LCD driver layers

This color: v1.1 extension
#define LCD_DRVTYPE_V1_1

Application

BSP_LCD... (stm32_adafruit_lcd.c)

```
uint8_t  BSP_LCD_Init(void);
uint32_t BSP_LCD_GetXSize(void);
uint32_t BSP_LCD_GetYSize(void);

uint16_t BSP_LCD_GetTextColor(void);
uint16_t BSP_LCD_GetBackColor(void);
void     BSP_LCD_SetTextColor(__IO uint16_t Color);
void     BSP_LCD_SetBackColor(__IO uint16_t Color);
void     BSP_LCD_SetFont(sFONT *fonts);
sFONT    *BSP_LCD_GetFont(void);

void     BSP_LCD_Clear(uint16_t Color);
void     BSP_LCD_ClearStringLine(uint16_t Line);
void     BSP_LCD_DisplayStringAtLine(uint16_t Line, uint8_t *ptr);
void     BSP_LCD_DisplayStringAt(uint16_t Xpos, uint16_t Ypos, uint8_t *Text, Line_ModeTypdef Mode);
void     BSP_LCD_DisplayChar(uint16_t Xpos, uint16_t Ypos, uint8_t Ascii);

void     BSP_LCD_DrawPixel(uint16_t Xpos, uint16_t Ypos, uint16_t RGB_Code);
void     BSP_LCD_DrawHLine(uint16_t Xpos, uint16_t Ypos, uint16_t Length);
void     BSP_LCD_DrawVLine(uint16_t Xpos, uint16_t Ypos, uint16_t Length);
void     BSP_LCD_DrawLine(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2);
void     BSP_LCD_DrawRect(uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height);
void     BSP_LCD_DrawCircle(uint16_t Xpos, uint16_t Ypos, uint16_t Radius);
void     BSP_LCD_DrawPolygon(pPoint Points, uint16_t PointCount);
void     BSP_LCD_DrawEllipse(int Xpos, int Ypos, int XRadius, int YRadius);
void     BSP_LCD_DrawBitmap(uint16_t Xpos, uint16_t Ypos, uint8_t *pBmp);
void     BSP_LCD_FillRect(uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height);
void     BSP_LCD_FillCircle(uint16_t Xpos, uint16_t Ypos, uint16_t Radius);
void     BSP_LCD_FillPolygon(pPoint Points, uint16_t PointCount);
void     BSP_LCD_FillEllipse(int Xpos, int Ypos, int XRadius, int YRadius);

void     BSP_LCD_DisplayOff(void);
void     BSP_LCD_DisplayOn(void);

uint16_t BSP_LCD_ReadID(void);
uint16_t BSP_LCD_ReadPixel(uint16_t Xpos, uint16_t Ypos);
void     BSP_LCD_DrawRGB16Image(uint16_t Xpos, uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, uint16_t *pData);
void     BSP_LCD_ReadRGB16Image(uint16_t Xpos, uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, uint16_t *pData);
```

```
typedef struct _tFont
{
  const uint8_t *table;
  uint16_t Width;
  uint16_t Height;

} sFONT;
```

```
typedef struct
{
  uint32_t TextColor;
  uint32_t BackColor;
  sFONT    *pFont;

}LCD_DrawPropTypeDef;
```

```
typedef struct
{
  int16_t X;
  int16_t Y;
}Point, * pPoint;
```

```
typedef enum
{
  CENTER_MODE = 0x01,
  RIGHT_MODE  = 0x02,
  LEFT_MODE   = 0x03
}Line_ModeTypdef;
```

BSP_TS... (stm32_adafruit_ts.c)

```
uint8_t BSP_TS_Init(uint16_t XSize, uint16_t YSize);
void    BSP_TS_GetState(TS_StateTypeDef *TsState);
```

```
typedef struct
{
  uint16_t TouchDetected;
  uint16_t X;
  uint16_t Y;
  uint16_t Z;
}TS_StateTypeDef;
```

LCDdriver, TSdriver („ili9325.c" or „st7783.c" or „hx8347g.c" or...)

LCD_DrvTypeDef (from lcd.h), TS_DrvTypeDef (from ts.h), BITMAPSTRUCT (from bmp.h)

```
typedef struct
{
  void     (*Init)(void);
  uint16_t (*ReadID)(void);
  void     (*DisplayOn)(void);
  void     (*DisplayOff)(void);
  void     (*SetCursor)(uint16_t, uint16_t);
  void     (*WritePixel)(uint16_t, uint16_t, uint16_t);
  uint16_t (*ReadPixel)(uint16_t, uint16_t);

   /* Optimized operation */
  void     (*SetDisplayWindow)(uint16_t, uint16_t, uint16_t, uint16_t);
  void     (*DrawHLine)(uint16_t, uint16_t, uint16_t, uint16_t);
  void     (*DrawVLine)(uint16_t, uint16_t, uint16_t, uint16_t);

  uint16_t (*GetLcdPixelWidth)(void);
  uint16_t (*GetLcdPixelHeight)(void);
  void     (*DrawBitmap)(uint16_t, uint16_t, uint8_t*);
  void     (*DrawRGBImage)(uint16_t, uint16_t, uint16_t, uint16_t, uint8_t*);
  void     (*ReadRGBImage)(uint16_t, uint16_t, uint16_t, uint16_t, uint8_t*);
  void     (*FillRect)(uint16_t, uint16_t, uint16_t, uint16_t, uint16_t);
}LCD_DrvTypeDef;

typedef struct
{
  void       (*Init)(uint16_t);
  uint16_t  (*ReadID)(uint16_t);
  void       (*Reset)(uint16_t);
  void       (*Start)(uint16_t);
  uint8_t    (*DetectTouch)(uint16_t);
  void       (*GetXY)(uint16_t, uint16_t*, uint16_t*);
  void       (*EnableIT)(uint16_t);
  void       (*ClearIT)(uint16_t);
  uint8_t    (*GetITStatus)(uint16_t);
  void       (*DisableIT)(uint16_t);
}TS_DrvTypeDef;
```

IO driver („lcdts io gpio8.c" or „lcd io spi.c" or „lcdts io fsmc8.c" or lcd[ts]...)

```
void     LCD_Delay (uint32_t delay);
void     LCD_IO_Init(void);
void     LCD_IO_Bl_OnOff(uint8_t Bl);

void     LCD_IO_WriteCmd8(uint8_t Cmd);
void     LCD_IO_WriteCmd16(uint16_t Cmd);
void     LCD_IO_WriteData8(uint8_t Data);
void     LCD_IO_WriteData16(uint16_t Data);

void     LCD_IO_WriteCmd8DataFill16(uint8_t Cmd, uint16_t Data, uint32_t Size);
void     LCD_IO_WriteCmd8MultipleData8(uint8_t Cmd, uint8_t *pData, uint32_t Size);
void     LCD_IO_WriteCmd8MultipleData16(uint8_t Cmd, uint16_t *pData, uint32_t Size);
void     LCD_IO_WriteCmd16DataFill16(uint16_t Cmd, uint16_t Data, uint32_t Size);
void     LCD_IO_WriteCmd16MultipleData8(uint16_t Cmd, uint8_t *pData, uint32_t Size);
void     LCD_IO_WriteCmd16MultipleData16(uint16_t Cmd, uint16_t *pData, uint32_t Size);

void     LCD_IO_ReadCmd8MultipleData8(uint8_t Cmd, uint8_t *pData, uint32_t Size, uint32_t DummySize);
void     LCD_IO_ReadCmd8MultipleData16(uint8_t Cmd, uint16_t *pData, uint32_t Size, uint32_t DummySize);
void     LCD_IO_ReadCmd8MultipleData24to16(uint8_t Cmd, uint16_t *pData, uint32_t Size, uint32_t DummySize);
void     LCD_IO_ReadCmd16MultipleData8(uint16_t Cmd, uint8_t *pData, uint32_t Size, uint32_t DummySize);
void     LCD_IO_ReadCmd16MultipleData16(uint16_t Cmd, uint16_t *pData, uint32_t Size, uint32_t DummySize);
void     LCD_IO_ReadCmd16MultipleData24to16(uint16_t Cmd, uint16_t *pData, uint32_t Size, uint32_t DummySize);

uint8_t  TS_IO_DetectToch(void);
uint16_t TS_IO_GetX(void);
uint16_t TS_IO_GetY(void);
uint16_t TS_IO_GetZ1(void);
uint16_t TS_IO_GetZ2(void);
```

(only lcdts...)

Hardware

GPIO, SPI, FSMC, LTDC...

Stm32f103 bluepill – st7735 spi setting

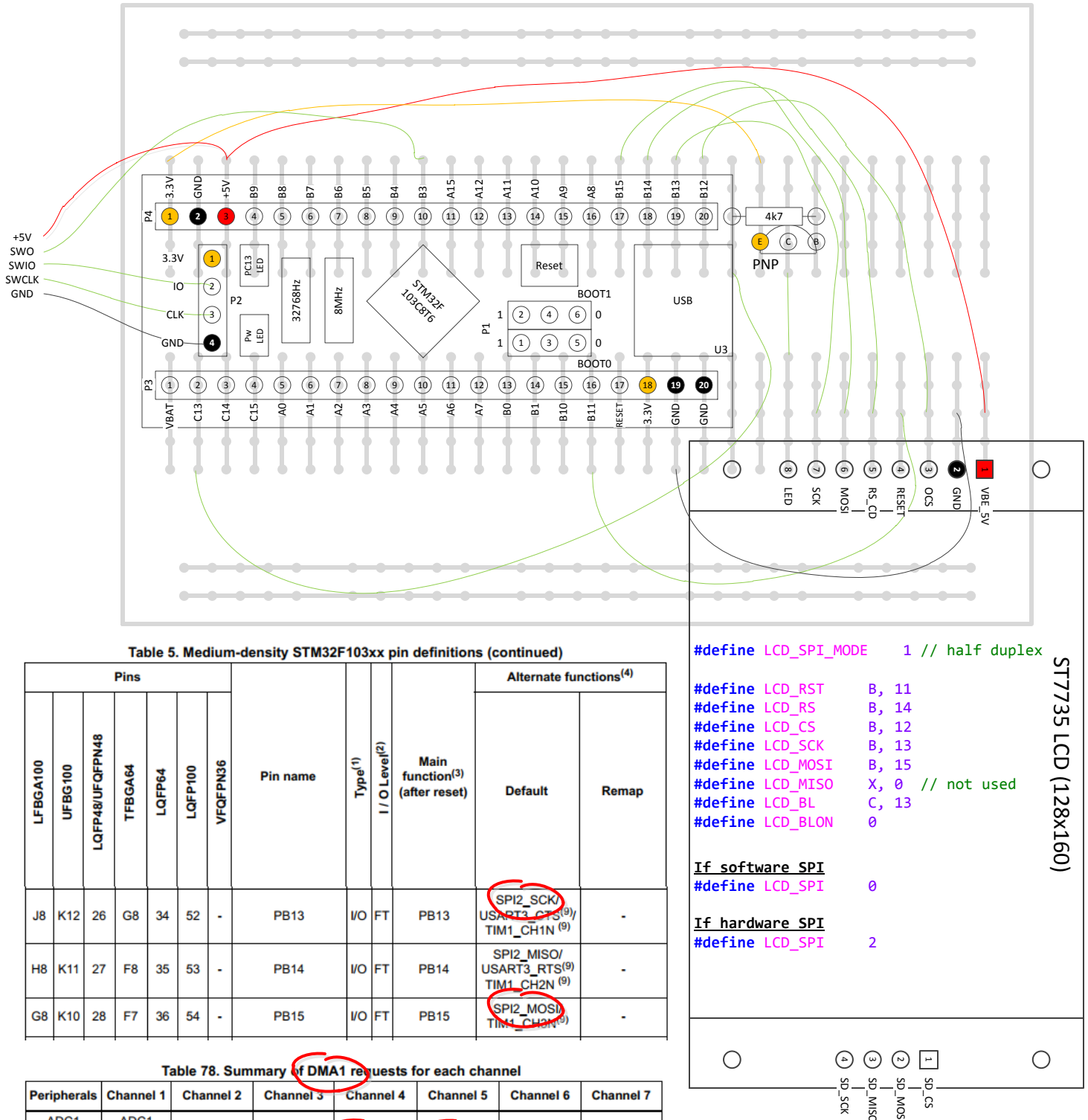


Table 5. Medium-density STM32F103xx pin definitions (continued)

