**Project 0**

**TFT Novatek Library за TFT дисплей**

Библиотеката е създадена за работа с TFT дисплей като е базирана на драйвър NV3041A. Този драйвър е създаден за TFT екрани с размери 480x272 като се използва основно за екрани с този размер и по-малки с по-малка консумация на електричество.

**1.Хардуер**

*1)ESP 32 Devkitc V4 WROOM 32E*

ESP32-DevKitC V4 е малка развойна платка, базирана на ESP32 и произведена от Espressif. Повечето I/O пинове са изведени към пин хедърите от двете страни за лесно свързване с периферни устройства. Разработчиците могат да свързват периферни устройства чрез проводници или да монтират ESP32-DevKitC V4 на експериментална платка.

*2)TFT резистивен дисплей*

1. Тип на Дисплея: 4.3 инчов

2. Подсветка: 7 Бели LED диоди, паралелна свързаност

3. Работна температура: -20°C~+70°C

4. Температура на съхранение: -30°c~+80°c

5. Резолюция: 480xRGBx272

*3)Окабеляване*

**2.CPP част**

Първоначално се правят всички дефиниции на команди които ще се използват част от тях се дефинират в head файлът също. Съответните команди се взимат от документацията на драйвъра като те основно са за ротацията, оразмеряването и записът в ram паметта на дисплея.

След това се добавя и head файлът чрез метода include за да може да се присвоят всички функционалности на дефинираните функции и също да се добавят останалите команди както споменахме по горе. Чрез същият метод се добавя и SPI библиотеката за да може да се дефинира комуникация използвана между ESP 32 платката и дисплея. Библиотеката си присвоява всички SPI дефинирани пинове на платката.

Създава се клас/шина, която описва всички параметри на пиновете като в самия клас се дефинират съответно наименованията на пиновете за основната шина. Следват всички функции нужни за инициализацията на дисплея:

*1)Функция Begin*

Тази функция инициализира SPI шината и TFT контролера. Следователно процесът се изпълнява по следния начин:

Стартира се SPI шината и последователно SPI транзакцията като в тази транзакция се конфигурират следните параметри:  
**1.MSBFIRST(Most Significant Bit First)**

Този параметър е и SPI режим като първо се изпраща най-значимият бит, като последователно се изпращат и останалите за да може да се определи значимостта. Това е стандартен режим за повечето SPI устройства.

**2.SPI Mode 0**

Тази настройка показва и уверява шината, че системният часовник (SCK) е настроен в LOW режим на работа.

След това се настройват CS и DCX пиновете съответно по следният начин:

**1.DCX(Data Command)** пинът се настройва за да изпраща информация съответно OUTPUT и стандартният си режим на работа при изпращане на данни съответно HIGH.

**2.CS(Chip Select)** се настройва да търси tft контролер като отново поставяме OUTPUT. Логиката на работа е, че при инициализация той е настроен в режим HIGH това означава че той премахва всички други избрани контролери за намери съответния с който се работи в момента.

Следва настройка на RESET пин като в този случай той прави reset контролера:

Пинът се настройва да изпраща сигнали съответно OUTPUT и при намерен контролер пинът е в HIGH режим на работа и изпълнява съответния reset като чака 10 милисекунди след това пинът се освобождава и минава в LOW режим на работа той отново изчаква 10 милисекунди. Процесът приключва като пинът се връща в HIGH режим на работа.

Извиква се tftInit функцията за да се инициализира дисплеят. Когато всички процеси в Begin функцията са преминали успешно програмата връща стойност true.

*2)RESET операция за дисплеят*

В този случай пинът се използва за reset на дисплея при изписване на съобщение и loop функция и не обхваща контролерът. Разликата този път е, че в първоначално положение пинът е поставен в LOW режим на работа като се изчакват 100 милисекунди пинът се прехвърля в HIGH режим на работа. По този начин се изпълнява reset минават 200 милисекунди за да се финализира процесът.

*3)Set rotation*

Това е показва в какво положение визуализира дисплея. Настройва се чрез методът switch case. Този метод разпределя всяко едно положение на различни опции като положенията започват от 0 до 270 градуса:

Опция 0: 0 градуса-нормална ориентация

Опция 1: 90 градуса-ротация 90 градуса

Опция 2: 180 градуса-ротация 180 градуса

Опция 3: 270 градуса-ротация 270 градуса

Този метод е лесен за използване от дисплея поради това че в зависимост от ротацията която ни е нужна дисплея може автоматично да избере съответната ротация като опция.

*4)Set text color*

С тази функция задаваме цвят на текста като поставяме параметър text color и стойност color.

*5)Set text size*

С тази функция задаваме големина на текста като поставяме параметър text size и стойност size.

*6)WriteAddrWindow*

Адрес на дисплей прозореца. Тази функция се извиква когато рисуваме фигури на екрана:

1.Първо се изчисляват координатите на прозореца съответно дължина и височина на дисплея.

2.След това се означава обхвата на адресите за колоната и съответно се назначава стойност. Като има начален голям и малък бит и краен голям и малък бит.

