#ifndef \_ADAFRUIT\_GFX\_H

#define \_ADAFRUIT\_GFX\_H

#if ARDUINO >= 100

#include "Arduino.h"

#include "Print.h"

#else

#include "WProgram.h"

#endif

#include "gfxfont.h"

#include <Adafruit\_I2CDevice.h>

#include <Adafruit\_SPIDevice.h>

/// A generic graphics superclass that can handle all sorts of drawing. At a

/// minimum you can subclass and provide drawPixel(). At a maximum you can do a

/// ton of overriding to optimize. Used for any/all Adafruit displays!

class Adafruit\_GFX : public Print {

public:

Adafruit\_GFX(int16\_t w, int16\_t h); // Constructor

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Draw to the screen/framebuffer/etc.

Must be overridden in subclass.

@param x X coordinate in pixels

@param y Y coordinate in pixels

@param color 16-bit pixel color.

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

virtual void drawPixel(int16\_t x, int16\_t y, uint16\_t color) = 0;

// TRANSACTION API / CORE DRAW API

// These MAY be overridden by the subclass to provide device-specific

// optimized code. Otherwise 'generic' versions are used.

virtual void startWrite(void);

virtual void writePixel(int16\_t x, int16\_t y, uint16\_t color);

virtual void writeFillRect(int16\_t x, int16\_t y, int16\_t w, int16\_t h,

uint16\_t color);

virtual void writeFastVLine(int16\_t x, int16\_t y, int16\_t h, uint16\_t color);

virtual void writeFastHLine(int16\_t x, int16\_t y, int16\_t w, uint16\_t color);

virtual void writeLine(int16\_t x0, int16\_t y0, int16\_t x1, int16\_t y1,

uint16\_t color);

virtual void endWrite(void);

// CONTROL API

// These MAY be overridden by the subclass to provide device-specific

// optimized code. Otherwise 'generic' versions are used.

virtual void setRotation(uint8\_t r);

virtual void invertDisplay(bool i);

// BASIC DRAW API

// These MAY be overridden by the subclass to provide device-specific

// optimized code. Otherwise 'generic' versions are used.

// It's good to implement those, even if using transaction API

virtual void drawFastVLine(int16\_t x, int16\_t y, int16\_t h, uint16\_t color);

virtual void drawFastHLine(int16\_t x, int16\_t y, int16\_t w, uint16\_t color);

virtual void fillRect(int16\_t x, int16\_t y, int16\_t w, int16\_t h,

uint16\_t color);

virtual void fillScreen(uint16\_t color);

// Optional and probably not necessary to change

virtual void drawLine(int16\_t x0, int16\_t y0, int16\_t x1, int16\_t y1,

uint16\_t color);

virtual void drawRect(int16\_t x, int16\_t y, int16\_t w, int16\_t h,

uint16\_t color);

// These exist only with Adafruit\_GFX (no subclass overrides)

void drawCircle(int16\_t x0, int16\_t y0, int16\_t r, uint16\_t color);

void drawCircleHelper(int16\_t x0, int16\_t y0, int16\_t r, uint8\_t cornername,

uint16\_t color);

void fillCircle(int16\_t x0, int16\_t y0, int16\_t r, uint16\_t color);

void fillCircleHelper(int16\_t x0, int16\_t y0, int16\_t r, uint8\_t cornername,

int16\_t delta, uint16\_t color);

void drawTriangle(int16\_t x0, int16\_t y0, int16\_t x1, int16\_t y1, int16\_t x2,

int16\_t y2, uint16\_t color);

void fillTriangle(int16\_t x0, int16\_t y0, int16\_t x1, int16\_t y1, int16\_t x2,

int16\_t y2, uint16\_t color);

void drawRoundRect(int16\_t x0, int16\_t y0, int16\_t w, int16\_t h,

int16\_t radius, uint16\_t color);

void fillRoundRect(int16\_t x0, int16\_t y0, int16\_t w, int16\_t h,

int16\_t radius, uint16\_t color);

