STM32 ile P4 LED display çalıştırmak

1. P4 Display ve STM32 Pin Bağlantıları

P4 LED displayler genellikle bir **HUB75** arayüzü kullanır. Bu arayüz, veri ve kontrol sinyallerini taşır. STM32 ile P4 displayi bağlarken aşağıdaki pinlere ihtiyacınız olacak:

- **R, G, B (Kırmızı, Yeşil, Mavi):** Renk verilerini taşır.
- **A, B, C, D (Satır Seçim Pinleri):** Displaydeki satırları seçmek için kullanılır. 4-bit adresleme ile 16 satır kontrol edilebilir ($2^4 = 16$). Bazı displaylerde daha fazla satır için E ve F gibi pinler de olabilir.
- **CLK (Clock - Saat):** Verilerin displaye yazılması için senkronizasyon sinyali.
- **LAT (Latch - Mandallama):** Verinin bir satır için tamamlandığını ve displaye gönderilmeye hazır olduğunu belirtir.
- **OE (Output Enable - Çıkış Etkinleştirme):** Displayin çıkışlarını etkinleştirir veya devre dışı bırakır (karartma/parlaklık kontrolü için kullanılabilir).

Bu pinleri STM32'nizin **GPIO** (General Purpose Input/Output) pinlerine bağlamanız gerekir. Hangi GPIO pinlerini kullanacağınız tamamen size kalmıştır. Ancak, hızlı veri transferi için **DMA (Direct Memory Access)** özelliğini kullanmak istiyorsanız, bazı GPIO pinlerinin belirli donanım modülleriyle eşleştiğini göz önünde bulundurmalısınız.

2. Güç Kaynağı

P4 LED displayler, özellikle tam parlaklıkta çalışırken, önemli miktarda akım çekerler. Bu nedenle, display için ayrı bir **5V güç kaynağı** kullanmalısınız. Displayin güç gereksinimlerini (genellikle 4A civarı) karşılayabilecek kapasitede bir adaptör seçtiğinizden emin olun. STM32'nizi bu güç kaynağına bağlamayın, STM32 kendi güç kaynağından beslenmelidir.

3. Yazılım Geliştirme Yaklaşımı

P4 displayleri kontrol etmek, sürekli ve hızlı bir şekilde veri göndermeyi gerektirir. Bu, mikrokontrolcü üzerinde bir dizi zorluğu beraberinde getirir. İki temel yazılım geliştirme yaklaşımı bulunmaktadır:

A. Timer ve Interrupt Kullanımı

Bu yaklaşım, mikrodenetleyicinin çekirdeğini meşgul etmeden veri göndermeyi sağlar.

1. **Timer Ayarı:** Yüksek frekansta çalışan bir timer (zamanlayıcı) ayarlayın. Bu timer, displayin her satırını yenilemek için kullanılacak döngüyü tetikleyecektir.
2. **Kesme Servisi (Interrupt):** Her timer kesmesinde, bir sonraki satırın verilerini displaye gönderecek bir kesme hizmet rutini (**ISR - Interrupt Service Routine**) yazın.
3. **Çift Tamponlama (Double Buffering):** Görüntü güncellemeleri sırasında ekranda titreme veya bozulma olmaması için çift tamponlama tekniğini kullanın. Bir tampon, displaye gönderilecek veriyi tutarken, diğer tamponda yeni görüntü verilerini hazırlarsınız. Hazırlık tamamlandığında, tamponları değiştirirsiniz.

Bu yöntem, işlemcinin diğer görevleri yapmasına olanak tanır, ancak hızlı bir şekilde veri aktarımı için donanım desteği olmadan yeterli olmayabilir.

B. DMA (Direct Memory Access) Kullanımı

P4 display kontrolü için **en etkili ve profesyonel yöntem** DMA kullanmaktır.

1. **DMA ve Timer/SPI:** DMA, işlemciye yük bindirmeden bellekten çevresel birime (bu durumda GPIO'lara) veri aktarımı yapar. Bir timer veya SPI modülü, DMA transferlerini tetiklemek için kullanılabilir.
2. **Transfer Şablonu:** DMA ile veri göndermek için, displayin gerektirdiği tüm sinyalleri (CLK, LAT, OE, R, G, B, A, B, C, D) bit seviyesinde kodlayan bir veri şablonu oluşturmalısınız.
3. **ISR ile Satır Kontrolü:** Her satır için DMA transferi tamamlandığında bir DMA kesmesi tetiklenir. Bu kesme içinde, bir sonraki satırın adres pinlerini (A, B, C, D) güncelleyip bir sonraki DMA transferini başlatırsınız.