Pins							Pin name	Type(1)	I / O Level(2)	Main function(3) (after reset)	Alternate functions(4)	
LFBGA100	UFBG100	LQFP48/UFQFPN48	TFBGA64	LQFP64	LQFP100	VFQFPN36					Default	Remap
J8	K12	26	G8	34	52	-	PB13	I/O	FT	PB13	SPI2_SCK/ USART3_CTS(9)/ TIM1_CH1N (9)	-
H8	K11	27	F8	35	53	-	PB14	I/O	FT	PB14	SPI2_MISO/ USART3_RTS(9) TIM1_CH2N (9)	-
G8	K10	28	F7	36	54	-	PB15	I/O	FT	PB15	SPI2_MOSI/ TIM1_CH3N(9)	-

Table 78. Summary of DMA1 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1	-	-	-	-	-	-
SPI/I²S	-	SPI1_RX	SPI1_TX	SPI2/I2S2_RX	SPI2/I2S2_TX	-	-
USART	-	USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I²C	-	-	-	I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM3	-	TIM3_CH3	TIM3_CH4 TIM3_UP	-	-	TIM3_CH1 TIM3_TRIG	-
TIM4	TIM4_CH1	-	-	TIM4_CH2	TIM4_CH3	-	TIM4_UP

```
If hardware SPI and DMA on
#define LCD_DMA_TX   1, 5, 0 // DMA number, channel, priority
#define LCD_DMA_RX   1, 4, 2 // DMA number, channel, priority
```

```
#define LCD_SPI_MODE    1 // half duplex

#define LCD_RST     B, 11
#define LCD_RS      B, 14
#define LCD_CS      B, 12
#define LCD_SCK     B, 13
#define LCD_MOSI    B, 15
#define LCD_MISO    X, 0  // not used
#define LCD_BL      C, 13
#define LCD_BLON    0

If software SPI
#define LCD_SPI     0

If hardware SPI
#define LCD_SPI     2
```

```
#define LCD_SCK_EXTRACLK  1 // ST7735: 1 extra clock (when data direction change from write to read)
```

Stm32f303cct – st7735 spi setting

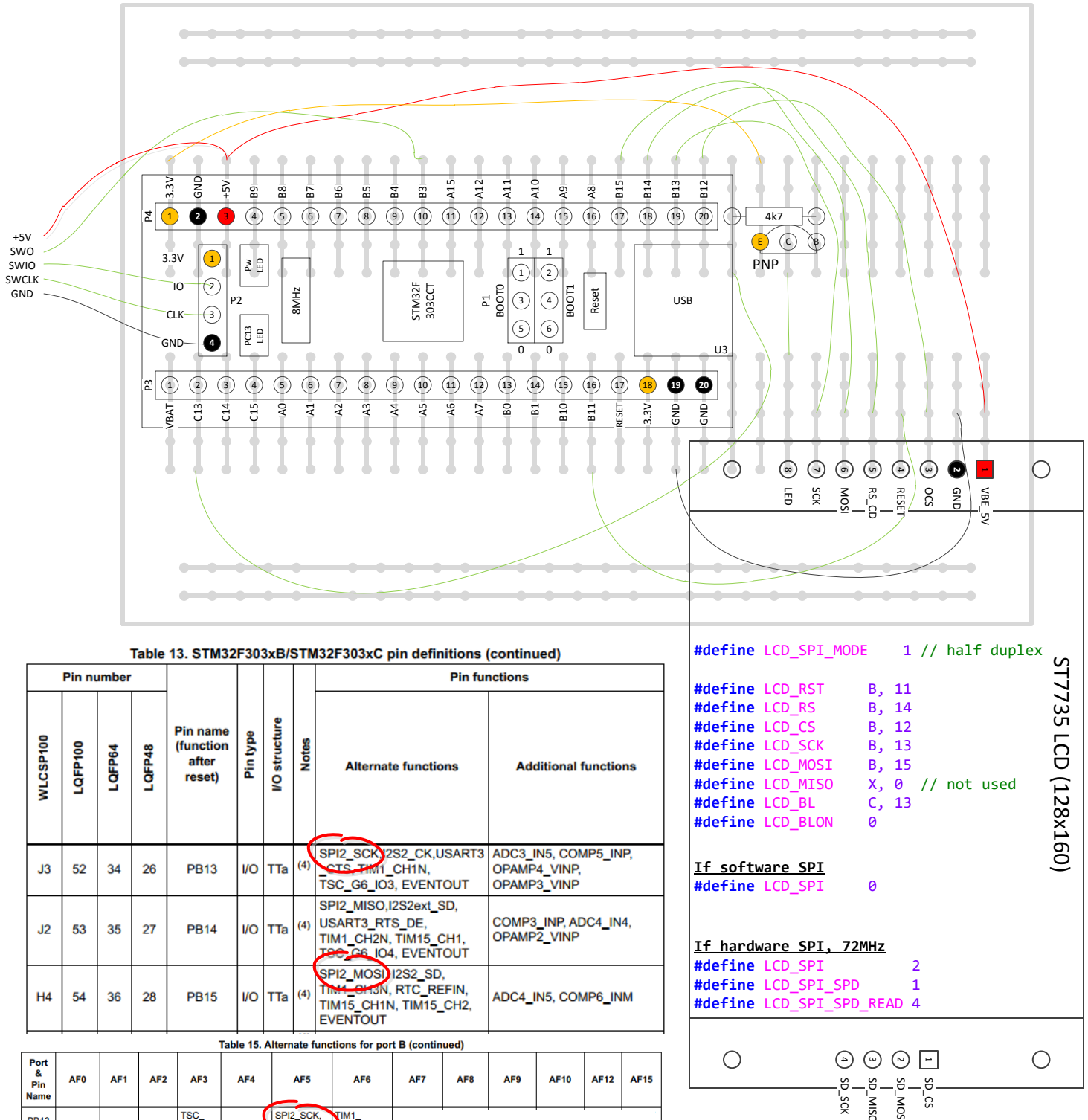


Table 13. STM32F303xB/STM32F303xC pin definitions (continued)

Pin number				Pin name (function after reset)	Pin type	I/O structure	Notes	Pin functions	
WLCSP100	LQFP100	LQFP64	LQFP48					Alternate functions	Additional functions
J3	52	34	26	PB13	I/O	TTa	(4)	SPI2_SCK,I2S2_CK,USART3_CTS, TIM1_CH1N, TSC_G6_IO3, EVENTOUT	ADC3_IN5, COMP5_INP, OPAMP4_VINP, OPAMP3_VINP
J2	53	35	27	PB14	I/O	TTa	(4)	SPI2_MISO,I2S2ext_SD, USART3_RTS_DE, TIM1_CH2N, TIM15_CH1, TSC_G6_IO4, EVENTOUT	COMP3_INP, ADC4_IN4, OPAMP2_VINP
H4	54	36	28	PB15	I/O	TTa	(4)	SPI2_MOSI,I2S2_SD, TIM1_CH3N, RTC_REFIN, TIM15_CH1N, TIM15_CH2, EVENTOUT	ADC4_IN5, COMP6_INM

Table 15. Alternate functions for port B (continued)

Port & Pin Name	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8	AF9	AF10	AF12	AF15
PB13	-	-	-	TSC_G6_IO3	-	SPI2_SCK, I2S2_CK	TIM1_CH1N						
PB14	-	TIM15_CH1	-	TSC_G6_IO4	-	SPI2_MISO, I2S2ext_SD	TIM1_CH2N						
PB15	RTC_REFIN	TIM15_CH2	TIM15_CH1N	-	TIM1_CH3N	SPI2_MOSI, I2S2_SD	-						

Table 78. STM32F303xB/C/D/E, STM32F358xC and STM32F398xE summary of DMA1 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel6	Channel7
ADC	ADC1	-	-			-	-
SPI	-	SPI1_RX	SP1_TX	SPI2_RX	SPI2_TX	-	-
USART	-	USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I2C	I2C3_TX(1)	I2C3_RX(1)	-	I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX

```
#define LCD_SPI_MODE    1 // half duplex

#define LCD_RST     B, 11
#define LCD_RS      B, 14
#define LCD_CS      B, 12
#define LCD_SCK     B, 13
#define LCD_MOSI    B, 15
#define LCD_MISO    X, 0  // not used
#define LCD_BL      C, 13
#define LCD_BLON    0

If software SPI
#define LCD_SPI     0


If hardware SPI, 72MHz
#define LCD_SPI           2
#define LCD_SPI_SPD       1
#define LCD_SPI_SPD_READ  4
```

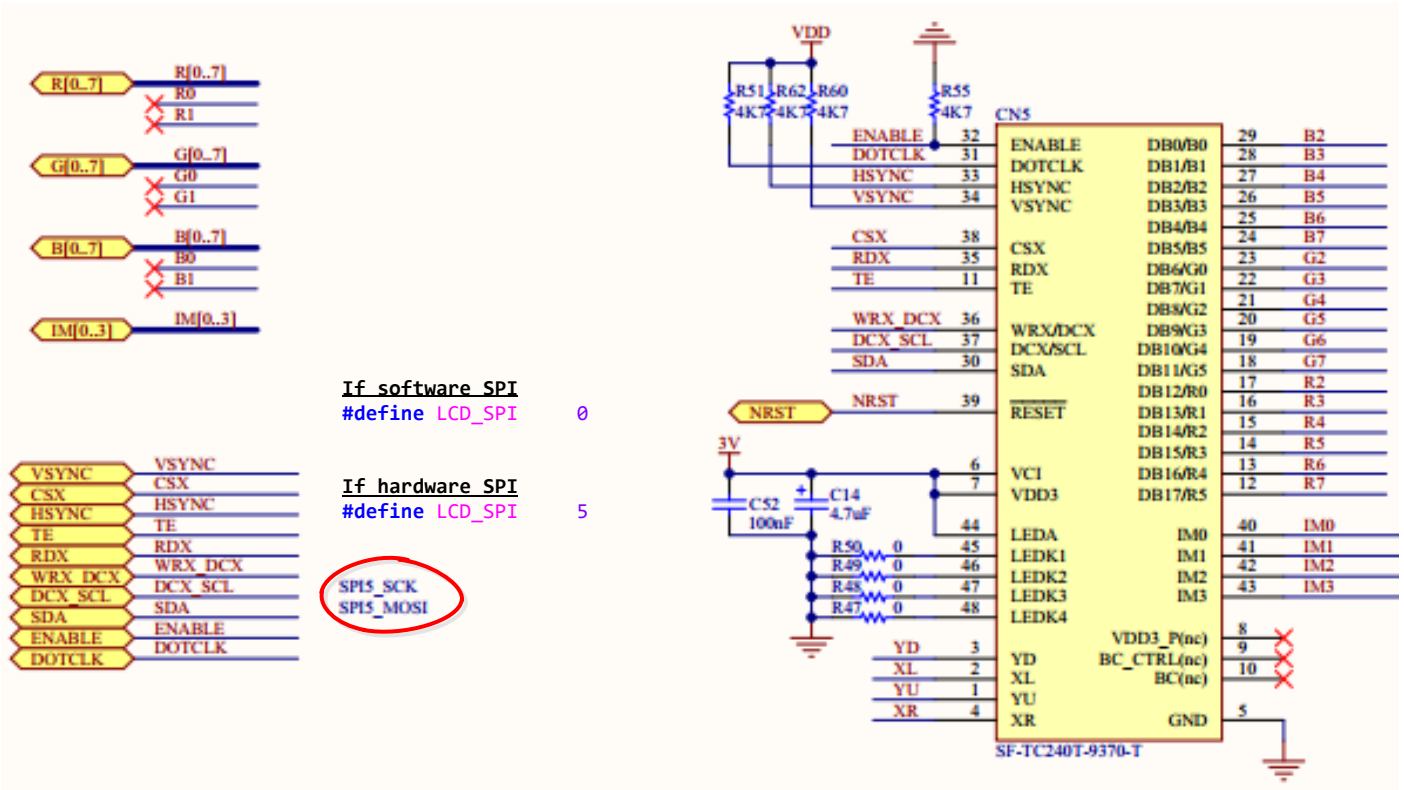
```
If hardware SPI 2
#define LCD_SPI_AFR     5
```

```
If hardware SPI and DMA on
#define LCD_DMA_TX  1, 5, 0 // DMA number, channel, priority
#define LCD_DMA_RX  1, 4, 2 // DMA number, channel, priority


#define LCD_SCK_EXTRACLK  1 // ST7735: 1 extra clock (when data direction change from write to read)
```

