对于资源有限的嵌入式来说是无法容纳的，所以我们选择了更适合应用的MINI SD卡作为图片数据的存储，通过SPI接口访问SD卡，硬件连接如下：



SPI驱动中我们首先构建如下函数：

```
void spi_init(SPI_MemMapPtr SPI,BOOLEAN bMode)；      //初始化SPI
u8 spi_send_byte(SPI_MemMapPtr SPI,u8 ucData)；       //SPI发送一个字节数据
u8 spi_read_byte(SPI_MemMapPtr SPI)；                 //SPI读取一个字节数据
```

SD卡驱动——初始化流程如下：

（1）首先是 74个clk，然后 CS_LOW；发送 CMD0，收到的应答是 0x01 ；

（2）接着发送CMD1，收到的应答应该是 0x00 ；最后 CS_HIGH 。

至此，初始化完成。

需要注意的问题：

（1） 初始化的时钟不宜太快，可以在SD卡初始化完成后可提高数据读写速度；

（2） 在发送命令之前和收到应答之后，主控制器应该发送8个时钟完成相应操作； CMD0的CRC是0x95 ，其余命令的CRC 无所谓。

（3） 读取单块数据流程：CS_LOW -->8个clk -->发送CMD17 -->接收响应 R1 -->接收读数据起始令牌0xFE -->接收数据 -->接收CRC -->8个clk -->CS_HIGH；

（4） 写入单块数据流程：CS_LOW -->8个clk -->发送CMD24 -->接收响应 R1 -->写入读数据起始令牌0xFE -->写入数据 -->接收CRC -->8个clk -->CS_HIGH；

（5） 读写操作指令：单块写命令CMD24，多块写命令CMD25；单块读命令CMD17，多块读命令CMD18。单块读写时，数据块的长度为512 字节，多块读写时SD卡收到1个停止命令CMD12后停止读写。

SD卡初始化后即可移植FatFS文件系统，通过它管理SD卡上的文件。 FatFs Module是一种完全免费开源的FAT文件系统模块，专门为小型的嵌入式系统而设计。它完全用标准C语言编写，所以具有良好的硬件平台独立性，可以移植到8051、PIC、AVR、SH、Z80、H8、ARM等系列单片机上，且只需做简单的修改。它支持FATl2、FATl6和FAT32，支持多个存储媒介；有独立的缓冲区，可以对多个文件进行读 / 写，并特别对8位单片机和16位单片机做了优化。FATFS源代码的获取，可以到官网下载：http://elm-chan.org/fsw/ff/00index_e.html

FatFS的移植主要通过SPI驱动构建如下几个函数：

```
1 DSTATUS disk_initialize (BYTE);                          //SD卡的初始化
2 DSTATUS disk_status (BYTE);                              //获取SD卡的状态
3 DRESULT disk_read (BYTE, BYTE*, DWORD, BYTE);            //从SD卡读取数据
4 DRESULT disk_write (BYTE, const BYTE*, DWORD, BYTE);     //将数据写入SD卡，若该文件系统为只读文件系统则不用实现该函数
5 DRESULT disk_ioctl (BYTE, BYTE, void*);                  //获取SD卡文件系统相关信息
```

