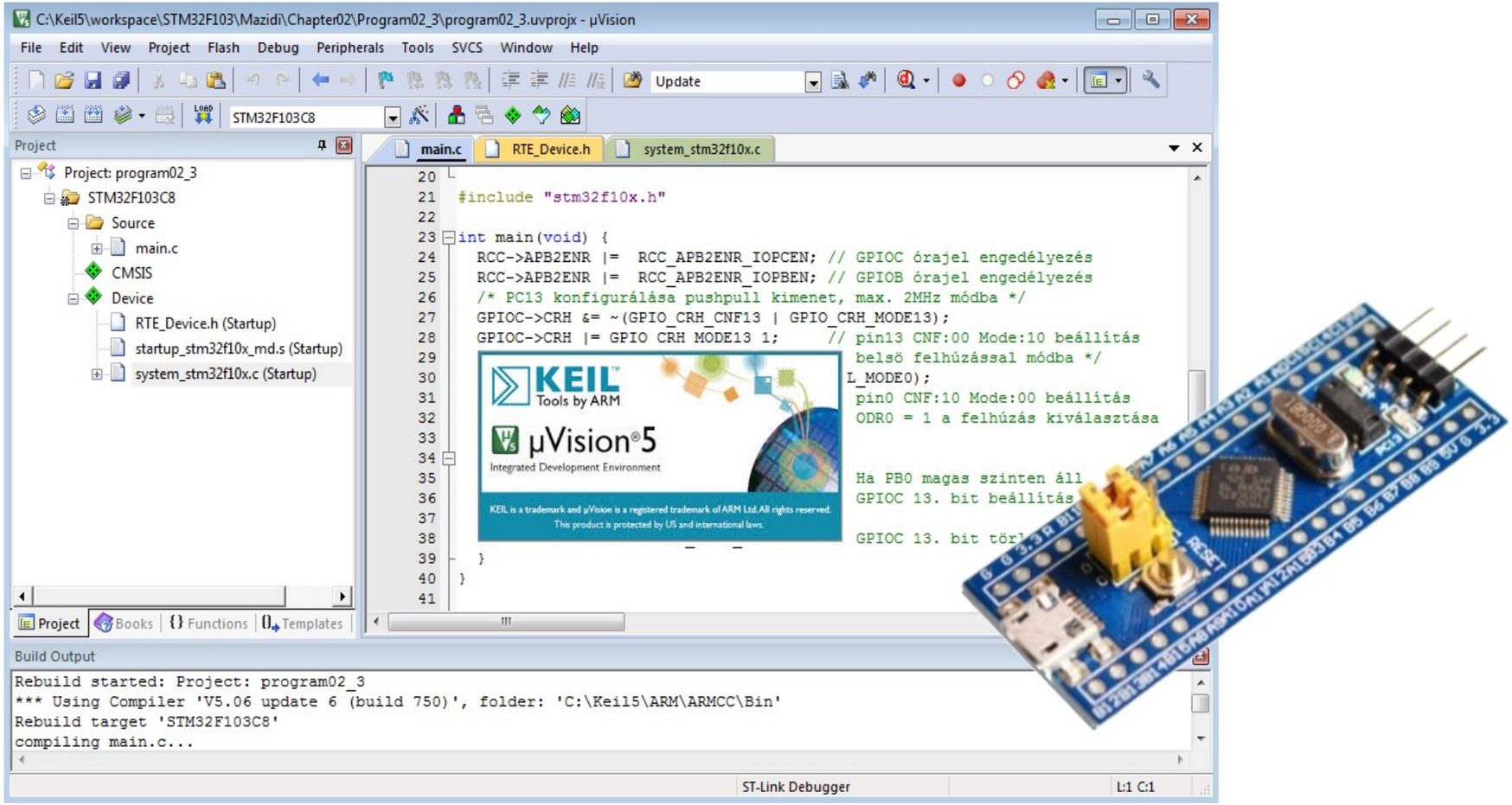
STM32 mikrovezérlők programozása ARM Keil környezetben



17. ST7735 1.8” színes TFT kijelzők vezérlése

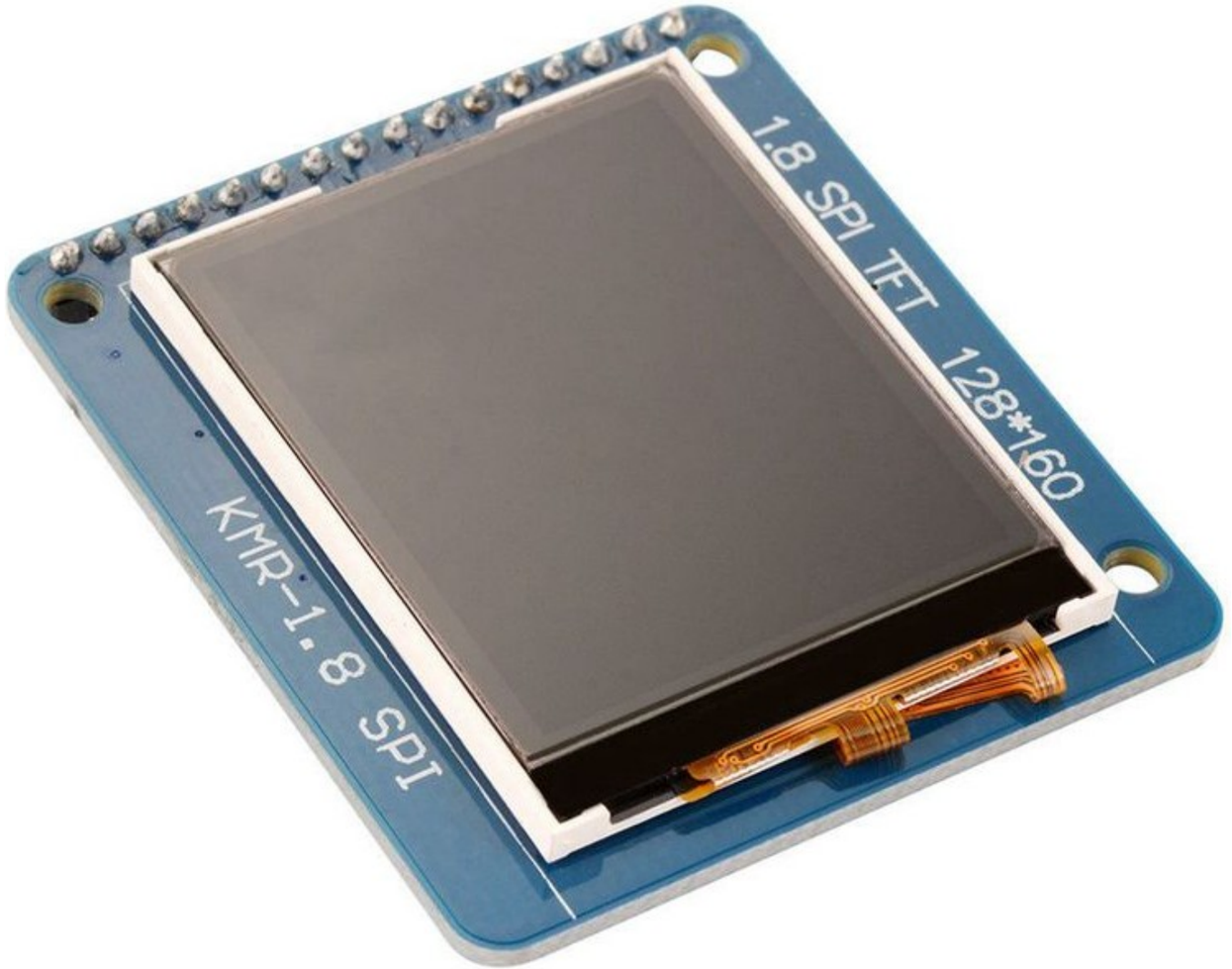
Felhasznált és ajánlott irodalom

- Joseph Yiu: The Definitive Guide To The ARM CORTEX-M3
- Muhammad Ali Mazidi, Shujen Chen, Eshragh Ghaemi: STM32 Arm Programming for Embedded Systems
- Alexander Tarasov: **Курс «Штурмуем STM32»**
- ARM Keil MDK Getting started
- **STM32F103C8** adatlap és termékinfo
- **STM32F103** Family Reference Manual

- Sitronix ST7735 adatlap
- Adafruit: Adafruit-ST7735-Library
- Dung-Yi Chao: STM32F4-ST7735 tutorial