Stm32f429 discovery - ili9341 spi setting

STM32 pin		Board function																	
Main function	LQFP144	System	VCP	SDRAM	LCD-TFT	LCD-RGB	LCD-SPI	L3GD20	USB	LED	Push-button	I²C Ext	Touch panel	Free I/O	Power supply	CN2	CN3	CN6	P1	P2
BOOT0	138	BOOT0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	21	-
NRST	25	NRST	-	-	RESET	RESET	RESET	-	-	-	B2	-	-	-	-	5	-	-	-	12
PA0	34	-	-	-	-	-	-	-	-	-	B1	-	-	-	-	-	-	-	-	18
PC2	28	-	-	-	CSX	CSX	CSX	-	-											6
PD13	82	-	-		WRX	-	DCX	-	-											7
PF7	19	-	-	-	DCX	-	SCL	SCK	-											
PF8	20	-	-	-	-	-	-	MISO	-	-	-	-	-	-	-	-	-	-	-	5
PF9	21	-	-	-	SDA	-	SDI/SDO	MOSI	-	-	-	-	-	-	-	-	-	-	-	8

```
#define LCD_SPI_MODE    1 // half duplex

#define LCD_RST     X, 0  // not used
#define LCD_RS      D, 13
#define LCD_CS      C, 2
#define LCD_SCK     F, 7
#define LCD_MOSI    F, 9
#define LCD_MISO    X, 0  // not used (half duplex)
#define LCD_BL      X, 0  // not used
#define LCD_BLON    0
```



Stm32f429 discovery - ili9341 spi setting

Table 12. STM32F427xx and STM32F429xx alternate function mapping (continued)

Port	AF0 SYS	AF1 TIM1/2	AF2 TIM3/4/5	AF3 TIM8/9/ 10/11	AF4 I2C1/ 2/3	AF5 SPI1/2/ 3/4/5/6	AF6 SPI2/3/ SAI1	AF7 SPI3/ USART1/ 2/3	AF8 USART6/ UART4/5/7 /8	AF9 CAN1/2/ TIM12/13/14 /LCD	AF10 OTG2_HS /OTG1_ FS	AF11 ETH	AF12 FMC/SDIO /OTG2_FS	AF13 DCMI	AF14 LCD	AF15 SYS
PF7	-	-	-	TIM11_ CH1	-	SPI5_ SCK	SAI1_ MCLK_B	-	UART7_Tx	-	-	-	FMC_ NREG	-	-	EVEN TOUT
PF8	-	-	-	-	-	SPI5_ MISO	SAI1_ SCK_B						FMC_ NIOWR	-	-	EVEN TOUT
PF9	-	-	-	-	-	SPI5_ MOSI	SAI1_ FS_B						FMC_CD	-	-	EVEN TOUT

```
If hardware SPI 5
#define LCD_SPI_AFR     5
```

Table 43. DMA2 request mapping

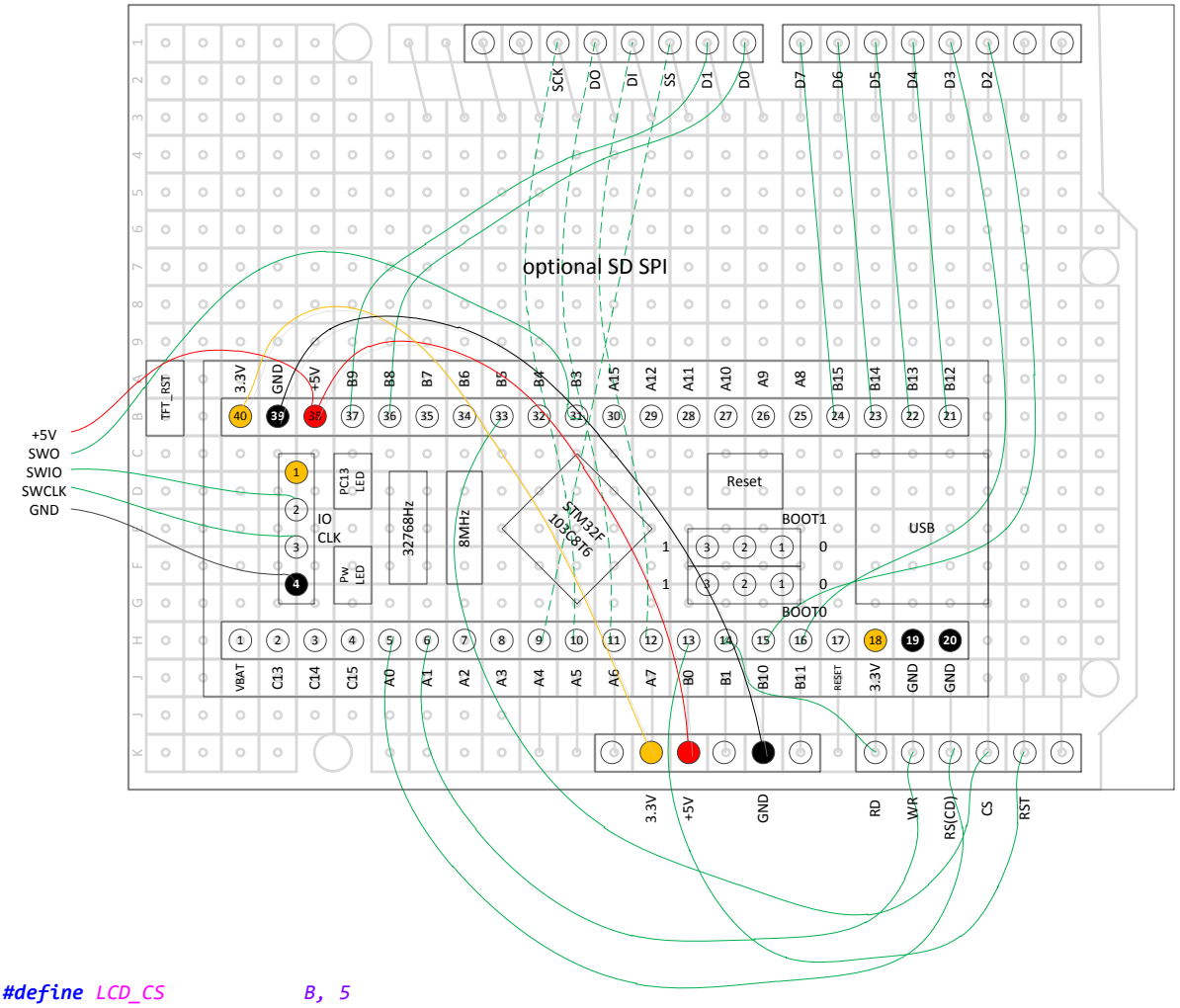
Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	ADC1	SAI1_A(1)	TIM8_CH1 TIM8_CH2 TIM8_CH3	SAI1_A(1)	ADC1	SAI1_B(1)	TIM1_CH1 TIM1_CH2 TIM1_CH3	
Channel 1		DCMI	ADC2	ADC2	SAI1_B(1)	SPI6_TX(1)	SPI6_RX(1)	DCMI
Channel 2	ADC3	ADC3		SPI5_RX(1)	SPI5_TX(1)	CRYP_OUT	CRYP_IN	HASH_IN
Channel 3	SPI1_RX		SPI1_RX	SPI1_TX		SPI1_TX		
Channel 4	SPI4_RX(1)	SPI4_TX(1)	USART1_RX	SDIO		USART1_RX	SDIO	USART1_TX
Channel 5								
Channel 6								
Channel 7		TIM8_UP	TIM8_CH1	TIM8_CH2	TIM8_CH3	SPI5_RX(1)	SPI5_TX(1)	TIM8_CH4 TIM8_TRIG TIM8_COM

```
If hardware SPI and DMA on
#define LCD_DMA_TX   2, 2, 4, 0 // DMA number, channel, stream, priority
#define LCD_DMA_RX   2, 2, 3, 2 // DMA number, channel, stream, priority
```

1. These requests are available on STM32F42xxx and STM32F43xxx.

```
#define LCD_SCK_EXTRACLK   0 // ILI9341: 0 extra clock (when data direction change from write to read)
```

Stm32f103c8t bluepill gpio 8bit setting

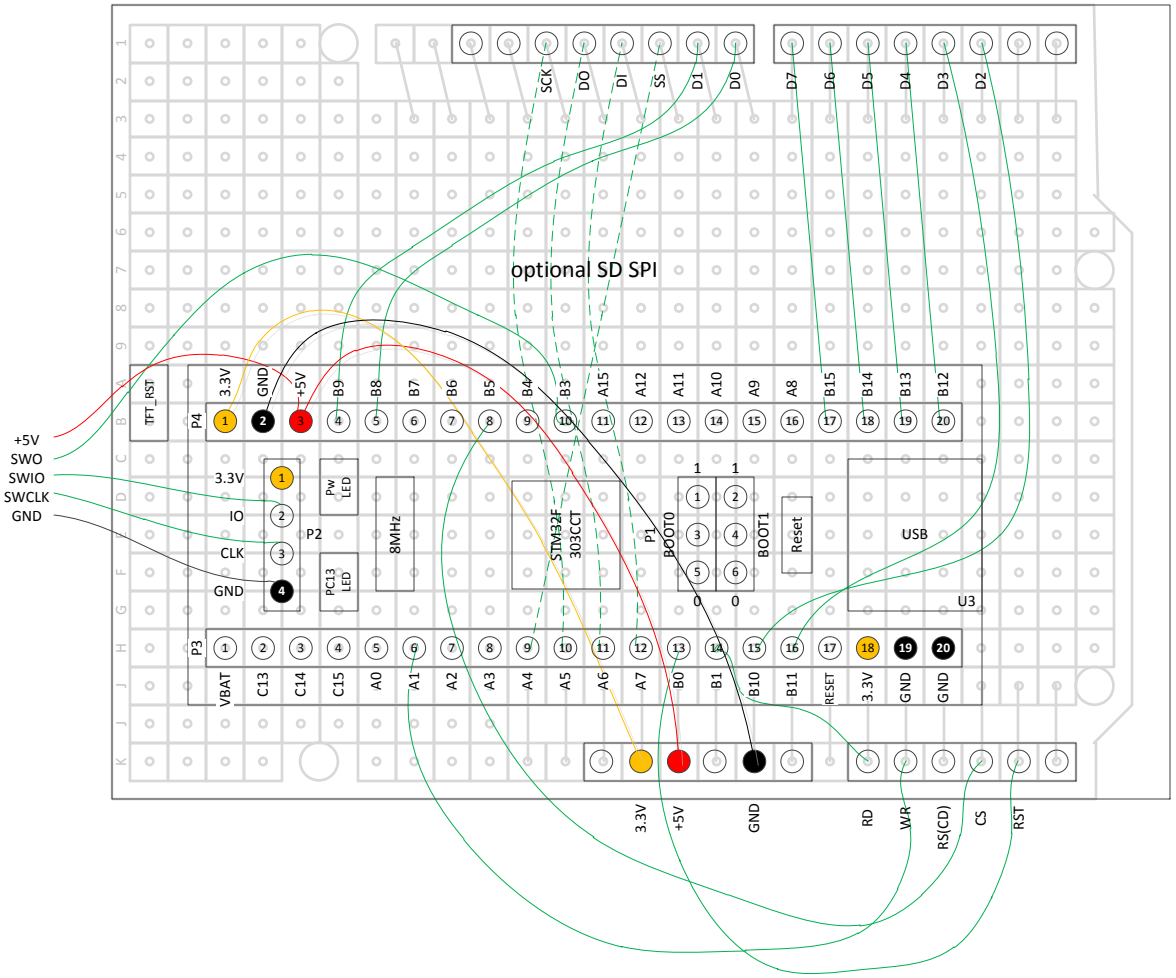



```
#define LCD_CS          B, 5
#define LCD_RS          A, 0 // TS_XM
#define LCD_WR          A, 1 // TS_YP
#define LCD_RD          B, 1
#define LCD_RST         B, 0

#define LCD_D0          B, 8
#define LCD_D1          B, 9
#define LCD_D2          B, 10
#define LCD_D3          B, 11
#define LCD_D4          B, 12
#define LCD_D5          B, 13
#define LCD_D6          B, 14 // TS_XP
#define LCD_D7          B, 15 // TS_YM

#define LCD_BL          X, 0  // not used
#define LCD_BLON        0

If analog touchscreen:
#define TS_ADC          1  // ADC1, ADC2
#define TS_XM_ADCCH     0  // ADC12_IN0
#define TS_YP_ADCCH     1  // ADC12_IN1

#define LCD_WRITE_DELAY   0  // (72MHz)
#define LCD_READ_DELAY    1  // (72MHz)
#define TS_AD_DELAY     500
```