void drawBitmap(int16\_t x, int16\_t y, const uint8\_t bitmap[], int16\_t w,

int16\_t h, uint16\_t color);

void drawBitmap(int16\_t x, int16\_t y, const uint8\_t bitmap[], int16\_t w,

int16\_t h, uint16\_t color, uint16\_t bg);

void drawBitmap(int16\_t x, int16\_t y, uint8\_t \*bitmap, int16\_t w, int16\_t h,

uint16\_t color);

void drawBitmap(int16\_t x, int16\_t y, uint8\_t \*bitmap, int16\_t w, int16\_t h,

uint16\_t color, uint16\_t bg);

void drawXBitmap(int16\_t x, int16\_t y, const uint8\_t bitmap[], int16\_t w,

int16\_t h, uint16\_t color);

void drawGrayscaleBitmap(int16\_t x, int16\_t y, const uint8\_t bitmap[],

int16\_t w, int16\_t h);

void drawGrayscaleBitmap(int16\_t x, int16\_t y, uint8\_t \*bitmap, int16\_t w,

int16\_t h);

void drawGrayscaleBitmap(int16\_t x, int16\_t y, const uint8\_t bitmap[],

const uint8\_t mask[], int16\_t w, int16\_t h);

void drawGrayscaleBitmap(int16\_t x, int16\_t y, uint8\_t \*bitmap, uint8\_t \*mask,

int16\_t w, int16\_t h);

void drawRGBBitmap(int16\_t x, int16\_t y, const uint16\_t bitmap[], int16\_t w,

int16\_t h);

void drawRGBBitmap(int16\_t x, int16\_t y, uint16\_t \*bitmap, int16\_t w,

int16\_t h);

void drawRGBBitmap(int16\_t x, int16\_t y, const uint16\_t bitmap[],

const uint8\_t mask[], int16\_t w, int16\_t h);

void drawRGBBitmap(int16\_t x, int16\_t y, uint16\_t \*bitmap, uint8\_t \*mask,

int16\_t w, int16\_t h);

void drawChar(int16\_t x, int16\_t y, unsigned char c, uint16\_t color,

uint16\_t bg, uint8\_t size);

void drawChar(int16\_t x, int16\_t y, unsigned char c, uint16\_t color,

uint16\_t bg, uint8\_t size\_x, uint8\_t size\_y);

void getTextBounds(const char \*string, int16\_t x, int16\_t y, int16\_t \*x1,

int16\_t \*y1, uint16\_t \*w, uint16\_t \*h);

void getTextBounds(const \_\_FlashStringHelper \*s, int16\_t x, int16\_t y,

int16\_t \*x1, int16\_t \*y1, uint16\_t \*w, uint16\_t \*h);

void getTextBounds(const String &str, int16\_t x, int16\_t y, int16\_t \*x1,

int16\_t \*y1, uint16\_t \*w, uint16\_t \*h);

void setTextSize(uint8\_t s);

void setTextSize(uint8\_t sx, uint8\_t sy);

void setFont(const GFXfont \*f = NULL);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Set text cursor location

@param x X coordinate in pixels

@param y Y coordinate in pixels

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setCursor(int16\_t x, int16\_t y) {

cursor\_x = x;

cursor\_y = y;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Set text font color with transparant background

@param c 16-bit 5-6-5 Color to draw text with

@note For 'transparent' background, background and foreground

are set to same color rather than using a separate flag.

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setTextColor(uint16\_t c) { textcolor = textbgcolor = c; }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Set text font color with custom background color

@param c 16-bit 5-6-5 Color to draw text with

@param bg 16-bit 5-6-5 Color to draw background/fill with

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setTextColor(uint16\_t c, uint16\_t bg) {

textcolor = c;

textbgcolor = bg;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Set whether text that is too long for the screen width should

automatically wrap around to the next line (else clip right).

@param w true for wrapping, false for clipping

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setTextWrap(bool w) { wrap = w; }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Enable (or disable) Code Page 437-compatible charset.

There was an error in glcdfont.c for the longest time -- one

character (#176, the 'light shade' block) was missing -- this

threw off the index of every character that followed it.