3.След това се означава обхвата на адресите за редовете и се назначава също стойност. Аналогично на колоните има начален голям и малък бит и краен голям и малък бит.

4. Всички координати и стойности както крайните и началните битове за колоните и редовете се записват в RAM паметта като функцията завършва с това.

*7)Set cursor*

Създава се курсор за да се определи позицията на текста. В този случай Addr Window се използва за присвояването на адрес на един пиксел на дисплея.

*8) TFT init(инициализация на дисплея)*

Тази функция се използва за цялостната инициализация на дисплея.

Тя започва да работи след като е извикана в begin метода.

1.Първо се изпращат командите за инициализация и верификация на TFT контролера. Команда 0x01прави системен рестарт като изтрива информацията от предишни инициализации, верификации и настройки от контролери. Изчаква 100 милисекунди.

2.Изпраща се команда 0x11, която изключва Sleep Mode, в случай че е включен тази проверка се прави независимо от състоянието на този режим.

3.Преминават още 100 милисекунди. Инициализацията приключва и се изпраща команда Display On чрез метода Write Command която съответно стартира дисплея.

*9)Fill Screen*

Функцията се използва за запълни екрана с цвят който дисплея да покаже след стартиране.

1.Задават се стойности за най-голям и най-малък бит на цвят или в случая пиксел на екрана.

2. Изпраща се команда за запис на стойностите в RAM паметта.

3.DCX пинът се настройва да бъде със стойност HIGH понеже цветът се изпраща като данни.

4.CS пинът се настройва да бъде LOW като по този начин той избира дисплея и контролера който използва съответния дисплей.

5.Чрез методът SPI transfer се изпраща първо най-големият след това останалите и последен се изпраща най-малкият бит.

6.След приключването на трансферът на данни CS пинът минава в стандартния си режим на работа а именно HIGH. По този начин той се откача от дисплея и контролера и изтрива данните за тях при следващо закачане с друг дисплей/контролер.

*10)Invert Display*

Тази функция контролира инверсията на дисплея. Функцията работи чрез if метод по следния начин:

Ако дисплеят бива инвертиран се изпраща команда 0x21 която включва инверсията. Ако не е инвертиран се изпраща команда 0x20 която изключва инверсията. Дори и да не се включва това се прави за проверка в случай че инверсията е включена по стандарт или заводски настройки.

*11) Display On*

Функция която изпраща командата DISP ON за да се включи дисплея.

*12) Display Off*

Функцията която изпраща командата DISP OFF за да се изключи дисплея.

*13)Методът Write Command*

Тази функция се използва за да се изпращат команди чрез SPI транзакции.

1.DCX пинът се поставя в LOW режим за да може да изпраща команди. Като тук той е по стандарт в OUTPUT режим от предишни настройки.

2.CS пинът се поставя в LOW режим за да може да избере и да намери използваният контролер.

3.Чрез SPI Transfer се изпраща съответната команда.

4.След като е изпратена командата CS пинът минава в HIGH режим на работа и по този начин прекъсва връзка с избраният TFT контролер.

*14)Методът Write Data*

Функцията се използва за да изпраща данни чрез SPI транзакции.

1.DCX пинът си остава в HIGH режим понеже се изпращат данни.

Като тук той е по стандарт в OUTPUT режим от предишни настройки.

2.CS пинът се поставя в LOW режим за да може да избере и да намери използваният контролер.

3.Чрез SPI Transfer се изпращат съответните данни.

4.След като е изпратена командата CS пинът минава в HIGH режим на работа и по този начин прекъсва връзка с избраният TFT контролер.

*15)Print*

Функцията се използва за да се изведе информация и текст на дисплей екрана.

1)Чрез Set Text Color и Set Cursor задаваме цвят и позиционираме текста на екрана.

2)Чрез For Loop се задава методът с който се принтира съобщението.

3)Чрез SPI Transfer се изпраща съобщението към дисплея.

**3.Заглавен (Head) файл**

В head файлът съдържа всички променливи функции и параметри както и няколко дефиниции в нашия случай. Там се съдържат всички функции под името чрез което се викат в основния код.

В този файл създаваме библиотеката с метода ifndef който след това проверява дали е създадена библиотеката.

А**)"ifndef**" означава "ако не е дефинирано" и е директива на препроцесора в програмните езици C и C++. Тя се използва, за да провери дали даден макрос или идентификатор е бил дефиниран преди в кода. Ако макросът или идентификаторът не е бил дефиниран, кодовият блок в рамките на "ifndef" и съответното "endif" ще бъде изпълнен.

След това се добавя с метода include библиотеката на SPI(Serial Peripheral Interface) за да може отново да си присвои всички SPI пинове и наименованията им. Добавя се по същия начин и Arduino библиотеката за да може да използваме нейния синтаксис за функциите които създаваме и именуваме в кода като по този начин не получаваме синтактични грешки.