Table 5. Medium-density STM32F103xx pin definitions (continued)

Pins							Pin name	Type(1)	I / O Level(2)	Main function(3) (after reset)	Alternate functions(4)	
LFBGA100	UFBG100	LQFP48/UFQFPN48	TFBGA64	LQFP64	LQFP100	VFQFPN36					Default	Remap
G2	L2	10	G2	14	23	7	PA0-WKUP	I/O	-	PA0	WKUP/ USART2_CTS(9)/ ADC12_IN0/ TIM2_CH1_ ETR(9)	-
H2	M2	11	H2	15	24	8	PA1	I/O	-	PA1	USART2_RTS(9)/ ADC12_IN1/ TIM2_CH2(9)	-

Stm32f303cct gpio 8bit setting

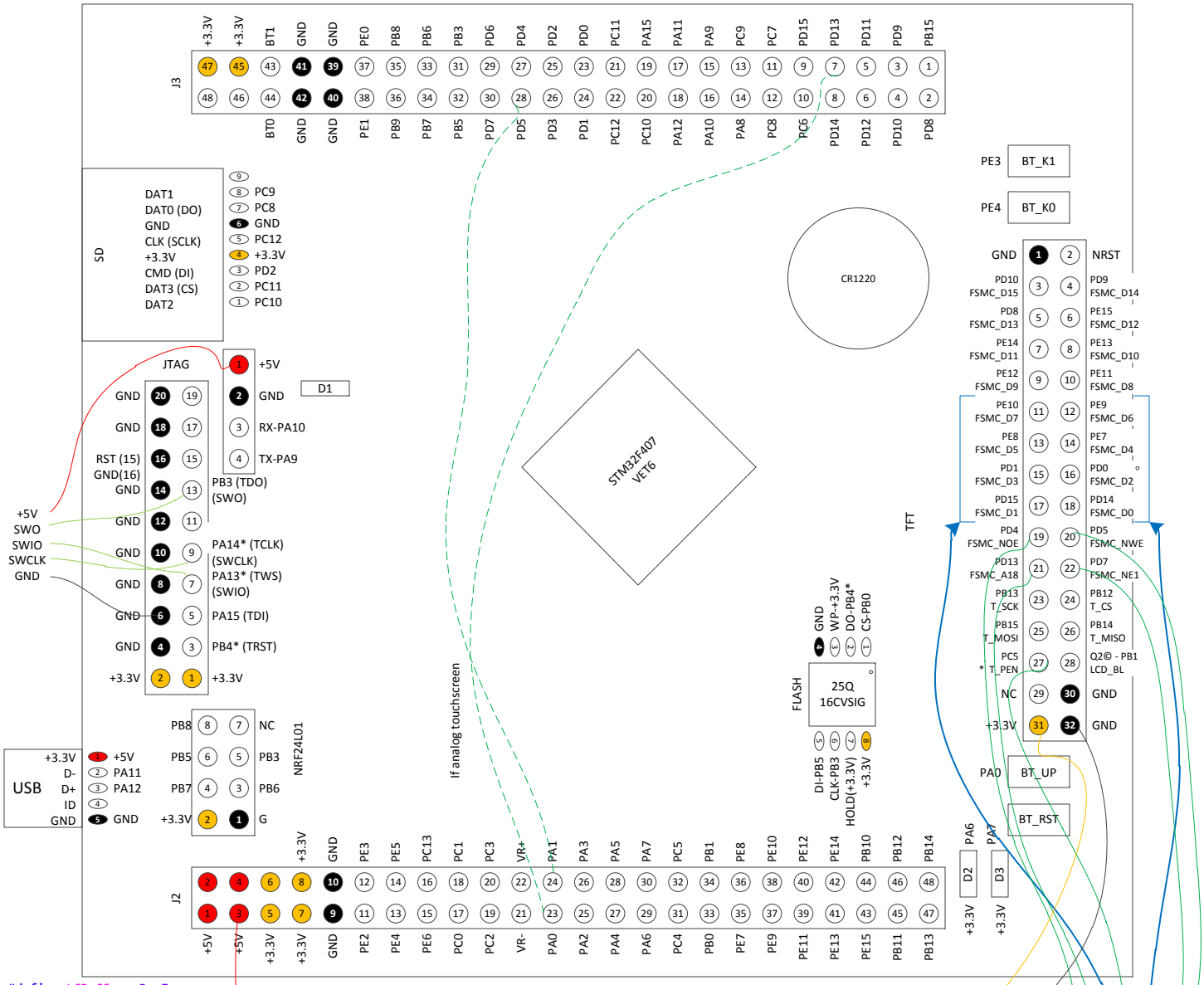



```
#define LCD_CS            B, 5
#define LCD_RS            A, 0 // TS_XM
#define LCD_WR            A, 1 // TS_YP
#define LCD_RD            B, 1
#define LCD_RST           B, 0

#define LCD_D0            B, 8
#define LCD_D1            B, 9
#define LCD_D2            B, 10
#define LCD_D3            B, 11
#define LCD_D4            B, 12
#define LCD_D5            B, 13
#define LCD_D6            B, 14 // TS_XP
#define LCD_D7            B, 15 // TS_YM

#define LCD_BL            X, 0  // not used
#define LCD_BLON          0

If analog touchscreen:
#define TS_ADC            1   // ADC1
#define TS_XM_ADCCH       1   // ADC1_IN1
#define TS_YP_ADCCH       2   // ADC1_IN2

#define LCD_WRITE_DELAY   1   // (72MHz)
#define LCD_READ_DELAY    2   // (72MHz)
#define TS_AD_DELAY     500
```

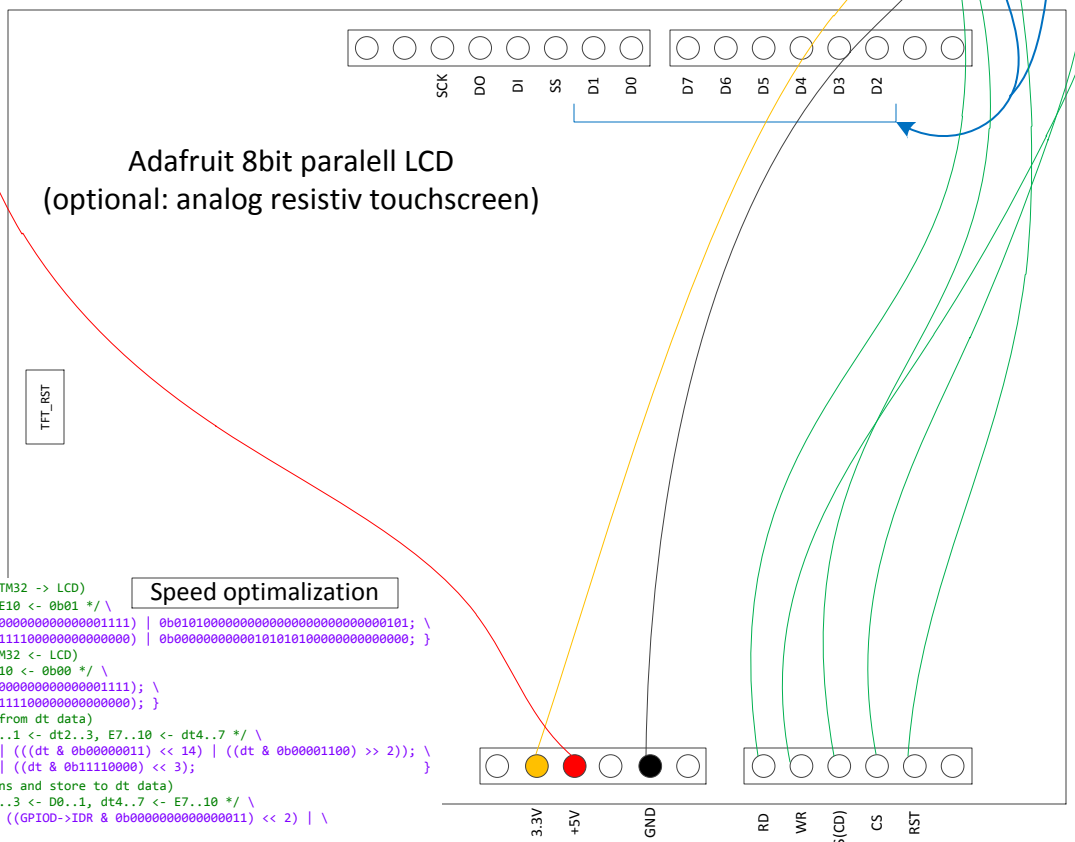
Table 13. STM32F303xB/STM32F303xC pin definitions (continued)

Pin number				Pin name (function after reset)	Pin type	I/O structure	Notes	Pin functions	
WLCSP100	LQFP100	LQFP64	LQFP48					Alternate functions	Additional functions
H9	23	14	10	PA0	I/O	TTa	(4)	USART2_CTS, TIM2_CH1_ETR,TIM8_BKIN, TIM8_ETR,TSC_G1_IO1, COMP1_OUT, EVENTOUT	ADC1_IN1, COMP1_INM, RTC_TAMP2, WKUP1, COMP7_INP
J9	24	15	11	PA1	I/O	TTa	(4)	USART2_RTS_DE, TIM2_CH2, TSC_G1_IO2, TIM15_CH1N, RTC_REFIN, EVENTOUT	ADC1_IN2, COMP1_INP, OPAMP1_VINP, OPAMP3_VINP