移植完成后即可通过 FatFs 读取 SD 卡上的文件了 ,FatFs 提供下面的函数可供使用：

f_mount - 注册/注销一个工作区域（Work Area）

f_open - 打开/创建一个文件f_close - 关闭一个文件

f_read - 读文件f_write - 写文件

f_lseek - 移动文件读/写指针

f_truncate - 截断文件

f_sync - 冲洗缓冲数据 Flush Cached Data

f_opendir - 打开一个目录

f_readdir - 读取目录条目

f_getfree - 获取空闲簇 Get Free Clusters

f_stat - 获取文件状态

f_mkdir - 创建一个目录

f_unlink - 删除一个文件或目录

f_chmod - 改变属性（Attribute）

f_utime - 改变时间戳（Timestamp）

f_rename - 重命名/移动一个文件或文件夹

f_mkfs - 在驱动器上创建一个文件系统

f_forward - 直接转移文件数据到一个数据流 Forward file data to the stream directly

f_gets - 读一个字符串

f_putc - 写一个字符

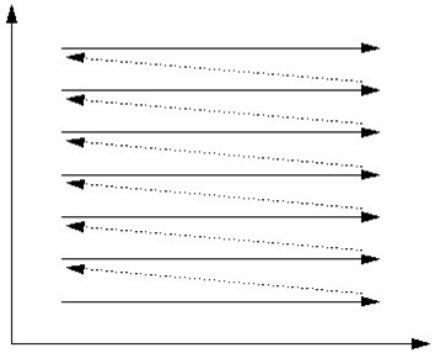
f_puts - 写一个字符传

f_printf - 写一个格式化的字符磁盘I/O接口

4-2 BMP文件的解码：

（参考小马哥的iBoard电子学堂：http://www.cnblogs.com/xiaomagee/archive/2012/11/27/2791539.html）

BMP（Bitmap）是 Windows 操作系统中的标准图像文件格式，与硬件设备无关，使用非常广泛。它采用位映射存储格式，BMP 文件的图像深度可选为 1、4、8、16 或 24bit。数据区里的数据是线性的，行主序，依次是点 1 的 B 值、点 1 的 G 值、点 1 的 R 值，点 2 的 B 值、点 2 的 G 值、点 2 的 R 值等等。需要注意的是：Windows 中普遍按照行倒向扫描的约定读取 BMP 文件，此时，图像的扫描方式是从左到右、从下到上。扫描方式如下图所示：



BMP 文件的数据格式一般由四部分组成：位图文件头、位图信息头、调色板及位图数据，如下表所示:

块名称	大小/bit
位图文件头	14
位图信息头	40
调色板（可选块）	8（1 bit 位图调色板） 64（4 bit 位图调色板） 1024（8 bit 位图调色板）
位图数据	由图像的实际尺寸决定

通过FatFs读取存储于SD卡上的BMP文件，并解码，然后调用 void LCD_Point(u16 xpos, u16 ypos,u16 color) 函数即可实现BMP图像文件的显示，程序中通过设置PIT定时1S中断，然后在定时器中做5S钟计数，即可循环播放存储于SD卡上的图片，实现数码相册的功能。

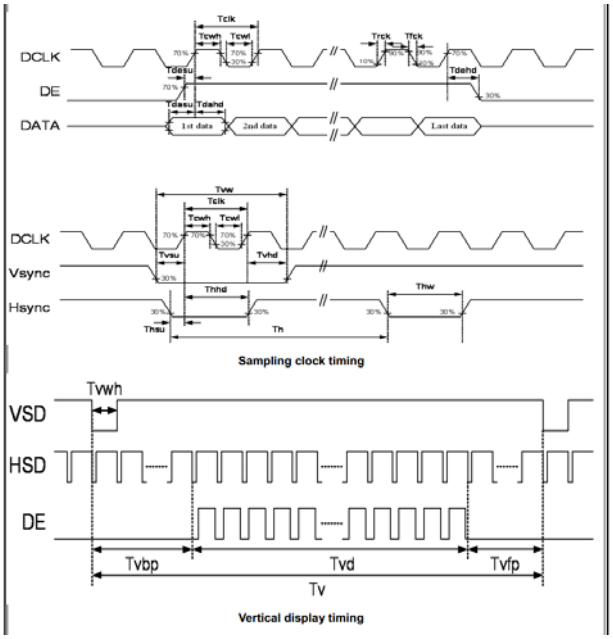
详细例程见附件。

第二章： K60 FlexBus 扩展 SSD1963 驱动 4.3 寸 TFT

一、介绍篇：

相比于 2.8 寸 TFT，4.3 寸更显高端大气上档次，可是 4.3 寸液晶屏基本上都不含控制器，需要支持 LCD 控制器的 MCU 来驱动，Freescale 的 K70 支持 DDR，支持 LCD 控制器，用它驱动 4.3 寸自然是小 case，不过 K70 高高在上的价格限制了它的应用，所以我们选择 K60 通过 FlexBus 总线外扩 SSD1963/RA8875 实现 4.3 寸及以上的屏的驱动。（事实上，如果不需要以太网功能，可以选择成本更低的 K10 或 K20 ）。