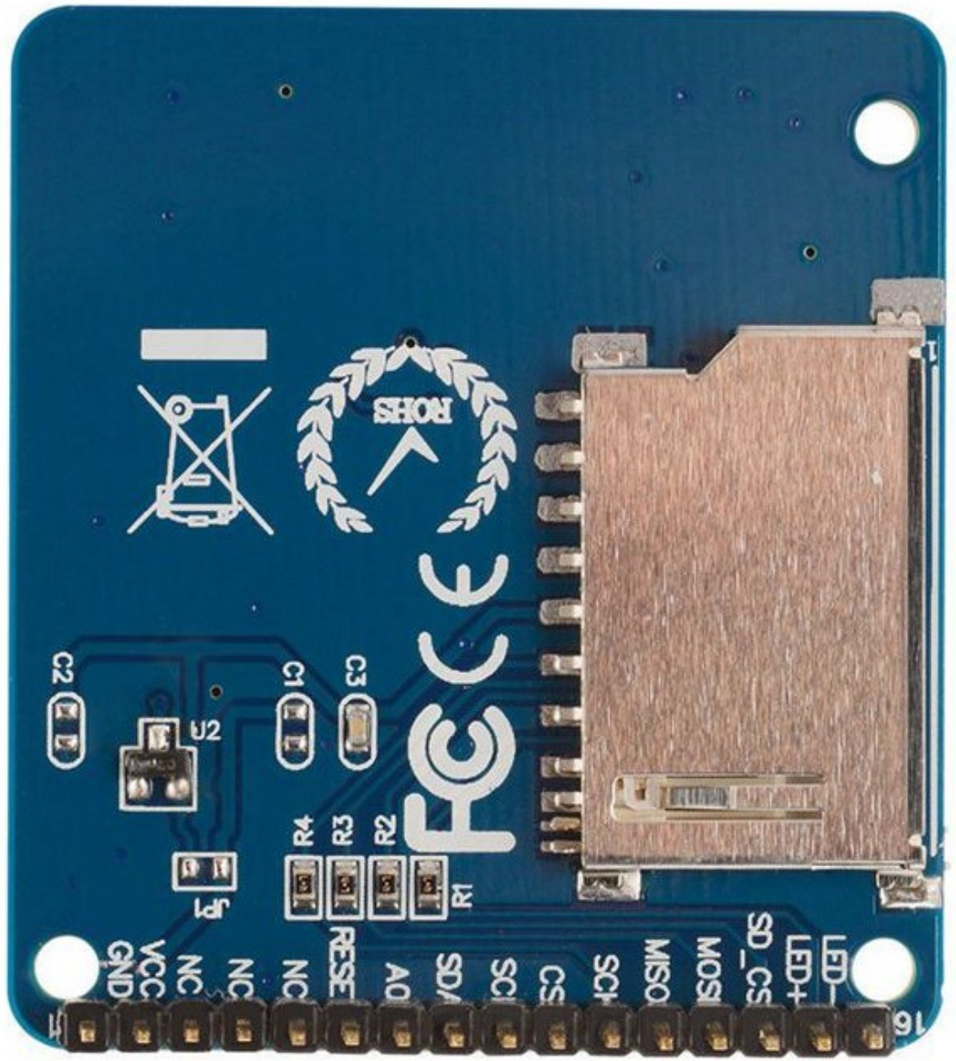
ST7735 1.8” színes LCD kijelző

- Vezérlő: **Sitronix ST7735**
- Interfész: SPI, csak írás (max. 10 – 15 MHz)
- Data/Command választás külön vonalon
- Kijelző: TFT, 65 536 szín (16 bit RGB, 5-6-5)
- Felbontás: **128 x 160** pixel
- SD kártya foglalat:
- Csak SPI mód kivezetések:
  - SD_CS
  - MOSI
  - MISO
  - SCK

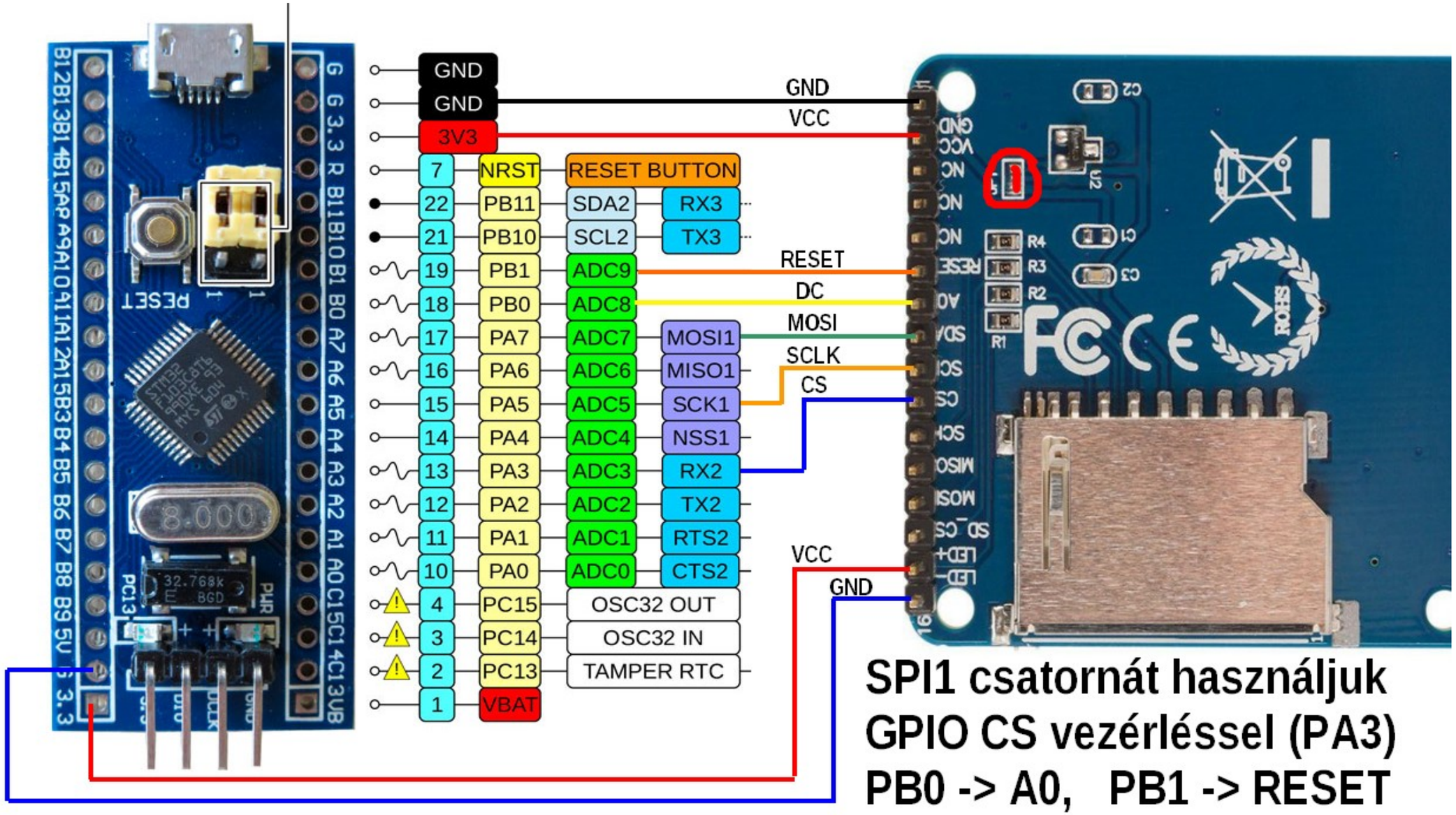


A kivezetések

- **GND** – a tápegység közös pontja
- **VCC** – tápfeszültség (5V, vagy **JP1** zárása után 3.3V)
- **NC** – nincs bekötve
- **RESET** – kijelző reset jel (0: aktív, 1: inaktív)
- **A0** – kijelző regisztertválasztó jel (RS, DC)
- **SDA** – kijelző SPI adatbemenet (MOSI jel)
- **SCL** – kijelző SPI órajel
- **CS** – kijelző SPI eszközválasztó jel (0: aktív)
- **SCK** – SD kártya SPI órajel
- **MISO** – SD kártya adatkimenete
- **MOSI** – SD kártya adatbemenete
- **SD_CS** – SD kártya eszközválasztó jel (0: aktív)
- **LED+** – kijelző háttérvilágítás (anód)
- **LED–** – kijelző háttérvilágítás (katód)