Създава се отново клас/шина както в Cpp файлът под наименованието на нашата библиотека като в него отново се дефинират параметрите, вида на променливите, техните имена и за някои от тях се поставят стойности.

B) В класът се добавят всички функции и променливи като се делят на 3 раздела public, private и protected.Чрез това деление се определят нивата на достъп също се контролира как могат да бъдат достъпвани и използвани съответните членове на класът. Тази концепция се е от съществено значение за инкапсулацията и скриването на информация, тъй като позволява да се изложат необходимите части на от интерфейса на класа, като се запазят останалите детайли на реализацията скрити.

**1) Private:**

Всички членове на класа декларирани в този раздел, са достъпни само в класа в който ги дефинират. Те не могат да бъдат достъпвани или модифицирани извън класа, включително и от производните класове(подкласове). Това инкапсулира вътрешните детайли на реализацията и защитава целостта на данните в класа. В нашата библиотека се дефинират променливи за SPI комуникацията, стойностите за цвят на текста, размер на текста както и методите writeCommand и writeData.

**2)Protected:**

Всички членове на класа декларирани в този раздел са достъпни в рамките на класа, който ги дефинира и всички възможни производни(подкласове).

Това позволява на производните класове да достъпват и модифицират защитените членове, което може да бъде позволено, когато искате да споделите данни между клас и неговите подкласове, но все още да контролирате достъпа. В нашият случай този раздел съдържа всички параметри за настройка на колоните и редовете на екрана както tftinit функцията за инициализация на дисплея.

**3)Public:**

Всички членове на класа декларирани в този раздел, са достъпни за от всяко едно място на библиотеката включително извън класа. Те могат да бъдат достъпвани от всякакъв код, стига да има инстанция на класа в него.

В нашия случай този раздел съдържа begin методът, както всички сетъри, функциите за включване и изключване на дисплея, функцията за ротация и за изпълване на екрана с цвят, print функцията и също функционалностите за адресиране на екрана.

**4.Ino File(главен код/проект/)**

Това е главният проект който се използва за да може дисплея да функционира като от там се задават функциите които трябва да изпълнява.

Тук е цялостното описание на кода като има описани абсолютно всички функционалности и как работят:

1)Кодът започва с включването на необходимия заглавен(head) файл "TFT\_Novatek\_NV3041A.h", който вероятно съдържа дефинициите и функциите, необходими за взаимодействие с TFT дисплея.

***2)Дефиниции на пинове:***

Кодът дефинира няколко константи, които съответстват на пиновете, свързани между Arduino платката и TFT дисплея. Това включва пинове за управление на подсветката (TFT\_BL), нулиране(reset) (RST), избор на чип(chip select) (TFT\_CS), системен часовник(system clock) (TFT\_SCK), серийно предаване на данни от контролера към дисплея (TFT\_MOSI) и предаване на данни/команди (TFT\_DCX).

***3)Инициализация на TFT\_Novatek\_NV3041A:***

Създава се инстанция на класа `TFT\_Novatek\_NV3041A` с конструктора. Конструкторът приема дефинициите на пиновете като параметри, заедно с други настройки като ротация и дали дисплеят използва IPS (In-Plane Switching) панел. Тази инстанция се използва за взаимодействие с дисплея в продължение на скрипта.

***4) Функция за настройка (setup):***

Функцията `setup` е мястото, където се извършва началната настройка на проекта. Започва с инициализация на серийната комуникация за отстраняване на грешки. След това се опитва да инициализира TFT дисплея чрез функцията `tft.begin()` и if метод:

А) Ако инициализацията се провали, се извежда съобщение за грешка, и дисплеят се изключва.

Б) Ако инициализацията е успешна, дисплеят се включва, а екранът се запълва с бял цвят чрез `tft.fillScreen(0xFFFF)`.

Кода също така конфигурира пина за управление на подсветката на TFT (`TFT\_BL`) като OUTPUT и го задава в режим HIGH, за да включи подсветката. Задава се позицията на текстовия курсор, цвета на текста и се извежда "Hello World" на екрана с помощта на функциите `tft.setCursor`, `tft.setTextColor` и `tft.print`.

Накрая, има изчакване 5000 милисекунди.

***5) Функция за цикъл (loop):***

Функцията `loop` е мястото, където се извършва основната операция на скрипта в непрекъснат цикъл. Вътре в цикъла се генерират произволни x и y позиции. Задава се позиция на курсора, цвят и размер на текста чрез функциите които създадохме.

След това скриптът извежда "Hello World!" на генерираната позиция на екрана чрез функцията `tft.print`. Изчакват се 2000 милисекунди

**6) Заключение:**

Кодът инициализира TFT дисплея, запълва екрана с бял цвят, извежда "Hello World" на фиксирана позиция и след това в безкраен цикъл извежда "Hello World!" на случайни позиции на екрана с променящи се цветове и размери на всеки 2 секунди.

Библиотеката може да се използва без проблеми стига да използва драйвър NV3041A за да функционират всички дефинирани команди. Тя може да се доработва още и да се развива за по големи екрани и проекти.