一般 4.3 寸通用时序如图：



根据上图可知，对于 4.3 寸屏的驱动需要不断给时钟、给数据，类似于 VGA 基于行场扫描的模式，如果用一般的 MCU 通过 GPIO 模拟时序，那么 MCU 基本上做不了其他事了。而 SSD1963/RA8875 恰恰是解决这个问题的方案，它的一侧是对 MCU 的，即 8080 时序，可通过 WR、RD、RS、CS 以及数据线对其进行读写，另一侧则是对通用 TFT 进行不断刷新的接口。我

们先看看 SSD1963 与 RA8875 的区别

SSD1963 VS RA8875

支持功能		SSD1963	RA8875	备注
类别	功能描述			
支持 LCD 规格	分辨率 800*480	支持	支持	RGB 接口
电源	单电源	不支持	支持	SSD1963 需要外供 1.2V 和 3.3V 两组电压
MCU 接口	8080/8bit	支持	支持	RA8875 支持多种串、并模式 MCU 输入接口
	8080/16bit	支持	支持	
	6800/8bit	支持	支持	
	6800/16bit	支持	支持	
	I2C	不支持	支持	
	3-wire SPI	不支持	支持	
	4- wire SPI	不支持	支持	
2D 图形绘制	绘制圆、椭圆、直线、矩形、三角形等	不支持	支持	RA8875 支持填充及非填充 2D 图形绘制
BTE	图形处理加速引擎	不支持	支持	RA8875 内部集成图形处理加速引擎，常规图形处理无需通过 MCU 程式运算实现，减轻 MCU 工作量
双图层	800*480@256 色	不支持	支持	RA8875 分辨率≤480*272 双图层模式支持 64K 色
	480*272@64K 色	不支持	支持	
字库	简体中文字库	不支持	支持	RA8875 可直接外接上海集通字库芯片，字符调用时只需通过 MCU 向 RA8875 写入 ASCII 码或中文区位码即可，RA8875 内部会自动完成字符显示数据向 DDRAM 的传输
	繁体中文字库	不支持	支持	
	ISO/IEC 8859-1/2/3/4 英欧文字库	不支持	支持	
	字符 1~4 倍放大缩小	不支持	支持	
	字符旋转	不支持	支持	
键盘	4*5 键盘控制接口	不支持	支持	RA8875 内建键盘控制器，通过对相关积存器操作即可实现对键盘的控制
触摸屏	4 线电阻触摸屏控制	不支持	支持	RA8875 内建 4 线触摸屏控制器，通过对相关积存器操作即可实现对触摸屏的控制
PWM	脉冲信号，一般用于背光的调节	支持	支持	R8875 有两组 PWM 输出，SSD1963 为一组
DMA	支持外接 Serial FLASH 用作 DMA 功能	不支持	支持	R8875 支持外接 Serial FLASH，用于预存图片数据
GPIO	扩展 GPIO 口	支持	支持	通过积存器操作读取或设置 GPIO 上的电平，一般用来模拟 LCD 屏或其他功能模块的控制信号，如 RESET、CS、SPI 时序

虽然性能上 RA8875 要优于 SSD1963 ,不过 SSD1963 的价格优势十分

明显，所以我们还是选择了 SSD1963 作为 4.3 寸 LCD 的驱动。

二、硬件篇：

让我们来看看如何和 K60 的 FlexBus 总线相连：

（1） CS：片选（低有效）

（2） RS：数据/指令选择（高：数据，低：命令）

（3） WR：写操作（低有效）

（4） RD：读操作（低有效）

（5） DB7~DB0 或者 DB15~DB0：数据线。

根据以上接口就很容易挂接到 FlexBus 总线上：

SSD1963 （8080）	MCU FlexBus
CS	FB_CS0
WR	FB_RW
RD	FB_OE
DB15~DB0(/DB7~DB0)	FB_AD15~FB_AD0(/FB_AD7~FB_AD0)