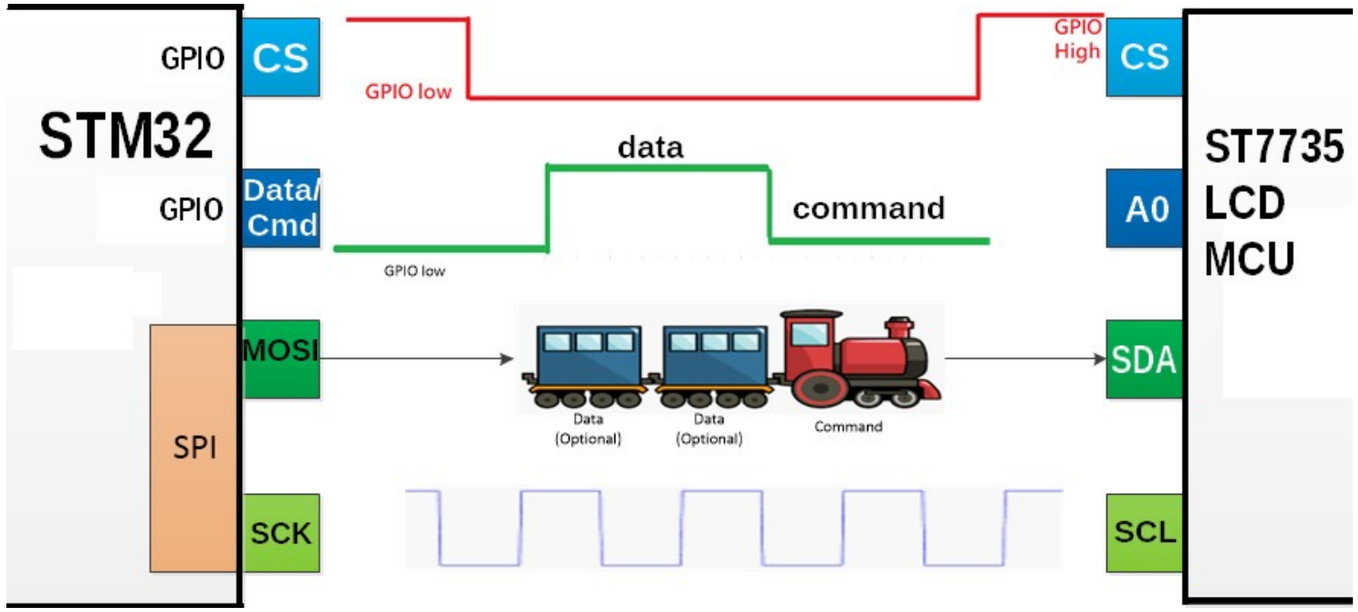


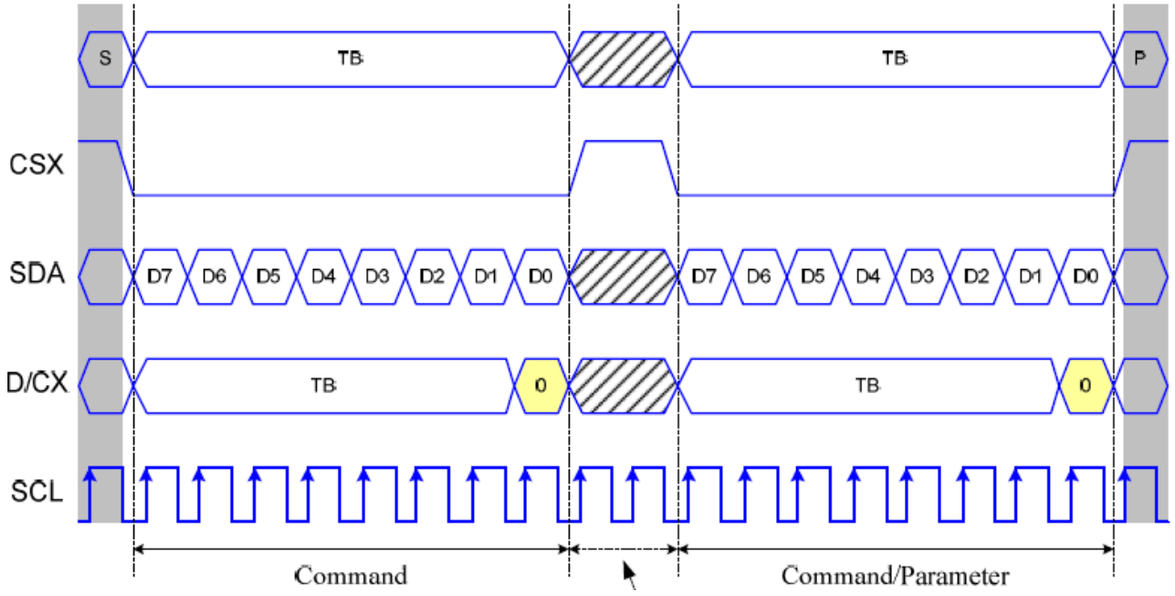
Bekötési vázlat



A kijelző modul vezérlése

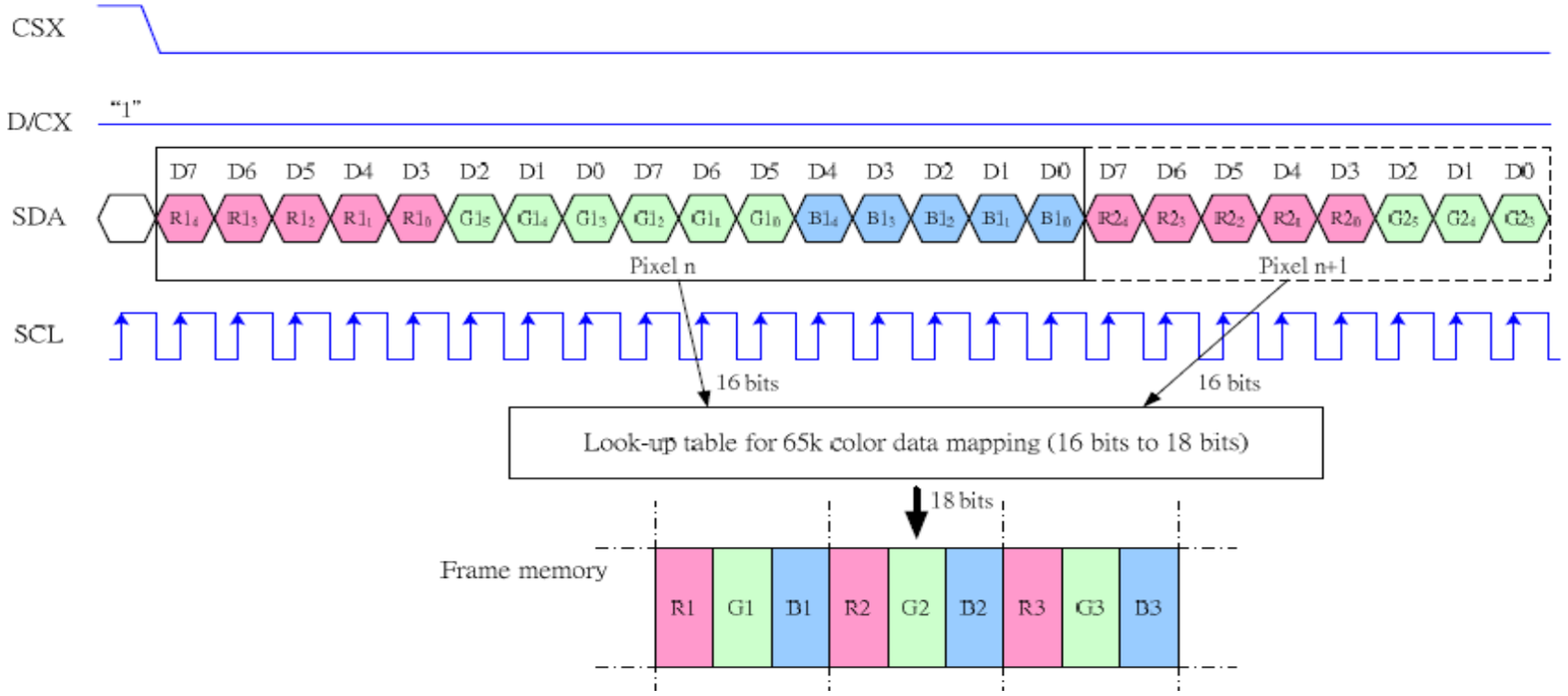
- A vezérlés azt jelenti, hogy parancsokat és adatokat küldünk az **ST7735** vezérlőnek
- A kommunikáció módja esetünkben 4-vezetékes (**CS, SCK, SDA, DC**), ahol a **DC** vonalon jelezzük, hogy parancs (DC = '0') vagy adat (DC = '1') következik
- A **CS** jel magas állapotában az **ST7735** bemenete inaktív, az adat és órajel bemenet változásait figyelmen kívül hagyja

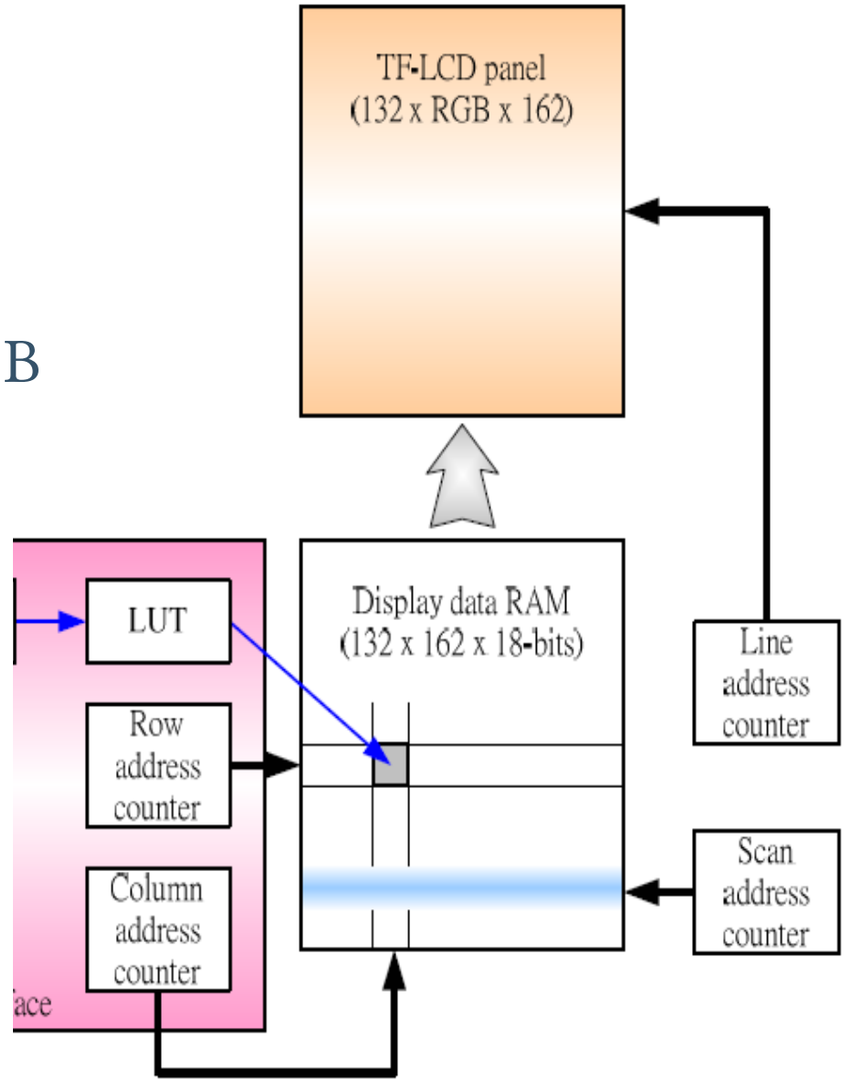




Színvezérlés 5-6-5 módban

- Az **ST7735** képmemóriája 132 x 162 x 18 bit (RGB)
- Színvezérlési módok: 12-, 16-, vagy 18 bites
- Mi a 16-bites (5-6-5) színvezérlési módot használjuk, amely a **COLMOD** (0x3A) = 0x05 paranccsal választható
- Minden pixelhez két bájtot kell küldeni: 5 bit R, 6 bit G, 5 bit B