Stm32f407vet gpio 8bit setting



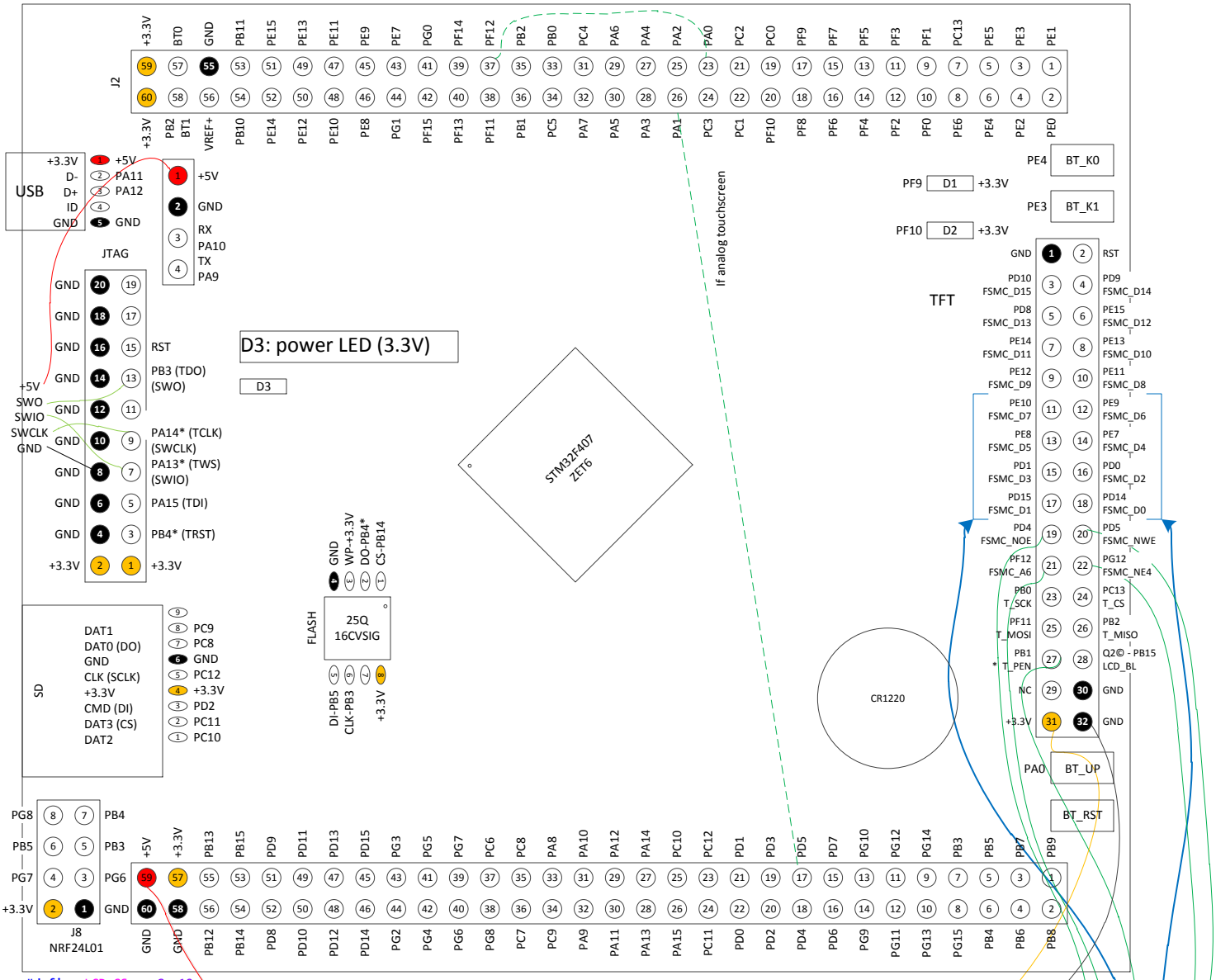

```
#define LCD_CS    D, 7
#define LCD_RS    D, 13
#define LCD_WR    D, 5
#define LCD_RD    D, 4
#define LCD_RST   C, 5

#define LCD_D0    D, 14
#define LCD_D1    D, 15
#define LCD_D2    D, 0
#define LCD_D3    D, 1
#define LCD_D4    E, 7
#define LCD_D5    E, 8
#define LCD_D6    E, 9
#define LCD_D7    E, 10

#define LCD_BL    B, 1
#define LCD_BLON  0

#define LCD_WRITE_DELAY 1
#define LCD_READ_DELAY  2

If analog touchscreen:
#define TS_XM_AN  A, 0
#define TS_YP_AN  A, 1

#define TS_XM_ADCCH  0
#define TS_YP_ADCCH  1

#define TS_AD_DELAY  500
```



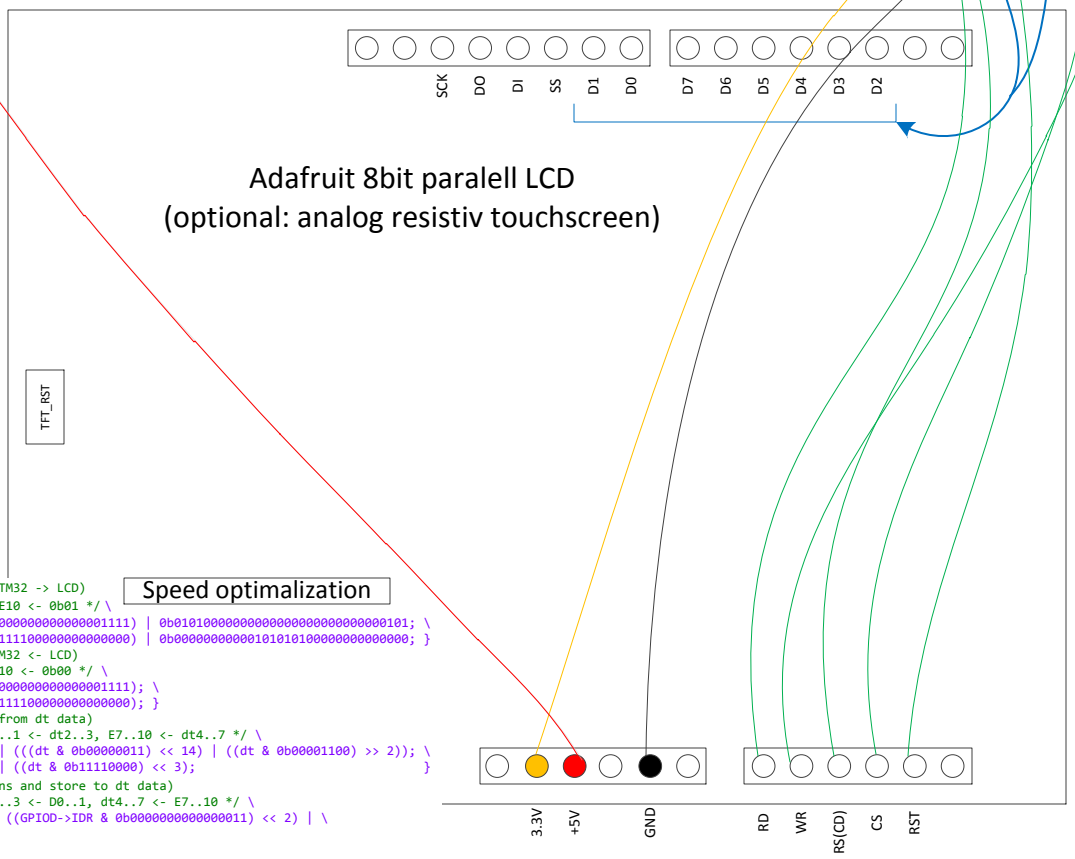
Speed optimalization

```
// datapins setting to output (data direction: STM32 -> LCD)
#define LCD_DIRWRITE { /* D0..D1, D14..D15, E7..E10 <- 0b01 */ \
GPIOD->MODER = (GPIOD->MODER & ~0b11110000000000000000000000001111) | 0b01010000000000000000000000000101; \
GPIOE->MODER = (GPIOE->MODER & ~0b00000000000011111111000000000000) | 0b00000000000001010101000000000000; }
// datapins setting to input (data direction: STM32 <- LCD)
#define LCD_DIRREAD { /* D0..D1, D14..D15, E7..E10 <- 0b00 */ \
GPIOD->MODER = (GPIOD->MODER & ~0b11110000000000000000000000001111); \
GPIOE->MODER = (GPIOE->MODER & ~0b00000000000011111111000000000000); }
// datapins write, STM32 -> LCD (write I/O pins from dt data)
#define LCD_WRITE(dt) { /* D14..15 <- dt0..1, D0..1 <- dt2..3, E7..10 <- dt4..7 */ \
GPIOD->ODR = (GPIOD->ODR & ~0b1100000000000011) | ((((dt & 0b00000011) << 14) | ((dt & 0b00001100) >> 2)); \
GPIOE->ODR = (GPIOE->ODR & ~0b0000011110000000) | ((dt & 0b11110000) << 3);                                 }
// datapins read, STM32 <- LCD (read from I/O pins and store to dt data)
#define LCD_READ(dt) { /* dt0..1 <- D14..15, dt2..3 <- D0..1, dt4..7 <- E7..10 */ \
dt = ((GPIOD->IDR & 0b1100000000000000) >> 14) | ((GPIOD->IDR & 0b0000000000000011) << 2) | \
     ((GPIOE->IDR & 0b0000011110000000) >> 3); }
```

Stm32f407zet gpio 8bit setting



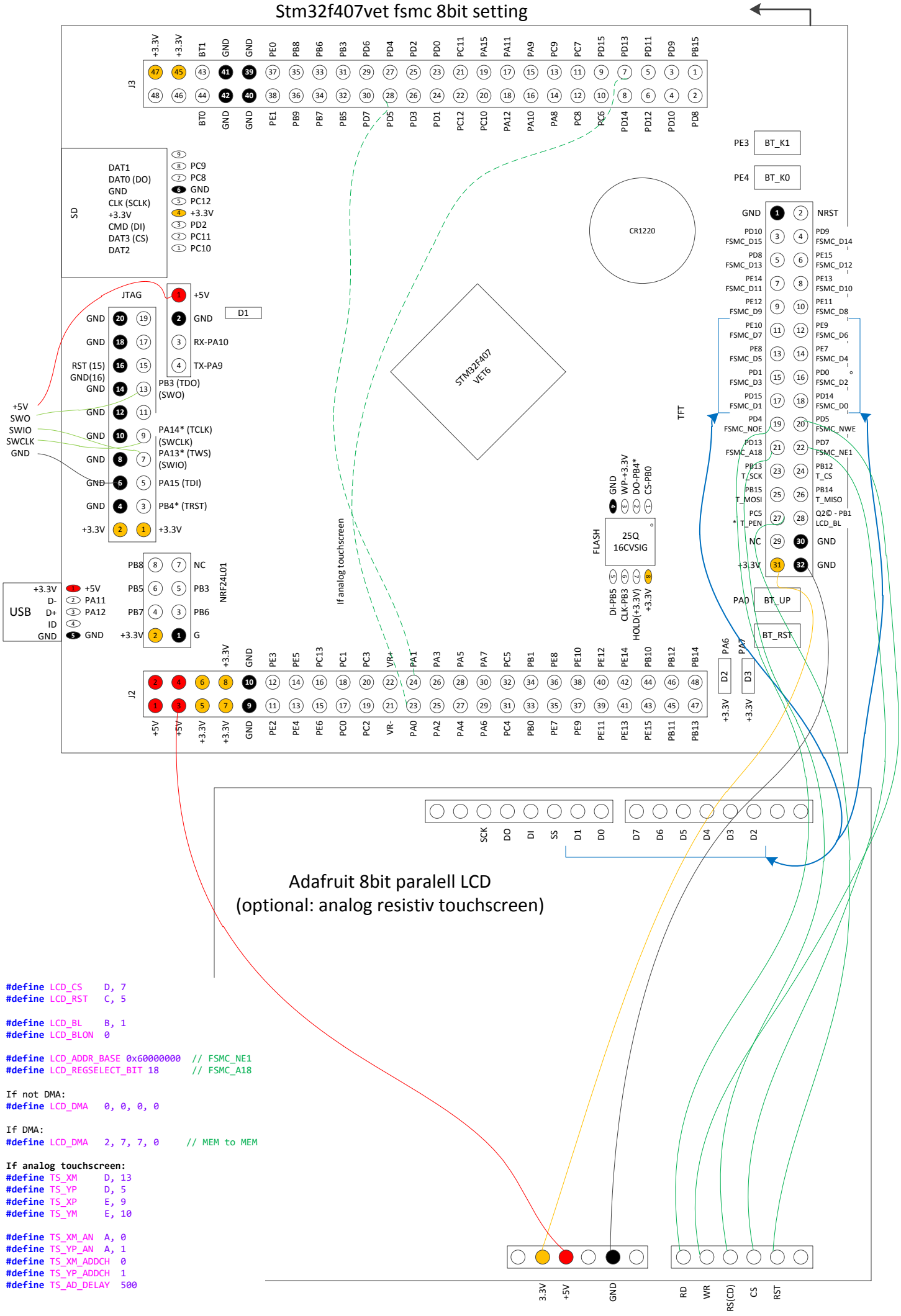

```
#define LCD_CS    G, 12
#define LCD_RS    F, 12
#define LCD_WR    D, 5
#define LCD_RD    D, 4
#define LCD_RST   B, 1

#define LCD_D0    D, 14
#define LCD_D1    D, 15
#define LCD_D2    D, 0
#define LCD_D3    D, 1
#define LCD_D4    E, 7
#define LCD_D5    E, 8
#define LCD_D6    E, 9
#define LCD_D7    E, 10

#define LCD_BL    B, 15
#define LCD_BLON  0

#define LCD_WRITE_DELAY 1
#define LCD_READ_DELAY  2

If analog touchscreen:
#define TS_XM_AN  A, 0
#define TS_YP_AN  A, 1

#define TS_XM_ADCCH  0
#define TS_YP_ADCCH  1

#define TS_AD_DELAY  500
```