问题是：RS 如何来接？Flexbus 总线中并没有 RS 接口。

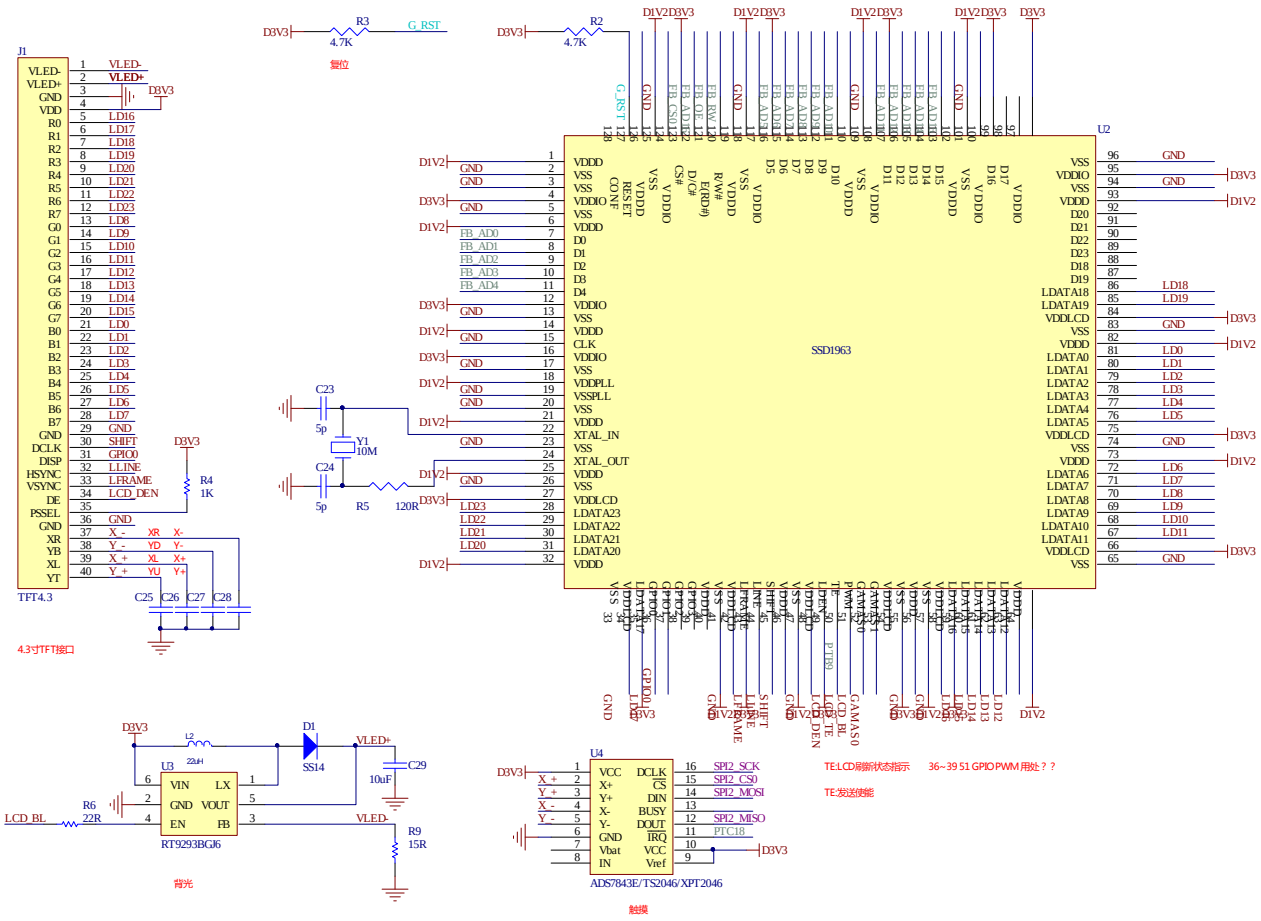
我们来看 Flexbus 总线时序：

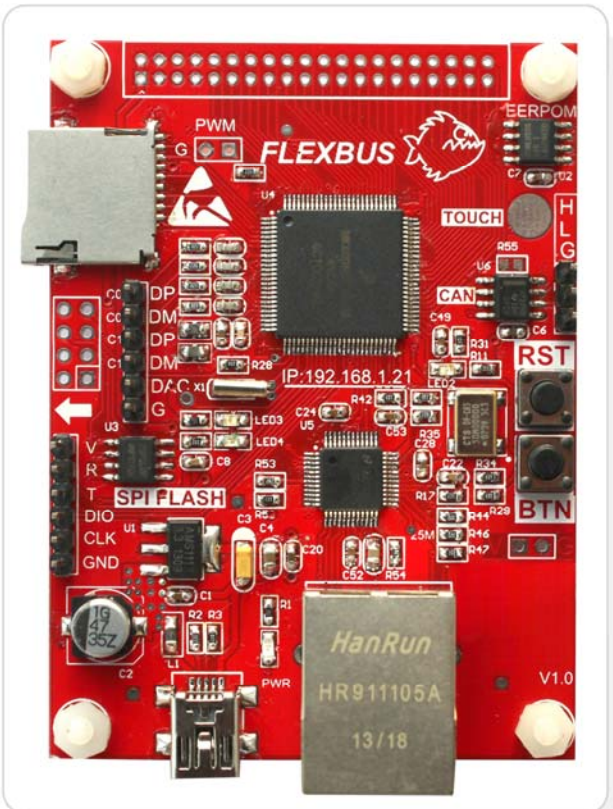
Flexbus 的基地址为 0x6000_0000，当我们执行：

*（volatile　unsigned short int *）(0x60000000) = dat 时，那么此时根据 Flexbus 的选址空间，CS0 将输出低，同时这是一个写操作，FB_WR 为低，因为地址线与数据线复用，所以在 FB_AD15~AD0 上会先出现地址 0000_0000,跟着出现数据 dat。所以我们可以使用一根地址