A képmemória

- A 132 x 162 képpontból a kijelzőnk csak 128 x 160-at tud megjeleníteni
- A memória és a kijelző képpontjainak összerendelése a címzést befolyásoló regisztereinek átírásával megváltoztatható
- Ugyancsak megváltoztatható az RGB sorrend BGR-re (bizonyos kijelzőtípusokhoz kell)

RA = Row Address,
CA = Column Address
SA = Scan Address
MX = Mirror X-axis
MY = Mirror Y-axis
ML = Scan direction parameter

When using 128RGB x 160 resolution (GM[2:0] = "011", SMX=SMY=SRGB= '0')

Gate Out	Source Out		S7	S8	S9	S10	S11	S12	------	S385	S386	S387	S388	S389	S390		
	RA		RGB=0 / RGB=N						RGB Order	RGB=0 / RGB=N						SA	
	MY=' 0 '	MY=' 1 '														ML=' 0 '	ML=' 1 '
2	0	159	R0	G0	B0	R1	G1	B1	------	R126	G126	B126	R127	G127	B127	0	159
3	1	158							------							1	158
4	2	157							------							2	157
5	3	156							------							3	156
6	4	155							------							4	155
7	5	154							------							5	154
8	6	153							------							6	153
9	7	152							------							7	152
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
154	152	7							------							152	7
155	153	6							------							153	6
156	154	5							------							154	5
157	155	4							------							155	4
158	156	3							------							156	3
159	157	2							------							157	2
160	158	1							------							158	1
161	159	0							------							159	0
	CA	MX=' 0 '	0			1			------	126			127				
		MX=' 1 '	127			126			------	1			0				

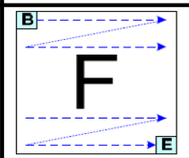
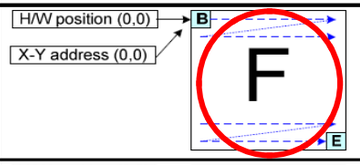
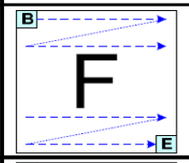
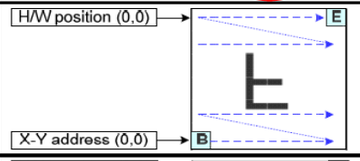
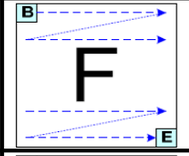
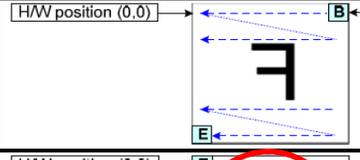
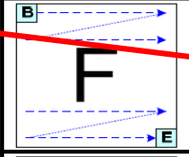
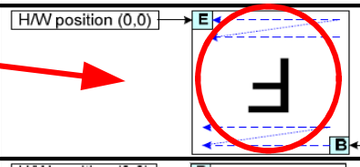
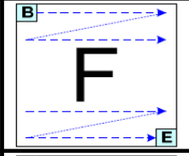
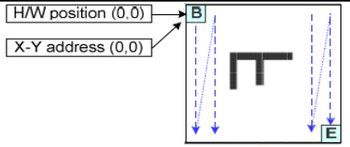
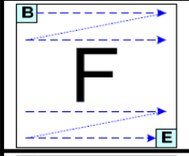
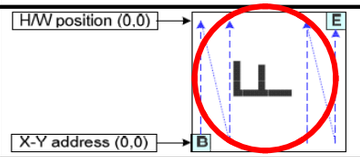
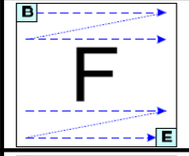
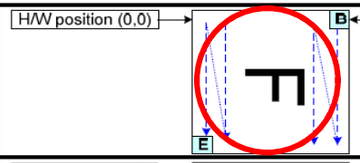
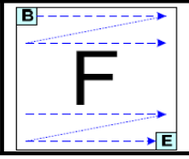
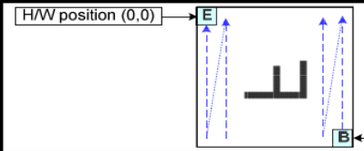
Képernyőtükrözés, forgatás

- Az *x* és *y* tengelyre történő tükrözések kombinálásával segítségével fejtetőre állíthatjuk a képet (180 fokos elforgatás)

  Mi is ezt a beállítást használjuk:
  MADCTL (0x36) = 0xC8 (MX =1, MY=1)
  Ha a 0xC8 paraméter helyett nullát írunk, megfordítható a kép!

- Az *x*-*y* koordináták felcserélésével és *x* vagy *y* tengelyre történő tükrözéssel elforgathatjuk a képet 90, illetve 270 fokkal is

- Léteznek olyan kijelzőmodulok, amelyeknél a kijelző képpontjai 1 - 2 képponttal eltolva vannak bekötve, ezt is lehet korrigálni a **CASET** és **RASET** parancsok kiadásánál

Display Data Direction	MADCTL Parameter MV	MX	MY	Image in the Host (MPU)	Image in the Driver (DDRAM)
Normal	0	0	0		
Y-Mirror	0	0	1		
X-Mirror	0	1	0		
X-Mirror Y-Mirror	0	1	1		
X-Y Exchange	1	0	0		
X-Y Exchange Y-Mirror	1	0	1		
X-Y Exchange X-Mirror	1	1	0		
X-Y Exchange X-Mirror Y-Mirror	1	1	1		

A hardver inicializálás lépései (spi.c, illetve st7735.c)

- **SPI1** órajelének engedélyezése (`RCC->APB2ENR SPI1EN` bit)
- **AFIO** modul órajelének engedélyezése (`RCC->APB2ENR AFIOEN` bit)
- **GPIOA** órajelének engedélyezés (`RCC->APB2ENR IOPAEN` bit – PA3, PA5, PA7 miatt)
- **GPIOB** órajelének engedélyezés (`RCC->APB2ENR IOPBEN` bit – PB0, PB1 miatt)
- **SPI1** remap tiltása (`AFIO->MAPR SPI1_REMAP` bit törlése)
- **PA5** és **PA7** konfigurálása, mint **SCK** és **MOSI** (`CNF=10 MODE=11` AltFunc pushpull kimenet)
- **PA3** konfigurálása, mint **CS** (`CNF=00 MODE=11 GPIO` pushpull kimenet)
- **SPI1->CR1** konfigurálása: Master mód=1, 8-bit, LSBfirst=0, BR = 011 (72/16 = 4.5 MHz)
- **SPI1->CR2** konfigurálása: SSOE=1
- **SPI1->CR1** SPE bit 1-be állítása (engedélyezés)
- **PB0** és **PB1** konfigurálása (`CNF=00 MODE=11 GPIO` pushpull kimenet)

Bővebben lásd a keil19_12, keil19_13, keil19_14 előadásokban *(2020. március-április)*

Az SPI1 port kezelése

spi.c

```
#include "spi.h"
void SPI1_enable() { GPIOA->BSRR = GPIO_BSRR_BR4 | GPIO_BSRR_BR3; }          // PA3, PA4 LOW
void SPI1_disable(void) { GPIOA->BSRR = GPIO_BSRR_BS4 | GPIO_BSRR_BS3; }     // PA3, PA4 HIGH

void SPI1_init(void) {
  RCC->APB2ENR |= RCC_APB2ENR_SPI1EN;  // SPI1 órajel engedélyezés
  RCC->APB2ENR |= RCC_APB2ENR_AFIOEN;  // AFIO órajel engedélyezés
  RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;  // GPIOA órajel engedélyezés
  AFIO->MAPR &= ~AFIO_MAPR_SPI1_REMAP; // Törli SPI1 REMAP bitjét
  GPIOA->CRL &= ~0xF0FFF000;           // CNF és MODE bitek törlése
  GPIOA->CRL |=  0xB0B33000;           // CNF=10 MODE=11 (PA5,PA7) AltFunc puspull
                                       // CNF=00 MODE=11 (PA4)   GPIO pushpull
  SPI1->CR1 = SPI_CR1_BR_1 |           // Baud rate 72/16 = 4.5 MHz
              SPI_CR1_bR_0 |
              SPI_CR1_MSTR;            // Master mód beállítás
  SPI1->CR2 = SPI_CR2_SSOE;            // Single Master mód
  SPI1->CR1 |= SPI_CR1_SPE;            // Enable SPI1 module
}

void SPI1_write(uint8_t data) {
  SPI1_enable();                       // SPI1 SS engedélyezése
  while (!(SPI1->SR & SPI_SR_TXE)) {}  // TXE jelre várunk
  SPI1->DR = data;                     // Adat küldése
  while (SPI1->SR & SPI_SR_BSY) {}     // Átvitel végére várunk
  SPI1_disable();                      // SPI1 SS letiltása
}
```

spi.h

```
#include "stm32f10x.h"
void SPI1_init(void) ;
void SPI1_enable(void);
void SPI1_disable(void);
void SPI1_write(uint8_t data);
```