Bu yaklaşım, çok daha akıcı ve titremesiz bir görüntü elde etmenizi sağlar çünkü veriyi çok daha hızlı ve kesintisiz bir şekilde aktarabilir. Bu yöntem genellikle C++ gibi dillerde STM32 HAL veya LL kütüphaneleriyle uygulanır.

4. Geliştirme Ortamı ve Kütüphaneler

- **IDE: STM32CubeIDE**, STM32 geliştirme için en yaygın kullanılan ücretsiz entegre geliştirme ortamıdır.
- **Geliştirme Kiti: STM32 Nükleo** veya **STM32 Keşif** kartları, başlamak için harika seçeneklerdir.
- **Kütüphane:** Piyasada bu iş için yazılmış hazır kütüphaneler de bulabilirsiniz. Örneğin, **`STM32F4-P4-Display-Driver`** gibi GitHub projeleri size iyi bir başlangıç noktası olabilir. Hazır kütüphaneler, DMA ve timer yapılandırması gibi karmaşık kısımları sizin yerinize halleder, böylece sadece görüntü verilerini hazırlamaya odaklanabilirsiniz.

Özetle İzlenecek Yol

1. **Bağlantıları Kurun:** P4 displayi, ayrı bir 5V güç kaynağı kullanarak STM32'nize bağlayın.
2. **Yazılım Yaklaşımınızı Belirleyin:** Başlangıç için Timer/Interrupt yaklaşımı basit olabilir. Profesyonel ve akıcı bir sonuç için DMA yaklaşımını tercih edin.
3. **Projenizi Oluşturun:** STM32CubeIDE'de yeni bir proje başlatın, pinleri ve donanım birimlerini (timer, DMA) yapılandırın.
4. **Kodu Yazın:** Displayi kontrol edecek C/C++ kodunu yazın veya hazır bir kütüphaneyi entegre edin.
5. **Test Edin ve Hata Ayıklayın:** Kodunuzu karta yükleyin, displayde görüntü olup olmadığını kontrol edin ve gerekirse hata ayıklama yaparak problemleri giderin.

En çok kullanılan ve kullanılması faydalı kütüphaneler

STM32 ile P4 LED display çalıştırmak için en çok kullanılan ve en faydalı kütüphaneler genellikle doğrudan bu işe özel yazılmış, düşük seviyeli donanım erişimine odaklanmış kütüphanelerdir. Bu tür kütüphaneler genellikle ya özel bir projeye ait ya da GitHub gibi platformlarda paylaşılan açık kaynaklı projelerdir.

İşte en faydalı kütüphane türleri ve bunlara örnekler:

---

1. STM32 HAL ve LL Kütüphaneleri

Bu kütüphaneler, STM32 geliştirme sürecinin temel taşlarıdır. STMicroelectronics tarafından sağlanır ve donanıma doğrudan erişim sağlar. Aslında, P4 display kontrolü için bir kütüphane kullanmasanız bile, bu kütüphanelerin fonksiyonlarına ihtiyacınız olacaktır.

- **STM32 HAL (Hardware Abstraction Layer - Donanım Soyutlama Katmanı):** Daha yüksek seviyeli ve kullanımı daha kolay bir arayüz sunar. Sadece fonksiyonları çağırarak donanım işlemlerini gerçekleştirebilirsiniz. `HAL_GPIO_WritePin`, `HAL_TIM_Base_Start_IT` gibi fonksiyonlar bu kütüphaneye aittir.
- **STM32 LL (Low-Level - Düşük Seviyeli):** Daha çok register seviyesinde programlama yapanlar için tasarlanmıştır. Çok daha hızlı ve düşük bellek tüketimine sahiptir. Eğer performans sizin için çok önemliyse, bu kütüphaneyi tercih edebilirsiniz.

Bu kütüphaneler, bir proje oluşturduğunuzda **STM32CubeIDE** içinde otomatik olarak gelir. Yani ekstra bir indirme yapmanız gerekmez.