Speed optimalization

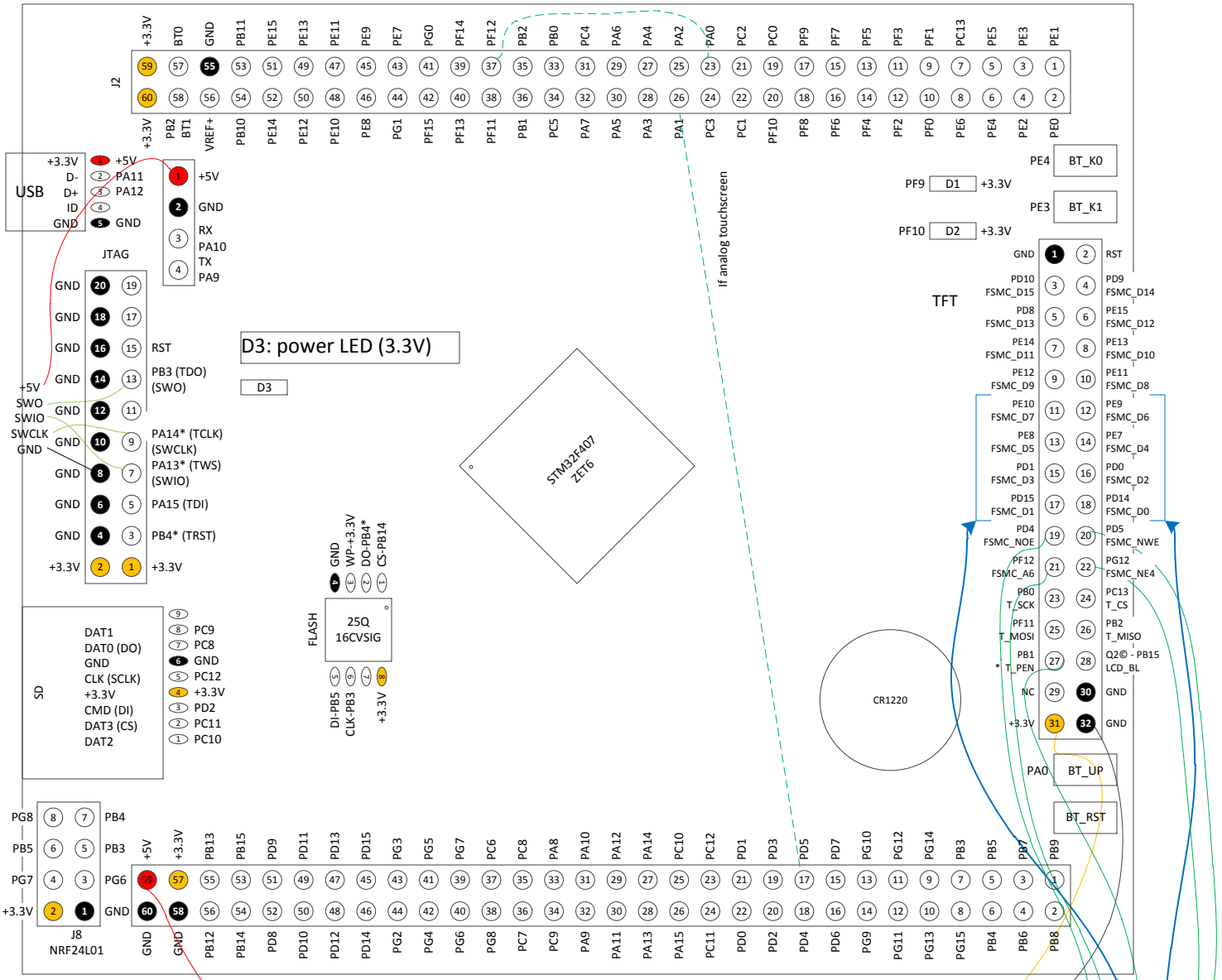
```
// datapins setting to output (data direction: STM32 -> LCD)
#define LCD_DIRWRITE { /* D0..D1, D14..D15, E7..E10 <- 0b01 */ \
GPIOD->MODER = (GPIOD->MODER & ~0b11110000000000000000000000001111) | 0b01010000000000000000000000000101; \
GPIOE->MODER = (GPIOE->MODER & ~0b00000000000011111111000000000000) | 0b00000000000001010101000000000000; }
// datapins setting to input (data direction: STM32 <- LCD)
#define LCD_DIRREAD { /* D0..D1, D14..D15, E7..E10 <- 0b00 */ \
GPIOD->MODER = (GPIOD->MODER & ~0b11110000000000000000000000001111); \
GPIOE->MODER = (GPIOE->MODER & ~0b00000000000011111111000000000000); }
// datapins write, STM32 -> LCD (write I/O pins from dt data)
#define LCD_WRITE(dt) { /* D14..15 <- dt0..1, D0..1 <- dt2..3, E7..10 <- dt4..7 */ \
GPIOD->ODR = (GPIOD->ODR & ~0b1100000000000011) | (((dt & 0b00000011) << 14) | ((dt & 0b00001100) >> 2)); \
GPIOE->ODR = (GPIOE->ODR & ~0b0000011110000000) | ((dt & 0b11110000) << 3);                               }
// datapins read, STM32 <- LCD (read from I/O pins and store to dt data)
#define LCD_READ(dt) { /* dt0..1 <- D14..15, dt2..3 <- D0..1, dt4..7 <- E7..10 */ \
dt = ((GPIOD->IDR & 0b1100000000000000) >> 14) | ((GPIOD->IDR & 0b0000000000000011) << 2) | \
     ((GPIOE->IDR & 0b0000011110000000) >> 3); }
```

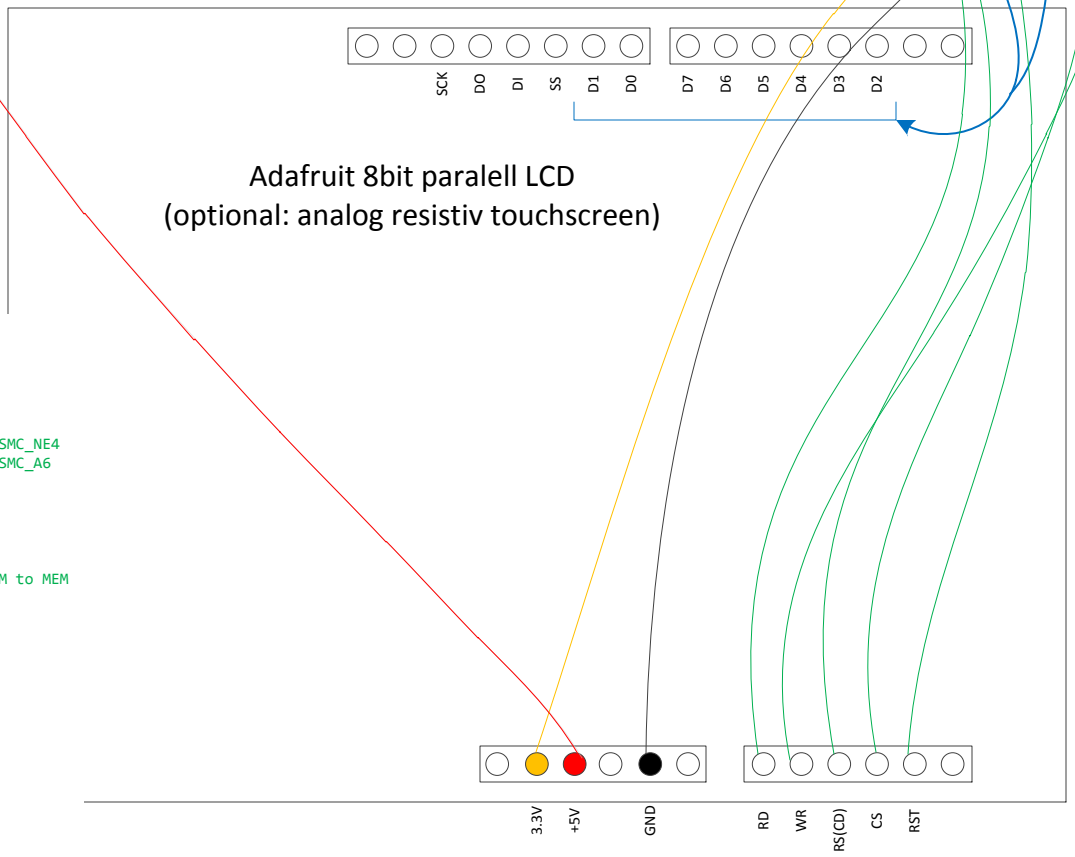
Stm32f407vet fsmc 8bit setting




```
#define LCD_CS    D, 7
#define LCD_RST   C, 5

#define LCD_BL    B, 1
#define LCD_BLON  0

#define LCD_ADDR_BASE 0x60000000  // FSMC_NE1
#define LCD_REGSELECT_BIT 18      // FSMC_A18

If not DMA:
#define LCD_DMA   0, 0, 0, 0

If DMA:
#define LCD_DMA   2, 7, 7, 0     // MEM to MEM

If analog touchscreen:
#define TS_XM     D, 13
#define TS_YP     D, 5
#define TS_XP     E, 9
#define TS_YM     E, 10

#define TS_XM_AN  A, 0
#define TS_YP_AN  A, 1
#define TS_XM_ADDCH  0
#define TS_YP_ADDCH  1
#define TS_AD_DELAY  500
```