Az ST7735 inicializálása és kezelése (st7735.h)

```
#include "stm32f10x.h"
#include <stdbool.h>
#include "spi.h"
//--  ST7735 RST pin control -----------------
#define LCD_RST1  GPIOB->BSRR = GPIO_BSRR_BS1;    // Set PB1
#define LCD_RST0  GPIOB->BSRR = GPIO_BSRR_BR1;    // Clear PB1
//-- ST7735 DC pin control -------------------
#define  LCD_DC1  GPIOB->BSRR = GPIO_BSRR_BS0;     // Set PB0
#define  LCD_DC0  GPIOB->BSRR = GPIO_BSRR_BR0;     // Clear PB0
#define ST7735_WIDTH  128
#define ST7735_HEIGHT 160
#define ST7735_XSTART 0
#define ST7735_YSTART 0
#define ST7735_ROTATION (ST7735_MADCTL_MX|ST7735_MADCTL_MY)

#define ST7735_NOP     0x00
#define ST7735_SWRESET 0x01
#define ST7735_RDDID   0x04
#define ST7735_RDDST   0x09
#define ST7735_SLPIN   0x10
#define ST7735_SLPOUT  0x11
#define ST7735_PTLON   0x12
#define ST7735_NORON   0x13
```

Itt lehet forgatni a képet
0x0 esetén a kivezetések felől van 0,0 pozíció
0xC8 esetén pedig 180°-kal el van forgatva a kép

```
#define ST7735_INVOFF  0x20
#define ST7735_INVON   0x21
#define ST7735_DISPOFF 0x28
#define ST7735_DISPON  0x29
#define ST7735_CASET   0x2A
#define ST7735_RASET   0x2B
#define ST7735_RAMWR   0x2C
#define ST7735_RAMRD   0x2E
#define ST7735_PTLAR   0x30
#define ST7735_COLMOD  0x3A
#define ST7735_MADCTL  0x36
#define ST7735_FRMCTR1 0xB1
#define ST7735_FRMCTR2 0xB2
#define ST7735_FRMCTR3 0xB3
#define ST7735_INVCTR  0xB4
#define ST7735_DISSET5 0xB6
#define ST7735_PWCTR1  0xC0
#define ST7735_PWCTR2  0xC1
#define ST7735_PWCTR3  0xC2
#define ST7735_PWCTR4  0xC3
#define ST7735_PWCTR5  0xC4
#define ST7735_VMCTR1  0xC5
...
```

Az ST7735 inicializálása és kezelése 3/1. (st7735.c)

```
#include "st7735.h"

void lcd7735_sendCmd(uint8_t cmd) {          // Parancsbájt küldése
  LCD_DC0; //Set DC low
  SPI1_write(cmd);
}

void lcd7735_sendData(uint8_t data) {        // Adatbájt küldése
  LCD_DC1;//Set DC HIGH
  SPI1_write(data);
}

void DelayMs(int n) {
  int i;
  SysTick->LOAD = SystemCoreClock/1000-1;  // Újratöltési érték 1 ms késleltetéshez
  SysTick->VAL  = 0;                       // Számláló törlése
  SysTick->CTRL = 0x5;                     // Engedélyezés, nincs interrupt, rendszer órajel
  for(i = 0; i < n; i++) {
    while((SysTick->CTRL & 0x10000) == 0); // A COUNT jelzöre várunk
  }
  SysTick->CTRL = 0;                       // SysTick leállítása
}
```

Az ST7735 inicializálása és kezelése 3/2. (st7735.c)

```
void ST7735_Init() {
  SPI1_enable();              //ST7735_Select();
  LCD_RST1; DelayMs(500);
  LCD_RST0; DelayMs(500);
  LCD_RST1; DelayMs(500);
  ST7735_Init_Command1();
  ST7735_Init_Command2();
  ST7735_Init_Command3();
  SPI1_disable();
}

void ST7735_Init_Command1(void)  {   // ST7735 initialization for Red Tab modules
  lcd7735_sendCmd(ST7735_SWRESET);   DelayMs(150);   //  1: Software reset
  lcd7735_sendCmd(ST7735_SLPOUT);    DelayMs(255);   //  2: Out of sleep mode
  ...

  lcd7735_sendCmd(ST7735_INVOFF);     // 13: Don't invert display
  lcd7735_sendCmd(ST7735_MADCTL);     // 14: Memory access control (directions)
    lcd7735_sendData(ST7735_ROTATION);//     row addr/col addr, bottom to top refresh
  lcd7735_sendCmd(ST7735_COLMOD);     // 15: set color mode
    lcd7735_sendData(0x05);           //      16-bit color (5-6-5)
}
```

Az inicializálás az Adafruit_ST7735 library RED_TAB változatú kijelzője konfigurálásának lépéseit követi

Az ST7735 inicializálása és kezelése 3/3. (st7735.c)

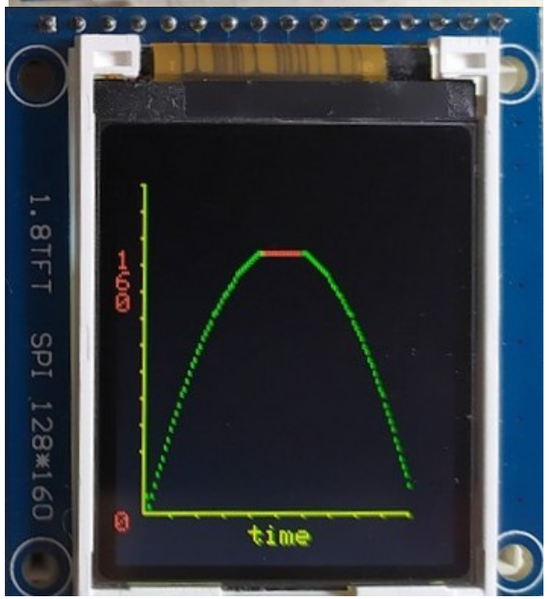
```
void ST7735_Init_Command2(void)
{
// Init for 7735R, part 2 (red tab only)
  lcd7735_sendCmd(ST7735_CASET);      //  1: Column addr set
  lcd7735_sendData(0x00);             //     XSTART = 0
  lcd7735_sendData(0x00);
  lcd7735_sendData(0x00);             //     XEND = 127
  lcd7735_sendData(0x7F);
  lcd7735_sendCmd(ST7735_RASET);      //  2: Row addr set
  lcd7735_sendData(0x00);             //     XSTART = 0
  lcd7735_sendData(0x00);
  lcd7735_sendData(0x00);             //     XEND = 127
  lcd7735_sendData(0x9F);
}
void ST7735_Init_Command3(void)
{
  lcd7735_sendCmd(ST7735_GMCTRP1);    //  Gamma ‘+’polarity Correction Characteristics Setting
  lcd7735_sendData(0x02); ...  és további 15 adat
  lcd7735_sendCmd(ST7735_GMCTRN1);    //  Gamma ‘-’polarity Correction Characteristics Setting
  lcd7735_sendData(0x03);  ...  és további 15 adat
  lcd7735_sendCmd(ST7735_NORON);  DelayMs(10);   // 3: Normal display on, no args, w/delay
  lcd7735_sendCmd(ST7735_DISPON); DelayMs(100);  // 4: Main screen turn on, no args w/delay
}
```