---

2. P4 Display Kontrol Kütüphaneleri (Üçüncü Parti)

Bu kütüphaneler, STM32'nin donanım özelliklerini (Timer, DMA, GPIO) kullanarak P4 displayleri sürmek için özel olarak yazılmıştır. Bu kütüphaneler, P4 displayin yenileme (refresh) döngüsü, renk verilerinin sıralanması ve DMA transferlerinin yönetimi gibi karmaşık işleri sizin için halleder.

- **Örnek Proje Kütüphaneleri:** Genellikle GitHub'da "STM32 P4 display driver" veya "STM32 HUB75 library" gibi aramalarla bulabileceğiniz açık kaynaklı projelerdir. Bu kütüphaneler, genellikle belirli bir STM32 mikrodenetleyici modeli için yazılmış olup, kodu inceleyerek kendi projenize kolayca uyarlayabilirsiniz.
    - **Neden Faydalı?** Bu kütüphaneler, DMA ve Timer ayarlarını nasıl yapacağınızı anlamanız için harika birer kaynaktır. Aynı zamanda, karmaşık bit manipülasyonları ve yenileme döngüsü mantığını önceden çözdükleri için size çok zaman kazandırırlar.

---

3. Grafik Kütüphaneleri

Eğer display üzerinde metin, şekiller veya basit animasyonlar göstermek istiyorsanız, bir grafik kütüphanesine ihtiyacınız olacaktır. Bu kütüphaneler, piksellerin tek tek nasıl çizileceğini yönetir ve P4 display sürücü kütüphanesinin üzerine oturur.

- **Adafruit GFX Library:** Popüler bir grafik kütüphanesidir. STM32 için doğrudan uyarlanabilir. Metin, çizgiler, daireler gibi temel grafik öğelerini çizmek için basit fonksiyonlar sunar.
- **LVGL (Light and Versatile Graphics Library):** Daha karmaşık bir grafik arayüzü oluşturmak için tasarlanmıştır. Eğer displayinizde menüler, butonlar ve dinamik arayüzler oluşturmak istiyorsanız, bu kütüphane çok güçlü bir seçenektir.

Bu kütüphaneler, çizmek istediğiniz şeyin piksel verisini oluşturur. Oluşturulan bu veriyi daha sonra P4 display kontrol kütüphanesine gönderirsiniz.

Önerilen Geliştirme Yolu

1. **STM32CubeIDE** ile bir proje başlatın ve **HAL** kütüphanesini kullanarak temel donanım ayarlarını yapın (GPIO pinlerini, Timer ve DMA'yı ayarlayın). Bu, P4 displayin temel sürüşü için en önemli adımdır.
2. **Üçüncü parti** bir P4 display sürücü kütüphanesi bulun (GitHub üzerinden). Bu kütüphanenin, **DMA** kullanarak veri aktarımını gerçekleştirdiğinden emin olun. Bu, en akıcı performansı sağlayacaktır.
3. Eğer sadece basit metin veya şekiller göstereceksenız, **Adafruit GFX** kütüphanesini projenize entegre edin. Eğer daha zengin bir arayüz hedefliyorsanız, **LVGL**'i araştırın.

DMA (Direct Memory Access)

DMA Neden Bu Kadar Önemli?

P4 LED displayler, satır-satır (row-by-row) mantığıyla çalışır. Ekranda 16 satır varsa, mikrodenetleyici bu 16 satırı çok hızlı bir şekilde art arda yenilemelidir. Bu yenileme döngüsü saniyede yüzlerce kez tekrarlanır. Her satır için:

1. Veri (renk bilgisi) serileştirilerek displaye gönderilir.
2. Saat (CLK) sinyali veriye eşlik eder.
3. Verinin displaye aktarıldığı belirtilir (LAT).
4. Çıkışlar etkinleştirilir (OE).

Bu işlemlerin her birini CPU'nun doğrudan yapması, işlemciyi çok büyük oranda meşgul eder ve başka hiçbir işe vakit ayıramaz hale getirir. İşte bu noktada DMA devreye girer.

DMA, bellekten bir çevre birimine (bu durumda GPIO'lara) veri transferini **CPU'ya ihtiyaç duymadan** yapabilen özel bir donanım birimidir. CPU bir DMA transferini başlattıktan sonra, kendi işlerine devam edebilir ve veri transferi tamamlandığında bir kesme (interrupt) alarak bilgilendirilir.