Stm32f407zet fsmc 8bit setting

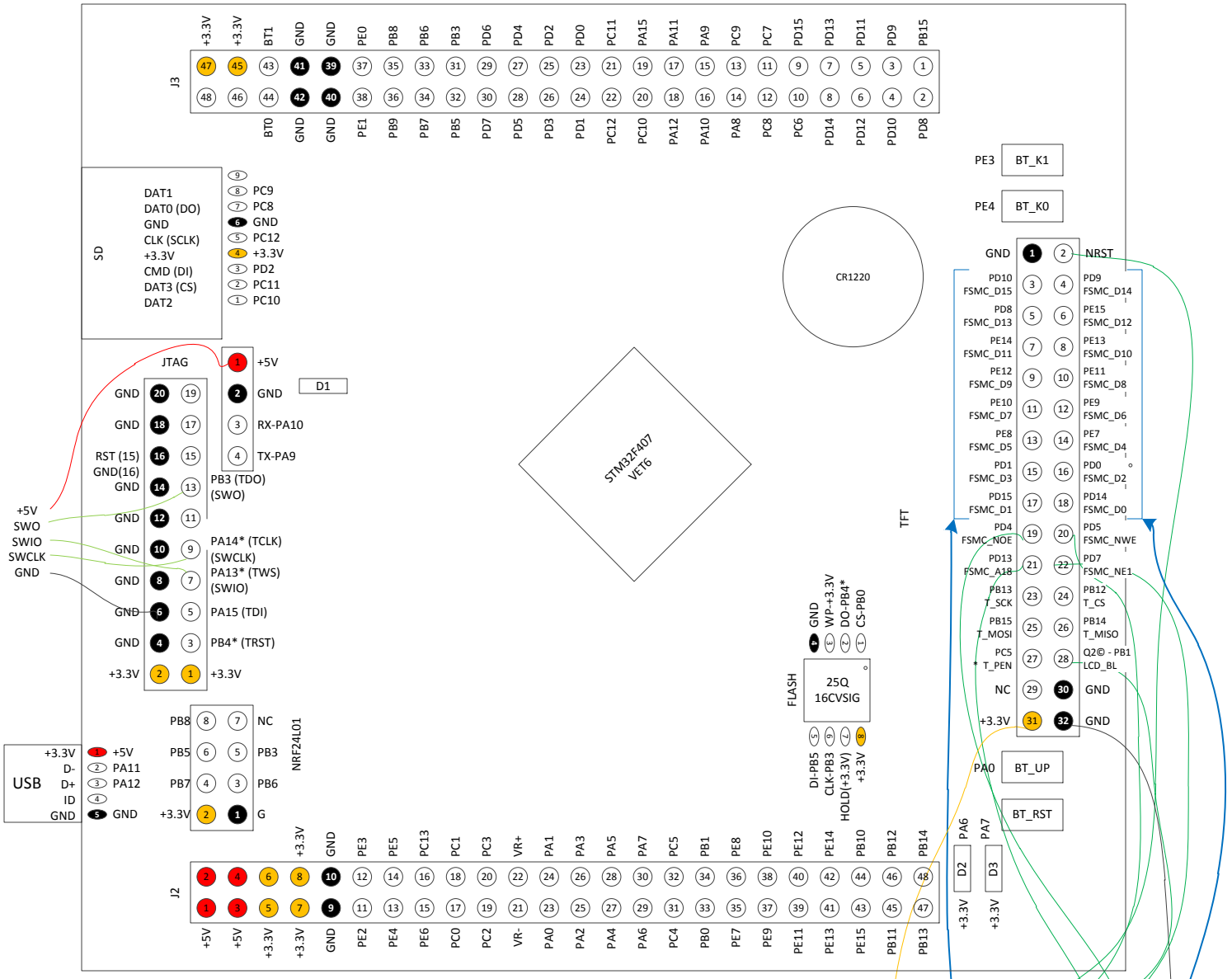




Adafruit 8bit paralell LCD
(optional: analog resistiv touchscreen)

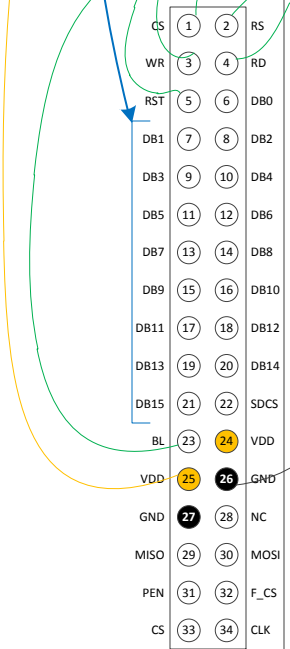
```
#define LCD_CS    G, 12
#define LCD_RST   B, 1

#define LCD_BL    B, 15
#define LCD_BLON  0

#define LCD_ADDR_BASE 0x6C000000 // FSMC_NE4
#define LCD_REGSELECT_BIT 6      // FSMC_A6

If not DMA:
#define LCD_DMA   0, 0, 0, 0

If DMA:
#define LCD_DMA   2, 7, 7, 0    // MEM to MEM

If analog touchscreen:
#define TS_XM     F, 12
#define TS_YP     D, 5
#define TS_XP     E, 9
#define TS_YM     E, 10

#define TS_XM_AN  A, 0
#define TS_YP_AN  A, 1
#define TS_XM_ADDCH  0
#define TS_YP_ADDCH  1
#define TS_AD_DELAY  500
```

Stm32f407vet gpio 16bit setting



ILI9341 (16bit)

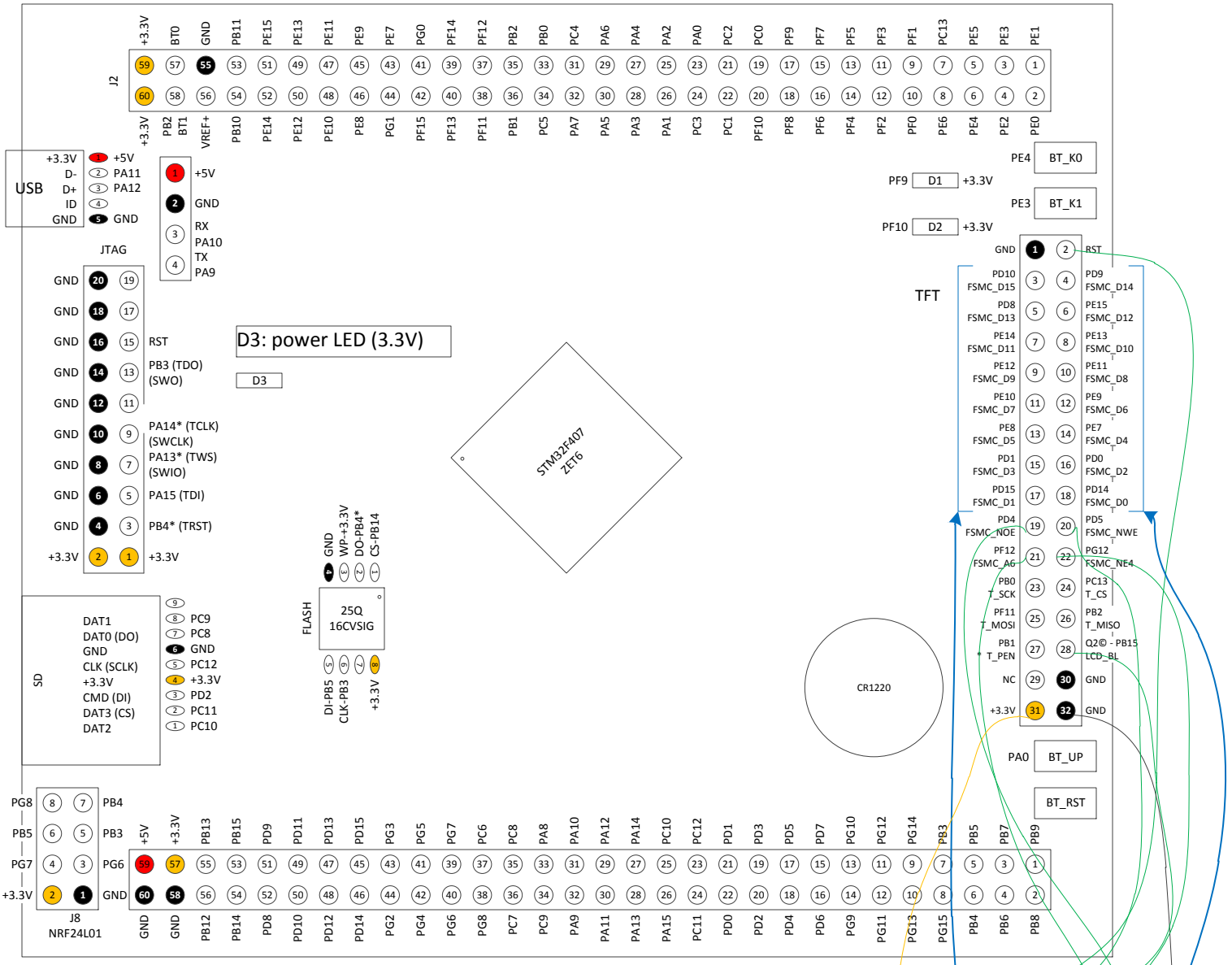
```
#define LCD_D0          D, 14
#define LCD_D1          D, 15
#define LCD_D2          D, 0
#define LCD_D3          D, 1
#define LCD_D4          E, 7
#define LCD_D5          E, 8
#define LCD_D6          E, 9
#define LCD_D7          E, 10
#define LCD_D8          E, 11
#define LCD_D9          E, 12
#define LCD_D10         E, 13
#define LCD_D11         E, 14
#define LCD_D12         E, 15
#define LCD_D13         D, 8
#define LCD_D14         D, 9
#define LCD_D15         D, 10

#define LCD_CS          D, 7
#define LCD_RS          D, 13
#define LCD_WR          D, 5
#define LCD_RD          D, 4
#define LCD_RST         X, 0

#define LCD_BL          B, 1
#define LCD_BLON        0

#define LCD_WRITE_DELAY 0
#define LCD_READ_DELAY  1
```



Speed optimalization

```
// datapins setting to output (data direction: STM32 -> LCD)
#define LCD_DIRWRITE { /* D0..D1, D8..D10, D14..D15, E7..E15 <- 0b01 */ \
GPIOD->MODER = (GPIOD->MODER & ~0b11110000001111110000000000001111) | 0b01010000000101010000000000000101; \
GPIOE->MODER = (GPIOE->MODER & ~0b11111111111111111100000000000000) | 0b01010101010101010100000000000000; }
// datapins setting to input (data direction: STM32 <- LCD)
#define LCD_DIRREAD  { /* D0..D1, D8..D10, D14..D15, E7..E15 <- 0b00 */ \
GPIOD->MODER = (GPIOD->MODER & ~0b11110000001111110000000000001111); \
GPIOE->MODER = (GPIOE->MODER & ~0b11111111111111111100000000000000); }

// datapins write, STM32 -> LCD (write I/O pins from dt data)
#define LCD_WRITE(dt) { /* D14..15 <- dt0..1, D0..1 <- dt2..3, D8..10 <- dt13..15, E7..15 <- dt4..12 */ \
GPIOD->ODR = (GPIOD->ODR & ~0b1100011100000011) | \
                          (((dt & 0b11) << 14) | ((dt & 0b1100) >> 2) | ((dt & 0b1110000000000000) >> 5)); \
GPIOE->ODR = (GPIOE->ODR & ~0b1111111110000000) | ((dt & 0b0001111111110000) << 3); }

// datapins read, STM32 <- LCD (read from I/O pins and store to dt data)
#define LCD_READ(dt) { /* dt0..1 <- D14..15, dt2..3 <- D0..1, dt13..15 <- D8..10, dt4..12 <- E7..15 */ \
dt = ((GPIOD->IDR & 0b1100000000000000) >> 14) | ((GPIOD->IDR & 0b0000000000000011) << 2) | \
     ((GPIOD->IDR & 0b0000011100000000) << 5)  | ((GPIOE->IDR & 0b1111111110000000) >> 3);  }
```

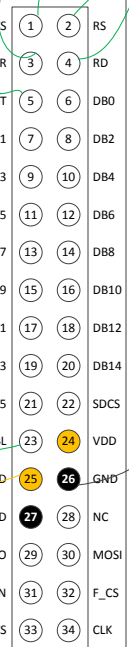
Stm32f407zet gpio 16bit setting



ILI9341 (16bit)

```
#define LCD_D0          D, 14
#define LCD_D1          D, 15
#define LCD_D2          D, 0
#define LCD_D3          D, 1
#define LCD_D4          E, 7
#define LCD_D5          E, 8
#define LCD_D6          E, 9
#define LCD_D7          E, 10
#define LCD_D8          E, 11
#define LCD_D9          E, 12
#define LCD_D10         E, 13
#define LCD_D11         E, 14
#define LCD_D12         E, 15
#define LCD_D13         D, 8
#define LCD_D14         D, 9
#define LCD_D15         D, 10
```