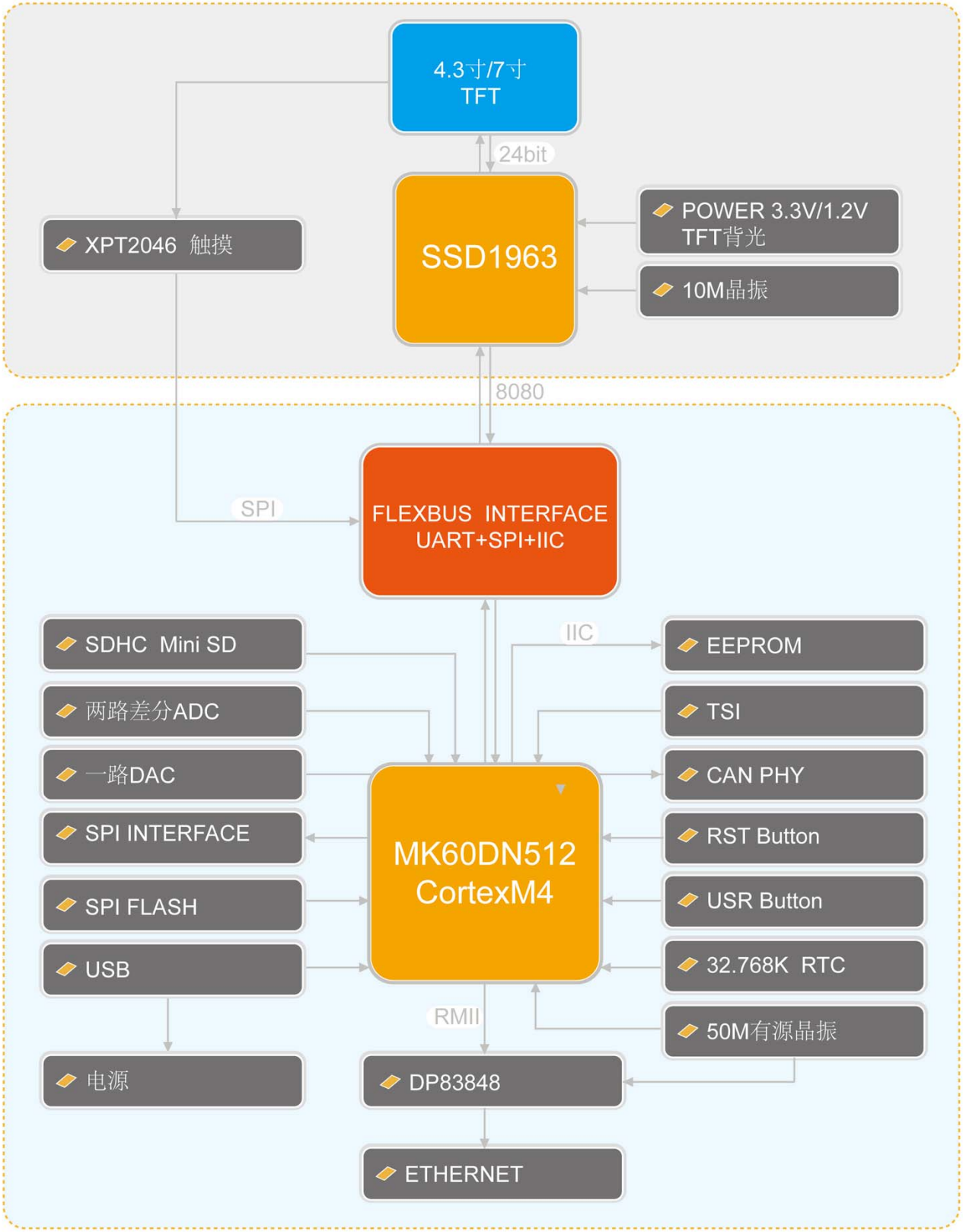
线来做 RS 的选择，如选择 FB_AD16，对于的地址为：0x60010000（0x60000000+2^16），当我们执行：*(volatile unsigned short int *) (0x60010000) = dat 时，此时地址线 FB_AD16 输出高，跟着 dat 会出现在 FB_AD15~FB_AD0 数据总线上，因此通过一条地址线实现了 RS 数据与指令的切换。

K60 与 SSD1963 驱动 4.3 寸 TFT（瀚彩 4.3 寸）原理图如下：（放大更清晰）


J1
TFT4.3
4.3寸TFT接口
SSD1963
U2
R3
4.7K
复位
R2
4.7K
C23
5p
Y1
10M
C24
5p
R5
120R
R4
1K
C25 C26 C27 C28
D1
SS14
L2
22uH
U3
RT9293BGJ6
R6
22R
LCD_BL
背光
C29
10uF
R9
15R
U4
ADS7843E/TS2046/XPT2046
触摸
TE:LCD刷新状态指示
TE:发送使能
36~39 51 GPIO PWM 用处？？


PWM
FLEXBUS
EERPOM
TOUCH
CAN
RST
BTN
SPI FLASH
DIO
CLK
GND
IP:192.168.1.21
HanRun
HR911105A
13/18
V1.0
PWR

设计架构图


4.3寸/7寸
TFT
24bit
SSD1963
POWER 3.3V/1.2V
TFT背光
10M晶振
XPT2046 触摸
8080
SPI
FLEXBUS INTERFACE
UART+SPI+IIC
IIC
SDHC Mini SD
两路差分ADC
一路DAC
SPI INTERFACE
SPI FLASH
USB
EEPROM
TSI
CAN PHY
RST Button
USR Button
32.768K RTC
50M有源晶振
MK60DN512
CortexM4
RMII
电源
DP83848
ETHERNET