Grafikus elemek kirajzolása

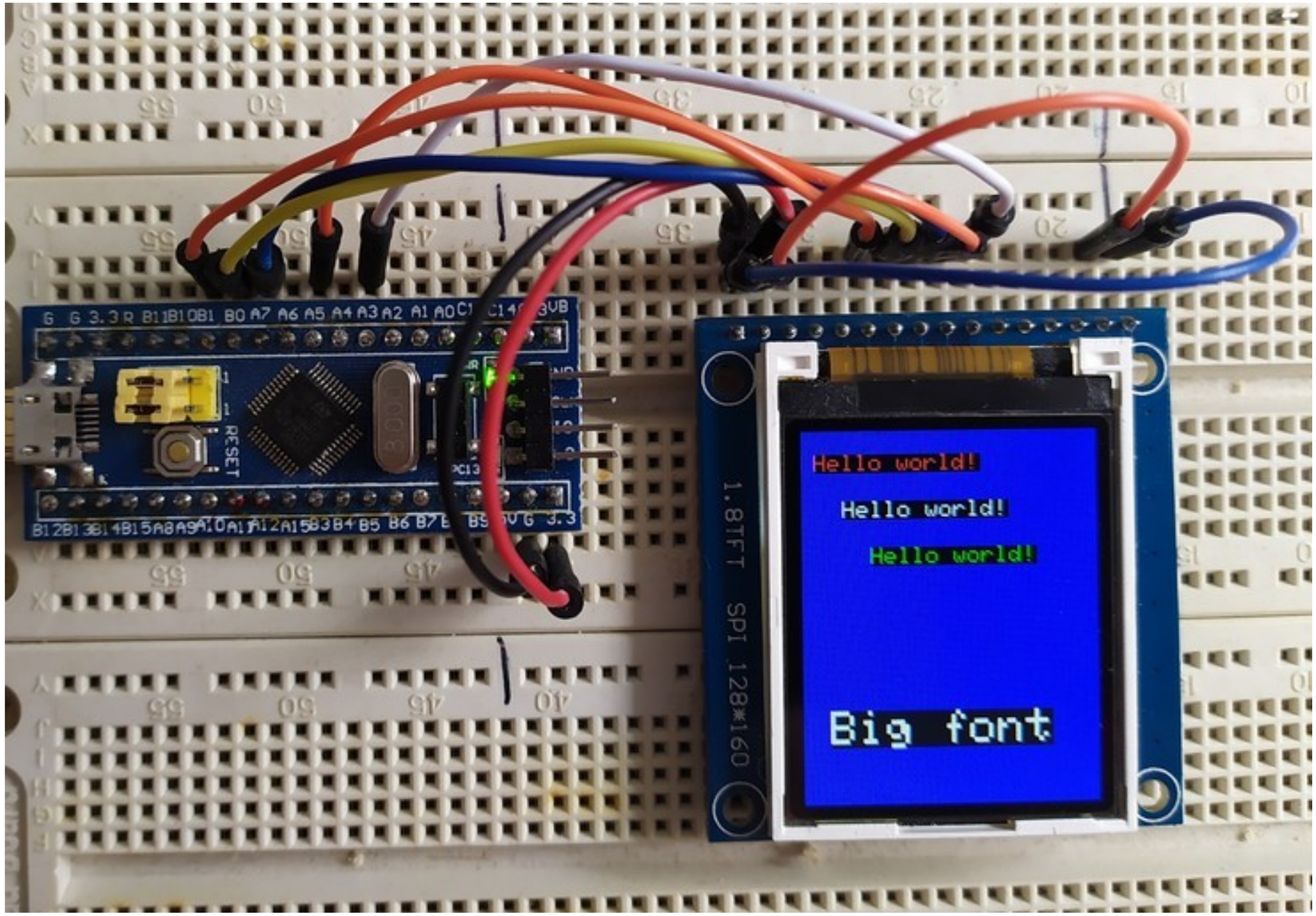
- Ha az inicializálás megtörtént, Dung-Yi Chao STM32F4-ST7735 programkönyvtárában az alábbi általános célú függvények állnak rendelkezésre a rajzoláshoz:
- void **ST7735_DrawPixel**(uint16_t x, uint16_t y, uint16_t color)
- uint32_t **ST7735_DrawString**(uint16_t x, uint16_t y, char *pt, int16_t textColor, uint8_t size);
  (ezt a függvényt módosítottuk, hogy a karakterméret változtatható legyen)
- void **ST7735_DrawCharS**(int16_t x, int16_t y, char c, int16_t textColor, int16_t bgColor, uint8_t size);
- void **ST7735_FillRectangle**(uint16_t x, uint16_t y, uint16_t w, uint16_t h, uint16_t color);
- void **ST7735_FillScreen**(uint16_t color);
- void **ST7735_InvertColors**(bool invert);
- void **ST7735_DrawFastHLine**(int16_t x, int16_t y, int16_t w, uint16_t color);
- void **ST7735_DrawFastVLine**(int16_t x, int16_t y, int16_t h, uint16_t color);

Az első próbálkozás (ST7735_1/main.c)

```c
#include "st7735.h"
int main(void) {
  /* PB0-PB1 konfigurálása digitális kimenetként */
  RCC->APB2ENR |= RCC_APB2ENR_IOPBEN; // GPIOB órajel engedélyezés
  GPIOB->CRL &= ~0x000000FF;          // CNF és Mode bitek törlése
  GPIOB->CRL |=  0x00000033;          // PB0,PB1 Pushpull GPIO kimenet
  SPI1_init();                        // Az SPI1 modul konfigurálása
  ST7735_Init();                      // A kijelző inicializálása
  while(1) {
    ST7735_FillRectangle(0,0,128,160,ST7735_BLUE);
    ST7735_DrawString(1, 1, "Hello world!", ST7735_RED,1);
    ST7735_DrawString(3, 3, "Hello world!", ST7735_WHITE,1);
    ST7735_DrawString(5, 5, "Hello world!", ST7735_GREEN,1);
    ST7735_DrawString(1, 6, "Big font", ST7735_WHITE,2);
    DelayMs(5000);
    ST7735_Drawaxes(ST7735_YELLOW, ST7735_BLACK, "time","0",
                    ST7735_RED,"160", ST7735_RED,160, 0);
    for(int i=0; i<100; i++) {
        ST7735_PlotIncrement();
        ST7735_PlotPoint((2600-(i-50)*(i-50))/16, ST7735_GREEN);
    }
    DelayMs(5000);
  }
}
```



A kísérleti elrendezés és a futási eredmény

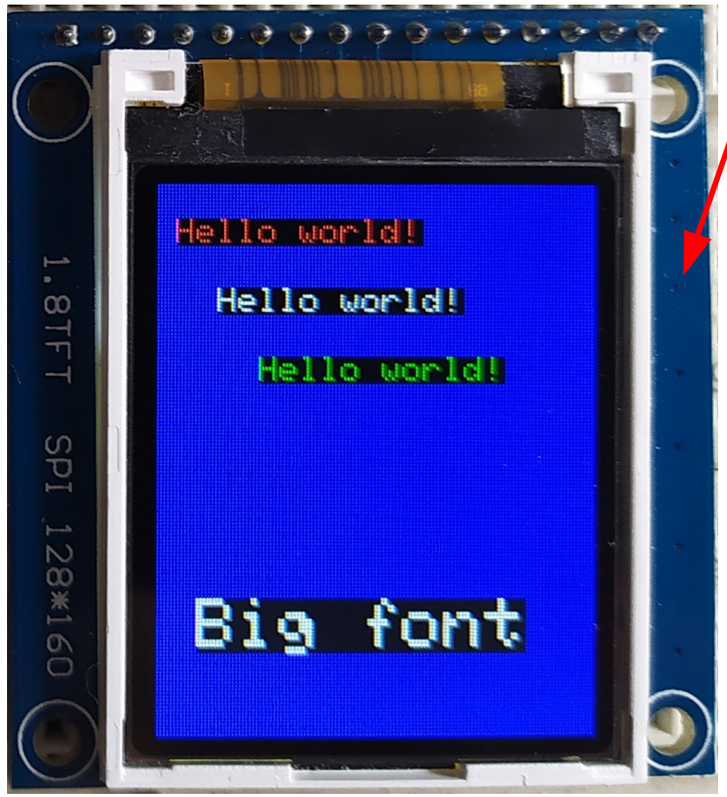


A forgatási paraméter itt nulla értékkel volt definiálva!

A kép elforgatása

- Az **st7735.h** fejléc állományban definiált `ST7735_ROTATION` paraméter átszerkesztésével „fejre állíthatjuk” a képet

```
#define ST7735_ROTATION (ST7735_MADCTL_MX|ST7735_MADCTL_MY)    // 0xC8

#define ST7735_ROTATION 0
```



Hogyan írhatunk a képmemóriába?

- A képmemóriába történő írás előtt ki kell jelölnünk egy téglalap alakú területet a sor- és oszlopindex tartományok megadásával (**RASET** és **CASET** parancsok). Majd kiadunk egy **RAMWR** parancsot és a kijelölt téglalap minden pixeléhez kiküldünk egy 16 bites színkódot
- Mivel a címbeállítás és a **RAMWR** parancs kiadása minden grafikus művelethez szükséges, ezért egy külön függvény szolgál ezek végrehajtására az **st7735.c** forráskódban
- Az **ST7735_XSTART** és **ST7735_YSTART** paraméterek estünkben nullák

```
static void ST7735_SetAddressWindow(uint8_t x0, uint8_t y0, uint8_t x1, uint8_t y1) {
  lcd7735_sendCmd(ST7735_CASET);           // Column addr set
  lcd7735_sendData(0x00);                  // XS15 ~ XS8
  lcd7735_sendData(x0+ST7735_XSTART);      // XSTART       XS7 ~ XS0
  lcd7735_sendData(0x00);                  // XE15 ~ XE8
  lcd7735_sendData(x1+ST7735_XSTART);      // XEND         XE7 ~ XE0
  lcd7735_sendCmd(ST7735_RASET);           // Row addr set
  lcd7735_sendData(0x00);
  lcd7735_sendData(y0+ST7735_YSTART);      // YSTART
  lcd7735_sendData(0x00);
  lcd7735_sendData(y1+ST7735_YSTART);      // YEND
  lcd7735_sendCmd(ST7735_RAMWR);
}
```

Elemi rajzműveletek: pont rajzolása

- **Pont rajzolása:**
  - Ellenőrizzük, hogy a megadott koordináták a képfelületre esnek-e
  - Beállítjuk az indexhatárokat
  - Kiküldünk egy 16 bites színkódot

```
void ST7735_DrawPixel(uint16_t x, uint16_t y, uint16_t color)  {
  if((x >= ST7735_WIDTH) || (y >= ST7735_HEIGHT)) return;
  SPI1_enable();
  ST7735_SetAddressWindow(x, y, x+1, y+1);
  uint8_t data[2];
  data[0] = color >> 8;
  data[1] = color & 0xFF;
  lcd7735_sendData(data[0]);
  lcd7735_sendData(data[1]);
  SPI1_disable();  //unselect
}
```

Elemi rajzműveletek: kitöltött téglalap rajzolása

- A határok ellenőrzése és a címtartományok beállítása után minden képpontot a megadott színkóddal töltjük fel
- Paraméterek: ***x,y*** – a bal felső sarok koordinátái, ***w*** és ***h*** – a téglalap szélessége és magassága, ***color*** – a kirajzolandó téglalap RGB színkódja 5-6-5 felbontásban

```
void ST7735_FillRectangle(uint16_t x, uint16_t y, uint16_t w, uint16_t h, uint16_t color) {
  if((x >= ST7735_WIDTH) || (y >= ST7735_HEIGHT)) return;
  if((x + w - 1) >= ST7735_WIDTH) w = ST7735_WIDTH - x;
  if((y + h - 1) >= ST7735_HEIGHT) h = ST7735_HEIGHT - y;
  ST7735_SetAddressWindow(x, y, x+w-1, y+h-1);
  LCD_DC1;
  SPI1_enable();
  for(y = h; y > 0; y--) {
    for(x = w; x > 0; x--) {
      SPI1_write(color>>8);
      SPI1_write(color);
    }
  }
  SPI1_disable();  //Unselect
}
```