DMA ile P4 Display Kontrolü Mantığı

Bu yöntemin temelinde, displaye gönderilecek verilerin bir "şablon" olarak bellekte hazırlanması yatar. Bu şablon, displayin pinlerine karşılık gelen bitleri içeren bir bayt dizisidir.

1. Veri Şablonunun Oluşturulması

Bu, projenin en karmaşık kısmıdır. P4 displayin bir satırına ait tüm bilgiyi tek bir veri yapısında toplarız. Bu yapı genellikle her pikselin rengini ve kontrol sinyallerini (CLK, LAT, OE) içerir.

- Her bir piksel için renk bilgisi (R, G, B) bayt dizisine yerleştirilir.
- Her pikselin ardından, bir saat (CLK) darbesini temsil eden bir bayt eklenir.
- Bir satırın tüm piksel verileri bu şekilde sıralandıktan sonra, mandallama (LAT) ve çıkış etkinleştirme (OE) sinyallerinin durumunu belirten kontrol baytları eklenir.

Örnek bir veri baytı yapısı:

Bit 7	Bit 6	...	Bit 3	Bit 2	Bit 1	Bit 0
OE	LAT	...	CLK	R	G	B

Her bir veri baytı, o an displayin hangi pinlerinin HIGH, hangilerinin LOW olacağını belirler.

2. DMA'nın Yapılandırılması

STM32CubeIDE'de, Timer (Zamanlayıcı) ve DMA birimini birlikte kullanmak için yapılandırmanız gerekir.

- **Timer (Zamanlayıcı):** Veri transferinin ne kadar hızlı yapılacağını belirlemek için bir timer kullanılır. Timer, belirli bir frekansta overflow yaparak (taşma yaparak) DMA transferini tetikler. Bu frekans, displayin piksel saat hızını (pixel clock rate) belirler.
- **DMA Kanalı:** Bir DMA kanalını GPIO portuna yönlendirilir. DMA'nın kaynak adresi, yukarıda hazırladığınız veri şablonunun başlangıç adresi olur. Hedef adres ise, GPIO çıkış veri kaydedicisidir (örneğin, `GPIOA->ODR`).
- **Transfer Tipi:** DMA'yı bellekten çevre birimine (Memory to Peripheral) veri aktarımı için yapılandırırsınız. Transferin bayt, yarım kelime (half-word) veya tam kelime (word) transferi olacağını belirtmeniz gerekir.

3. Yenileme Döngüsünün Yönetimi

Bu döngü, timer ve DMA kesmeleriyle yönetilir.

1. **İlk DMA Transferi Başlatılır:** CPU, ilk satırın verilerini içeren DMA transferini başlatır.
2. **DMA Veriyi Aktarır:** CPU başka işler yaparken, DMA donanımı veriyi hızla GPIO'lara aktarır. Her veri baytı, displayin pinlerinin durumunu anlık olarak değiştirir.
3. **DMA Kesmesi (DMA Interrupt):** Tüm bir satırın verisi aktarıldığında, DMA transferi tamamlanır ve bir kesme (interrupt) tetiklenir.

4. **Kesme Servis Rutini (ISR):** Bu kesme rutini içinde,
    - Bir sonraki satırın adres pinleri (A, B, C, D) güncellenir.
    - Bir sonraki satırın verisini içeren DMA transferi başlatılır.
    - Bu döngü, tüm satırlar yenilenene kadar devam eder.

DMA Kullanımının Faydaları

- **Yüksek Kare Hızı:** CPU'ya yük binmediği için, çok yüksek hızda veri aktarımı sağlanır. Bu, akıcı animasyonlar ve video benzeri görüntüler elde etmenizi sağlar.
- **Düşük CPU Yükü:** CPU, diğer görevleri (sensör okuma, haberleşme protokollerini çalıştırma vb.) yapmak için serbest kalır.
- **Azaltılmış Titreme (Flicker):** Görüntü, kesintisiz bir şekilde yenilendiği için insan gözü tarafından algılanan titreme en aza iner.

Sonuç olarak, STM32 ile P4 display kontrolü için DMA kullanımı sadece bir seçenek değil, yüksek performanslı bir uygulama için bir zorunluluktur. Bu yöntem, mikrokontrolcünün tüm potansiyelini kullanarak profesyonel sonuçlar elde etmenizi sağlar.