为了能够更好的发挥K60的功能，扩展了其他丰富的外设，可以作为Kinetis K系列的评估板使用。

三、软件篇：

根据 4.3 寸 LCD 的规格书，需要先定义相关的配置参数：

```
/************ 4.3 TFT 参数配置 ***************/
#define     HDP          479
#define     HT           531
#define     HPS          43
#define     LPS          8
#define     HPW          10

#define     VDP          271
#define     VT           288
#define     VPS          12
#define     FPS          4
#define     VPW          10
/*********************************************/
```

然后根据地址空间及 RS 定义写指令与写数据,

```
#define FLEXBUS_BASE_ADDRESS            0x60000000
#define LCD_COMMAND_ADDRESS             *(volatile unsigned short *)0x60000000
#define LCD_DATA_ADDRESS                *(volatile unsigned short *)0x60010000
      //FB_AD16
#define LCD_WR_REG(reg)                 LCD_COMMAND_ADDRESS = reg;
#define LCD_WR_DAT(dat)                 LCD_DATA_ADDRESS = dat;
```

接下来是 K60 Flexbus 的 GPIO 配置：

```
void SSD1963_Flexbus_Init(void)
{
 SIM->SCGC5 |= SIM_SCGC5_PORTB_MASK;
```

```
SIM->SCGC5 |= SIM_SCGC5_PORTC_MASK;
SIM->SCGC5 |= SIM_SCGC5_PORTD_MASK;
SIM->SOPT2 |= SIM_SOPT2_FBSL(3);
SIM->SCGC7 |= SIM_SCGC7_FLEXBUS_MASK;

SIM->CLKDIV1 |= SIM_CLKDIV1_OUTDIV3(2);                    //分频
PORTB->PCR[17] = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  //fb_ad[16]   FB_AD16
as LCD_RS
PORTB->PCR[18] = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  // fb_ad[15]
PORTC->PCR[0]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  // fb_ad[14]
PORTC->PCR[1]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  // fb_ad[13]
PORTC->PCR[2]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  //fb_ad[12]
PORTC->PCR[4]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  // fb_ad[11]
PORTC->PCR[5]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  //fb_ad[10]
PORTC->PCR[6]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  //fb_ad[9]
PORTC->PCR[7]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  //fb_ad[8]
PORTC->PCR[8]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  //fb_ad[7]
PORTC->PCR[9]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  //fb_ad[6]
PORTC->PCR[10] = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  //fb_ad[5]
PORTD->PCR[2]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  // fb_ad[4]
PORTD->PCR[3]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  //fb_ad[3]
PORTD->PCR[4]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  //fb_ad[2]
PORTD->PCR[5]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  //fb_ad[1]
PORTD->PCR[6]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;  //fb_ad[0]
//control signals
PORTB->PCR[19] = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;   // fb_oe_b
PORTD->PCR[1]  = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;   // fb_cs0_b
PORTC->PCR[11] = PORT_PCR_MUX(5)|PORT_PCR_DSE_MASK;   // fb_wr
FB->CS[0].CSAR = FB_CSAR_BA(0x6000);
//FLEXBUS_BASE_ADDRESS;
FB->CS[0].CSMR  =  FB_CSMR_BAM(0x0001) | FB_CSMR_V_MASK;
FB->CS[0].CSCR  = FB_CSCR_BLS_MASK   //右对齐，数据出现在FB_AD15~FB_AD0
```

```
        | FB_CSCR_PS(2)                    //16Byte数据接口
        | FB_CSCR_AA_MASK;
    }
```

然后实现基本的画点函数：

```
    void LCD_Draw_Point(u16 x, u16 y,int color) {      //向指定的x、y坐标显示color颜色的点
        LCD_SetWindow(x,y,x,y);
        LCD_WR_REG(LCD_START_REG);
        LCD_WR_DAT(color);
    }
```

基于该函数，又可扩展实现基本的画线、画圆、矩形填充以及 BMP 填充等功能。

应用扩展篇：移植 ucGUI 至 K60。见附件程序。

移植后的视频效果见如下链接：

http://v.youku.com/v_show/id_XNzAwMTE5OTA0.html?firsttime=15

Pixels/sec: 16397220！