But a TON of code has been written with the erroneous

character indices. By default, the library uses the original

'wrong' behavior and old sketches will still work. Pass

'true' to this function to use correct CP437 character values

in your code.

@param x true = enable (new behavior), false = disable (old behavior)

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void cp437(bool x = true) { \_cp437 = x; }

using Print::write;

#if ARDUINO >= 100

virtual size\_t write(uint8\_t);

#else

virtual void write(uint8\_t);

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Get width of the display, accounting for current rotation

@returns Width in pixels

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int16\_t width(void) const { return \_width; };

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Get height of the display, accounting for current rotation

@returns Height in pixels

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int16\_t height(void) const { return \_height; }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Get rotation setting for display

@returns 0 thru 3 corresponding to 4 cardinal rotations

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t getRotation(void) const { return rotation; }

// get current cursor position (get rotation safe maximum values,

// using: width() for x, height() for y)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Get text cursor X location

@returns X coordinate in pixels

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int16\_t getCursorX(void) const { return cursor\_x; }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Get text cursor Y location

@returns Y coordinate in pixels

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int16\_t getCursorY(void) const { return cursor\_y; };

protected:

void charBounds(unsigned char c, int16\_t \*x, int16\_t \*y, int16\_t \*minx,

int16\_t \*miny, int16\_t \*maxx, int16\_t \*maxy);

int16\_t WIDTH; ///< This is the 'raw' display width - never changes

int16\_t HEIGHT; ///< This is the 'raw' display height - never changes

int16\_t \_width; ///< Display width as modified by current rotation

int16\_t \_height; ///< Display height as modified by current rotation

int16\_t cursor\_x; ///< x location to start print()ing text

int16\_t cursor\_y; ///< y location to start print()ing text

uint16\_t textcolor; ///< 16-bit background color for print()

uint16\_t textbgcolor; ///< 16-bit text color for print()

uint8\_t textsize\_x; ///< Desired magnification in X-axis of text to print()

uint8\_t textsize\_y; ///< Desired magnification in Y-axis of text to print()

uint8\_t rotation; ///< Display rotation (0 thru 3)

bool wrap; ///< If set, 'wrap' text at right edge of display

bool \_cp437; ///< If set, use correct CP437 charset (default is off)

GFXfont \*gfxFont; ///< Pointer to special font

};

/// A simple drawn button UI element

class Adafruit\_GFX\_Button {

public:

Adafruit\_GFX\_Button(void);

// "Classic" initButton() uses center & size

void initButton(Adafruit\_GFX \*gfx, int16\_t x, int16\_t y, uint16\_t w,

uint16\_t h, uint16\_t outline, uint16\_t fill,

uint16\_t textcolor, char \*label, uint8\_t textsize);

void initButton(Adafruit\_GFX \*gfx, int16\_t x, int16\_t y, uint16\_t w,

uint16\_t h, uint16\_t outline, uint16\_t fill,

uint16\_t textcolor, char \*label, uint8\_t textsize\_x,

uint8\_t textsize\_y);

// New/alt initButton() uses upper-left corner & size

void initButtonUL(Adafruit\_GFX \*gfx, int16\_t x1, int16\_t y1, uint16\_t w,

uint16\_t h, uint16\_t outline, uint16\_t fill,

uint16\_t textcolor, char \*label, uint8\_t textsize);

void initButtonUL(Adafruit\_GFX \*gfx, int16\_t x1, int16\_t y1, uint16\_t w,

uint16\_t h, uint16\_t outline, uint16\_t fill,

uint16\_t textcolor, char \*label, uint8\_t textsize\_x,

uint8\_t textsize\_y);

void drawButton(bool inverted = false);

bool contains(int16\_t x, int16\_t y);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Sets button state, should be done by some touch function

@param p True for pressed, false for not.