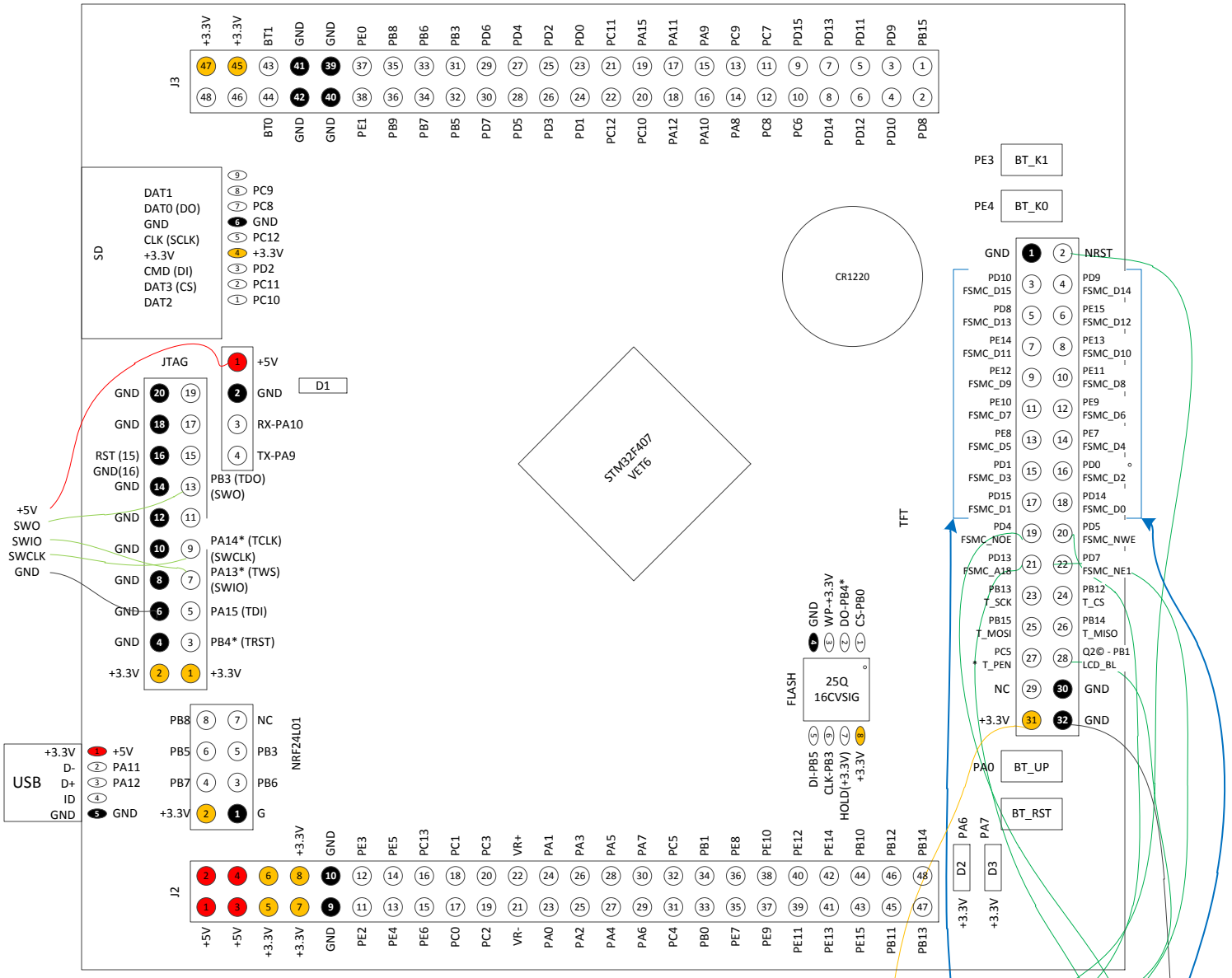
```
#define LCD_CS          G, 12
#define LCD_RS          F, 12
#define LCD_WR          D, 5
#define LCD_RD          D, 4
#define LCD_RST         X, 0

#define LCD_BL          B, 15
#define LCD_BLON        0

#define LCD_WRITE_DELAY 0
#define LCD_READ_DELAY  1
```



Speed optimalization

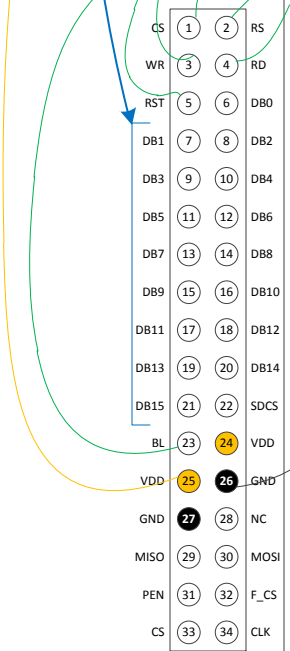
```
// datapins setting to output (data direction: STM32 -> LCD)
#define LCD_DIRWRITE { /* D0..D1, D8..D10, D14..D15, E7..E15 <- 0b01 */ \
GPIOD->MODER = (GPIOD->MODER & ~0b11110000001111110000000000001111) | 0b01010000000101010000000000000101; \
GPIOE->MODER = (GPIOE->MODER & ~0b11111111111111111100000000000000) | 0b01010101010101010100000000000000; }
// datapins setting to input (data direction: STM32 <- LCD)
#define LCD_DIRREAD  { /* D0..D1, D8..D10, D14..D15, E7..E15 <- 0b00 */ \
GPIOD->MODER = (GPIOD->MODER & ~0b11110000001111110000000000001111); \
GPIOE->MODER = (GPIOE->MODER & ~0b11111111111111111100000000000000); }

// datapins write, STM32 -> LCD (write I/O pins from dt data)
#define LCD_WRITE(dt) { /* D14..15 <- dt0..1, D0..1 <- dt2..3, D8..10 <- dt13..15, E7..15 <- dt4..12 */ \
GPIOD->ODR = (GPIOD->ODR & ~0b1100011100000011) | \
                          (((dt & 0b11) << 14) | ((dt & 0b1100) >> 2) | ((dt & 0b1110000000000000) >> 5)); \
GPIOE->ODR = (GPIOE->ODR & ~0b1111111110000000) | ((dt & 0b0001111111110000) << 3); }

// datapins read, STM32 <- LCD (read from I/O pins and store to dt data)
#define LCD_READ(dt) { /* dt0..1 <- D14..15, dt2..3 <- D0..1, dt13..15 <- D8..10, dt4..12 <- E7..15 */ \
dt = ((GPIOD->IDR & 0b1100000000000000) >> 14) | ((GPIOD->IDR & 0b0000000000000011) << 2) | \
     ((GPIOD->IDR & 0b0000011100000000) << 5)  | ((GPIOE->IDR & 0b1111111110000000) >> 3);  }
```

Stm32f407vet fsmc 16bit setting

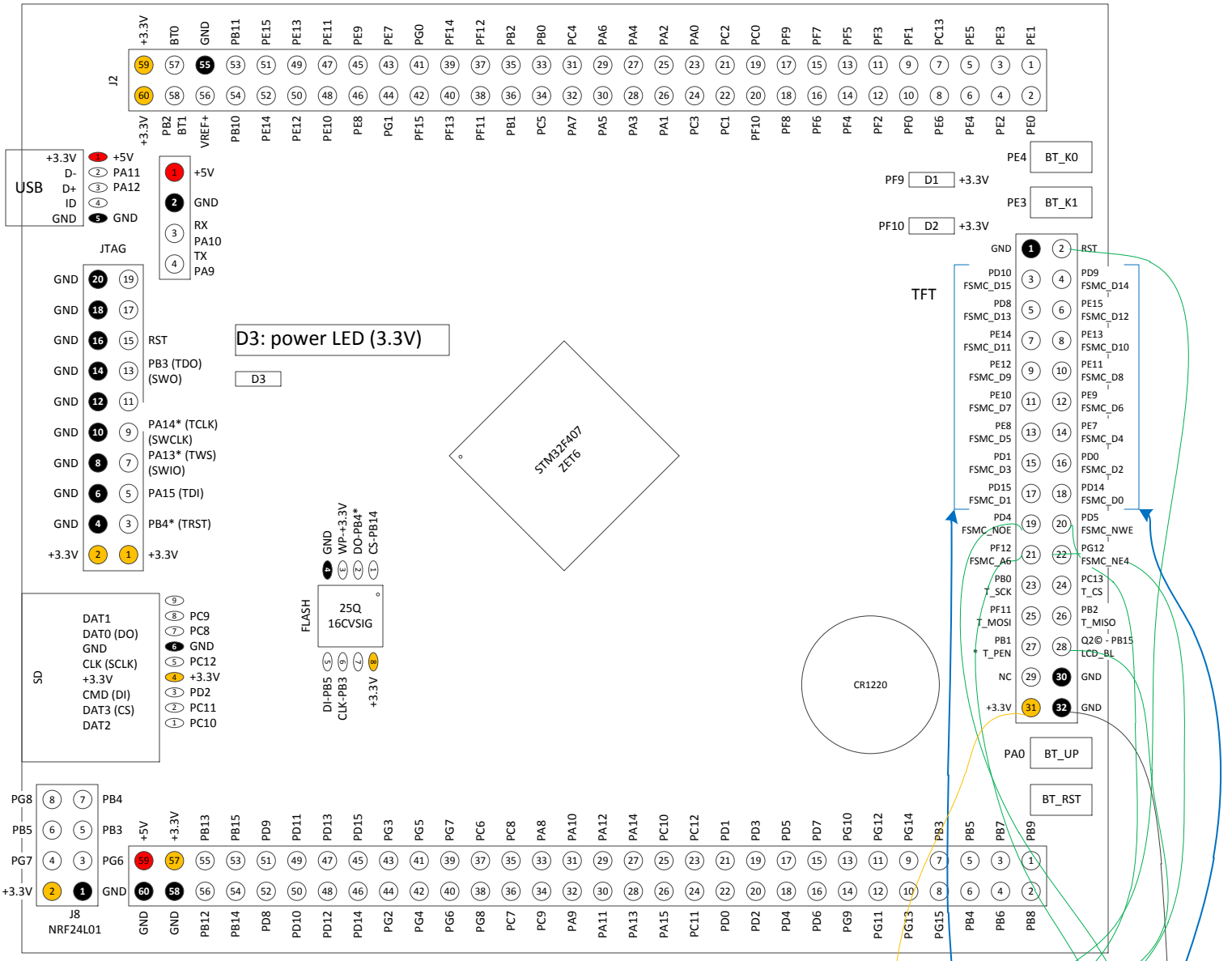


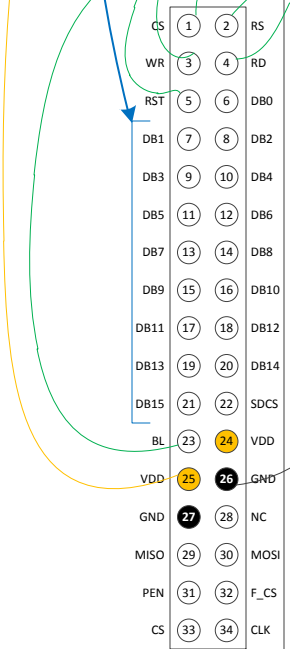
ILI9341 (16bit)

```
#define LCD_RST           X, 0

#define LCD_BL            B, 1
#define LCD_BLON          0

#define LCD_ADDR_BASE     0x60000000 // FSMC_NE1
#define LCD_REGSELECT_BIT 18         // FSMC_A18

If not used DMA:
#define LCD_DMA           0, 0, 0, 0

If used DMA:
#define LCD_DMA           2, 7, 7, 0 // MEM to MEM
```



Stm32f407zet fsmc16bit setting



ILI9341 (16bit)

```
#define LCD_RST           X, 0

#define LCD_BL            B, 15
#define LCD_BLON          0

#define LCD_ADDR_BASE     0x6C000000 // FSMC_NE4
#define LCD_REGSELECT_BIT 6          // FSMC_A6

If not used DMA:
#define LCD_DMA           0, 0, 0, 0

If used DMA:
#define LCD_DMA           2, 7, 7, 1 // MEM to MEM
```