Elemi rajzműveletek: egy karakter kirajzolása

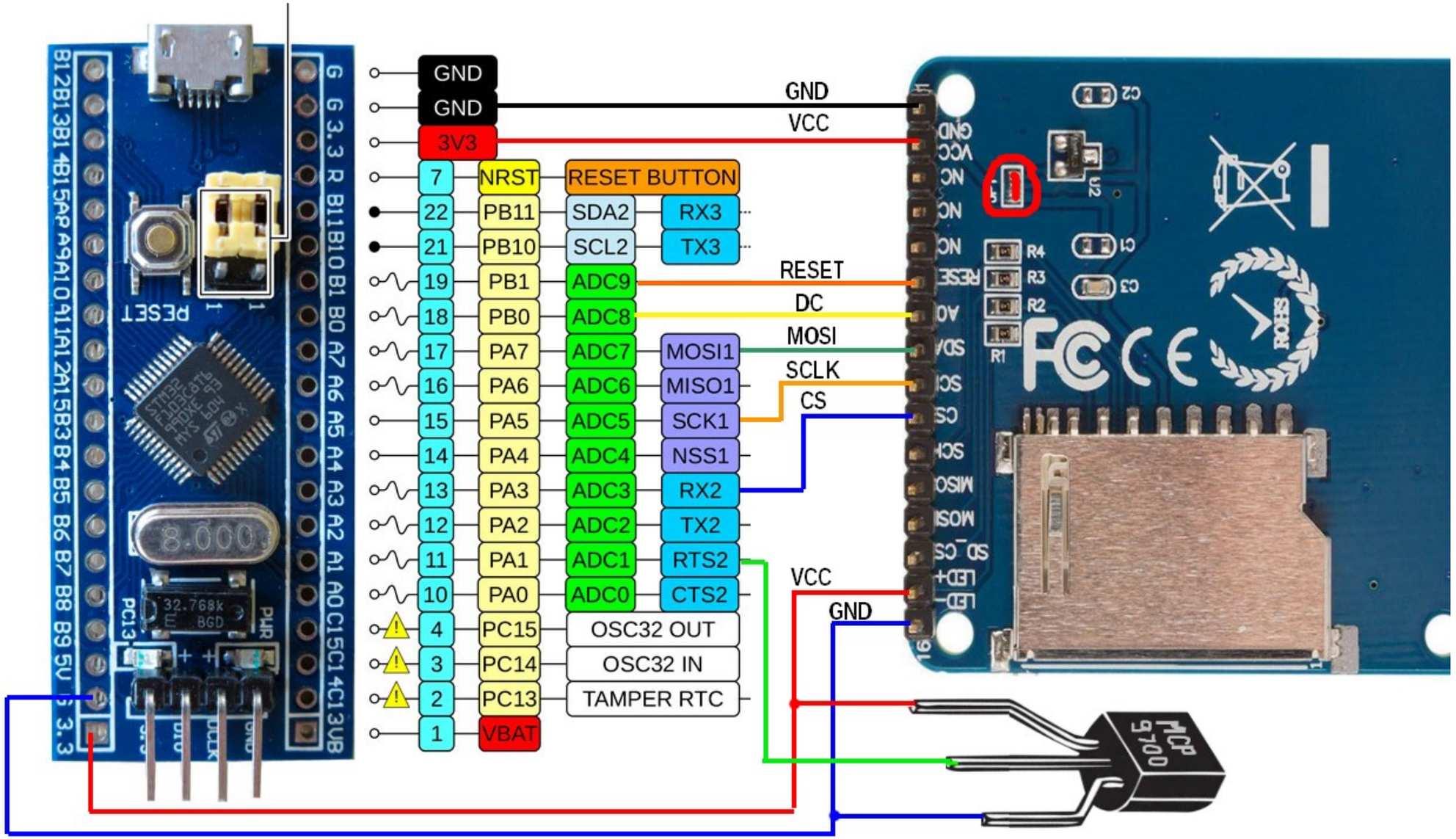
- A `Font[]` tömböt az **st7735.h** fejléc állomány definiálja (5x8 font)
- A ***size*** paraméterrel a pixelek skálázhatók (**1**: 1x1, **2**: 2x2, **3**: 3x3)

```
void ST7735_DrawCharS(int16_t x, int16_t y, char c, int16_t textColor, int16_t bgColor, uint8_t size) {
  uint8_t line; int32_t i, j;
  if((x>=ST7735_WIDTH) || (y>=ST7735_HEIGHT) ||
     ((x+5*size-1)>=ST7735_WIDTH) || ((y+8*size-1) >= ST7735_HEIGHT) ) return;
  for (i=0; i<6; i++ ) {
    if (i == 5) line = 0x0;
    else line = Font[(c*5)+i];
    for (j = 0; j<8; j++) {
      if (line & 0x1) {
        if (size == 1) ST7735_DrawPixel(x+i, y+j, textColor);
        Else { ST7735_FillRectangle(x+(i*size), y+(j*size), size, size, textColor); }
      } else if (bgColor != textColor) {
        if (size == 1) ST7735_DrawPixel(x+i, y+j, bgColor);
        else { ST7735_FillRectangle(x+i*size, y+j*size, size, size, bgColor); }
      }
      line >>= 1;
    }
  }
}
```

Bemutató célú függvények és egy program

- A **Dung-Yi Chao** GitHUB tárhelyén közzétett **STM32F4-ST7735 tutorial** egy egyszerű mintaprogram mellett az alábbi demó célú függvényeket biztosítja:
  - void **ST7735_Drawaxes**(*uint16_t axisColor, uint16_t bgColor, char *xLabel,char *yLabel1, uint16_t label1Color, char *yLabel2, uint16_t label2Color,int32_t ymax, int32_t ymin*) – koordinátatengelyek kirajzolása, feliratozása, skálázása
  - void **ST7735_PlotIncrement**(*void*) – az időskála léptetése, körülfordulás után az új pont helyének törlése
  - void **ST7735_PlotPoint**(*int32_t data1, uint16_t color1*) – egy mérési eredmény kirajzolása
  - void **ST7735_DrawImage**(*uint16_t x, uint16_t y, uint16_t w, uint16_t h, const uint16_t* data*) – bitkép kiírása
- A fenti függvényeket használja az a demó program is, amely a fent említett tutorialban található, s amelyet némiképp átalakítva most bemutatunk (**ST7735_2** projekt néven). A program a **PA1** bemeneten az **ADC1** segítségével 500 ms-onként méri egy **MCP9700** analóg hőmérő jelét, majd az eredményt kijelzi az **ST7735** vezérlőjű kijelzőn

Bekötési vázlat



ST7735_2/main.c 2/1. oldal

Az **ADC1** konfigurálását a **keil19_07** előadás (2019. dec. 5.) **Program07_1** mintaprojektjében mutattuk be

