LCD driver layers

This color: v1.1 extension
#define LCD_DRVTYPE_V1_1

Application

BSP_LCD... (stm32_adafruit_lcd.c)

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

} sFONT;
```

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

}LCD_DrawPropTypeDef;
```

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

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

```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);
```

BSP_TS... (stm32_adafruit_ts.c)

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

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

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

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

```c
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_io8p_gpio.c” or „lcd_io_spi.c” or „lcdts_io8p_fsmc.c”)

```c
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...