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void press(bool p) {

laststate = currstate;

currstate = p;

}

bool justPressed();

bool justReleased();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Query whether the button is currently pressed

@returns True if pressed

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool isPressed(void) { return currstate; };

private:

Adafruit\_GFX \*\_gfx;

int16\_t \_x1, \_y1; // Coordinates of top-left corner

uint16\_t \_w, \_h;

uint8\_t \_textsize\_x;

uint8\_t \_textsize\_y;

uint16\_t \_outlinecolor, \_fillcolor, \_textcolor;

char \_label[10];

bool currstate, laststate;

};

/// A GFX 1-bit canvas context for graphics

class GFXcanvas1 : public Adafruit\_GFX {

public:

GFXcanvas1(uint16\_t w, uint16\_t h);

~GFXcanvas1(void);

void drawPixel(int16\_t x, int16\_t y, uint16\_t color);

void fillScreen(uint16\_t color);

void drawFastVLine(int16\_t x, int16\_t y, int16\_t h, uint16\_t color);

void drawFastHLine(int16\_t x, int16\_t y, int16\_t w, uint16\_t color);

bool getPixel(int16\_t x, int16\_t y) const;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Get a pointer to the internal buffer memory

@returns A pointer to the allocated buffer

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t \*getBuffer(void) const { return buffer; }

protected:

bool getRawPixel(int16\_t x, int16\_t y) const;

void drawFastRawVLine(int16\_t x, int16\_t y, int16\_t h, uint16\_t color);

void drawFastRawHLine(int16\_t x, int16\_t y, int16\_t w, uint16\_t color);

uint8\_t \*buffer; ///< Raster data: no longer private, allow subclass access

private:

#ifdef \_\_AVR\_\_

// Bitmask tables of 0x80>>X and ~(0x80>>X), because X>>Y is slow on AVR

static const uint8\_t PROGMEM GFXsetBit[], GFXclrBit[];

#endif

};

/// A GFX 8-bit canvas context for graphics

class GFXcanvas8 : public Adafruit\_GFX {

public:

GFXcanvas8(uint16\_t w, uint16\_t h);

~GFXcanvas8(void);

void drawPixel(int16\_t x, int16\_t y, uint16\_t color);

void fillScreen(uint16\_t color);

void drawFastVLine(int16\_t x, int16\_t y, int16\_t h, uint16\_t color);

void drawFastHLine(int16\_t x, int16\_t y, int16\_t w, uint16\_t color);

uint8\_t getPixel(int16\_t x, int16\_t y) const;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Get a pointer to the internal buffer memory

@returns A pointer to the allocated buffer

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t \*getBuffer(void) const { return buffer; }

protected:

uint8\_t getRawPixel(int16\_t x, int16\_t y) const;

void drawFastRawVLine(int16\_t x, int16\_t y, int16\_t h, uint16\_t color);

void drawFastRawHLine(int16\_t x, int16\_t y, int16\_t w, uint16\_t color);

uint8\_t \*buffer; ///< Raster data: no longer private, allow subclass access

};

/// A GFX 16-bit canvas context for graphics

class GFXcanvas16 : public Adafruit\_GFX {

public:

GFXcanvas16(uint16\_t w, uint16\_t h);

~GFXcanvas16(void);

void drawPixel(int16\_t x, int16\_t y, uint16\_t color);

void fillScreen(uint16\_t color);

void byteSwap(void);

void drawFastVLine(int16\_t x, int16\_t y, int16\_t h, uint16\_t color);

void drawFastHLine(int16\_t x, int16\_t y, int16\_t w, uint16\_t color);

uint16\_t getPixel(int16\_t x, int16\_t y) const;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Get a pointer to the internal buffer memory

@returns A pointer to the allocated buffer

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint16\_t \*getBuffer(void) const { return buffer; }

protected:

uint16\_t getRawPixel(int16\_t x, int16\_t y) const;

void drawFastRawVLine(int16\_t x, int16\_t y, int16\_t h, uint16\_t color);

void drawFastRawHLine(int16\_t x, int16\_t y, int16\_t w, uint16\_t color);

uint16\_t \*buffer; ///< Raster data: no longer private, allow subclass access

};

#endif // \_ADAFRUIT\_GFX\_H