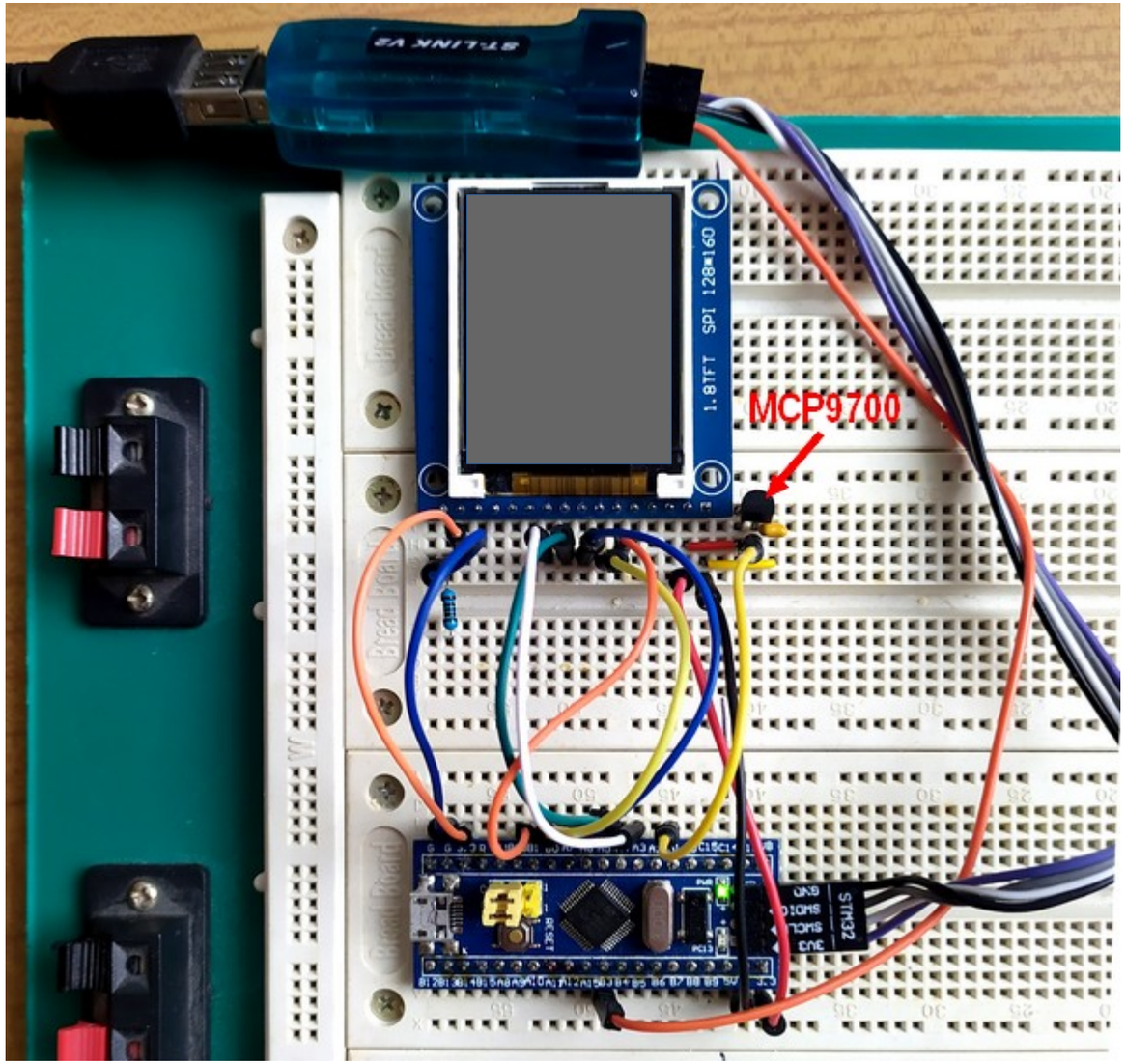
```
#include "st7735.h"
#define YMAX 300
#define YMIN 200

void ADC1_Init() {
  RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;     // GPIOA engedélyezés
  GPIOA->CRL &= ~GPIO_CRL_MODE1;          // PA1 analóg bemenet
  GPIOA->CRL &= ~GPIO_CRL_CNF1;           // CNF=00, Mode=00
  RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;     // ADC1 órajel engedélyezés
  RCC->CFGR &= ~RCC_CFGR_ADCPRE_0;        // ADC prescaler = 6
  RCC->CFGR |= RCC_CFGR_ADCPRE_1;         // ADCCLK = 72/6 = 12 MHz
  ADC1->CR2 = ADC_CR2_EXTTRIG |           // External trigger engedélyezés
              ADC_CR2_EXTSEL;             // EXTSEL=111 SW trigger választása
  ADC1->SQR3 = 1;                         // Ch 1-gyel indul a konverziós sorozat
  ADC1->SQR1 = 0;                         // A konverzió sorozat hossza = 1
  ADC1->CR2 |= ADC_CR2_ADON;              // ADC1 engedélyezése
}

uint16_t ADC1_GetValue(void) {
    ADC1->CR2 |= ADC_CR2_SWSTART;         // Konverzió indítása
    while(!(ADC1->SR & ADC_SR_EOC));      // Konverzió végére várunk
    return ADC1->DR;                      // Az eredmény kiolvasása
}
```

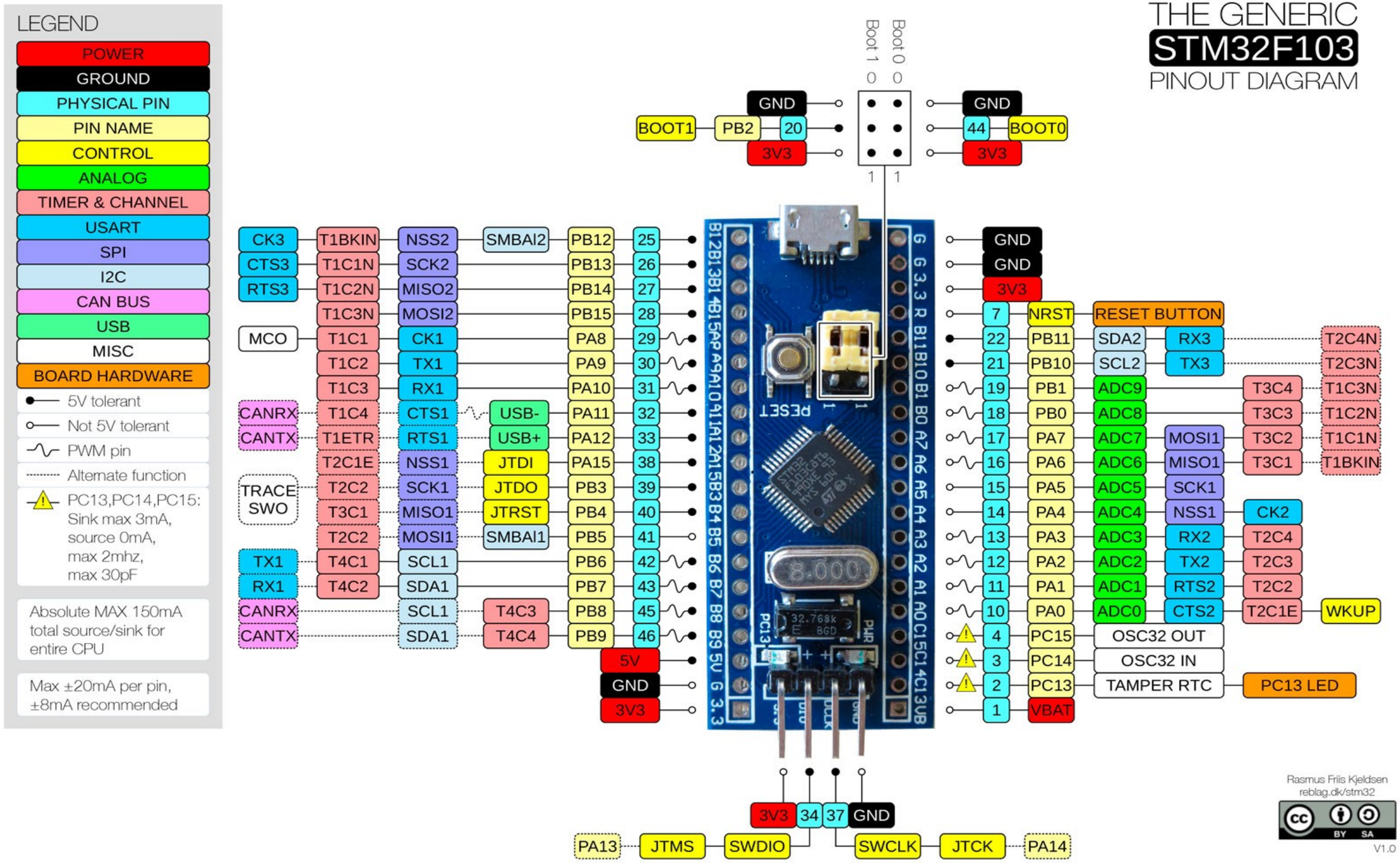
ST7735_2/main.c 2/2. oldal

- Az **ADC1** jelét millivoltokra számítjuk át, és levonjuk az 500 mV-os nullapont eltolást.
- A megjelenítés határa: YMAX = 300 mV, YMIN = 200 mV (20 – 30 °C közötti ábrázolunk)

```
int main(void) {
  /* PB0-PB1 konfigurálása digitális kimenetként */
  RCC->APB2ENR |= RCC_APB2ENR_IOPBEN; // GPIOB órajel engedélyezés
  GPIOB->CRL &= ~0x000000FF;          // CNF és Mode bitek törlése
  GPIOB->CRL |=  0x00000033;          // PB0 és PB1 Pushpull kimenet, max. 50 MHz
  SPI1_init();                        // Az SPI1 modul konfigurálása
  ADC1_Init();
  ST7735_Init();                      // A kijelző inicializálása
  ST7735_FillScreen(ST7735_BLACK);
  ST7735_Drawaxes(AXISCOLOR, BGCOLOR, "Time", "ADC", LIGHTCOLOR, "", 0, YMAX, YMIN);
  while(1) {
    int sensorValue = ADC1_GetValue();
    sensorValue = sensorValue*3300/4096;
    ST7735_PlotPoint(sensorValue-500,ST7735_GREEN);
    ST7735_PlotIncrement();
    DelayMs(500);
  }
}
```

A megépített kapcsolás




LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
THE GENERIC
STM32F103
PINOUT DIAGRAM
Boot 1
Boot 0
GND
BOOT1
PB2
20
3V3
GND
44
BOOT0
3V3
CK3
CTS3
RTS3
MCO
CANRX
CANTX
TRACE SWO
TX1
RX1
CANRX
CANTX
T1BKIN
T1C1N
T1C2N
T1C3N
T1C1
T1C2
T1C3
T1C4
T1ETR
T2C1E
T2C2
T3C1
T2C2
T4C1
T4C2
NSS2
SCK2
MISO2
MOSI2
CK1
TX1
RX1
CTS1
RTS1
NSS1
SCK1
MISO1
MOSI1
SCL1
SDA1
SCL1
SDA1
SMBAI2
USB-
USB+
JTDI
JTDO
JTRST
SMBAI1
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PA11
PA12
PA15
PB3
PB4
PB5
PB6
PB7
PB8
PB9
25
26
27
28
29
30
31
32
33
38
39
40
41
42
43
45
46
5V
GND
3V3
GND
GND
3V3
7
22
21
19
18
17
16
15
14
13
12
11
10
4
3
2
1
NRST
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
VBAT
RESET BUTTON
SDA2
SCL2
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
OSC32 OUT
OSC32 IN
TAMPER RTC
RX3
TX3
MOSI1
MISO1
SCK1
NSS1
RX2
TX2
RTS2
CTS2
T3C4
T3C3
T3C2
T3C1
CK2
T2C4
T2C3
T2C2
T2C1E
PC13 LED
T2C4N
T2C3N
T1C3N
T1C2N
T1C1N
T1BKIN
WKUP
3V3
34
37
GND
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14
Rasmus Friis Kjeldsen
reblag.dk/stm32
V1.0